VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE ANOMÁLIÍ V TEMPORÁLNÍCH DATECH

ANOMALY DETECTION IN TEMPORAL DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JOZEF ONDRIA**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

160619

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Ondria Jozef, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Informační systémy a databáze
Název:	**Detekce anomálií v temporálních datech**
Kategorie:	Data mining
Akademický rok:	2023/24

Zadání:

1. Seznamte se s problematikou získávání znalostí z dat, zaměřte se na problematiku časových řad a na detekci odlehlých hodnot a neobvyklých událostí.
2. Seznamte se s dostupnými datovými sadami temporálního charakteru a prostudujte knihovny poskytující vhodné metody pro detekci anomálií v časových řadách.
3. Po dohodě s vedoucím zvolte konkrétní algoritmy pro detekci neobvyklých událostí v temporálních datech a prostudujte je.
4. Zvolené algoritmy implementujte.
5. Implementované algoritmy experimentálně ověřte a porovnejte na vhodné datové sadě.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování v tomto projektu.

Literatura:

- Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A Review on Outlier/Anomaly Detection in Time Series Data. ACM Comput. Surv. 54, 3, Article 56 (April 2022), 33 pages.
- H. -S. Wu: A survey of research on anomaly detection for time series," 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2016, pp. 426-431.
- Aggarwal, C.: Outlier Analysis. Springer, 2013.

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:	1.11.2023
Termín pro odevzdání:	17.5.2024
Datum schválení:	13.2.2024

Abstrakt

Cieľom tejto práce je výber moderných techník a algoritmov pre detekciu anomálií v temporálnych dátach. Vybrané algoritmy je potrebné implementovať a následne experimentálne overiť a porovnať ich funkčnosť na vhodnej dátovej sade.

Abstract

This thesis aims to select modern techniques and algorithms for anomaly detection in temporal data. It is necessary to implement the selected algorithms and then experimentally verify and compare them on a suitable dataset.

Kľúčové slová

získavanie znalostí z databáz, dolovanie z dát, detekcia anomálií, detekcia odľahlých hodnôt, temporálne dáta, časové rady

Keywords

knowledge discovery in databases, data mining, anomaly detection, outlier detection, temporal data, time series

Citácia

ONDRIA, Jozef. *Detekce anomálií v temporálních datech*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Detekce anomálií v temporálních datech

Prehlásenie

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

.....
Jozef Ondria
13. mája 2024

Poďakovanie

Týmto ďakujem vedúcemu práce Ing. Vladimírovi Bartíkovi, Ph.D. za odborné vedenie a pomoc pri tvorbe tejto diplomovej práce.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Spoločnosť sa posledné roky nachádza v období, ktoré môžeme nazvať ako doba veľkých dát. Je to tak preto, že z každej strany, resp. z každého odboru prichádza veľké množstvo údajov, čím je každým rokom ťažšie a ťažšie tieto údaje analyzovať. Aj kvôli tomu sa veľmi dôležitým odborom v informatike stalo získavanie znalostí z databáz, ktoré má za úlohu z týchto veľkých dát získať potrebné informácie.

V tomto odbore existuje niekoľko typov úloh a jedným z nich je detekcia odľahlých hodnôt. Síce sú vo väčšine prípadov tieto anomálne údaje nepotrebné, prípadne sú znakom zlej kvality vstupných údajoch, tak existujú prípady, kde sú práve anomálie bodom záujmu.

Cieľom tejto práce je vytvoriť aplikáciu s niekoľkými významnými experimentami, ktorých úlohou bude detekcia odľahlých hodnôt v temporálnych dátach, a to s využitím moderných dostupných algoritmov určených na riešenie daného problému. U týchto algoritmov je následne potrebné overiť ich funkčnosť a vzájomne ich porovnať na vhodnej dátovej sade.

Obsah tejto práce je logicky usporiadaný v niekoľkých kapitolách. V kapitole 2 sa nachádza teoretický základ pre pochopenie problematiky získavania znalostí z databáz zo všeobecného hľadiska. Je v nej okrem definície tohto odboru popísaný jeho proces a nakoniec aj najčastejšie typy úloh, ktoré tento odbor zahŕňa.

Nadväzujúca kapitola 3 tiež poskytuje teoretický výklad. V tomto prípade však ide o teóriu špecializovanú na riešený problém, ktorým je detekcia anomálií v temporálnych údajoch. V tejto časti textu sú najprv predstavené základne pojmy, a následne popísané praktické využitie i vlastnosti techník pre detekciu anomálií v časových radoch.

V úvode kapitoly 4 sú popísané vybrané dátové sady, ktoré sú vhodné pre experimentálne overenie funkcionality detekčných algoritmov. Následne sú predstavené technológie, ktoré umožňujú detekciu odľahlých hodnôt v časových radoch. Nakoniec sú vymenované vybrané detekčné algoritmy spolu s ich stručným popisom.

V kapitole 5 je uvedený stručný návrh webovej aplikácie a jej implementácia. V závere tejto časti je popísaná finálna funkčnosť implementovanej webovej aplikácie.

Ďalšou časťou práce je kapitola 6. V nej sú popísané vykonané experimenty pre overenie a porovnanie funkčností implementovaných algoritmov. Tieto experimenty sú popísané v rámci dvoch hlavných experimentálnych úloh. Záverečnou časťou kapitoly sú príklady praktického využitia experimentov detekcie anomálií v časových radoch.

Poslednou kapitolou je kapitola 7. Ide o záver práce v ktorom sú spomenuté výsledky práce. Okrem toho je v závere tiež uvedený námet na ďalšie pokračovanie v tejto diplomovej práci.

Kapitola 2

Získavanie znalostí z databáz

Táto kapitola je zameraná na vysvetlenie teoretického základu ohľadom získavania znalostí z databáz. Je dôležitá pre pochopenie ďalších časti tejto práce.

Prvá časť textu je venovaná definícii získavania znalostí z databáz, a to hlavne vymedzením toho, kedy sa dá hovoriť o získavaní znalosti z dát a kedy nie. Následne je podrobne vysvetlený proces získavania znalostí z databáz, a nakoniec sú popísané najrozšírenejšie typy dolovacích úloh v rámci tohto odboru.

2.1 Definícia získavania znalostí z databáz

Získavanie znalostí z databáz (angl. *knowledge discovery in databases*) je populárny odbor informačných technológií, ktorý nadobúda na dôležitosti práve v tomto období, ktoré je možné nazvať ako obdobie **veľkých dát** (angl. *big data*). Dáta sú dnes v každom sektore produkované v obrovských objemoch a je veľmi užitočné z nich získať hodnotné informácie na analýzu alebo rozhodovanie v súkromných i verejných organizáciach. A o toto sa práve usiluje odbor získavania znalostí z databáz.

V odbornej literatúre je tento termín často zamieňaný za termín **dolovanie z dát** (angl. *data mining*), pričom je chápaný ako jeho synonymum. V tejto práci sa však oba termíny rozlišujú – pojem dolovanie z dát je v nej chápaný ako jeden z konkrétnych krokov tvoriacich práve proces získavania znalostí z databáz. Tento krok procesu je podrobnejšie vysvetlený v ďalšej časti tejto kapitoly.

Formálnejšie môžeme získavanie znalostí z databáz definovať ako netriviálny proces identifikácie nových, platných, potenciálne užitočných a v konečnom dôsledku pochopiteľných vzorov. Mieri na vzory, ktoré [15]:

- nevedú k priamemu výpočtu preddefinovaných veličín (netriviálne),
- sa môžu použiť na nové dáta s nejakou určitosťou (platné),
- boli doteraz neznáme (nové),
- poskytujú istý úžitok k ďalším úlohám (užitočné),
- vedú k novým užitočným poznatkom (pochopiteľné).

Správnou interpretáciou spomínaných vzorov sú následne nadobudnuté samotné znalosti. Zaujímavé a zároveň mätúce býva určenie hranice toho, čo môžeme označiť za tento

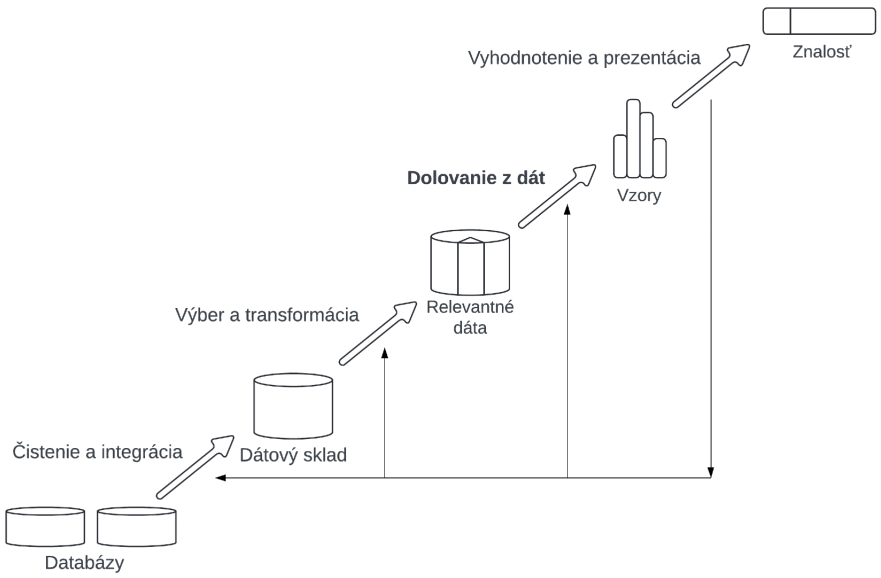
proces a čo už nie. Napríklad platí, že výsledky, ktoré sa dajú dostať prostredníctvom príkazu `SELECT` programovacieho jazyka SQL, neoznačujeme za získavanie znalostí z databáz. Tento postup je príliš jednoduchý, a teda porušuje podmienku netriviálnosti.

2.2 Proces získavania znalostí z databáz

Ako už bolo spomenuté, získavanie znalostí z databáz je proces, ktorý pozostáva z niekoľkých konkrétnych krokov. V rôznej literatúre je počet týchto krokov rozdielny, keďže čiastkové kroky rôzne spájajú. V tejto práci sú preto tieto kroky rozdelené do štyroch väčších krokov, ktoré postupne zodpovedajú krokom z obrázku 2.1.

Úlohou prvého z nich je výber, čistenie a integrácia vhodných dát z viacerých databáz do dátového skladu tak, aby boli pripravené pre vykonanie rôznych úloh dolovania. Ďalším krokom je výber a transformácia dát z dátového skladu do podoby vhodnej pre konkrétnu úlohu dolovania. Uvedené prvé dva kroky sa však zvyknú uvádzať aj ako jeden komplexnejší krok, ktorý sa nazýva **predspracovanie dát**. Tretím krokom je samotné dolovanie z dát, a to pomocou nejakého dolovacieho algoritmu. Výstupom tohto kroku sú vzory, ktoré sa v úplne poslednom kroku pretvoria na znalosti. Ide o krok vyhodnotenia a prezentácie. Všetky tieto kroky sú bližšie rozobrané v nasledujúcom texte.

Dôležité je tiež poznamenať, že tento proces je iteratívny. Jeho vykonávanie sa opakuje s cieľom získať čo najkvalitnejšie znalosti. Príkladom môže byť úprava vstupných parametrov dolovacieho algoritmu pre získanie lepších výsledkov.



Obr. 2.1: Proces získavania znalostí z databáz (zdroj: [9])

2.2.1 Čistenie a integrácia dát

Čistenie a integrácia sú čiastkové kroky, ktoré sú prvou časťou predspracovania dát pre dolovanie.

Cieľom **integrácie dát** je naplnenie dátového skladu dátami z viacerých zdrojov. Na druhej strane **čistenie dát** rieši problém nekvality dát. Existujú tri základné prejavy nekvality dát, ktoré práve čistenie rôznymi formami rieši: [27]:

- **Dáta sú nekompletné** – Môže ísť o jednotlivé hodnoty ale aj celé atribúty, potrebné pre dolovanie.
- **Dáta sú zašumené** – Atribúty obsahujú nesprávne alebo odľahlé hodnoty. Príkladom môže byť hodnota teploty pacienta $10°C$.
- **Dáta sú nekonzistentné** – Potenciálnym zdrojom je každá redundancia dát, ku ktorej môže dôjsť pri použití dát z niekoľkých dátových zdrojov.

Čiastkové kroky rozoberané v tejto podkapitole sú typicky vykonávané pomocou procesu nazývaného skratkou **ETL** (angl. *extraction, transformation and loading*), čiže procesu extrakcie, transformácie a načítania.

ETL proces sa používa na extrakciu dát z interných a externých zdrojov organizácie, transformáciu týchto dát a ich načítanie do dátového skladu. Ako už naznačuje samotný názov, ide o proces pozostávajúci z týchto troch krokov [24]:

- **Extrakcia** zhromažďuje dáta z viacerých rôznorodých zdrojov dát. Týmito zdrojmi môžu byť operačné databázy, ale aj súbory v rôznych formátoch. Aby sa vyriešili problémy s kompatibilitou, sú dáta extrahované pomocou aplikačného rozhrania, ako je napríklad *JDBC*.
- **Transformácia** upravuje dáta z formátu dátových zdrojov do formátu dátového skladu. To zahŕňa typicky čistenie a redukciu dát.
- **Načítanie** zásobuje dátový sklad transformovanými dátami. To zahŕňa aj obnovenie dátového skladu, čo znamená šírenie aktualizácií z dátových zdrojov na dátový sklad v pravidelných intervaloch alebo ihneď po zmene v dátovom zdroji.

2.2.2 Výber a transformácia

Druhou časťou predspracovania dát sú čiastkové kroky výberu a transformácie dát z dátového skladu pre ľubovoľnú dolovaciu úlohu. Zatiaľ čo pri **výbere** ide len o spôsob výberu vhodných atribútov pre úlohu, **transformácia** je krok, ktorý môže pozostávať z viacerých transformačných dátových úkonov. Dôležité je uvedomiť si, že tento krok závisí vo veľkej miere na nasledujúcom kroku dolovania z dát. Dôvodom je, že rôzne dolovacie algoritmy môžu mať rôzne nároky na vstupné dáta, ktoré im pomôžu získať kvalitnejšie výsledky.

Ide teda o dôležité procesy, ktoré môžu podstatne zvýšiť úspešnosť pri použití techník strojového učenia pre praktické problémy dolovania z dát. Predstavujú druh dátového inžinierstva – konštruovanie vstupných údajov do formy vhodnej pre zvolenú schému učenia a konštruovanie výstupov tak, aby bola efektívnejšia. Dá sa na ne pozrieť ako na sadu trikov, ktoré je možné použiť na praktické problémy dolovania znalostí aj pre zvýšenie úspešnosti výsledkov. Aby boli dáta pre dolovacie metódy použiteľné, existuje niekoľko spôsobov transformácie dát [25]:

- **Výber atribútov** – V mnohých praktických situáciách existuje príliš veľa atribútov, ktoré musia algoritmy zvládnuť. Niektoré z nich môžu byť irelevantné alebo nadbytočné. V dôsledku toho musia byť údaje vopred spracované, aby sa vybrala podmnožina atribútov, ktoré sa majú použiť pri učení. Mnohé algoritmy sa snažia vybrať

vhodné atribúty svojpomocne. Napriek tomu je možné predvýberom atribútov zvýšiť výkon týchto algoritmov. Experimenty ukazujú, že pridávanie zbytočných atribútov spôsobuje zhoršenie výkonnosti niektorých algoritmov.

- **Diskretizácia atribútov** – Diskretizácia numerických atribútov je absolútne kľúčová v prípade, ak dáta obsahujú numerické atribúty, ale zvolená metóda strojového učenia dokáže zvládnuť len kategorické atribúty. Dokonca aj metódy, ktoré dokážu spracovať číselné atribúty, prinášajú často lepšie výsledky alebo pracujú rýchlejšie, ak sú atribúty už diskretizované. Nastáva aj opačná situácia, v ktorej musia byť kategorické atribúty reprezentované numericky. Tento prípad však nenastáva často.
- **Projekcia dát** – Projekcia dát zahŕňa rôzne techniky. Ide o pridanie nových syntetických atribútov, ktorých účelom je prezentovať existujúce informácie vo forme vhodnej pre metódu strojového učenia. Príkladom môže byť normalizácia, čo je úprava hodnôt atribútu do nejakého intervalu (napr. $\langle 0, 1 \rangle$).
- **Vzorkovanie dát** – Úlohou je predovšetkým redukcia dát. Vzorkovanie vstupu je dôležitým krokom v mnohých praktických aplikáciách dolovania údajov a často je to jediný spôsob, ako sa dajú riešiť skutočne rozsiahle problémy. Príkladom môže byť náhodné vzorkovanie.
- **Čistenie dát** – Problematika čistenia dát je rozobraná v podkapitole 2.2.1. Je možné ho použiť aj v tomto kroku, hlavne ak čistenie nebolo vykonané pri predchádzajúcom kroku, alebo je nutné ho použiť pre potreby nejakej metódy strojového učenia.

2.2.3 Dolovanie z dát

Dolovanie z dát je krokom získavania znalostí z databáz, o ktorom sa dá tvrdiť, že je jadrom celého tohto procesu. Ide o použitie konkrétneho dolovacieho algoritmu, pričom používané dolovacie algoritmy vychádzajú ako zo štatistiky, tak aj zo strojového učenia.

Štatistika študuje zber, analýzu, interpretáciu a prezentáciu údajov. Dolovanie z dát má so štatistikou prirodzenú spojitosť. Štatistický model je súbor matematických funkcií popisujúcich správanie objektov v cieľovej triede z hľadiska náhodných premenných a ich pravdepodobnostných rozdelení. Štatistické modely sú široko používané na modelovanie dát a dátových tried. Napríklad pri úlohách dolovania, ako je charakterizácia a klasifikácia údajov, možno zostaviť štatistické modely cieľových tried. Inými slovami, takéto štatistické modely môžu byť výsledkom úlohy dolovania údajov. [16]

Strojové učenie je rýchlo sa rozvíjajúca disciplína, ktorá skúma to, ako sa počítače môžu učiť (alebo zlepšiť svoj výkon) na základe dát. Hlavnou oblasťou výskumu je, aby sa počítačové programy automaticky naučili rozpoznávať zložité vzory a robiť inteligentné rozhodnutia na základe údajov. Napríklad typickým problémom strojového učenia je naprogramovať počítač tak, aby dokázal automaticky rozoznať ručne písané poštové smerovacie čísla po tom, čo sa to naučil z dodaných príkladov (čiže z nejakých trénovacích dát). [16]

Algoritmy strojového učenia sa delia do štyroch základných skupín [16]:

- **Učenie s učiteľom** (angl. *supervised learning*) je v podstate synonymom klasifikácie (angl. *classification*) a regresie (angl. *regression*). Učenie prebieha z dát tréningovej sady, ktoré sú označené triedou. Vhodným príkladom je opäť problém rozpoznávania poštových smerovacích čísel. V ňom sa ako tréningová sada dát používa množina obrázkov ručne písaných poštových smerovacích čísel, a zároveň ich zodpovedajúce strojovo čitateľné preklady, ktoré predstavujú spomínaného učiteľa.

- **Učenie bez učiteľa** (angl. *unsupervised learning*) je v podstate synonymom pre zhlukovanie (angl. *clustering*). Proces učenia je označovaný bez učiteľa, pretože vstupné dáta nie sú označené triedou, do ktorej patria. Typicky môžeme použiť zhlukovanie na nájdenie tried v údajoch. Napríklad metóda učenia bez učiteľa môže mať ako vstup množinu obrázkov ručne písaných číslic. Predpokladajme, že nájde desať zhlukov údajov. Tieto zhluky môžu zodpovedať desiatim rôznym čísliciam od 0 do 9. Keďže však trénovacie údaje nie sú označené, naučený model nám nemôže povedať sémantický význam nájdených zhlukov.

- **Kombinácia učenia s učiteľom a bez učiteľa** (angl. *semi-supervised learning*) je skupina techník strojového učenia využívajúcich pri učení modelu dáta označené či neoznačené triedou, do ktorej patria. V tomto prístupe sa označené údaje používajú na učenie modelov tried, zatiaľ čo údaje bez označenia sa používajú na spresnenie hraníc medzi triedami.

- **Aktívne učenie** (angl. *active learning*) je skupina, ktorá používateľom umožňuje hrať aktívnu úlohu v procese učenia. Aktívny prístup k učeniu môže požiadať používateľa (napr. doménového experta), aby označil príklad, ktorý môže byť zo sady neoznačených príkladov alebo syntetizovaný učiacim programom. Cieľom je optimalizovať kvalitu modelu aktívnym získavaním poznatkov od ľudských používateľov vzhľadom na obmedzenie počtu príkladov. Toto obmedzenie určuje, koľkokrát môžu byť používatelia požiadaní o označenie údajov.

Typy dolovacích úloh, ktoré boli spomenuté v tejto časti, a mnohé ďalšie sú detailnejšie vysvetlené v sekcii 2.3.

2.2.4 Vyhodnotenie a prezentácia dát

Úlohou posledného kroku získavania znalostí je prostredníctvom vyhodnotenia previesť vydolované výsledné vzory na pochopiteľné a užitočné informácie pre používateľa.

Vyhodnotenie vydolovaných dát je identifikácia skutočne zaujímavých vzorov. Zvyčajne len malá časť z výsledných vzorov je pre koncového používateľa zaujímavá. Zaujímavosť vzorov určujú štyri základne vlastnosti - pochopiteľnosť, platnosť, užitočnosť a novosť (všetky popísané v sekcii 2.1). [27]

Cieľom **prezentácie** je prezentácia výsledkov dolovania používateľovi, a to s využitím technik vizualizácie a reprezentácie znalosti. [27]. Výsledky je možné odprezentovať textovou, tabuľkovou alebo grafickou formou, ku ktorej patria rôzne diagramy, ako napríklad bodový alebo stĺpcový diagram.

2.3 Typy dolovacích úloh

V rámci získavania znalostí je možné vykonávať pomerne veľké množstvo rôznych dolovacích úloh. Niektoré z nich sú priblížené v tejto podkapitole. Skôr než príde na rad predstavenie jednotlivých typov úloh, bude uvedené základné rozdelenie týchto úloh podľa typu využitého modelu.

Vytvorený dolovací model môže byť [13]:

- **Predikatívny** – Predikatívny model predpovedá hodnoty údajov pomocou známych výsledkov zistených z iných údajov. Napríklad sa predikatívne modelovanie môže ro-

biť na základe použitia historických údajov. Patria sem úlohy klasifikácie, regresie, analýzy časových radov a predikcia.

- **Deskriptívny** – Deskriptívny model identifikuje vzory alebo vzťahy v údajoch. Na rozdiel od predikatívneho modelu deskriptívny model slúži ako spôsob na skúmanie vlastností údajov nie na predpovedanie nových vlastností. Patrí sem zhlukovanie, sumarizácia, asociačné pravidla a objavovanie sekvencií.

2.3.1 Popis triedy/konceptu

Prvým predstaveným typom dolovaciej úlohy je popis triedy/konceptu (angl. *class/concept description*).

Napríklad, v obchode s elektronikou môžu existovať triedy položiek *počítač* a *tlačiareň*, a tiež môžu byť zavedené pojmy *zákazník, ktorý veľa míňa* a *zákazník, ktorý sa drží svojho rozpočtu*. Potom môže byť užitočné popísať triedy a koncepty nejakým súhrnným, stručným, a pritom dostatočne presným spôsobom. Takýto spôsob budeme nazývať popisom triedy/konceptu, ktorý môžeme získať jedným z dvoch nasledujúcich spôsobov [16]:

- **Charakterizácia dát** – Charakterizácia dát značí sumarizáciu všeobecných vlastností analyzovanej triedy. Dáta zodpovedajúce triede je možné z databázy vybrať spravidla jednoduchým dotazom. Napríklad triedou by mohli byť softvérové produkty, ktorých predaj sa v minulom roku zvýšil o 10%.

- **Diskriminácia dát** – Diskriminácia údajov je porovnanie všeobecných vlastností dátových objektov cieľovej triedy so všeobecnými vlastnosťami objektov z jednej alebo viacerých kontrastných tried. Cieľové a kontrastné triedy môže špecifikovať používateľ a príslušné dátové objekty je možné získať prostredníctvom databázových príkazov. Používateľ môže napríklad chcieť porovnať všeobecné vlastnosti softvérových produktov, ktorých predaj sa minulý rok zvýšil o 10% s produktami, ktorých predaj sa za rovnaké obdobie znížili aspoň o 30%.

2.3.2 Frekventované vzory, asociačná a korelačná analýza

Frekventované vzory, asociačná a korelačná analýza sú úlohy, vďaka ktorým sme schopní nájsť vzťahy a závislosti medzi údajmi.

Frekventované vzory, ako už názov napovedá, sú vzory, ktoré sa v údajoch vyskytujú často. Existuje mnoho druhov frekventovaných vzorov, ako sú frekventované množiny, frekventované podpostupnosti (známe aj ako sekvenčné vzory) a ďalších frekventované podštruktúry. *Frekventovaná množina* položiek je definovaná ako množina položiek, ktoré sa často vyskytujú spolu v transakčnej dátovej sade – napríklad mlieko a chlieb, ktoré mnohí zákazníci často nakupujú v rámci jedného nákupu v potravinách. Často sa vyskytujúca podpostupnosť, ako napríklad vzor, kde majú zákazníci tendenciu kúpiť najprv notebook, potom digitálny fotoaparát a potom pamäťovú kartu (v rámci troch rôznych nákupov), je (frekventovaný) *sekvenčný vzor*. Podštruktúra sa môže vzťahovať na rôzne štrukturálne formy (napr. grafy, stromy), ktoré možno kombinovať s položkami alebo podsekvenciami. Ak sa podštruktúra vyskytuje často, nazýva sa (frekventovaný) *štruktúrovaný vzor*. Dolovanie frekventovaných vzorov vedie k objaveniu zaujímavých **asociácií** v podobe asociačných pravidiel a **korelácií**. [16]

Príkladom **asociačnej analýzy** môže byť to, ako chce marketingový manažér v obchode s elektronikou vedieť, ktoré položky sa často nakupujú spolu (to znamená v rámci toho

istého nákupu). Príkladom takého asociačného pravidla, získaného z transakčnej databázy obchodu, je:

$$\begin{aligned} kupuje(X, počítač) &\implies kupuje(X, softvér) \\ &[podpora = 1\%, spoľahlivosť = 50\%], \end{aligned} \tag{2.1}$$

kde X je premenná predstavujúca zákazníka. Spoľahlivosť 50% znamená, že ak si zákazník kúpi počítač, je 50% pravdepodobnosť, že si kúpi aj softvér. Podpora 1% znamená, že 1% všetkých analyzovaných nákupov obsahuje zároveň počítač aj softvér. Toto pravidlo asociácie zahŕňa jediný atribút alebo predikát (t.j. nakupuje), ktorý sa v pravidle opakuje. Asociačné pravidlá, ktoré obsahujú jeden predikát, sa označujú ako *jednorozmerné asociačné pravidlá.* [16]

Predpokladajme, že systém dolovania z dát môže nájsť v relačnej databáze obchodu spojenej s nákupmi asociačné pravidla takého typu:

$$\begin{aligned} vek(X, \langle 20, 29\rangle) \wedge príjem(X, \langle 40K, 49K\rangle) &\implies kupuje(X, notebook) \\ &[podpora = 2\%, spoľahlivosť = 60\%]. \end{aligned} \tag{2.2}$$

Pravidlo hovorí, že zo skúmaných zákazníkov obchodu sú 2% vo veku 20 až 29 rokov s príjmom 40 000 Kč až 49 000 Kč, ktorí si zároveň kúpili notebook. Je 60% pravdepodobnosť, že si zákazník v tejto vekovej a príjmovej skupine zakúpi notebook. Ide o asociačné pravidlo zahŕňajúce viac ako jeden atribút alebo predikát (t.j. vek, príjem a kupuje), a tým pádom možno vyššie uvedené pravidlo označiť ako *viacrozmerné asociačné pravidlo.* [16]

Asociačné pravidlá sú zvyčajne označené ako nezaujímavé a následne vyradené, ak nespĺňajú minimálny prah podpory ani minimálny prah spoľahlivosti. Je možné vykonať dodatočnú analýzu na odhalenie zaujímavých **štatistických korelácií** medzi súvisiacimi dvojicami atribút-hodnota. [16]

2.3.3 Klasifikácia

Klasifikácia je proces hľadania modelu, ktorý popisuje a rozlišuje dátové triedy alebo koncepty. Model je odvodený na základe analýzy množiny trénovacích dát (tzn. dátových príkladov, ktoré sú označené triedou, do ktorej patria). Model sa používa na predpovedanie označenia triedy objektov dát, pre ktoré je označenie triedy neznáme. Odvodený model môže byť reprezentovaný rôznymi formami, ako sú klasifikačné pravidlá, rozhodovacie stromy, matematické vzorce alebo neurónové siete. [16]

Rozhodovací strom je stromová štruktúra, kde každý uzol označuje test hodnoty atribútu. Následne každá vetva uzlu predstavuje výsledok testu a listy stromu predstavujú triedy alebo distribúcie tried, do ktorých sú objekty zaradzované. Rozhodovacie stromy môžu byť ľahko prevedené na klasifikačné pravidlá. Ak je na klasifikáciu použitá **neurónová sieť**, tak je typicky súborom neurónov s váhovými spojeniami medzi jednotlivými neurónmi. [16]

Tento typ dolovacej úlohy je dvojkrokový proces, ktorý pozostáva z kroku učenia, v ktorom sa vytvorí klasifikačný model a kroku klasifikácie, kde sa model používa na predpovedanie tried objektov. [16]

V prvom kroku sa vytvorí klasifikátor popisujúci vopred určenú množinu dátových tried. Je to **krok učenia**, kde klasifikačný algoritmus zostavuje klasifikátor analyzovaním, resp. učením sa z *tréningovej sady* tvorenej databázovými n-ticami a ich pridruženými triedami. N-tica X je n-rozmerný *vektor atribútu* zobrazujúci n meraní vykonaných na n-tici n databázových atribútov. O každej n-tici X sa predpokladá, že patrí do preddefinovanej triedy

určenej iným databázovým atribútom, ktorý sa nazýva *atribút označenia triedy*. Atribút označenia triedy nadobúda diskrétne hodnoty a je neusporiadaný. Je kategorický, a teda každá hodnota slúži ako kategória alebo trieda. Jednotlivé n-tice tvoriace trénovaciu množinu sa označujú ako trénovacie n-tice a sú náhodne odoberané z analyzovanej databázy. [16]

V druhom kroku sa model použije na klasifikáciu. Ide teda o **krok testovania**. Najprv sa odhadne predikatívna presnosť klasifikátora. Ak by sme použili trénovaciu sadu na meranie presnosti klasifikátora, tento odhad by bol pravdepodobne optimistický, preto sa používa *testovacia sada*, ktorá sa skladá z testovacích n-tíc a ich priradených tried. Sú nezávislé od trénovacích n-tíc, čo znamená, že neboli použité na konštrukciu klasifikátora. Presnosť klasifikátora na danej testovacej sade je percento n-tíc testovacej sady, ktoré klasifikátor správne klasifikuje. Ak sa presnosť klasifikátora považuje za prijateľnú, klasifikátor sa môže použiť na klasifikáciu reálnych dátových objektov, pre ktoré nie je známe označenie triedy. [16]

Príkladom klasifikácie by mohla byť situácia, kedy chce manažér predaja v obchode s elektronikou klasifikovať veľkú množinu produktov na základe troch druhov reakcií na predajnú kampaň: dobrá odozva, mierna odozva a žiadna odozva. Je potrebné vytvoriť model pre každú z týchto troch tried na základe vlastností produktov, ako sú cena, značka, miesto výroby, typ a kategória. Takýto klasifikátor môže obchodu pomôcť pochopiť účinnosť kampane, a tiež pomôcť v budúcnosti navrhnúť lepšiu. [16]

2.3.4 Regresia

Zatiaľ čo klasifikácia predikuje kategorické (diskrétne, neusporiadané) označenia tried, regresia modeluje spojité funkcie. To znamená, že regresia sa používa na predikciu chýbajúcich alebo nedostupných hodnôt číselných údajov. Regresná analýza je štatistická metodológia, ktorá sa najčastejšie používa na numerickú predikciu, hoci existujú aj iné metódy. [16]

Pri **jednoduchej lineárnej regresii** sa údaje modelujú tak, aby zodpovedali priamke. Napríklad náhodnú premennú y, nazývanú ako premennú odozvy, je možné modelovať ako lineárnu funkciu inej náhodnej premennej x, nazývanej ako premenná predikátora, pomocou rovnice:

$$y = wx + b, \tag{2.3}$$

s predpokladom, že rozptyl y je konštantný. V kontexte dolovania údajov sú x a y numerické databázové atribúty. Koeficienty w a b, nazývané ako regresné koeficienty, špecifikujú smernicu priamky a priesečník y. Tieto koeficienty je možné vypočítať metódou najmenších štvorcov. **Viacnásobná lineárna regresia** je rozšírením jednoduchej lineárnej regresie, ktorá umožňuje modelovať premennú odozvy y ako lineárnu funkciu dvoch alebo viacerých premenných predikátora. [16]

Príkladom použitia regresie by mohol byť prípad, kedy chce manažér predaja obchodu predpovedať výšku výnosu, ktorý každý produkt vygeneruje za nasledujúci kalendárny rok, a to na základe dát výnosov jednotlivých produktov za minulé roky. Keďže takýto predikatívny model predpovedá spojitú funkciu, ide o model regresnej analýzy. [16]

2.3.5 Zhlukovanie

Na rozdiel od klasifikácie, ktorá analyzuje množiny údajov označených triedou z testovacej dátovej sady, zhlukovanie analyzuje dátové objekty bez označení tried, a teda bez akýchkoľvek trénovacích dát. V mnohých praktických prípadoch totiž údaje už označené triedou jednoducho neexistujú. Na označenie objektov triedami je možné použiť zhlukovanie. [16]

Objekty sú zhlukované alebo zoskupované na základe princípu maximalizácie vnútrotriednej podobnosti a minimalizácie medzitriednej podobnosti. To znamená, že zhluky objektov sú tvorené tak, aby objekty v zhluku mali vysokú podobnosť, ale aby boli odlišné od objektov v iných zhlukoch. Každý takto vytvorený zhluk objektov možno považovať za triedu objektov, z ktorých je možné odvodiť rozhodovacie pravidlá. [16]

Zhlukovacie metódy je možné rozdeliť do niekoľkých kategórií [23]:

- **Hierarchické metódy** – Hierarchické metódy vytvárajú hierarchický rozklad množiny dátových objektov zdola-nahor alebo zhora-nadol na základe matice vzdialeností. Ukončenie hierarchickej metódy je možne vykonať pomocou definovania počtu tried alebo definovania ukončujúcej podmienky.
- **Metódy založené na rozdeľovaní** – Metódy založené na rozdeľovaní vykonávajú rozklad n objektov do vopred stanoveného počtu k tried. Platí, že $k \leq n$, kde každý zhluk obsahuje minimálne jeden objekt a každý objekt patrí iba do jednej triedy.
- **Metódy založené na hustote** – Metódy založené na hustote vykonávajú rozdelenie 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.
- **Metódy založené na mriežke** – Metódy založené na mriežke vytvárajú rozklad priestoru do buniek mriežky. Samotné zhlukovanie nie je realizované nad objektami, ale nad bunkami tejto mriežky.

Príkladom pre zhlukovú analýzu môže byť vykonanie zhlukovej analýzy na dátach zákazníkov obchodu pre identifikáciu homogénnych skupín populácií zákazníkov. Tieto zhluky môžu predstavovať jednotlivé zákaznícke cieľové skupiny daného obchodu. [16]

2.3.6 Detekcia anomálií, resp. odľahlých hodnôt

Posledným typom úlohy dolovania, ktorý je v tomto prehľade spomenutý, je detekcia odľahlých hodnôt. Keďže je detekcia anomálií predmetom tejto práce, detailnejšie je tento typ úlohy predstavený v kapitole 3.1.2.

Kapitola 3

Detekcia anomálií v temporálnych dátach

Úlohou nasledujúceho textu je ponúknuť čitateľovi podrobný teoretický základ pre pochopenie riešeného problému.

V tejto kapitole sú najprv definované základné pojmy, často spomínané v tejto práci. V ďalšej sekcii je spomenutý význam detekcie anomálií v časových radoch v praxi s praktickými príkladmi. Nasledujú základné vlastnosti metód pre analýzu anomálií v časových radoch. Nakoniec sú stručne popísané spôsoby detekcie anomálií v časových radoch, ktoré mnohé detekčné algoritmy využívajú.

3.1 Definícia základných pojmov

Pre pochopenie problematiky, ktorou sa práca zaoberá je dôležité vôbec pochopiť samotný názov tejto práce, a teda porozumieť pojmom detekcia anomálií a temporálne dáta. Nasledujúci text je preto venovaný vysvetleniu tejto terminológie.

3.1.1 Temporálne dáta

Termín temporálny pochádza z anglického slova *temporal*, čo v preklade znamená *časový*. Je teda z názvu možné predpokladať, že ide o dáta, ktoré majú časový charakter, a teda obsahujú nejakú časovú zložku. V odbornej literatúre sú temporálne dáta často nazývané ako *časové sekvencie*.

Príkladom temporálnych dát sú pravidelné časové rady (napr. ceny akcií), sekvencie udalostí (napr. lekárske záznamy) a temporálne databázy (napr. verzovaná databáza). Spoločným znakom všetkých týchto typov sekvencií je úplné usporiadanie ich elementov. Líšia sa v type primárnej informácie, v pravidelnosti elementov sekvencie a v tom, či obsahujú explicitnú časovú informáciu priradenú ku každému elementu (napr. časové razítko). [21]

Temporálne dáta majú niekoľko vlastností, podľa ktorých ich je možné popísať i klasifikovať. Obzvlášť dôležitou vlastnosťou pre túto prácu je už spomínaná **pravidelnosť** elementov v časovej sekvencii.

Podľa pravidelnosti je možné časové sekvencie rozdeliť na [21]:

- **Pravidelné** – Zodpovedajú dátam, ktoré sú nazývané termínom *časové rady*. Časové rady môžu byť charakterizované časovým krokom medzi dvomi časovými bodmi. Hodnoty časových radov sú teda zaznamenané v pravidelných časových intervaloch.

- **Nepravidelné** – Nepravidelné časové sekvencie sú vhodné pre zaznamenávanie dát udalostí, ktoré sa vyskytujú nepravidelne, teda v náhodných časových intervaloch.

Táto práca sa bude zaoberať detekciou anomálií v pravidelných časových sekvenciach, resp. časových radoch. Keďže tento typ časových údajov bude vstupom experimentov práce, je bližšie špecifikovaný v ďalšom texte.

Časový rad

Súčasne technologické pokroky nám umožňujú zbierať veľké množstvo údajov v priebehu času v rôznych oblastiach výskumu. Pozorovania zaznamenané usporiadane a korelované v čase tvoria **časový rad**. [10]

Časové rady delíme na dve základné skupiny [10]:

- **Jednorozmerné časové rady** – Formálne môžeme jednorozmerný časový rad $X = \{x_t\}_{t \in T}$ definovať ako usporiadanú množinu pozorovaní, kde každé pozorovanie obsahuje jednu reálnu nameranú hodnotu, a zároveň je každé pozorovanie zaznamenané v špecifickom čase $t \in T \subseteq \mathbb{Z}^+$. Potom x_t je bod alebo pozorovanie zaznamenané v čase t. $S = x_p, x_{p+1}, ..., x_{p+n-1}$ je podsekvencia o dĺžke $n \leq |T|$ začínajúca na pozícii p časového radu X, kde $p, t \in T$ a zároveň $p \leq |T| - n + 1$.
- **Viacrozmerné časové rady** – U viacrozmerného časového radu je oproti jednorozmernému rozdiel v tom, že ide o usporiadanú množinu pozorovaní, kde každé pozorovanie pozostáva z k-rozmerného vektoru, kde sa každý vektor skladá z k reálnych, nameraných hodnôt, a teda platí, že pozorovanie $x_t = (x_{1t}, ..., x_{kt})$. Pre každú dimenziu $j \in \{1, ..., k\}$ tiež platí, že $X_j = \{x_{jt}\}_{t \in T}$ je jednorozmerný časový rad.

Príkladom jednorozmerného časového radu by mohli byť údaje zo senzora merajúceho teplotu v priestore každú hodinu. Na druhú stranu údaje senzora, ktorý každú hodinu meria okrem teploty aj vlhkosť, sú vhodným príkladom pre dvojrozmerný časový rad.

Dekompozícia časového radu

Časový rad sa skladá z niekoľkých komponentov. Tieto komponenty súvisia so vzormi časových radov, ktoré môžu výrazne ovplyvniť hodnoty časového radu, a tým pádom môžu negatívne ovplyvniť výsledky rôznych úloh dolovania z dát.

Ide o komponenty [17]:

- trend-cyklicita,
- sezónnosť,
- zvyšok.

Ako už názov napovedá, komponent **trend-cyklicita** (často nazývaný len ako *trend*) kombinuje dva vzory časových radov. Vzor **trendu** sa v časovom rade nachádza v prípade, že v dátach je dlhodobý rast alebo pokles. Niekedy sa trend označuje ako zmena smeru, a to napríklad pri prechode z klesajúceho do rastúceho trendu. Na druhej strane **cyklicita** je vzor charakterizovaný cyklami. Cyklus nastáva, keď dáta vykazujú rasty a poklesy, ktoré nemajú pevnú frekvenciu. Tieto výkyvy môžu byť spôsobené napríklad ekonomickými podmienkami, často súvisiacimi s obchodným cyklom. [17]

Sezónnosť je komponent pomenovaný po ďalšom vzore časových radov. Ten nastáva v prípade, ak sú dáta ovplyvnené napríklad časom v roku alebo dňom v týždni. Sezónnosť má vždy pevnú a dobre známu periódu. [17]

Posledným komponentom je **zvyšok**, ktorý už nesúvisí so žiadnym vzorom časových radov. Naopak obsahuje dáta po odstránení spomínaných vzorov. [17]

Je bežnou praxou, že sa úlohy dolovania vykonávajú na komponente zvyšku. Keďže ide o časový rad očistený od vzorov, je takto možné dôjsť k platným výsledkom. Aby bolo možné tento komponent získať, je nutné rozložiť časový rad na všetky spomenuté komponenty. Tento proces je nazývaný ako **dekompozícia časového radu**.

Dekompozíciu je možné vykonať v dvoch rôznych modeloch. Prvým modelom je **aditívny** model, ktorý je definovaný rovnicou:

$$y_t = S_t + T_t + R_t, \tag{3.1}$$

kde y_t sú dáta, S_t je sezónny komponent, T_t je komponent trend-cyklicita, R_t je komponent zvyšku a t je perióda. Druhým modelom je **multiplikatívny** model, definovaný rovnicou:

$$y_t = S_t \times T_t \times R_t, \tag{3.2}$$

kde y_t sú dáta, S_t je sezónny komponent, T_t je komponent trend-cyklicita, R_t je komponent zvyšku a t je perióda. [17]

3.1.2 Detekcia anomálií

Ako už bolo spomenuté v kapitole 2.3.6, detekcia anomálií je jedna zo základných úloh dolovania z dát.

Dátová sada môže obsahovať objekty, ktoré nie sú v súlade so všeobecným správaním alebo s modelom údajov dátovej sady. Tieto dátové objekty sú odľahlé. Mnohé metódy dolovania v rámci iných typoch úloh dolovania odstraňujú takéto odľahlé objekty ako šum alebo výnimky. V niektorých praktických prípadoch však môžu byť zriedkavé udalosti zaujímavejšie ako tie, ktoré sa častejšie vyskytujú. Vhodným príkladom pre takýto prípad je detekcia podvodov s kreditnými kartami. Analýza odľahlých hodnôt sa označuje ako **analýza odľahlých hodnôt** alebo **detekcia anomálií**. [16]

Odľahlé hodnoty je možné nájsť pomocou **štatistických testov**, ktoré predpokladajú model distribúcie alebo pravdepodobnosti dát, alebo pomocou **zhlukovania**, kde sa objekty vzdialené od akéhokoľvek iného zhluku považujú za odľahlé hodnoty. Namiesto použitia štatistických testov alebo zhlukovania môžu **metódy založené na hustote** detekovať anomálie v lokálnom priestore, hoci z pohľadu globálneho štatistického rozloženia vyzerajú normálne. [16]

Je dôležité si uvedomiť, že v tejto práci nebude nazerané na odľahlé hodnoty ako na šum, ktorý je potrebné odstrániť pre dosiahnutie presnejších výsledkov iných úloh dolovania z dát. Naopak, v experimentálnej časti práce bude detekcia anomálií bodom záujmu, a to ako skúmanie neobvyklého správania nejakého systému v rámci porovnania rôznych detekčných algoritmov.

3.2 Techniky detekcie anomálií v časových radoch

V tejto sekcii sú definované a popísané základné vlastnosti techník pre detekciu odľahlých hodnôt v časových radoch. V texte je však možné nájsť aj dôležité informácie o typoch

odľahlých hodnôt, ktoré sú tieto techniky schopné detekovať, a to spolu s ich praktickou demonštráciou.

3.2.1 Praktické využitie

Celosvetovo mali dáta dosiahnuť objem viac ako 40ZB už v roku 2020. Aj preto je možné povedať, že nastala doba veľkých dát (angl. *big data*). Kvôli tomu je potrebné urýchlene vyriešiť problém ako získať potenciálne informácie z tak veľkého množstva dát. Veľkú časť týchto dát tvoria aj časové rady. [26]

Časové rady sú tak široko používané v mnohých oblastiach, ako armáda, ekonómia a vedecké pozorovania, že vzbudili veľkú pozornosť výskumníkov. Aj keď je počet abnormálnych údajov v bežných časových radoch veľmi malý, neznamená to, že abnormálne údaje nie sú dôležité. Naopak, za týmito pár abnormálnymi údajmi sa môžu ukrývať niektoré dôležité informácie. V lekárskej oblasti by sa pomocou nich mohol detektovať abnormálny srdcový rytmus, aby mohli lekári pomocou EKG zistiť ochorenie včas. [26]

Okrem toho môže byť detekcia anomálií časových radov využitá aj pri monitorovaní prevádzky motora, detekcii narušenia informačnej siete, praní špinavých peňazí, podvodoch s kreditnými kartami, analýze akciového trhu, monitorovaní nesprávneho daňového správania, analýza prírodných katastrof a pod. Je zrejmé, že štúdium detekcie anomálií časových radov má vysokú teoretickú hodnotu a praktický význam. [26]

3.2.2 Vlastnosti techník

V tejto podkapitole sú postupne predstavené vlastnosti techník pre analýzu odľahlých hodnôt v časových radoch, podľa ktorých je možné dané techniky aj klasifikovať. V rámci týchto vlastností sú definované aj dôležité pojmy týkajúce sa tejto problematiky, ako sú typy odľahlých hodnôt v časových radoch.

Techniky pre detekciu anomálií v časových radoch sa líšia v závislosti od [10]:

- typu vstupných údajov,
- typu odľahlých hodnôt,
- charakteru metódy.

Typy vstupných údajov

Časové rady delíme na dve základné skupiny, ktoré zároveň zodpovedajú typom vstupných dát, s ktorými sú metódy schopné pracovať. Ide o **jednorozmerné** a **viacrozmerné** časové rady. [10]

Táto klasifikácia časových radov s vysvetlením je popísaná v kapitole 3.1.1.

Typy odľahlých hodnôt

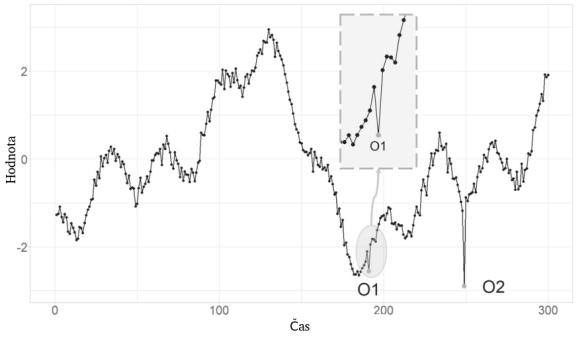
Pod pojmom odľahlá hodnota alebo anomália si človek pri časových radoch väčšinou predstaví jeden bod, ktorý viditeľne leží ďaleko od ostatných nameraných hodnôt. Každopádne anomáliou môže byť aj viac ako jeden konkrétny bod.

V časových radoch rozlišujeme tri typy anomálii, ktoré sú spomínané techniky schopné detekovať [10]:

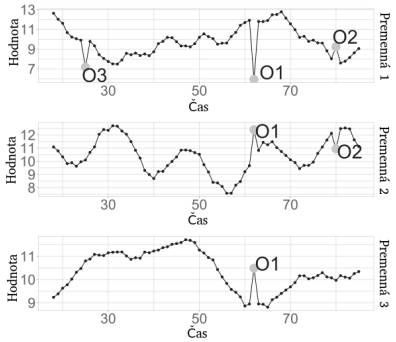
- odľahlý bod,

- odľahlá podsekvencia,
- odľahlý časový rad.

Odľahlý bod (angl. *point outlier*) je údaj, ktorý sa v konkrétnom časovom okamihu správa nezvyčajne v porovnaní s ostatnými hodnotami v časovom rade (globálna odľahlá hodnota) alebo so susednými bodmi (lokálna odľahlá hodnota). Bodové odľahlé hodnoty môžu byť jednorozmerné alebo viacrozmerné v závislosti od toho, či ovplyvňujú jednu alebo viac časovo závislých premenných. Napríklad obrázok 3.1 obsahuje dve jednorozmerné bodové odľahlé hodnoty $O1$ a $O2$, zatiaľ čo viacrozmerný časový rad zložený z troch premenných na obrázku 3.2 jednu jednorozmernú bodovú odľahlú anomáliu $O3$, a tiež dve viacrozmerné bodové odľahlé anomálie $O1$ a $O2$. [10]

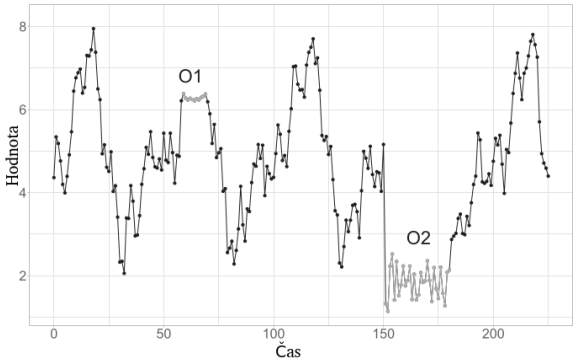


Obr. 3.1: Odľahlé body v jednorozmernom časovom rade (zdroj: [10])

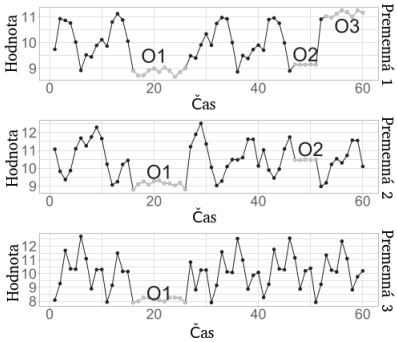


Obr. 3.2: Odľahlé body vo viacrozmernom časovom rade (zdroj: [10])

Termín **odľahlá podsekvencia** (angl. *subsequence outlier*) sa vzťahuje na body po sebe idúce v čase, ktorých spoločné správanie je nezvyčajné, hoci každé pozorovanie nemusí byť nutne odľahlým bodom. Podsekvenčné odľahlé hodnoty môžu byť tiež globálne alebo lokálne, a tiež môžu ovplyvňovať jednu (jednorozmerná odľahlá podsekvencia) alebo viacero (viacrozmerná odľahlá podsekvencia) časovo závislých premenných. Obrázok 3.3 poskytuje príklad jednorozmerných podsekvenčných anomálii $O1$ a $O2$. Obrázok 3.4 udáva tiež príklad jednorozmernej odľahlej podsekvencie $O3$, no okrem toho aj príklad viacrozmerných podsekvenčných anomálií $O1$ a $O2$. [10]

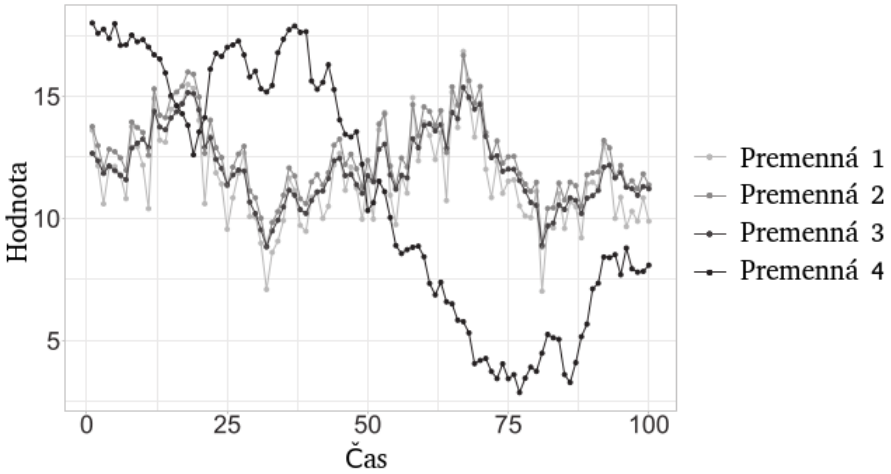


Obr. 3.3: Odľahlé podsekvencie v jednorozmernom časovom rade (zdroj: [10])



Obr. 3.4: Odľahlé podsekvencie vo viacrozmernom časovom rade (zdroj: [10])

Odľahlý časový rad (angl. *outlier time series*) je anomália, u ktorej je odľahlý celý časový rad. Tento prípad je však možný len v prípade, že na vstupe je viacrozmerný časový rad. Takýto časový rad sa skladá z viacero jednorozmerných radov a z nich môže byť niektorý odľahlý. Príklad je možné vidieť na obrázku 3.5, kde je odľahlý časový rad premennej 4. [10]



Obr. 3.5: Odľahlý časový rad vo viacrozmernom časovom rade(zdroj: [10])

Je podstatné si uvedomiť, že typy odľahlých hodnôt, ktoré je metóda schopná detekovať, úzko súvisia s typom vstupných údajov. Ak metóda umožňuje ako vstup iba jednorozmerné časové rady, potom nie je možné identifikovať viacrozmerný bod ani viacrozmernú podsekvenciu. Okrem toho odľahlé časové rady je možné nájsť len v časových radoch s viacerými premennými. Nakoniec treba poznamenať, že odľahlé hodnoty závisia od kontextu. Ak teda metóda detekcie používa celý časový rad ako kontextovú informáciu, potom sú zistené odľahlé hodnoty globálne. V opačnom prípade, ak metóda používa iba segment radu (časové okno), potom sú zistené odľahlé hodnoty lokálne, pretože sú odľahlé v rámci svojho susedstva. Globálne odľahlé hodnoty sú tiež lokálne, ale nie všetky lokálne odľahlé hodnoty sú globálne. Inými slovami sa niektoré lokálne odľahlé hodnoty môžu zdať normálne, ak je analyzovaný celý časový rad, ale môžu byť anomálne, ak sa zameriame iba na ich okolie (ide napr. o odľahlý bod $O1$ na obrázku 3.1). [10]

Charakter metódy

Charakter metódy hovorí, či je metóda detekcie jednorozmerná alebo viacrozmerná. Jednorozmerná metóda detekcie zohľadňuje iba jednu časovo závislú premennú, zatiaľ čo metóda viacrozmernej detekcie je schopná súčasne pracovať s viac ako jednou časovo závislou premennou. Je dôležite uvedomiť si, že metóda detekcie môže byť jednorozmerná, aj keď vstupom je viacrozmerný časový rad, pretože pre každú časovo závislú premennú možno vykonať individuálnu analýzu bez zohľadnenia závislostí, ktoré môžu existovať medzi premennými. Naopak, viacrozmernú techniku nemožno použiť, ak je vstupom jednorozmerný časový rad. [10]

3.2.3 Typy techník podľa spôsobu detekcie

Na základe princípu, na ktorom je samotná detekcia anomálií založená, je možné metódy rozdeliť do piatich kategórií nasledovne [26]:

- detekcia anomálií založená na štatistike,
- detekcia anomálií založená na zhlukovaní,
- detekcia anomálií založená na odchýlke,
- detekcia anomálií založená na vzdialenosti,
- detekcia anomálií založená na hustote.

V tejto sekcii sú postupne vysvetlené všetky tieto spôsoby detekcie odľahlých hodnôt.

Detekcia anomálií založená na štatistike

Detekcia anomálií založená na štatistike je najstarší a najviac študovaný prístup. Predpokladá, že ak je rozdiel medzi údajmi a štatistickým rozdelením alebo daným modelom väčší ako konkrétna hodnota alebo rozsah, tak ide o anomáliu. [26]

Túto metódu možno rozdeliť na dve kategórie: metóda založená na štatistickom rozdelení a metóda založená na hĺbke. [26]

Cieľom **metódy založenej na štatistickom rozdelení** je snaha priradiť vstupným údajom štatistické rozdelenie (ako napr. normálne rozdelenie, Poissonovo rozdelenie atď.) a následné detekovanie anomálie metódou kontroly konzistencie. Prevažná väčšina metód detekcie anomálií založených na štatistickom rozdelení je vhodná len pre jednorozmerné časové rady, zatiaľ čo v skutočnosti veľký počet dátových množín používaných pre detekovanie anomálií obsahuje viacrozmerné údaje, a teda aj viacrozmerné časové rady. [26]

Metóda založená na hĺbke predpokladá, že každý objekt je bodom v n-rozmernom priestore, každý bod má jednu nastavenú hĺbku a anomáliou je objekt, ktorý má menšiu hĺbku. Tento proces zabraňuje problémom s prispôsobením rozdelenia údajov, tak ako to je v metóde založenej na rozdelení. Zároveň má vyššiu účinnosť pri detekcii v jednorozmerných časový radoch. Je tu však potrebné vypočítavať konvexný uzáver n-rozmerného priestoru, a tiež má väčšiu zložitosť výpočtu, čím je vhodná len pre málorozmerné časové rady, ako sú dvojrozmerné alebo trojrozmerné časové rady. Rovnako ma nižšiu účinnosť na väčších množinách údajov. [26]

Detekcia anomálií založená na zhlukovaní

Detekciu anomálií založenú na zhlukovaní dát je možné vykonať priamo alebo nepriamo pomocou existujúcich zhlukovacích algoritmov, ako sú *DBSCAN*, *ROCK*, *K-Means* atď. Analýza zhlukovania sa však líši od detekcie anomálií, keďže zhlukovanie sa zameriava na nájdenie zhlukov údajov a detekcia anomálií na nájdenie odľahlých hodnôt. Detekcia anomálií je len „doplnkovým produktom“ zhlukovania. V obyčajnom zhlukovacom algoritme nie je žiadna špeciálna optimalizácia pre detekciu anomálií, takže je dosiahnutá nízka účinnosť. Vo väčšine prípadov je definícia a testovanie anomálie implicitné, zatiaľ čo v zhlukovaní nemôžu byť jasne vyjadrené. [26]

Detekcia anomálií založená na odchýlke

Detekcia anomálií založená na odchýlke je rozdelená do troch kategórií: metóda detekcie sekvenčných anomálií, metóda *OLAP* dátovej kocky a metóda modelu predikcie. [26]

Metóda detekcie sekvenčných anomálií vznikla návrhom sekvenčnej výnimky pánom *Agrawalom* a ďalšími v roku 1996. Tá považovala dátové body so zjavnou odchýlkou od priľahlých sekvencií za anomálie s využitím mechanizmu skenovania dátových množín. Výpočtová zložitosť tohto algoritmu závisí lineárne od veľkosti dátovej sady a má vysokú účinnosť. Jeho predpoklad výnimky je však príliš ideálny a koncepčne nesprávny, a tiež nie je vhodný pre viacrozmerné časové rady. [26]

Pán *Sarasangi* a ďalší použili **techniku OLAP dátovej kocky** na detekciu anomálií na rozsiahle dátové sady. Hodnoty dátovej kocky sa považujú za anomáliu, ak sa výrazne líšia od očakávaných hodnôt štatistického modelu. [26]

Metóda modelu predikcie je o použití modelov ako *SVM*, *ARAM*, *Bayesovská sieť* a ďalšie modely na štúdium neznámeho vzťahu v údajoch časových radov a následnom zostavení predikatívneho modelu. Nakoniec sa anomálie posudzujú prostredníctvom odchýlky predikatívnej a aktuálnej hodnoty. Táto metóda má dobrú efektivitu pri málorozmerných údajoch. Naopak nie je vhodná pre viacrozmerné vstupné dáta. [26]

Detekcia anomálií založená na vzdialenosti

Základnou myšlienkou detekcie anomálií založenej na vzdialenosti je vypočítať vzdialenosť medzi dátovými bodmi v dátovom priestore, a to zvolením vhodnej funkcie vzdialenosti. Za anomáliu sa považuje, keď je medzi dátovým objektom a ostatnými veľká vzdialenosť. [26]

V súčasnosti existuje niekoľko metód detekcie anomálií založených na vzdialenosti, ako je algoritmus založený na indexe, algoritmus založený na vnorenej slučke a algoritmus založený na prvku. Detekcia anomálií založená na vzdialenosti kombinuje myšlienky založené na štatistickom rozdelení, prekonáva hlavné nevýhody detekcie anomálií založenej na štatistickom rozdelení, je ľahšie realizovateľná i pochopiteľná a je široko používaná. Každopádne má tiež svoje nevýhody. Po prvé, komplexnosť algoritmu je relatívne vysoká a nevie brať do úvahy veľkosť dátových sád ani dimenzionálnu škálovateľnosť. Časová zložitosť je príliš vysoká. Druhou hlavnou nevýhodou je, že detekcia anomálií založená na vzdialenosti je pri spracovaní dátových sád so zjavnými rozdielmi vo vnútornej hustote chybná. Buď považuje údaje v oblasti malej hustoty za anomálie, alebo naopak nevie nájsť nejaké anomálie. [26]

Detekcia anomálií založená na hustote

Všetky vyššie uvedené metódy majú spoločný problém, ktorým je, že kritérium globálnej vzdialenosti sa považuje za základ pre detekciu anomálií. V skutočnosti je anomália zvyčajne detekovaná z pohľadu jednotlivca, čo znamená, že odľahlý bod je ďaleko od svojho susedného zhluku. Nie je teda vhodné používať globálnu vzdialenosť. Na vyriešenie tohto problému Breunig a ďalší vedci navrhli algoritmus detekcie anomálií založený na hustote. Jeho základnou myšlienkou je detekcia anomálie porovnaním hustoty objektu a jeho suseda. [26]

Zavádza lokálny odľahlý faktor (*LOF – Local Outlier Factor*), pričom pracuje s úvahou, že anomália nie je binárnou vlastnosťou, ale meraním. Čím vyššia je hodnota *LOF*, tým je pravdepodobnejšie, že dáta sú abnormálne. [26]

Myšlienka založená na hustote je bližšie k *Hawkinsovej* definícii výnimky ako myšlienka založená na vzdialenosti. Tým dokáže odhaliť lokálnu anomáliu a znížiť chybu detekcie,

ktorá obsahuje rôzne distribúcie. V metóde založenej na hustote sú však určité problémy, a to napríklad, že jej časová zložitosť je stále vysoká, výsledky detekcie sú citlivé na výber parametrov a parametre je ťažké určiť. [26]

Kapitola 4

Dátové sady a technológie

Predchádzajúce kapitoly rozoberali základné teoretické poznatky, či už všeobecnejšie z odboru dolovania znalostí z dát, alebo špecifickejšie k problému detekovania anomálií z časových radov. V tejto časti je, aj na základe týchto teoretických princípov, zachytený výber konkrétnych technológií i dát, ktoré budú použité v praktickej časti tejto práce.

V prvej časti je opísaný výber konkrétnych dátových sád s odôvodnením ich vhodnosti. Ďalšia časť je venovaná preskúmaniu dostupných technológií, ktoré poskytujú podporu pre riešený problém a nakoniec spomenutý výber a opodstatnenosť konkrétnych detekčných algoritmov s ich základnou charakterizáciou.

4.1 Dátové sady

Na základe dohody s vedúcim práce bolo stanovené, že hlavnou doménou dát bude počasie. Toto rozhodnutie vychádzalo z predpokladu, že dátových sád s meteorologickými údajmi existuje v dostatočnom množstve, a zároveň sú široko dostupné. Ďalším dôležitým aspektom zvolenia tejto domény bola vysoká pravdepodobnosť, že takéto dáta obsahujú významné odľahlé hodnoty, čo otvára experimentálnej časti tejto práce široké možnosti pre rôzne experimentálne úlohy.

Dátové sady boli získané zo zdroja **ECA&D**[1] (skratka pre anglický názov *European Climate Assessment & Dataset*), ktorý poskytuje voľne dostupné denné meteorologické dáta o počasí z rôznych meteorologických staníc v Európe a v oblasti Stredozemného mora. Tieto dáta je možné stiahnuť v preddefinovaných dátových sadách ale aj vytvorením vlastného príkazu výberu. Dáta je možné získať v niekoľkých meteorologických parametroch, ako sú minimálna, maximálna a stredná teplota vzduchu, množstvo zrážok, rýchlosť vetra, vlhkosť a mnohé ďalšie.

V nasledujúcej časti textu je zoznam vybraných dátových sád, ktoré boli vybrané pre vykonanie experimentov tejto práce. Každá dátová sada je charakterizovaná polohou meteorologickej stanice dát, meteorologickým parametrom nameraných hodnôt, jednotkou nameraných hodnôt, časovým intervalom dát a časovou frekvenciou dát. Vybrané dátové sady sú:

- Praha, stredná hodnota teploty vzduchu, $\times 10^{-1}\ °C$, $\langle 01.01.1979, 31.12.2023 \rangle$, denná frekvencia,

[1] https://www.ecad.eu/dailydata/index.php

- Aachen, stredná hodnota teploty vzduchu, $\times 10^{-1}\ °C$, $\langle 01.04.2011, 31.01.2023 \rangle$, denná frekvencia,
- Aachen, množstvo zrážok, $\times 10^{-1}\ mm$, $\langle 01.04.2011, 31.01.2023 \rangle$, denná frekvencia,
- Hamburg, rýchlosť vetra, $\times 10^{-1}\ m/s$, $\langle 01.01.1890, 31.01.2023 \rangle$, denná frekvencia.

Dôležité je poznamenať, že vybrané dátové sady sú reálne dáta o počasí, u ktorých namerané hodnoty nie sú označené ako odľahlé alebo neodľahlé. Ďalším dôležitým aspektom je, že ide o dáta jednorozmerných časových radov. Tieto fakty budú zohľadnené v ďalšom pokračovaní tejto práce, či už pri výbere vhodných technológií pre riešenie daného problému, ale aj v spôsobe zhodnotenia experimentálnych výsledkov.

4.2 Dostupné technológie

V tejto práci bol po dohode s vedúcim práce zvolený programovací jazyk *Pyhon* ako implementačný nástroj, keďže ponúka jednoznačne najväčšiu podporu pre problém detekcie anomálií v časových radoch.

V programovacom jazyku *Python* je dostupných hneď niekoľko balíčkov, ktoré poskytujú algoritmy pre riešený problém, a z ktorých je niekoľko predstavených v nasledujúcom texte. [22]

4.2.1 Balíček *Luminaire*

Luminaire je balíček, ktorý poskytuje riešenia strojového učenia na monitorovanie údajov časových radov. Poskytuje niekoľko funkcií na detekciu anomálií a predikatívne schopnosti, ktoré zahŕňajú korelačné a sezónne vzory v časových údajoch. [1]

Čo sa týka samotnej detekcie anomálií, *Luminaire* generuje model pre daný časový rad na základe jeho súčasných vzorov. Implementuje niekoľko techník modelovania na učenie sa rôznych variačných vzorov údajov, ktoré zahŕňajú *ARIMA, filtračné modey* a *Fourierovú transformáciu*. Tiež zahŕňa globálne charakteristiky, a zároveň sa učí lokálne vzory, aby bol proces učenia odolný voči akýmkoľvek lokálnym výkyvom a rýchlejšie sa vykonával. [1]

4.2.2 Balíček *TODS*

TODS (skratka pre anglický názov *Time-series Outlier Detection System*) je kompletný automatizovaný systém strojového učenia na detekciu odľahlých hodnôt v jednorozmerných a viacrozmerných časových radoch. Poskytuje moduly na budovanie systémov detekcie odľahlých hodnôt založených na strojovom učení vrátane spracovania údajov, spracovania časových radov, analýzy prvkov (extrakcia), detekčných algoritmov a modulu zosilnenia. Je možné vykonať tri možnosti detekcie odľahlých hodnôt na údajoch časových radov, a to detekciu jednotlivých odľahlých bodov, detekciu podsekvencie odľahlých bodov a detekciu odľahlých celých časových radov. Práve k týmto možnostiam je v *TODS* k dispozícii široká škála zodpovedajúcich algoritmov. [4]

Tento balíček je postavený na *Python* balíčku *PyOD*, ktorý je populárny pre riešenie úloh detekcie anomálií v rôznych dátach. *TODS* využíva mnoho jeho algoritmov (algoritmy pomenované s prefixom *Pyod*), ktorých funkcionalitu prispôsobuje pre potreby časových radov. K týmto algoritmom sú pridané aj implementácie ďalších vlastných algoritmov.

4.2.3 Knižnica *Orion*

Orion je knižnica strojového učenia vytvorená pre detekciu anomálií časových radov bez učiteľa. Táto knižnica bola vytvorená aby [3]:

- poskytla jedno miesto, kde môžu používatelia nájsť to najnovšie a najlepšie zo sveta strojového učenia a hlbokého učenia vrátane vlastných inovácií,
- poskytla systematický spôsob hodnotenia najnovších a najlepších metód strojového učenia,
- poskytla výskumníkom v oblasti strojového učenia spôsob, ako prispieť, aby boli ich inovácie okamžite dostupné koncovým používateľom.

4.2.4 Balíček *Kats*

Kats je súprava nástrojov na analýzu údajov časových radov. Je to ľahký, ľahko použiteľný a zovšeobecniteľný rámec na vykonávanie analýzy časových radov. Analýza časových radov je základnou súčasťou práce, vedy a priemyslu, a to od pochopenia kľúčových štatistík a charakteristík cez odhaľovanie anomálií až po predpovedanie budúcich trendov. *Kats* si kladie za cieľ poskytnúť jednotné kontaktné miesto pre analýzu časových radov vrátane detekcie, predpovedí, extrakcie/vkladania funkcií, viacrozmernej analýzy atď. Podporuje tiež funkcie na zisťovanie rôznych vzorov v časových radoch vrátane sezónnosti, odľahlých hodnôt, bodu zmeny a pomalých zmien trendov. [2]

4.2.5 Iné balíčky

Okrem spomenutých balíčkov resp. knižníc existuje mnoho ďalších, ktoré je tiež možné nájsť v citovanom zozname. [22]

Ide napríklad o balíčky: *MatrixProfile, Merlion, SaxPy* a *Sktime*.

4.3 Vybrané algoritmy

V predchádzajúcom texte bolo predstavených niekoľko balíčkov resp. knižníc, ktoré poskytujú podporu pre detekciu anomálií v časových radách. Keďže balíček ***TODS*** popísaný v sekcii 4.2.2 je špecializovaný na detekciu anomálií v časových radoch, a zároveň poskytuje mnoho detekčných algoritmov, ktoré sú založené na rôznych princípoch detekcie, bol tento balíček po dohode s vedúcim práce vybraný pre riešenie zadaného problému.

Ďalším dôležitým aspektom tejto voľby bol fakt, že veľká väčšina jeho algoritmov patrí do skupiny algoritmov **učenia bez učiteľa** (angl. *unsupervised learning*). Keďže vybrané dátové sady obsahujú dáta, ktoré nezahŕňajú informáciu o triede odľahlosti, je skupina týchto algoritmov ideálna pre experimenty tejto práce.

Zo spomínanej technológie bolo pre experimentovanie vybraných desať algoritmov. Ide o tieto algoritmy, ktorých názvy sú prebrané priamo z dokumentácie[2] balíčka:

- PyodIsolationForest,
- PCAODetect,
- DeepLog,

[2] https://tods-doc.github.io/tods.detection_algorithm.html

- PyodKNN,
- LSTMODetect,
- PyodABOD,
- PyodLOF,
- PyodAE,
- PyodSoGaal,
- KDiscordODetect.

V nasledujúcom texte sú tieto detekčné algoritmy bližšie vysvetlené. Je však dôležité v tomto momente poznamenať, že veľkým problémom pri študovaní týchto algoritmov bola slabá dostatočnosť informácii v dokumentácii balička *TODS* k jednotlivým algoritmom.

Jedným z problémov bolo, že sa v dokumentácii detekčných algoritmov nachádzali nefunkčné odkazy na odbornú literatúru, ktorá by mala ozrejmiť princíp, na ktorom sú jednotlivé algoritmy postavené. V niektorých prípadoch však bolo možné z malého množstva informácií nachádzajúcich sa v názvoch nefunkčných odkazov danú literatúru nájsť. V iných prípadoch bolo nutné k odbornej literatúre pristúpiť cez dokumentáciu už spomínaného balíčka *PyOD*, ak to teda u daného algoritmu bolo možné. V niektorých prípadoch však bola nutná analýza zdrojového kódu.

Druhým, oveľa závažnejším problémom študovanej dokumentácie bolo, že u jednotlivých algoritmov nebol uvedený typ odľahlých hodnôt, ktoré detekujú. V úvode tejto dokumentácie je pritom uvedené, že systém poskytuje algoritmy pre detekciu všetkých typov odľahlých hodnôt. Typy odľahlých hodnôt boli definované v kapitole 3.2.2 tejto práce. Po konzultácii s vedúcim práce bolo rozhodnuté, že súčasťou experimentálnej časti tejto práce budú experimenty pre získanie týchto informácií.

4.3.1 PyodIsolationForest

Isolation Forest je algoritmus pre detekciu anomálií, ktorý sa líši od iných detekčných metód tým, že namiesto modelovania normálnych inštancií explicitne izoluje anomálie. Tento prístup využíva stromové štruktúry nazvané izolačné stromy, ktoré sú efektívne v izolovaní jednotlivých prvkov najmä preto, že anomálie sú zvyčajne menej časté a líšia sa od normálnych inštancií, čo ich robí ľahko izolovateľnými. [19]

Algoritmus buduje súbor izolačných stromov na danej dátovej sade, pričom anomálie sú identifikované ako inštancie s kratšími priemernými cestami v týchto stromoch. *Isolation Forest* vykazuje lineárnu časovú zložitosť s nízkymi konštantami a nízkymi pamäťovými požiadavkami, čo ho robí vhodným pre prácu s veľkými dátovými sadami. [19]

Na rozdiel od iných modelov založených na detekcii anomálií, *Isolation Forest* nepotrebuje žiadne merania vzdialenosti alebo hustoty, čo eliminuje veľké výpočtové nároky spojené s týmito metódami. Tento algoritmus je obzvlášť účinný pri viacrozmerných dátach, ktoré obsahujú veľké množstvo nepodstatných atribútov. [19]

4.3.2 PCAODetect

PCAODetect je detekčný algoritmus založený na algoritme *PCA* (skratka pre anglický názov *Principal Component Analysis*). [4]

PCA je technika na znižovanie dimenzionality dátových sád transformáciou pôvodných premenných na nový súbor premenných, ktoré sú lineárne nezávislé a zachytávajú maximum variability v dátach. [7]

Funkcionalita *PCA* spočíva vo výpočte kovariančnej matice dát a následnom získavaní jej vektorov a hodnôt. Vektory predstavujú smer hlavných komponentov dát, zatiaľ čo hodnoty určujú mieru rozptylu, ktorú každý komponent zachytáva. Výsledkom je, že dáta sú transformované na nový koordinovaný systém, kde sú hlavné osi zarovnané s novými hlavnými komponentmi. [7]

Tento algoritmus dokáže detekovať odľahlé hodnoty na základe vzdialeností od hyperroviny s najmenšími vlastnými vektormi. Táto metóda je účinná, pretože odľahlé body majú zvyčajne veľké projekčné chyby z dôvodu, že nespadajú do hlavného rozptylu dát. Tieto projekčné chyby možno použiť ako skóre pre detekciu odľahlých bodov. Na identifikáciu odľahlých bodov je možné použiť vzdialenosť bodov od tejto k-rozmernej hyperroviny prechadzajúcej stredom dát. Body, ktoré majú veľkú vzdialenosť od tejto hyperroviny, sú identifikované ako odľahlé. [7]

Hlavnými výhodami *PCA* sú schopnosť odstrániť redundantné informácie (viacnásobnú kolinearitu medzi premennými) a zvýšiť interpretabilitu dát tým, že sa zameriava len na tie komponenty, ktoré majú vysoký význam pre rozptyl dát. [7]

4.3.3 DeepLog

DeepLog je navrhnutý na detekciu anomálií v systémových logoch pomocou metód hlbokého učenia s využitím neurónových sietí typu *LSTM* (skratka pre anglický názov *Long Short-Term Memory*). Tieto siete umožňujú algoritmu modelovať logy ako sekvencie prirodzeného jazyka, čo *DeepLog* využíva na automatické učenie sa vzorcov z normálnych logov a identifikáciu odchýlok od týchto vzorcov. Tieto odchýlky sa považujú za anomálie. [12]

Algoritmus funguje na princípe učenia sa z normálneho správania systému zaznamenaného v logoch. Po natrénovaní modelu na dátach je schopný identifikovať odchýlky, ktoré sa vyskytujú v reálnom čase. Tento prístup umožňuje algoritmu prispôsobiť sa novým vzorcom v logoch a postupne aktualizovať svoj model na základe nových dát. [12]

Medzi hlavné výhody *DeepLog* patrí jeho schopnosť adaptácie na nové vzory bez nutnosti pretrénovania celého modelu, efektivita v identifikácii aj zložitejších vzorcov anomálií a vysoká presnosť detekcie vďaka schopnosti *LSTM* sietí zapamätať si dlhodobé závislosti v dátach. [12]

4.3.4 PyodKNN

Algoritmus *KNN* (skratka pre anglický názov *K-Nearest Neighbor*) je známa metóda pre úlohy dolovania klasifikácie. Je však možné ju použiť aj na detekciu anomálii, čo je popísane v nasledujúcom texte.

Algoritmus *KNN* sa používa na detekciu odľahlých bodov tak, že zvažuje súčet vzdialeností bodu k jeho k-najbližším susedom. Body sa považujú za odľahlé, ak je ich súčet vzdialeností medzi najvyššími. [8]

Funkcionalita tohto algoritmu zahŕňa efektívne nájdenie k-najbližších susedov pomocou novej metódy, ktorá zahŕňa lineárne zobrazenie vyhľadávacieho priestoru cez *Hilbertovu priestorovú krivku*. Táto metóda výrazne zrýchľuje proces znížením dimenzionality vyhľadávacieho priestoru. [8]

Výhodami použitia *KNN* pre detekciu odľahlých bodov sú jeho efektívnosť vo vysokodimenzionálnych priestoroch a schopnosť lineárne škálovať s veľkosťou dátového súboru.

Použitie *Hilbertovej krivky* pre priestorové indexovanie znižuje zložitosť pri hľadaní najbližších susedov. [8]

4.3.5 LSTMODetect

LSTM (skratka pre anglický názov *Long Short-Term Memory*) architektúra bola vyvinutá ako riešenie problému zanikajúcich gradientov, ktorý je bežný u štandardných rekurentných neurónových sietí. Tento problém spôsobuje, že vplyv daného vstupu na skryté vrstvy aj na výstup siete buď exponenciálne klesá alebo narastá, čo znefunkčňuje učenie modelu siete. [14]

LSTM sieť sa skladá zo sady rekurentne prepojených podsietí známych ako pamäťové bloky. Tieto bloky obsahujú jednu alebo viac pamäťových buniek a tri multiplikatívne jednotky: vstupné, výstupné a zabúdacie brány. Tieto brány umožňujú vykonávanie operácií písania, čítania a resetovania v analogickej forme. To umožňuje *LSTM* bunkám ukladať a pristupovať k informáciám po dlhé obdobie. [14]

LSTM siete úspešne riešia množstvo syntetických úloh, ktoré vyžadujú dlhodobú pamäť, ako je učenie bezkontextových jazykov, zapamätávanie si vysoko presných čísel v dlhotrvajúcich šumových sekvenciách a rôzne úlohy, ktoré vyžadujú presné časovanie a počítanie. Okrem toho boli *LSTM* siete aplikované na riešenie rôznych problémov reálneho sveta, ako je predpoveď sekundárnej štruktúry proteínov a generovanie hudby. [14]

4.3.6 PyodABOD

Algoritmus *ABOD* (skratka z anglického názvu *Angle-Based Outlier Detection*) vyhodnocuje odľahlé hodnoty tak, že namiesto spoliehania sa len na merania vzdialenosti, analyzuje rozptyl uhlov medzi rozdielovými vektormi bodu a ostatnými bodmi v dátovej sade. Táto metóda je navrhnutá tak, aby bola menej ovplyvnená dimenzionalitou, čo ju práve robí vhodnou pre dáta s vysokou rozmernosťou. [18]

Funkcionalita tohto algoritmu zahŕňa výpočet uhlového faktora odľahlosti pre každý dátový bod. Toto sa dosahuje skúmaním uhlov medzi každým párom bodov vo vzťahu k určitému bodu a stanovením, ako veľmi sa tieto uhly líšia. Bod s vysokým rozptylom týchto uhlov je pravdepodobne odľahlý. [18]

Hlavnou výhodou *ABOD* oproti tradičným metódam založeným na vzdialenosti je jeho robustnosť vo vysokorozmerných priestoroch, kde merania vzdialenosti môžu stratiť význam. Okrem toho nevyžaduje výber parametrov, ktoré by mohli ovplyvniť kvalitu výsledkov, čo poskytuje jednoduchšiu a potenciálne objektívnejšiu analýzu. [18]

4.3.7 PyodLOF

Algoritmus *LOF* (skratka pre anglický názov *Local Outlier Factor*) identifikuje odľahlé hodnoty v dátovej sade tým, že každému objektu priradí faktor lokálnej odľahlosti. Ten reprezentuje mieru jeho odchýlky od ostatných objektov v jeho lokálnom okolí. Tento prístup sa líši od tradičných metód, ktoré považujú status odľahlého bodu za binárnu vlastnosť, čo je obzvlášť účinné v zložitých dátových sadách, kde je dôležitý skôr kontext lokálneho okolia. [11]

Algoritmus vypočíta faktor odľahlosti na základe lokálnej odchýlky hustoty daného objektu v porovnaní s jeho susedmi. Tento prístup umožňuje *LOF* detekovať odľahlé body, ktoré by globálne metódy zamerané len na analýzu extrémnych hodnôt v celkovej dátovej sade nemuseli zachytiť. [11]

LOF je veľmi efektívny v dátových sadách, kde sú anomálie lokálneho charakteru, čo znamená že sa výrazne odchýlia od svojho lokálneho okolia, pričom v globálnom kontexte dát nemusia byť výraznými odľahlými hodnotami. To robí *LOF* vhodným pre aplikácie ako detekcia podvodov, kde sú takéto lokálne anomálie relevantnejšie než globálne odľahlé body. [11]

4.3.8 PyodAE

AE (skratka z anglického názvu *Auto Encoders*) sú typom neurónových sietí, ktoré sa používajú na zníženie dimenzionality dát pomocou procesu kódovania a dekódovania. Kóder transformuje vstupné dáta na komprimovanú reprezentáciu a dekodér sa pokúša rekonštruovať pôvodné dáta z tejto komprimovanej reprezentácie. Tento proces umožňuje modelu identifikovať a ignorovať šum alebo menej dôležité vzory v dátach. [7]

Algoritmus sa učí rekonštruovať vstupy z ich latentných reprezentácií, pričom chyba rekonštrukcie slúži na tréning siete. Latentné reprezentácie sú naučené tak, aby zachytili podstatné charakteristiky vstupných dát, čo umožňuje *AutoEncoderu* efektívne redukovať dimenzie a tiež identifikovať odľahlé hodnoty, ktoré sa ťažko rekonštruujú. [7]

AutoEncodery poskytujú flexibilný spôsob na učenie hlbokých reprezentácií dát, ktoré môžu byť užitočné pre rôzne typy úloh vrátane detekcie anomálií a redukcie šumu. Výhodou je ich schopnosť zachytiť nelineárne vzťahy v dátach, čo ich robí vhodnými pre komplexné dátové sady, kde tradičné metódy ako *PCA* môžu zlyhať. [7]

4.3.9 PyodSoGaal

Algoritmus *SoGaal* (skratka z anglického názvu *Single-Objective Generative Adversarial Active Learning*) je založený na princípoch generatívneho adversatívneho učenia a aktívneho učenia. Algoritmus sa snaží generovať syntetické vzorky, ktoré sú podobné reálnym odľahlým vzorkám v dátovej sade. [20]

Algoritmus využíva dve hlavné komponenty: generátor a diskriminátor. Generátor sa snaží vytvárať dáta, ktoré sú nerozoznateľne od reálnych dát, zatiaľ čo diskriminátor sa snaží rozlíšiť medzi skutočnými dátami a generovanými dátami. Tento proces pomáha modelu naučiť sa identifikovať charakteristiky, ktoré definujú odľahlé vzorky. [20]

Jednou z hlavných výhod *SoGaal* je jeho schopnosť generovať nové vzorky, ktoré pomáhajú modelu lepšie pochopiť rozhranie medzi normálnymi dátami a odľahlými dátami. Toto je obzvlášť užitočné v scenároch, kde sú odľahlé dáta vzácne alebo keď je ťažké získať označené dáta pre tréning. [20]

4.3.10 KDiscordODetect

KDiscord najprv rozdelí viacrozmerné časové rady na podsekvenčné matice, na ktorých používa detekciu odľahlých hodnôt pomocou *KNN* z knižnice *PyOD*. [4]

Algoritmus *KNN* je detailnejšie špecifikovaný v podkapitole 4.3.4.

Kapitola 5

Aplikácia

Zadanie tejto práce z pohľadu implementácie hovorí len o realizácii vybraných detekčných algoritmov. Na základe dohody s vedúcim práce je však ďalším cieľom vytvoriť univerzálnu webovú aplikáciu, prostredníctvom ktorej si bude môcť používateľ vykonať vlastné experimenty detekcie odľahlých hodnôt na dátach, ktoré majú charakter jednorozmerných, numerických časových radov.

V tejto kapitole je popísaná tvorba tejto aplikácie, rozdelená do časti návrhu a časti implementácie. Poslednou časťou textu je predstavenie použitia a finálnej funkčnosti aplikácie.

5.1 Návrh

V tejto sekcii je popísaný jednoduchý návrh webovej aplikácie, ktorého výsledkom je niekoľko diagramov *UML* (skratka pre anglický názov *Unified Modeling Language*), či návrhov pohľadov pre používateľské rozhranie. Všetky grafické návrhy boli vytvorené v softvéri *Visual Paradigm*[1].

5.1.1 Neformálna špecifikácia požiadaviek

Ako už bolo spomenuté v úvode tejto kapitoly, hlavným cieľom navrhovanej webovej aplikácie je, aby si mohol ľubovoľný používateľ vykonať vlastné experimenty detekcie anomálií v časových radoch na vlastných dátach. Dôležité však je aj to, aby si vedel z obdržaných výsledkov experimentov odniesť nejakú znalosť. Pre vytvorenie takejto aplikácie je dôležitých hneď niekoľko funkcionalít.

Prvou takouto funkcionalitou je nahranie vlastných dátových sád, ktoré majú charakter jednorozmerných numerických časových radov. Takéto dáta sa nachádzajú hlavne v tabuľkových formách, v ktorých môžu byť rôzne aj nepotrebné stĺpce. Preto je dôležité, aby mal používateľ možnosť pri nahrávaní súboru možnosť zvoliť dôležité stĺpce, a to hlavne stĺpec obsahujúci časové dáta a stĺpec obsahujúci namerané numerické hodnoty. Po načítaní vstupných dát by mal byť používateľ v systéme schopný nájsť všetky jeho už nahrané dáta v prehľadnej tabuľke. Vhodné by bolo, ak by používateľ vedel zobraziť detail konkrétneho nahrania svojich dát. V tomto detaile by si vedel pozrieť detailné informácie, ktoré pri nahraní zadal, spolu s obsahom nahraného súboru. V danom detaile by mohol zo systému nahrané dáta odstrániť, poprípade stiahnuť zodpovedajúci súbor dát.

[1] https://www.visual-paradigm.com/

Druhou dôležitou skupinou funkcionalít navrhovaného systému sú požiadavky na samotné experimentovanie. Tým najdôležitejším je samotné vykonanie experimentu. Pred jeho vykonaním by mal používateľ zadať niekoľko vstupných konfiguračných údajov. Malo by ísť o výber dátovej sady, zvolenie možnosti pre sezónnu dekompozíciu vstupných dát a výber detekčného algoritmu spolu s nastavením jeho parametrov. Po spustení experimentu by mal systém vykonať detekciu a zobraziť jej výsledky používateľovi. Súčasťou výsledkov by mali byť informácie o vykonanej dekompozícii časového radu v prípade, ak bolo jej vykonanie používateľom zvolené. Hlavným výsledkom by mal byť podrobný súhrn o detekovaných anomáliách, či už v tabuľke alebo v grafe. Okrem vykonania experimentu by mal byť používateľ schopný vykonané experimenty prehľadávať, a tiež zobraziť detail s výsledkami ľubovoľného experimentu.

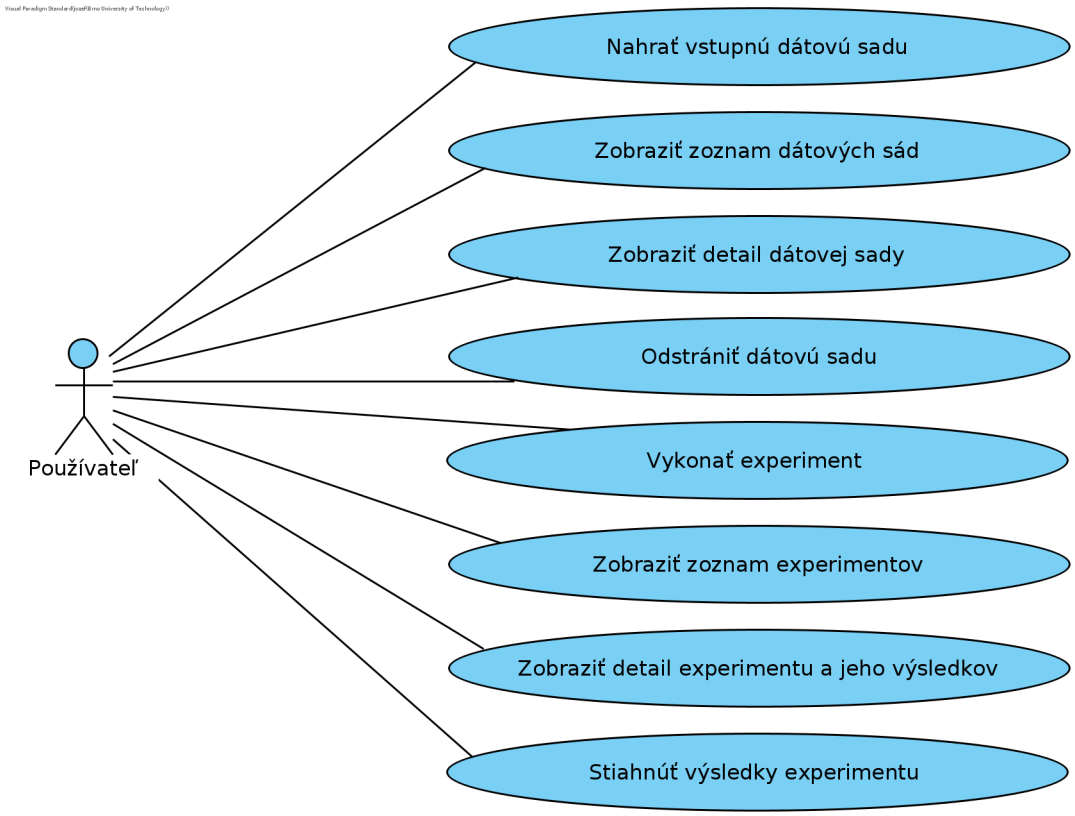
Technickými požiadavkami aplikácie sú jednoduchosť inštalácie či spustenia aplikácie, a aby išlo o webovú aplikáciu. Dôležitá ja aj jednoduchosť a intuitívnosť používateľského rozhrania.

5.1.2 Analýza požiadaviek

Z neformálnej špecifikácie požiadaviek vyplýva, že v systéme by sa mal nachádzať jeden *aktér*, ktorým je **používateľ**. Používateľ by mal mať v systéme možnosť vykonať tieto základné úkony:

- spravovať načítané dáta,
- vykonať experiment,
- a prehľadávať už vykonáne experimenty.

Na základe týchto poznatkov bol v ďalšom kroku analýzy požiadaviek vytvorený podrobnejší návrh úkonov vo forme diagramu prípadov použitia, ktorý je možné vidieť na obrázku 5.1.



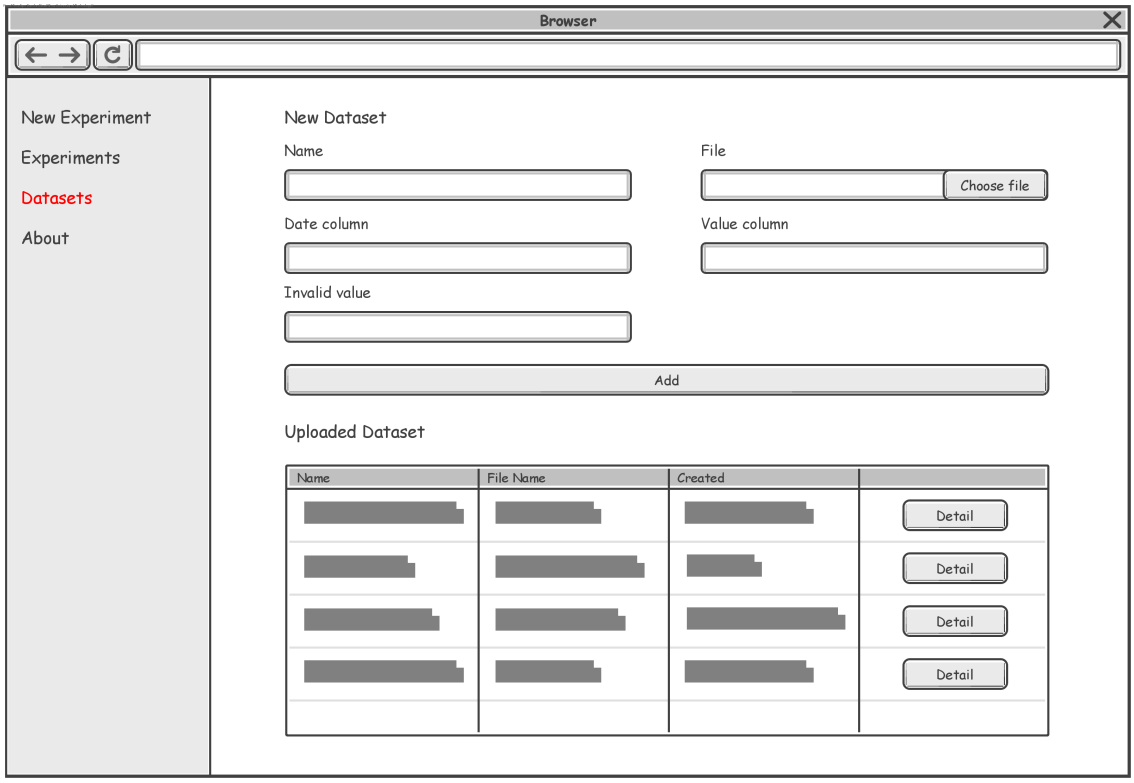
Obr. 5.1: Diagram prípadov použitia

5.1.3 Návrh používateľského rozhrania

Ďalším krokom návrhu systému je vytvorenie návrhu základných pohľadov, resp. obrazoviek používateľského rozhrania. Tento návrh bol vytvorený na základe vykonanej analýzy požiadaviek. Ide o jednoduchý návrh, ktorého úlohou je predstaviť základné elementy jednotlivých podstránok webu.

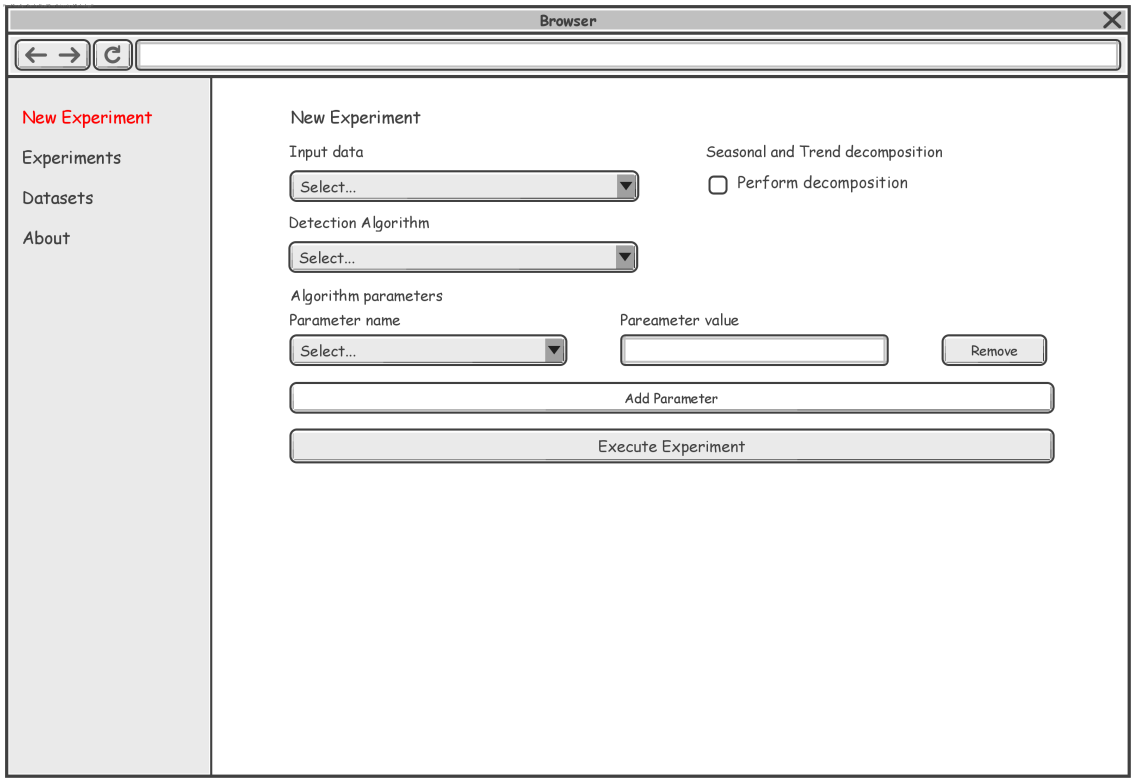
Prvým návrhom používateľského rozhrania je pohľad **nahraných dátových sád**, ktorého návrh je zobrazený na obrázku 5.2. Je potrebné zdôrazniť, že ľavý bočný panel je fixné menu, ktoré je spoločné pre všetky obrazovky aplikácie. Pomocou neho má používateľ možnosť prepínať medzi základnými podstránkami aplikácie.

Pohľad nahraných dát je rozdelený do dvoch hlavných častí. Vo vrchnej časti je formulár, ktorý slúži na už spomínané nahranie vstupných dátových sád. V ňom môže používateľ zvoliť vlastný názov, súbor s tabuľkovými vstupnými dátami, názvy stĺpcov tabuľky pre čas i hodnotu vstupných údajov (definujúce časový rad) a hodnotu definujúcu neplatný údaj. Naopak v spodnej časti pohľadu je tabuľka nahraných dátových sád, ktorá obsahuje informácie o zvolenom názve, názvu načítaného súboru a čase pridania. Tiež je možné pomocou tlačidla v poslednom stĺpci zobraziť detail konkrétneho načítania.



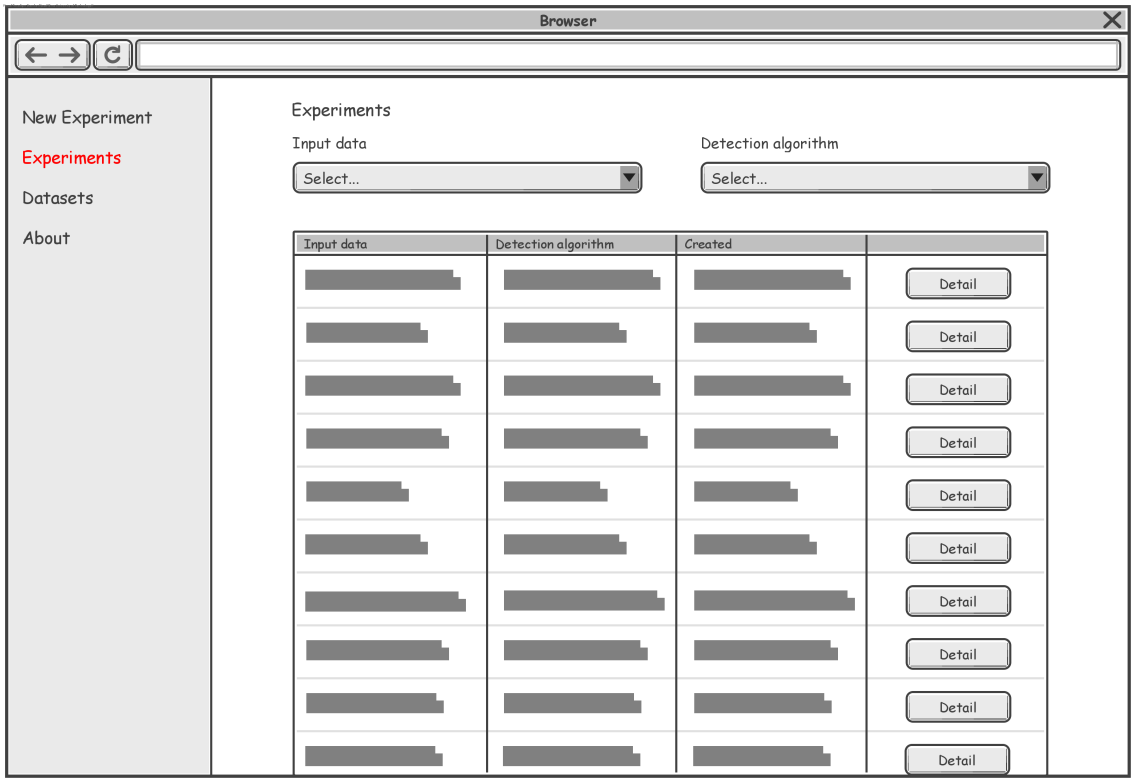
Obr. 5.2: Pohľad dátových sád

Na obrázku 5.3 je zobrazený návrh používateľského rozhrania pre **vykonanie nového experimentu**. Skladá sa z formulára, ktorým používateľ môže nakonfigurovať svoj experiment. Konfigurácia spočíva vo výbere dátovej sady a zvolením toho, či sa na vstupných dátach vykoná dekompozícia časového radu. Rovnako si používateľ môže vybrať algoritmus, ktorým bude detekcia anomálií vykonaná, spolu s nastavením vstupných parametrov detekčného algoritmu. Na konci je potvrdzovacie tlačidlo, ktorým je možné experiment spustiť. Po vykonaní experimentu by sa mali používateľovi zobraziť výsledky.



Obr. 5.3: Pohľad nového experimentu

Posledným základným návrhom používateľského rozhrania je pohľad pre **prehľadávanie vykonaných experimentov**. Ten je možné vidieť na obrázku 5.4. Jeho základom je tabuľka v druhej časti stránky, ktorý obsahuje informácie o experimentoch. Táto tabuľka poskytuje informácie o vstupnej dátovej sade experimentu, vybranom detekčnom algoritme experimentu a čase vykonania experimentu. Okrem toho dovoľuje používateľovi zobraziť detail konkrétneho experimentu prostredníctvom tlačidla v poslednom stĺpci tabuľky. Súčasťou stránky je tiež jednoduchý formulár v hornej časti, pomocou ktorého môže používateľ filtrovať zobrazené experimenty, a to zvolením dátovej sady alebo konkrétneho detekčného algoritmu.

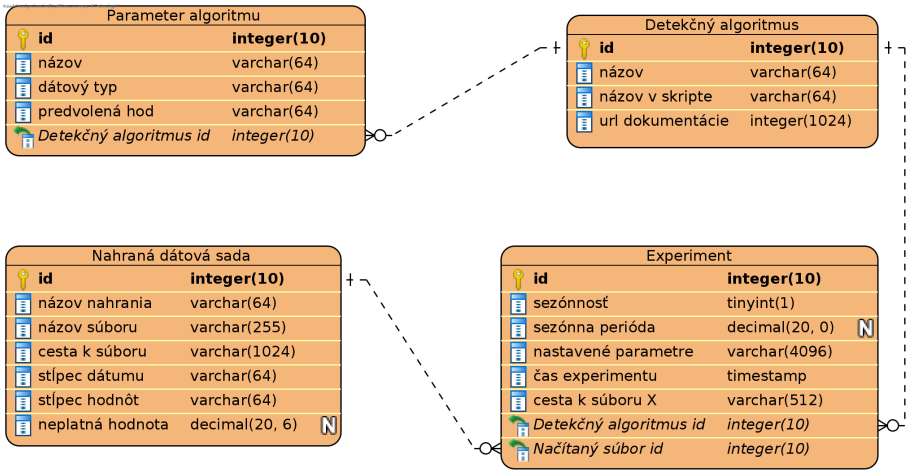


Obr. 5.4: Pohľad prehľadávania vykonaných experimentov

5.1.4 Návrh databázovej schémy

Z analýzy požiadaviek na systém vyplýva, že aplikácia má byť perzistentná. Aplikácia má byť teda schopná ukladať jednotlivé dáta, s ktorými používateľ pracuje tak, aby ich bol schopný prehľadávať alebo inak spravovať. Preto je tiež súčasťou návrhu aplikácie jednoduchá databázová schéma, na ktorej základe vznikne relačná databáza, schopná spomínané údaje ukladať. Jej návrh je možné vidieť na obrázku 5.5. Schéma obsahuje štyri relačné tabuľky.

Jednou relačnou tabuľkou je **načítaný súbor**, pričom by táto tabuľka mala uchovávať údaje o nahraných dátových sadách. Príkladom týchto údajov je cesta k uloženému súboru dát a vlastný názov nahrania dát. Ďalšími dôležitými tabuľkami sú **detekčný algoritmus** a **parameter algoritmu**, ktoré majú uchovávať správne údaje o detekčných algoritmoch. Tieto údaje by mali byť predvyplnené už od spustenia systému. Vďaka nim bude mať používateľ informácie o dostupných algoritmoch a ich vstupných parametroch, ako je predvolená hodnota každého parametra. Poslednou relačnou tabuľkou je **experiment**. Táto tabuľka ukladá údaje o vykonaných experimentoch. Obsahuje napríklad stĺpec *cesta k súboru X*, čo označuje údaj o ceste k výstupnému súboru. V systéme však bude takýchto súborov oveľa viac. Pôjde hlavne o tabuľkové súbory a obrázkové súbory, ktorých obsahom budú výsledne grafy experimentu. Pre jednoduchosť a intuitívnosť návrhu je však pre výstupné súbory uvedený len tento jeden stĺpec.



Obr. 5.5: Návrh databázovej schémy

5.2 Implementácia

V tejto sekcii sú popísané základné princípy implementácie webovej aplikácie pre detekciu anomálií z jednorozmerných časových radov.

V prvej časti je popísaná architektúra, špecifikácia a nosné technológie implementovanej aplikácie. Nasleduje súhrnný popis jednotlivých časti implementácie, ktorými sú frontend, backend z pohľadu serverovej obsluhy klientských požiadaviek a backend z pohľadu vykonávania detekcie anomálií v časových radoch.

Tento text sa často odkazuje na konkrétne súbory implementácie, ktorých prehľad je vypísaný v stromovej súborovej štruktúre uvedenej v prílohe B.

5.2.1 Architektúra a špecifikácia

Implementovaná webová aplikácia je postavená na architektúre **klient-server** typu **tenký klient**. To znamená, že na strane klienta je vykonávaná minimálna aplikačná logika. Naopak väčšina tejto logiky je vykonávaná na strane servera.

Na základe toho je tiež možné aplikáciu klasifikovať ako **multistránkovú** (angl. *Multi Page Application)*. To znamená, že prepínanie jednotlivých stránok resp. pohľadov aplikácie je vykonávané prostredníctvom *HTTP* požiadaviek, ktoré klient posiela serveru. Na obdržané požiadavky následne server odpovedá *HTML* kódom novej stránky.

Aplikácia využíva architektonický vzor, nazývaný ako ***MVC*** (skratka pre anglický názov *Model-View-Controller*), ktorý poskytuje organizáciu systému medzi rôzne komponenty. Týmito komponentami sú pohľady (angl. *views*), ktoré zobrazujú informácie, modely (angl. *models*), ktoré spracovávajú dáta a ovládače (angl. *controllers*), ktoré spracovávajú požiadavky klientov.

Všetky spomenuté aspekty aplikácie vychádzajú z použitej technológie, ktorou je framework *CodeIgniter 3*[2]. Ide o framework postavený na programovacom jazyku *PHP*[3], pričom poskytuje nástroje a knižnice pre rýchlejší vývoj webových stránok s využitím už spomínaného architektonického vzoru *MVC*.

[2] https://codeigniter.com/userguide3/general/welcome.html
[3] https://www.php.net/

Dôležitou vlastnosťou implementácie je tiež použitie nástroja *Docker Compose*[4]. Ide o nástroj, ktorý umožňuje spúšťať softvérové aplikácie v niekoľkých virtuálnych prostrediach – kontajneroch, čím sa často predchádza problémom s inštaláciou a spustením týchto aplikácií. Pri inštalácii detekčného nástroja *TODS*, definovanom v kapitole 4.2.2, vznikalo hneď niekoľko problémov. Preto bol *Docker Compose* využitý pre vytvorenie virtuálneho kontajnera so správnou konfiguráciou prostredia pre tento detekčný nástroj. V tom istom kontajneri je inštalovaný aj *HTTP* server *Apache 2*[5], ktorý umožňuje prevádzku celej webovej aplikácie. Súčasťou je aj druhý virtuálny kontajner, na ktorom sa spúšťa databázový systém *MySQL*[6] pre uchovávanie údajov. Výhodou použitia nástroja *Docker Composer* je aj jednoduchá inštalácia, resp. jednoduché spustenie implementovanej aplikácie. Postup spustenia sa nachádza v prílohe C.

5.2.2 Frontend

Frontendová časť aplikácie je implementovaná v súboroch jazyka *PHP*, ktoré sú obsiahnuté v adresári `web-app/application/views`. Tento adresár je frameworkom *CodeIgniter* určený pre pohľady architektonického vzoru *MVC*.

Jednotlivé pohľady sú implementované prostredníctvom niekoľkých technológií. Samozrejmosťou je využitie značkovacieho jazyka *HTML*[7] (skratka pre anglický názov *Hypertext Markup Language*) pre hlavnú štruktúru jednotlivých stránok či štýlovacieho jazyka *CSS*[8] (skratka pre anlgický názov *Cascading Style Sheets*) pre štýlovanie daných stránok. Pre jednoduchšie štýlovanie stránok bola zvolená téma *flatly* štýlovacieho frameworku *Bootswatch*[9], ktorého štýlovací súbor je prístupný na ceste `web-app/css/bootstrap.min.css`. Ten obsahuje štýly pre základné elementy *HTML*. Štýlovanie aplikácie je tiež doplnené o vlastnú implementáciu štýlov nachádzajúcich sa v súbore `web-app/css/custom.css`. Ten obsahuje napríklad triedu `custom-nav` pre vzhľad ľavého bočného menu, ktoré je súčasťou každej stránky aplikácie.

Ďalšími využitými technológiami pri frontende sú programovací jazyk *JavaScript*[10] spolu s knižnicou *jQuery*[11]. Tie sa využívajú na dynamické zmeny vzhľadu stránky, ako je napríklad implementácia pridania nového parametra pri konfigurácii experimentu vo funkcii `addParameter()` pohľadu `experimentPrepare2.php`. Ďalším využitím týchto technológií je aj dynamické načítanie obsahu stránky, kedy sa zo servera nestiahne celá stránka ale len jej časť. Toto je vykonávané v spolupráci s technológiou *AJAX*. Príkladom môže byť funkcia `getAlgoParams()` pohľadu `experimentPrepare.php`. V nej sa pre konkrétny detekčný algoritmus vybraný používateľom stiahne zo servera stránka pre pridanie parametrov algoritmu. Táto stránka je následne vložená do príslušného elementu pôvodne načítanej stránky.

Programovací jazyk, ktorý sa tiež podieľa na tvorení pohľadov, je *PHP*. Vďaka tomu je v pohľade možné použiť údaje zo spracovania požiadavky. To funguje tak, že ovládač prostredníctvom príkazu `load->view()` vytvorí stránku z príslušného pohľadu a odošle ju klientovi.

[4] https://docs.docker.com/compose/
[5] https://httpd.apache.org/
[6] https://www.mysql.com/
[7] https://developer.mozilla.org/en-US/docs/Web/HTML/
[8] https://developer.mozilla.org/en-US/docs/Web/CSS/
[9] https://bootswatch.com/flatly/
[10] https://www.javascript.com/
[11] https://jquery.com/

5.2.3 Backend – spracovanie klientskych požiadaviek

Pre potreby aplikácie je vytváraná relačná databáza, do ktorej sa ukladajú dôležité údaje. Databáza sa vytvára pri spustení aplikácie, a to vykonaním skriptu jazyka *SQL*. Tento skript je implementovaný v súbore `db.sql`. Skript tiež naplní údajmi relačné tabuľky, ktoré uchovávajú údaje o implementovaných detekčných algoritmoch.

V adresári `web-app/application/models` sú implementované triedy, ktoré zodpovedajú modelom architektonického vzoru *MVC*. Implementované sú dva modely. Tým prvým je `ExperimentsMod.php`, ktorého úlohou je vklad a výber experimentálnych dát do databázy. Druhým modelom je `UploadMod.php`, ktorého úlohou je správa databázových dát o importovaných dátových sadách používateľa. *CodeIngiter* poskytuje podporu jednoduchých príkazov pre prácu s relačnou databázou. Vhodnou ukážkou vloženia dát do databázy je funkcia `insert()` v modeli `UploadMod.php`, ktorá do databázy vkladá údaje o nahraných sadách používateľa. Vykonáva sa pomocou príkazu `db->insert()`. Príkladom výberu dát z databázy môže byť funkcia `getExperiments()` z `ExperimentsMod.php`, ktorá vyberá z databázy experimenty na základe zvoleného detekčného algoritmu a zvolených vstupných dát experimentu. To je implementované príkazom `db->get()`, pred ktorým sa však dotaz upresní pomocou príkazov `db->select()`, `db->from()` a `db->join()`.

Komponentami vzoru *MVC*, ktoré riadia celý proces obsluhy klientských požiadaviek sú ovládače. Tie sú implementované v adresári `web-app/application/controllers`. Pri spracovaní požiadaviek je často potrebné získať vstupné dáta od klienta. Tie sa získavajú pomocou knižnice *input*, ktorá je vo frameworku *CodeIgniter* štandardne načítaná. Po spracovaní požiadavky sa výsledok klientovi posiela prostredníctvom knižnice *loader* v prípade, že výsledkom je *HTML* stránka. V prípade, že výsledkom je chyba s konkrétnym *HTTP* kódom, výsledok sa pošle prostredníctvom knižnice *output*.

Jednou z náročnejších implementovaných akcií systému je pridanie vstupnej dátovej sady spolu s ďalšími informáciami o príslušnej sade. Táto požiadavka je spracovávaná vo funkcii `add()` v triede ovládača `Upload.php`. Nahranie a uloženie dátovej sady na server je realizované pomocou knižnice *upload*.

Ďalšou náročnou funkciou systému je spustenie samotného experimentu, ktorý sa vykonáva v priloženom skripte jazyka *Python*. Obsluha tejto klientskej požiadavky je implementovaná v ovládači `Experiment.php` presnejšie vo funkcii `execute_experiment()`. V nej sa najskôr na základe klientského vstupu zhromaždia všetky podstatné vstupné údaje experimentu. Ide hlavne o vstupnú dátovú sadu, na ktorej sa bude experiment vykonávať, ale tiež o údaje vybraného detekčného algoritmu. Následne je spustený spomínaný experimentálny skript, čo je implementované funkciou `exec()`. V prípade, že výsledkom skriptu nie je chybový kód, spracujú sa jeho výstupné údaje. Keďže ide o výsledky experimentu, tieto údaje sa najprv uložia do databázy a nakoniec sú vrátené klientovi.

5.2.4 Backend – detekcia anomálií v časových radoch

Ako je spomenuté v predchádzajúcej podkapitole, o experimentovanie sa stará samostatný skript implementovaný v programovacom jazyku *Python*. Jeho zdrojový kód je obsiahnutý v súbore `web-app/scripts/detection_app.py`. Je dôležité poznamenať, že jeden beh detekčného programu vykoná jeden experiment. To znamená detekciu anomálií na jednej dátovej sade s využitím jedného detekčného algoritmu.

Program najprv spracuje vstupné dáta zo štandardného vstupu. Sú to informácie o experimente, ktorý má vykonať. V nich sú zahrnuté detailné údaje o súbore dátovej sady vrátane cesty, údaje špecifikujúce časový interval dát dátovej sady, na ktorom sa má experiment

vykonať, údaje o vykonaní dekompozície časového radu vrátane dĺžky sezónnej periódy a detailné údaje o vybranom detekčnom algoritme spolu s používateľským nastavením jeho vstupných parametrov. Tieto vstupné údaje programu sú očakávane vo formáte *JSON*[12], pričom sú spracované prostredníctvom funkcie `loads()` balíčka *json*[13]. Implementácia tohto kroku programu sa nachádza v metóde `main`.

Druhým krokom hlavnej metódy je zavolanie funkcie `perform_experiment()`, ktorá riadi celý proces experimentu. Jej prvým dôležitým krokom je načítanie dát zo súboru dátovej sady a ich následné čistenie. Tento krok je zabezpečený funkciou `load_clean_data()`. Úlohou čistenia v tejto funkcii je nahradenie neplatných hodnôt priemernou hodnotou. Ide o priemernú hodnotu v rámci daného kalendárneho mesiaca alebo celého kalendárneho roka. Hodnoty sú neplatné, ak sa rovnajú neplatnej hodnote, alebo sú takto označené na základe stĺpca definujúceho kvalitu hodnôt. Pre všetky úkony nad dátami časových radov poskytuje podporu balíček *pandas*[14]. Ďalšou dôležitou volanou funkciou je `filter_time_series_by_time()`. Tá je ďalším krokom procesu experimentu, ktorý filtruje údaje časového radu na základe časového intervalu, ktorý bol zadaný používateľom. Pred detekciou sú hodnoty časového radu ešte normalizované do intervalu $\langle 0, 1 \rangle$. Zároveň sa v prípade potreby vykoná dekompozícia časového radu, ktorá je realizovaná funkciou `statsmodels_seasonal_decomposition()`. Tá využíva dekompozíciu `STL()` v rámci použitého balíčka *statsmodels*[15]. O samotnú detekciu sa stará funkcia `po_perform_detection()`. V jej úvode sa vyberie správny detekčný algoritmus balíčka *TODS* (popísaný v kapitole 4.2.2) a prostredníctvom funkcií `fit()` a `predict()` sa detekujú odľahlé hodnoty. Následne sa už len generujú výstupné súbory tabuliek či grafov, ktoré detailne popisujú výsledky experimentu. Grafy sú generované pomocou balíčka *matplotlib*[16].

Posledným krokom detekčného programu je vrátenie údajov na štandardný výstup. Tieto údaje obsahujú informácie o výsledkoch experimentu, a to hlavne vo forme absolútnych ciest k výsledným súborom. Výstup je štrukturovaný vo formáte *JSON*, čo je aj v tomto prípade zabezpečené balíčkom *json* a jeho funkciou `dumps()`.

5.3 Finálna funkčnosť a použitie

V nasledujúcom texte je zobrazená výsledná podoba aplikácie a rovnako popísaná jej funkčnosť. V tejto sekcii sú uvedené aj funkcionality, ktoré boli v rámci implementácie pridané k návrhu pre rozšírenie používateľských možností.

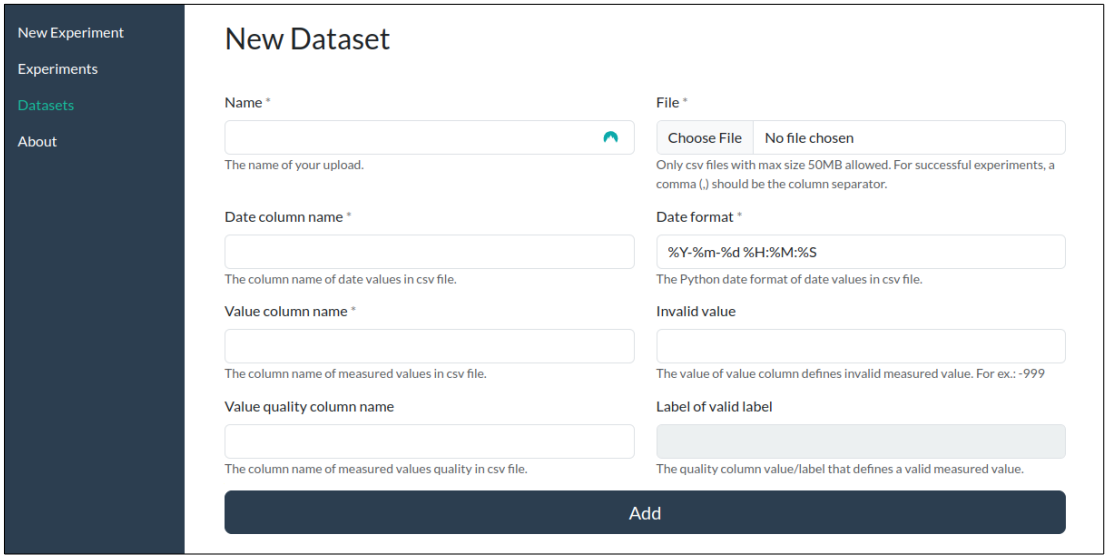
Na ukážke obrázku 5.6 je vidieť ľavý bočný panel, ktorý je súčasťou každej stránky aplikácie. Ide o hlavné menu aplikácie, prostredníctvom ktorého je používateľ schopný dostať sa do všetkých hlavných častí aplikácie.

Prvou predstavenou hlavnou časťou webovej aplikácie je **stránka dátových sád**. Stránka sa skladá z dvoch častí. V hornej časti sa nachádza formulár pre nahranie dátovej sady. Ukážka tejto časti je zobrazená na obrázku 5.6. Základnými prvkami formulára je zadanie vlastného názvu a vloženie súboru dátovej sady vo formáte *CSV*. Ďalšie elementy formulára predstavujú informácie o vloženom súbore. Je nutné zadať názov stĺpca pre časy i namerané hodnoty časového radu. Ďalším povinným údajom je formát, v ktorom sú uvedené časové značky. Tento formát musí zodpovedať časovému formátu programovacieho jazyka

[12] https://www.json.org/json-en.html
[13] https://docs.python.org/3/library/json.html
[14] https://pypi.org/project/pandas/
[15] https://www.statsmodels.org/stable/index.html
[16] https://matplotlib.org/

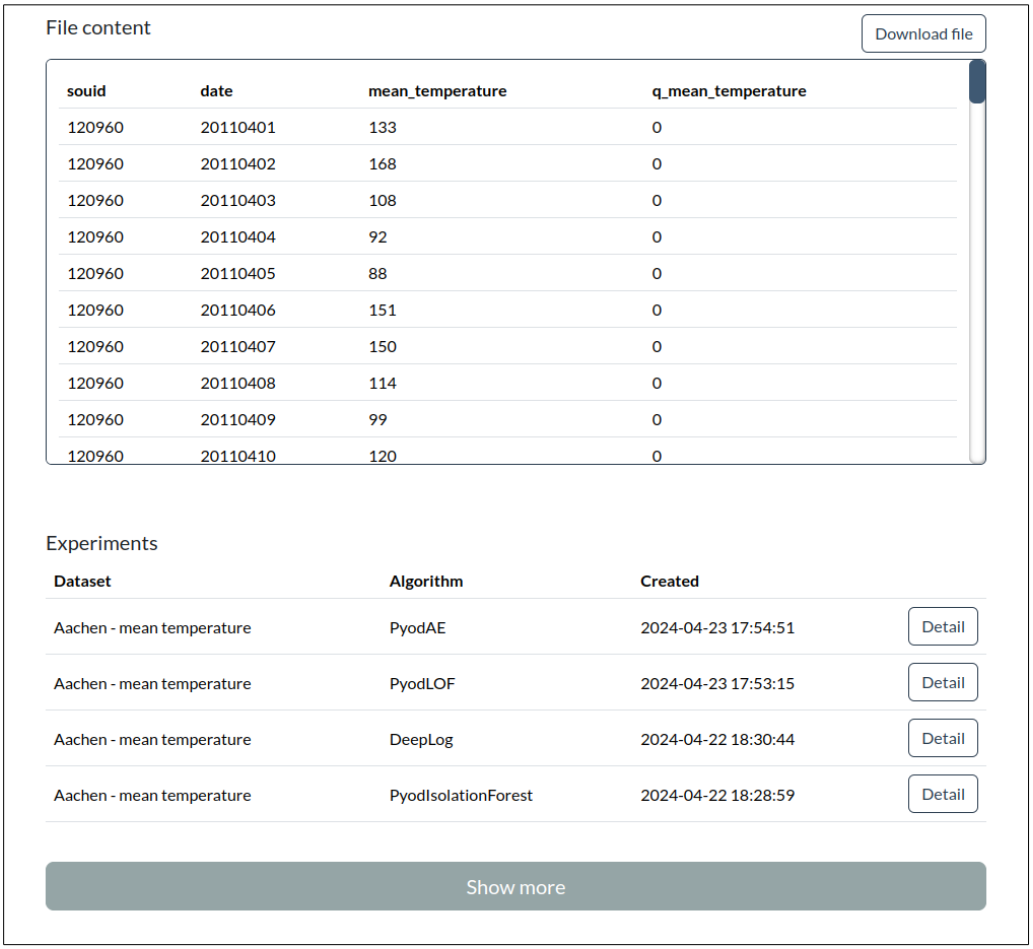
Python. Ak sa v dátovej sade nachádzajú neplatné hodnoty, používateľ ich má možnosť definovať. Prvou možnosťou je zadať číselnú hodnotu, ktorá neplatnú hodnotu predstavuje. Druhou možnosťou je určiť názov stĺpca, ktorý obsahuje informácie o platnosti jednotlivých nameraných hodnôt. Zároveň je potrebné, pre tento stĺpec zadať hodnotu označenia, ktorá symbolizuje platnú nameranú hodnotu.

Druhá dolná časť stránky obsahuje výpis pridaných dátových sád do systému. Ide o jednoduchú tabuľku, pomocou ktorej je možné identifikovať konkrétnu dátovú sadu a zobraziť jej detail.



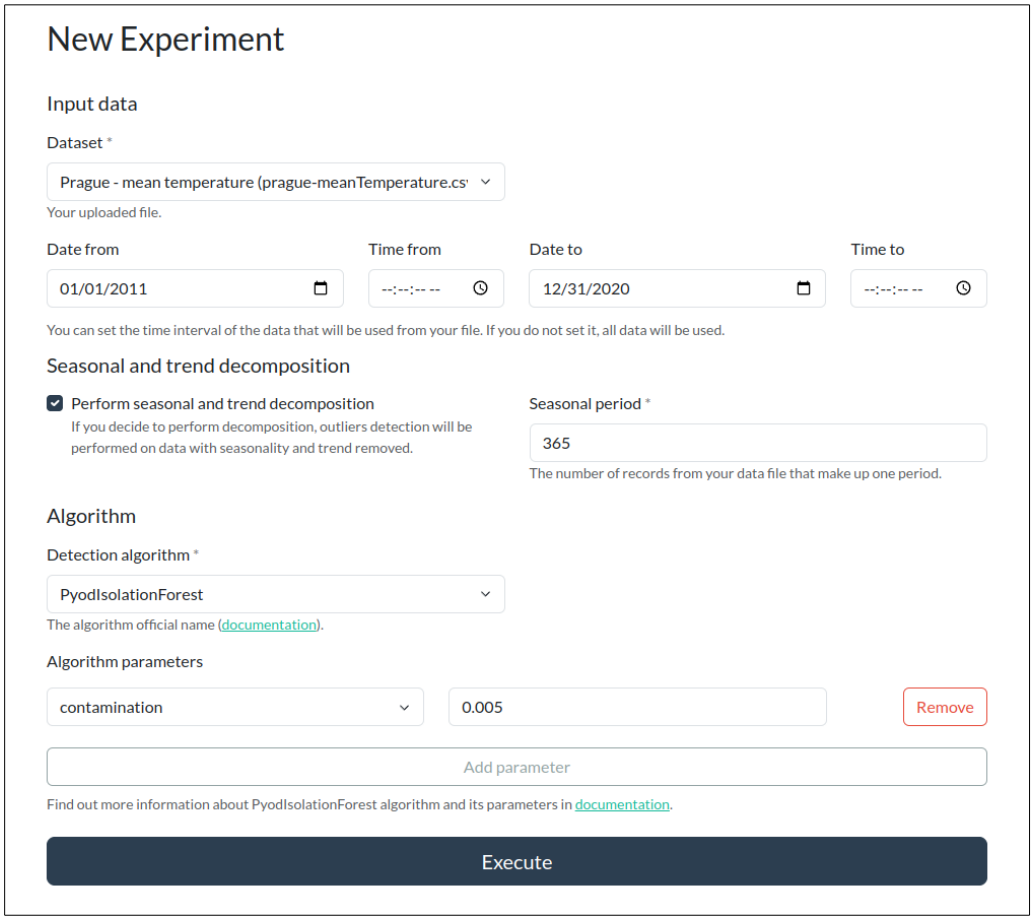
Obr. 5.6: Pohľad dátových sád – nahranie dátovej sady

Detail dátovej sady je ďalšou stránkou webovej aplikácie. Jej prvou časťou je výpis údajov, ktoré používateľ zadal do formulára pri nahrávaní zodpovedajúcej dátovej sady. Druhou časťou stránky je obsah nahraného súboru dátovej sady. Keďže takéto súbory môžu obsahovať príliš veľa záznamov, prvotné načítanie obsahuje len časť týchto záznamov. Po každom dosiahnutí konca zobrazeného obsahu sa dynamicky načítajú ďalšie záznamy. Používateľ má tiež možnosť stiahnuť celý obsah súboru na svoj lokálny stroj. Poslednou časťou stránky je tabuľkový zoznam vykonaných experimentov na danej dátovej sade. Druhú a tretiu časť stránky je možné vidieť na ukážke obrázku 5.7.



Obr. 5.7: Detail nahranej dátovej sady – obsah súboru a experimenty

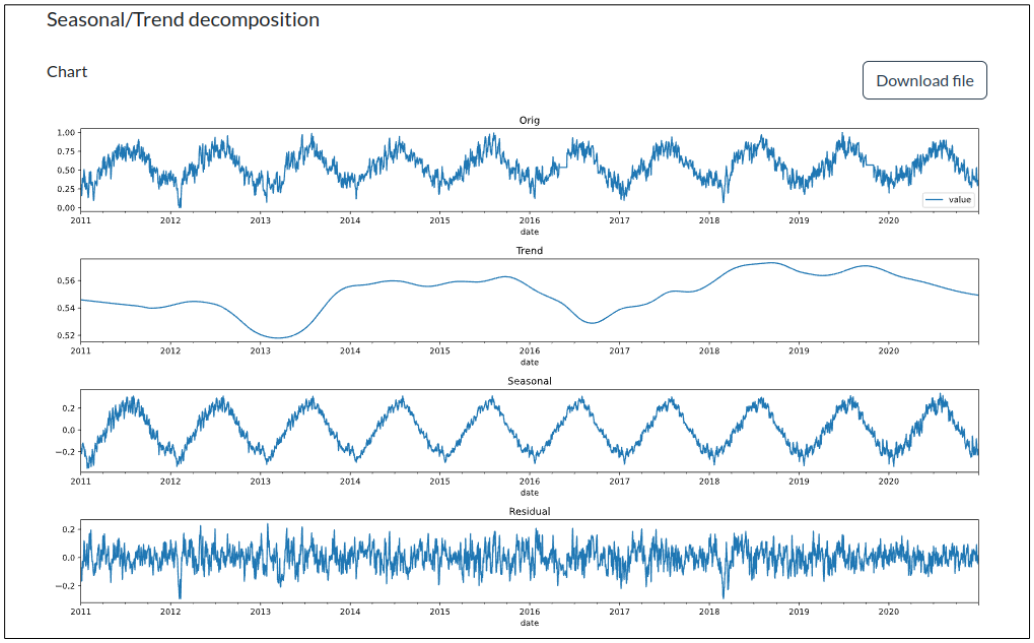
Nasledujúci prezentovaný pohľad je pohľad **nového experimentu**. Jeho základom je formulár pre konfiguráciu experimentu. V úvode formulára si používateľ volí vstupnú dátovú sadu a časový interval dát, ktorým špecifikuje vstupné dáta z dátovej sady. Vo formulári je tiež možné určiť, či bude vykonaná sezónna a trendová dekompozícia vstupného časového radu. V prípade, že používateľ potvrdí vykonanie dekompozície, je povinný zadať aj počet záznamov tvoriacich jednu sezónnu periódu. Na konci formulára je volený detekčný algoritmus, ktorým bude experiment vykonaný. Následne má používateľ možnosť nastaviť konfiguračné parametre zvoleného algoritmu. V prípade, že niektorý parameter nastavený nie je, použitá bude jeho predvolená hodnota. K dispozícií je tiež odkaz do oficiálnej dokumentácie k danému algoritmu, ktorá poskytuje detailné informácie o samotnom algoritme a jeho konfiguračných parametroch. Ukážka formulára pre nový experiment je na obrázku 5.8.



Obr. 5.8: Pohľad nového experimentu – konfiguračný formulár

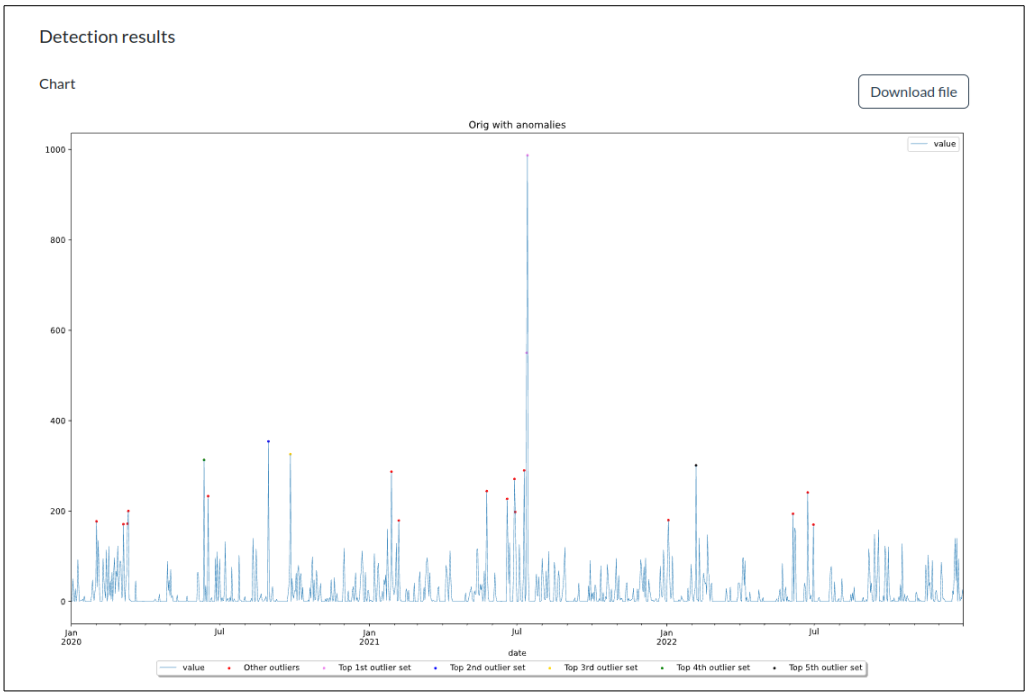
Po spustení a vykonaní experimentu sa pod formulárom zobrazia výsledky experimentu. Všetky vygenerované výsledky experimentu má používateľ možnosť stiahnuť na svoj lokálny stroj.

Prvou časťou výsledkov sú výstupné dáta čistenia a filtrácie. Tieto výsledky sú zobrazované vo forme grafu a tabuľky časového radu. V prípade, že používateľ povolil vykonanie dekompozície časového radu, tak jej výstupy sú ďalšou časťou experimentálnych výsledkov. Ide o súbor štyroch grafov: graf vstupných dát, graf trendového komponentu, graf sezónneho komponentu a graf dát zvyšku, ktoré sú neskôr použité ako vstup detekcie. Ukážku je vidieť na obrázku 5.9. Súčasťou dekompozície je aj tabuľka dát zvyšku.

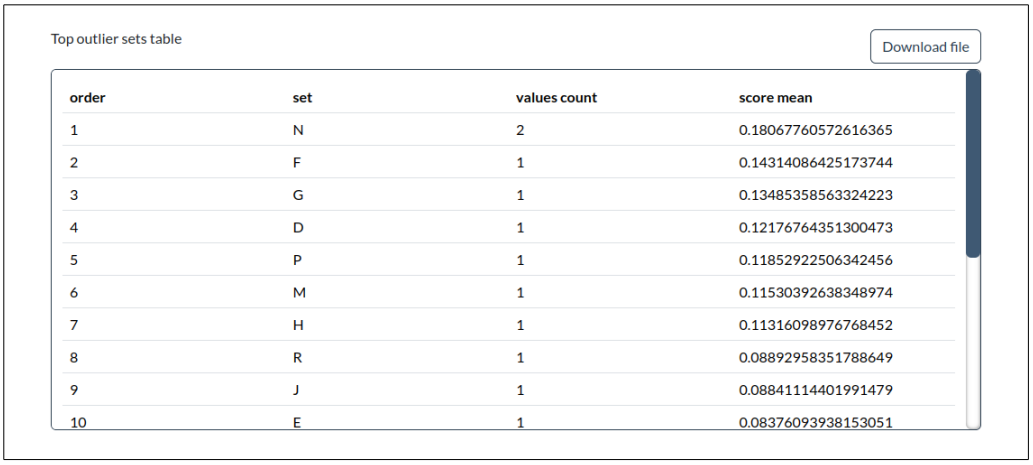


Obr. 5.9: Výsledky experimentu – dekompozícia časového radu

Poslednou časťou experimentálnych výsledkov sú výstupy samotnej detekcie anomálií. Tie pozostávajú z niekoľkých súborov. Zoznam a popis týchto súborov je možné vidieť v prílohe C, ktorá obsahuje tiež zoznam a popis výstupných súborov predspracovania dát, ktoré boli spomenuté v predchádzajúcom texte. Príkladmi výsledných súborov samotnej detekcie anomálií sú ukážky na obrázkoch 5.10 a 5.11.



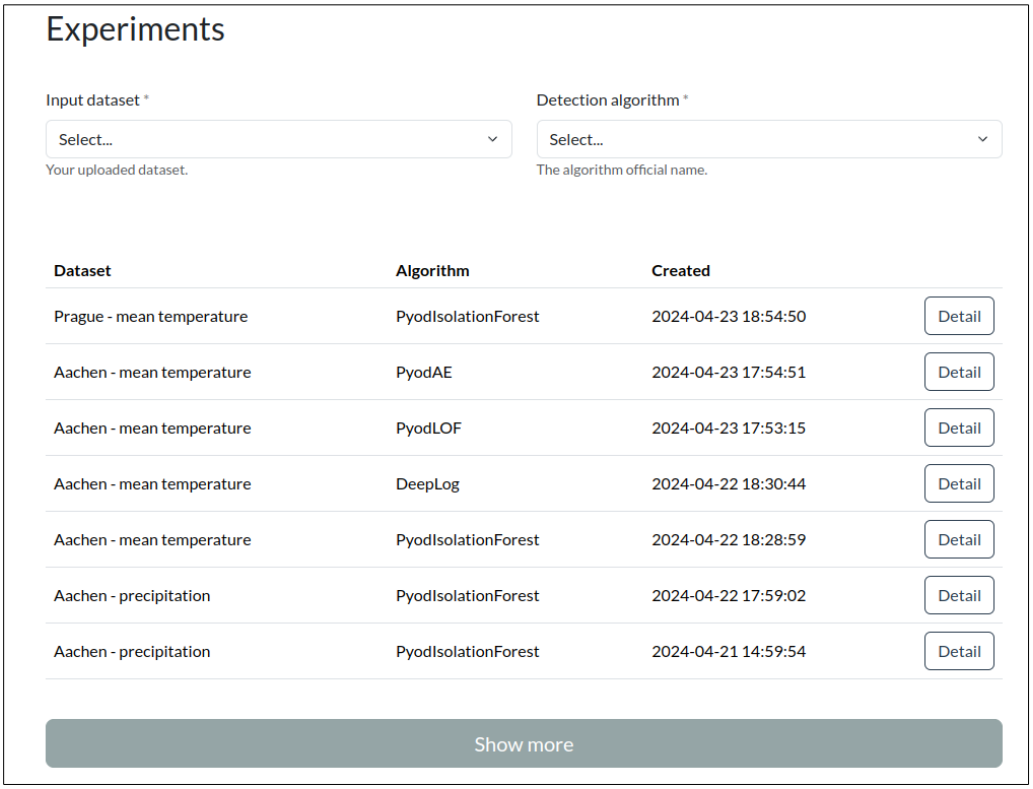
Obr. 5.10: Výsledky experimentu – súbor grafov s vyznačenými anomáliami



order	set	values count	score mean
1	N	2	0.18067760572616365
2	F	1	0.14314086425173744
3	G	1	0.13485358563324223
4	D	1	0.12176764351300473
5	P	1	0.11852922506342456
6	M	1	0.11530392638348974
7	H	1	0.11316098976768452
8	R	1	0.08892958351788649
9	J	1	0.08841114401991479
10	E	1	0.08376093938153051

Obr. 5.11: Výsledky experimentu – najvýznamnejšie odľahlé množiny

Treťou hlavnou časťou aplikácie je stránka **vykonaných experimentov**, ktorej úlohou je poskytnúť prehľad už vykonaných experimentov. Hlavnou časťou je tabuľka samotného prehľadu. Tá poskytuje základné informácie pre identifikáciu konkrétneho experimentu s tlačidlom pre zobrazenie jeho detailu. Aj v tomto prípade sa v prvotnom načítaní stránky zobrazuje len časť záznamov. Ďalšie záznamy sú dynamicky doplňované po každom kliknutí na tlačidlo pod tabuľkou. Záznamy experimentov v tabuľke je možné aj filtrovať pomocou jednoduchého formuláru vo vrchnej časti stránky. Filtrovať je možné podľa dátovej sady a detekčného algoritmu. V oboch prípadoch sa po výbere jednej z možností automaticky aktualizuje tabuľka záznamov.



Obr. 5.12: Pohľad vykonaných experimentov

Posledným pohľadom aplikácie je stránka **detailu experimentu**. V jej vrchnej časti sa nachádza výpis informácií, ktoré zadal používateľ do formulára pri konfigurácii experimentu. Druhou časťou tejto stránky sú výsledky daného experimentu. Táto časť zodpovedá výsledkom experimentu z pohľadu nového experimentu, ktorý bol popísaný v texte vyššie.

Kapitola 6

Experimenty

Táto kapitola obsahuje dve experimentálne úlohy, ktoré boli v rámci tejto práce vykonané. Cieľom prvej experimentálnej úlohy je porovnať detekčné algoritmy a zaradiť ich podľa typov anomálií, ktoré dokážu detekovať. Cieľom druhej experimentálnej úlohy je porovnať dané algoritmy na základe kvality, resp. významnosti detekovaných anomálií. V závere textu je uvedených pár príkladov praktického využitia detekcie anomálií v jednorozmerných časových radoch.

V tejto kapitole sa často používa terminológia spojená s anomáliami, ktorá môže miasť. Preto je v následujúcich riadkoch táto terminológia upresnená. Pri popise experimentov sa často vyskytuje slovné spojenie *odľahlá hodnota*. Týmto slovným spojením sa bude rozumieť jedna hodnota vstupného časového radu, ktorá bola detekovaná ako odľahlá. Ďalším často sa vyskytujúcim pojmom je termín *odľahlá množina (hodnôt)*. Ide o termín zvolený pre jasnejšiu interpretáciu výsledkov, pričom symbolizuje množinu pozostávajúcu z odľahlých hodnôt. Tento termín je v nasledujúcom texte často zamieňaný za pojem *anomália*. Jedna odľahlá množina zároveň zodpovedá *odľahlému bodu* (v prípade jednoprvkovej množiny) alebo *odľahlej podsekvencii* (v prípade množiny s počtom prvkom väčším ako 1). Odľahlý bod a odľahlá podsekvenica pritom zodpovedajú typom anomálií, ktoré boli definované v sekcii 3.2.2.

6.1 Prvá experimentálna úloha

V sekcii 4.3 tejto práce bolo uvedené, že jednou z experimentálnych úloh práce bude zaradenie vybraných algoritmov podľa typu anomálií, ktorý detekujú. Keďže sa v tejto práci pracuje s jednorozmernými časovými radmi, algoritmy môžu detekovať len dve druhy anomálií. Z troch typov anomálií definovaných v sekcii 3.2.2 ide o **odľahlé body** a **odľahlé podsekvencie**. Cieľom tejto úlohy je teda porovnať algoritmy a určiť, ktorý z týchto typov anomálií dokáže každý algoritmus detekovať.

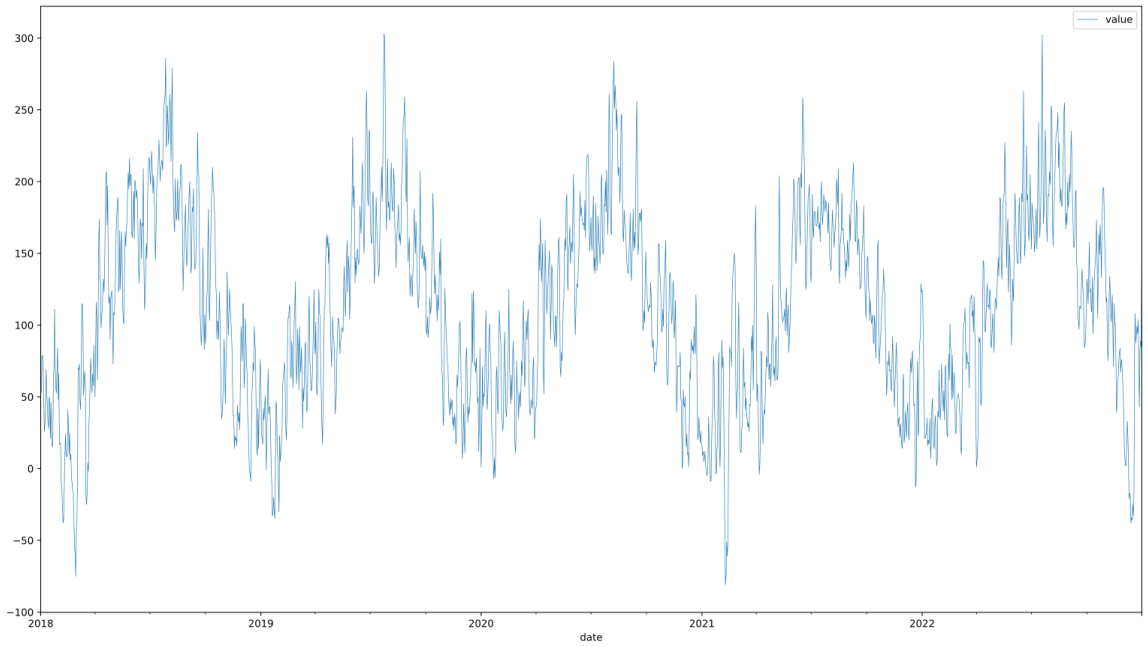
Ako už bolo spomenuté, v dokumentácii vybraného nástroja *TODS* pre detekciu anomálií v časových radoch nie je uvedené, aké typy anomálií dokážu jednotlivé algoritmy detekovať. Napriek tomu sa v spomínanej dokumentácii dá všimnúť, že tri z vybraných detekčných algoritmov obsahujú parameter veľkosti posuvného okna (oficiálny názov: *window_size*). Ide o algoritmy *PCAODetect*, *LSTMODetect* a *KDiscordODetect*. V odbore detekcie anomálií v časových radoch sa posuvné okno často využíva pri výpočte skóre odľahlosti konkrétnej hodnoty. Skóre hodnoty je v takomto prípade vypočítané aj na základe okolitých hodnôt v danom okne. Takýto prístup je účinný hlavne pri detekcii odľahlých pod-

sekvencií. Na základe tohto poznatku možno určiť predpoklad, že tri spomenuté detekčné algoritmy by mohli byť vhodnejšie pre detekciu odľahlých podsekvencií. Naopak ostatné algoritmy by mohli byť vhodnejšie pre detekciu odľahlých bodov.

Algoritmy používaného systému *TODS* vracajú výsledok ako pole hodnôt, v ktorom hodnota každého elementu definuje odľahlosť pre zodpovedajúcu hodnotu vstupného časového radu. Toto výstupné pole obsahuje hodnoty 0, ktoré charakterizujú neodľahlé hodnoty, a hodnoty 1, ktoré charakterizujú odľahlé hodnoty. Klasifikácia typu odľahlej hodnoty je vykonaná naivným spôsobom. Samostatná odľahlá hodnota je klasifikovaná ako odľahlý bod, zatiaľ čo susediace odľahlé hodnoty sú spolu klasifikované ako odľahlá podsekvencia.

6.1.1 Dátová sada a predspracovanie dát

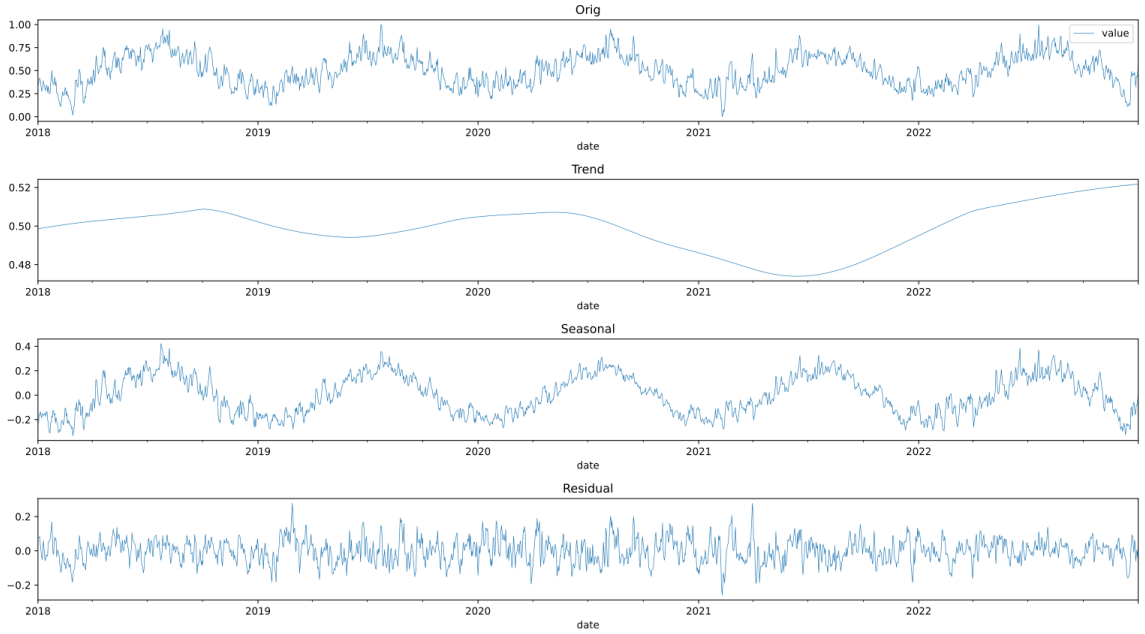
Pre túto experimentálnu úlohu bola vybraná dátová sada, ktorá popisuje priemernú dennú hodnotu teploty vzduchu v nemeckom meste Aachen. Táto dátová sada, ktorá bola popísaná v kapitole 4.1, neobsahuje žiadne neplatné namerané hodnoty. Napriek tomu však prešla procesom čistenia a filtrácie. Pre jednoduchú názornosť bol zvolený päťročný interval (od roku 2018 do roku 2022), na základe ktorého boli dáta filtrované. Ukážku čistených a filtrovaných dát je možné vidieť na obrázku 6.1. Z ukážky je možné usúdiť, že údaje vykazujú nemalé množstvo anomálií. Na základe toho sú tieto dáta vhodným vstupom pre následujúce experimenty tejto úlohy.



Obr. 6.1: Dáta dátovej sady po čistení a filtrácii

Druhou časťou predspracovania vybraných dát je dekompozícia časového radu. Je nesporným faktom, že vývoj teploty vzduchu v exteriéri je v rámci jedného roka ovplyvnená ročnými obdobiami. Z tohto dôvodu tento meteorologický parameter vykazuje údaje s výraznou sezónnou závislosťou, ktorej perióda je rovná jednému roku. To je tiež vidieť na predchádzajúcej ukážke vybraných dát. Pre správnu detekciu anomálií je teda nutné vykonať dekompozíciu časového radu, ktorá bola definovaná v kapitole 3.1.1. Dekompozícia

daného časového radu je zobrazená na obrázku 6.2. Nachádza sa v ňom súbor štyroch grafov, ktoré postupne reprezentujú vstupné dáta, trendovú zložku, sezónnu zložku a zvyšok po dekompozícii. Na údajoch zvyšku je jasne vidieť, že dáta boli zbavené od sezónnosti a trendu. Tieto údaje je teda vhodné použiť v ďalšom kroku získavania znalostí, ktorým je dolovanie z dát. To znamená, že spomínané dáta budú vstupom vybraných detekčných algoritmov.



Obr. 6.2: Dekompozícia dát časového radu

6.1.2 Popis vykonaných experimentov

Pre každý algoritmus (okrem algoritmu *Deeplog*) boli vykonané dva experimenty. V nich boli detekčné algoritmy spúšťané s predvolenými hodnotami takmer všetkých svojich konfiguračných parametrov. Jediným nastavovaným konfiguračným parametrom bol parameter s oficiálnym názvom *contamination*, ktorý určuje podiel odľahlých hodnôt v dátovej sade prostredníctvom desatinného čísla. Tento konfiguračný parameter je spoločný pre všetky vybrané algoritmy s výnimkou algoritmu *Deeplog*, ktorý neposkytuje žiadne konfiguračné parametre. Dôvodom nastavenia spomínaného parametra je predpoklad, že jeho hodnota môže výrazne ovplyvniť typ detekovaných anomálií – jeho vyššia hodnota by teoreticky mohla spôsobiť detekciu vyššieho podielu odľahlých podsekvencií. Preto bola pri experimentoch použitá nižšia hodnota tohto parametra. V prvom experimente každého algoritmu (okrem algoritmu *Deeplog*) bola jeho zvolená hodnota 0.03, čo značí, že v dátovej sade sú zo všetkých hodnôt odľahlé 3%. Keďže detekčný algoritmus *Deeplog* neponúka žiadne parametre, bol na ňom vykonaný experiment bez jeho akejkoľvek konfigurácie. Po jeho vykonaní bolo z výsledkov tabuľky štatistík jasné, že algoritmus detekuje 10% hodnôt dátovej sady ako odľahlých. Aby bolo porovnanie tohto algoritmu s ostatnými čo najobjektívnejšie, bol pre každý algoritmus vykonaný tiež druhý experiment s hodnotou parametra *contamination* 0.1. Navyše toto nastavenie algoritmov môže poskytnúť čiastočné potvrdenie experimen-

tálnych výsledkov predošlej konfigurácie algoritmov, a zároveň dať predstavu o závislosti podielu detekovaných odľahlých podsekvencií od parametru *contamination*.

Pre jasné pochopenie následujúceho textu bude každý experiment s nastavením konfiguračného parametru na hodnote 0.03 označovaný ako prvý experiment príslušného algoritmu. Naopak každý experiment s nastavením daného parametru na hodnote 0.1 bude označovaný ako druhý experiment príšlušného algoritmu.

6.1.3 Analýza výsledkov experimentov

V tejto sekcii sú podrobnejším spôsobom popísané a interpretované zaujímavé výsledky vykonaných experimentov. Interpretované výsledky sú získané z výsledných grafov s vyznačenými anomáliami a z tabuliek štatistík, ktoré sú výsledkom každého vykonaného experimentu vo webovej aplikácii. Názvy súborov, ktoré zodpovedajú týmto tabuľkám a grafom možno nájsť v prílohe C. Prehľadný súhrn spomínaných výsledkov možno vidieť v tabuľke 6.1, z ktorej vychádza aj nasledujúci text.

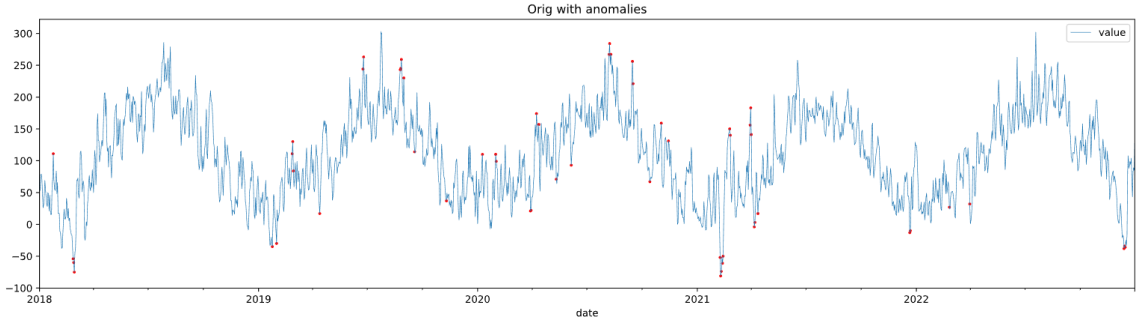
Algoritmus	Nastavený podiel anomálií	Odľahlé množiny	Priem. poč. prvkov množiny	Odľahlé body	Odľahlé podsekvencie
PyodIsolationForest	0.03	34	1.62	19 (56%)	15 (44%)
	0.1	96	1.9	52 (54%)	44 (46%)
PCAODetect	0.03	12	4.58	3 (25%)	9 (75%)
	0.1	23	7.91	2 (9%)	21 (91%)
Deeplog	0.03				
	0.1	148	1.24	122 (82%)	26 (18%)
PyodKNN	0.03	25	1.56	15 (60%)	10 (40%)
	0.1	78	1.58	49 (63%)	29 (37%)
LSTMODetect	0.03	6	9.17	0 (0%)	6 (100%)
	0.1	14	13.07	1 (7%)	13 (93%)
PyodABOD	0.03	36	1.47	25 (69%)	11 (31%)
	0.1	111	1.53	75 (68%)	36 (32%)
PyodLOF	0.03	33	1.21	29 (88%)	4 (12%)
	0.1	115	1.24	91 (79%)	24 (21%)
PyodAE	0.03	36	1.53	23 (64%)	13 (36%)
	0.1	88	2.08	46 (52%)	42 (48%)
PyodSoGaal	0.03	33	1.67	19 (58%)	14 (42%)
	0.1	81	2.26	43 (53%)	38 (47%)
KDiscordODetect	0.03	7	6.14	2 (29%)	5 (71%)
	0.1	20	7.7	2 (10%)	18 (90%)

Tabuľka 6.1: Tabuľka výsledkov vykonaných experimentov

Algoritmus *PyodIsolationForest* vo svojom prvom experimente detekoval celkom 34 odľahlých množín, z čoho 56% sú odľahlé body a zvyšných 44% sú odľahlé podsekvencie. Priemerný počet hodnôt jednej odľahlej množiny je pritom 1.62. To robí z algoritmu *PyodIsolationForest* pomerne vyvážený detektor z pohľadu typu detekovaných anomálií. Na obrázku 6.3 je zobrazený výsledný graf tohto experimentu. Ide o graf časového radu z prvej časti predspracovania spolu s vyznačenými detekovanými odľahlými hodnotami. Z ukážky je možné vidieť spomínanú vyváženosť algoritmu medzi detekovanýcmi odľahlými bodmi

a odľahlými podsekvenciami. Pri druhom experimente tohto algoritmu, s nastavením parametru *contamination* na hodnote 0.1, sa výsledky markantne nezmenili. Podiel odľahlých podsekvencií v odľahlých množinách stúpol len o 2%.

Na základe sumarizačnej tabuľky je možné k podobne vyváženým algoritmom zaradiť aj algoritmy *PyodSoGaal* a *PyodKNN*. Oba tieto algoritmy pri prvom experimente dosiahli podiel detekovaných odľahlých podsekvencií v hodnote 40% a viac. Dokonca aj priemerný počet prvkov v jednej odľahlej množine sa medzi spomenutými algoritmami líši len o 0.11. Zaujímavosťou však je, že zatiaľ čo pri druhom experimente algoritmu *PyodSoGaal* podiel detekovaných podsekvencií stúpol o 5%, pri algoritme *PyodKNN* tento podiel klesol o 3%. Keďže aj priemerný počet prvkov odľahlej množiny sa pri druhom experimente algoritmu *PoydKNN* zmenil len minimálne, možno tento algoritmus označiť za konzistentný voči zmene hodnoty konfiguračného parametra *contamination*.



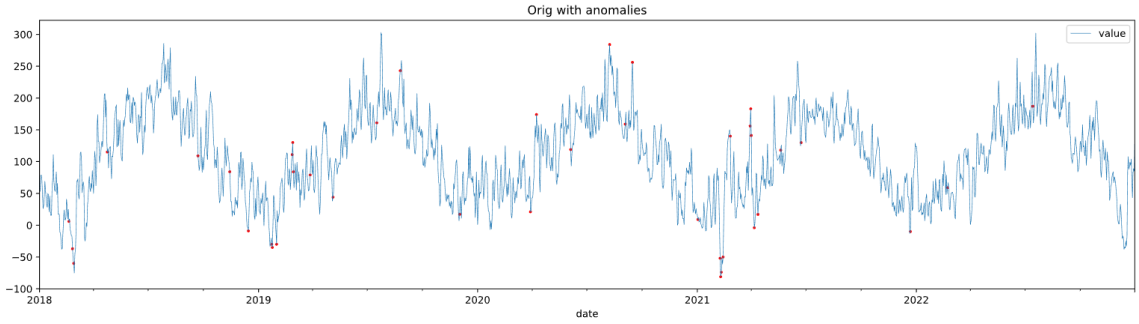
Obr. 6.3: Výsledný graf prvého experimentu algoritmu *PyodIsolationForest*

Zaujímavé výsledky dosiahol algoritmus *PyodAE*. Ten oproti predošlým algoritmom dosiahol pri prvom experimente podiel odľahlých podsekvencií na úrovni 36%. Pri druhom experimente sa však tento podiel zvýšil na hodnotu 48%. K zvýšeniu došlo aj pri priemernej mohutnosti jednej odľahlej množiny. Na základe uvedeného nárastu podielu odľahlých podsekvencií možno tento algoritmus tiež označiť za vyvážený z pohľadu typu detekovaných anomálií.

Algoritmus *PyodABOD* dosiahol pri prvom experimente podiel detekovaných odľahlých podsekvencií na úrovní 32% s priemerným počtom prvkov jednej odľahlej množiny v hodnote 1.53. Pri druhom vykonanom experimente sa obe tieto metriky zmenili minimálne. Týmto možno aj tento algoritmus označiť za konzistentný z pohľadu zmeny hodnoty konfiguračného parametra *contamination*. Čo sa však týka typu detekovaných anomálií, je možné o tomto algoritme tvrdiť, že detekuje predovšetkým odľahlé body.

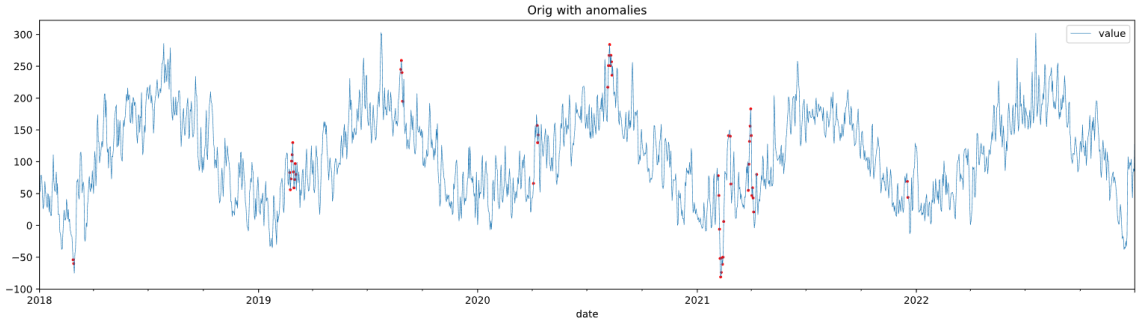
Do skupiny algoritmov, ktoré detekujú predovšetkým odľahlé body je určite možné zaradiť aj algoritmy *PyodLOF* a *DeepLog*. Prvý menovaný dosiahol vo svojom prvom experimente podiel detekovaných odľahlých podsekvencií len na úrovni 12%, čo znamená detekciu odľahlých bodov až na úrovni 88% pri priemernej mohutnosti odľahlej množiny v hodnote 1.21. Pri druhom experimente síce podiel odľahlých bodov klesol o 9%, napriek tomu však ide stále o vysokú úroveň. Lepší výsledok pri tomto experimente však dosiahol algoritmus *DeepLog*. Ako už bolo povedané, vzhľadom na absenciu konfiguračných parametrov algoritmu ide o jediný experiment vykonaný na tomto algoritme. V ňom však dosiahol podiel detekovaných odľahlých bodov na úrovni 82%, čo je najlepší výsledok pre detekekciu odľahlých bodov v časových radoch. Na obrázku 6.4 je ukážka prvého experimentu algoritmu

PyodLOF, na ktorej možno vidieť, že počet detekovaných odľahlých bodov jasne prevažuje nad počtom odľahlých podsekvencií.



Obr. 6.4: Výsledný graf prvého experimentu algoritmu *PyodLOF*

Poslednými tromi algroritmami, na ktorých boli vykonané experimenty sú *PCAODetect*, *LSTMODetect* a *KDiscordODiscord*. V tabuľke výsledkov je vidieť, že ide o algoritmy, detekujúce hlavne odľahlé podsekvencie. *PCAODetect* dosiahol v prvom experimente podiel detekovaných odľahlých podsekvencií v hodnote 75%, zatiaľ čo algoritmus *KDiscordODetect* o 4% menej. Zaujímavým údajom v tomto porovnaní je priemerná veľkosť odľahlej množiny, ktorá súvisí s počtom detekovaných odľahlých množín, resp. odľahlých podsekvencií. U algoritmu *KDiscordODetect* je táto priemerná hodnota vyššia o 1.56, čo znamená, že tento algoritmus detekoval menej odľahlých podsekvencií s väčšou dĺžkou. Pútavý výsledok v svojom prvom experimente dosiahol tiež algoritmus *LSTMODetect*, ktorého podiel detekovaných odľahlých podsekvencií nadobudol hodnotu rovných 100% s priemernou mohutnosťou odľahlnej množiny v hodnote 9.17. Aj keď detekoval len o jednu odľahlú množinu menej, priemerne obsahuje jeho jedna odľahla množina o vyše tri odľahlé hodnoty viac ako v prípade algoritmu *KDiscordODetect*. To indikuje ešte dlhšie odľahlé podsekvencie ako pri predchádzajúcich dvoch detekčných algoritmoch. V druhom experimente všetky tieto tri algoritmy dosiahli podobný podiel detekovaných odľahlých podsekvencií s hodnotami nad 90%. V tomto prípade sa výsledkom odlišuje len *LSTMODetect*, a to priemerným počtom prvkom odľahlých množín, ktorý je o vyše päť prvkov vyšší ako u algoritmov *PCAODetect* a *KDiscordODiscord*. Na obrázku 6.5 je ukážka výsledného grafu prvého experimentu algoritmu *PCAODetect*, na ktorej možno vidieť prevažnú detekciu odľahlých podsekvencií.



Obr. 6.5: Výsledný graf prvého experimentu algoritmu *PCAODetect*

6.1.4 Záver

V tejto experimentálnej úlohe boli vybrané detekčné algoritmy porovnané na základe typu odľahlých hodnôt, ktoré detekujú. Po analýze výsledkov vykonaných experimentov je možno učiniť záver, že každý z algoritmov dokáže detekovať oba typy anomálií, ktoré sa v jednorozmernom časovom rade môžu vyskytovať. Týmito typmi sú odľahlý bod a odľahlá podsekvencia. Algoritmy sa však líšia hodnotami podielov detekovaných typov anomálií. Na základe týchto rozdielov boli algoritmy zaradené do troch skupín nasledovne:

- algoritmy detekujúce prevažne odľahlé body: PyodLOF, DeepLog, PyodABOD,
- algoritmy s vyváženou detekciou oboch typov: PyodIsolationForest, PyodKNN, PyodAE, PyodSoGaal,
- algoritmy detekujúce prevažne odľahlé podsekvencie: PCAODetect, LSTMODetect, KDiscordODetec.

Na základe tohto rozdelenia možno potvrdiť správnosť predpokladu, uvedeného v úvode tejto experimentálnej úlohy, o existencii konfiguračného parametra *window_size* a jeho vplyvu na detekciu odľahlých podsekvencií.

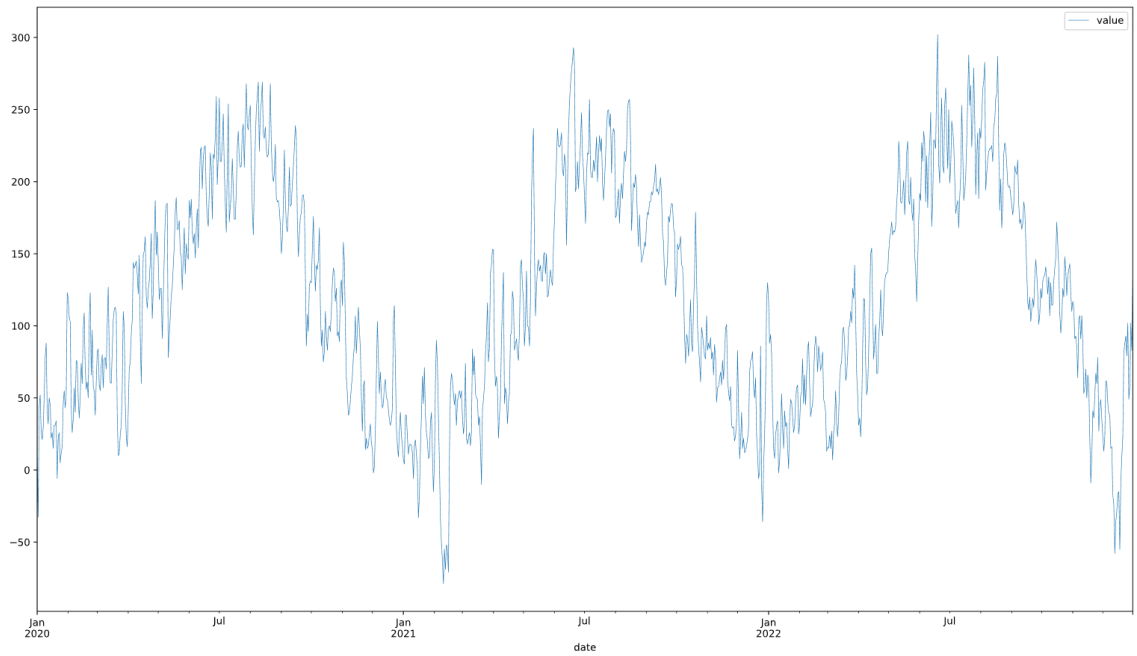
6.2 Druhá experimentálna úloha

Úlohou tejto experimentálnej úlohy je porovnať vybrané detekčné algoritmy na základe kvality konkrétnych detekovaných anomálií. To znamená, že u každého algoritmu budú popísané a rozoberané odľahlosti, ktoré algoritmus určil ako najvýznamnejšie. Významnosť jednotlivých anomálií je získaná z hodnoty ich skóre, ktoré pri procese detekovania získali od príslušného detekčného algoritmu. V detekčnom systéme *TODS* totiž platí, že čím vyššie skóre odľahlá hodnota získa, tým viac odľahlá táto hodnota je. Zároveň je potrebné zdôrazniť, že každý algoritmus má vlastnú škálu ohodnocovania skóre, preto nie je možné porovnať dve ľubovoľné anomálie z dvoch rôznych detekčných algoritmov. Preto bude táto experimentálna úloha zameraná na umiestnenie anomálií v rebríčkoch významnosti každého experimentu.

6.2.1 Dátová sada a predspracovanie dát

Pre vykonanie experimentov tejto úlohy boli vybrané dve dátové sady, ktorých dáta majú odlišný charakter. Ide o dátovú sadu dennej priemernej teploty v českom hlavnom meste Praha a zároveň dátovú sadu denného dopadnutého množstva zrážok v nemeckom meste Aachen. Obe tieto dátové sady boli popísane v sekcii 4.1.

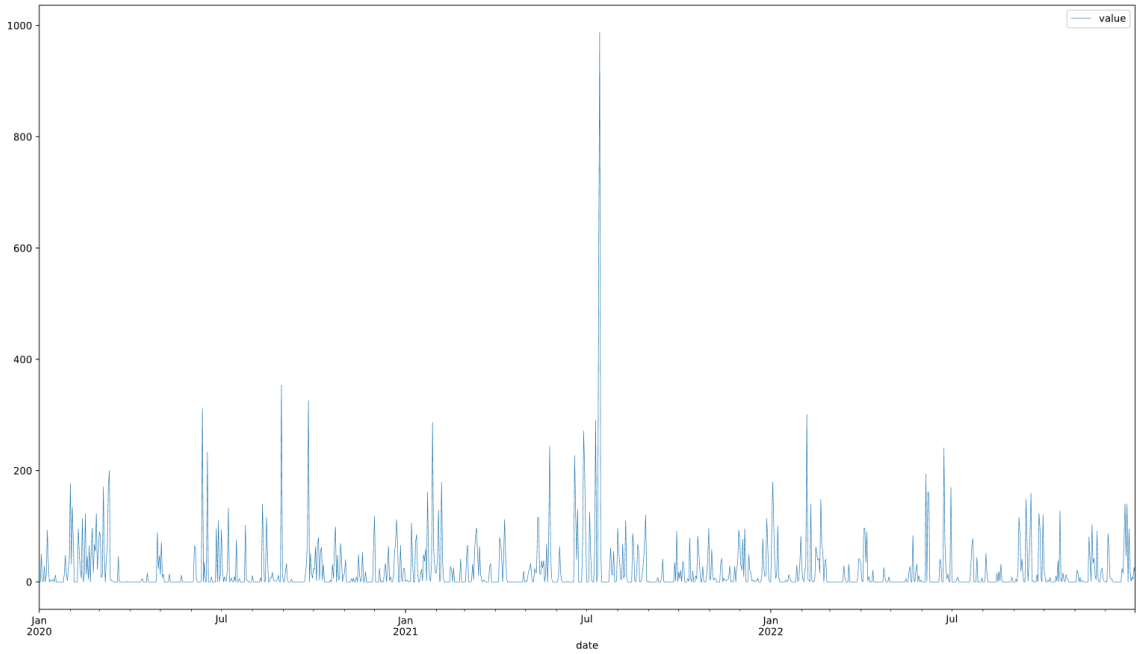
Ako prvá dátová sada bude v nasledujúcom texte označovaná sada určujúca priemernú teplotu v Prahe. Dáta tejto sady obsahujú aj neplatné hodnoty. Presnejšie ide o dva kalendárne mesiace, v ktorých nie je určená platná hodnota. V procese čistenia dát bola všetkým dňom týchto kalendárnych mesiacov pridelená priemerná hodnota ich zodpovedajúcich kalendárnych rokov. Následne bol filtrovaný trojročný interval dát tejto dátovej sady (od roku 2020 do roku 2022). Ako je vidno na obrázku 6.6 tieto dáta, podobne ako v predchádzajúcej experimentálnej úlohe, vykazujú silnú sezónnu závislosť. Preto je súčasťou predspracovania týchto dát aj dekompozícia časového radu.



Obr. 6.6: Prvá vstupná dátová sada experimentov po čistení a filtrácii

Dátová sada, ktorá bude v nasledujúcom texte označená ako druhá, popisuje denné množstvo dopadnutých zrážok v Aachene. Táto dátová sada neobsahuje žiadne neplatné hodnoty. Filtráciou dát bol opäť vybraný trojročný interval údajov (od roku 2020 do roku 2022). Ako možno vidieť na obrázku 6.7, tieto dáta vykazujú veľmi nízku sezónnu a trendovú závislosť. Preto pri predspracovaní týchto údajov nebola vykonaná dekompozícia časového radu.

Oproti dátam priemernej dennej teploty ide o dáta so skutočne iným charakterom. Zatiaľ čo sa hodnota priemernej teploty s pribúdajúcim časom a rôznymi výkyvmi plynule mení, u týchto údajov ide skôr o nárazové vzostupy z minimálnej hodnoty príslušného oboru hodnôt. Tento odlišný charakter dátových sád môže prispieť ku kvalite porovnania detekčných algoritmov.



Obr. 6.7: Druhá vstupná dátová sada experimentov po čistení a filtrácii

6.2.2 Popis vykonaných experimentov

Aj v tejto experimentálnej úlohe sú pre každý detekčný algoritmus vykonané dva experimenty. Dôvodom je zvolenie dvoch dátových sád. Prvým experimentom algoritmu sa bude rozumieť experiment vykonaný na prvej dátovej sade, ktorou je vývoj priemernej dennej teploty v Prahe. Naopak ako druhý experiment konkrétneho algoritmu bude označený experiment vykonaný na druhej vybranej dátovej sade, ktorou je vývoj dopadnutých denných zrážok v Aachene.

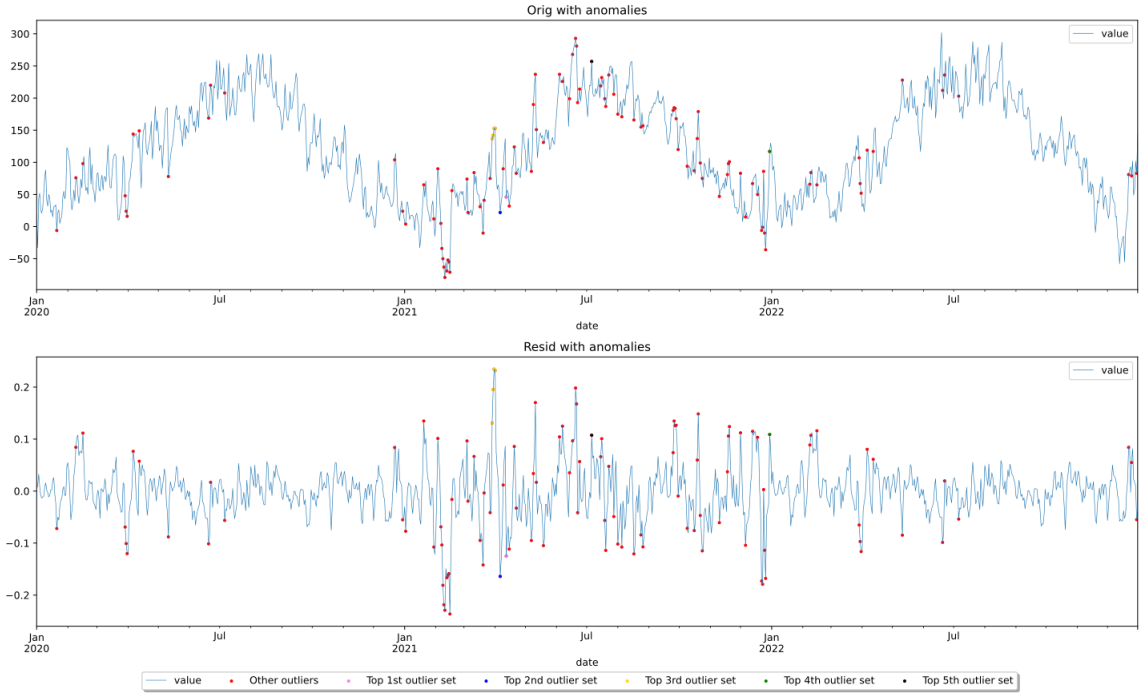
Všetky experimenty boli vykonané s predvoleným nastavením konfiguračných parametrov algoritmov. Jedinou výnimkou je aj v tomto prípade parameter *contamination*, ktorý je spoločný pre všetky algoritmy okrem algoritmu *DeepLog*. Ten bude spúšťaný bez akejkoľvek zmeny konfigurácie. V tejto experimentálnej úlohe je parameter *contamination* nastavovaný na hodnotu 0.02, čo značí podiel odľahlých hodnôt v dátovej sade na úrvoni 2%. Táto nízka hodnota bola zvolená z dôvodu, aby výsledky experimentov neboli zahltené príliš veľkým množstvom anomálií. Druhým dôvodom je, že všeobecne anomálie nedosahujú v dátach príliš veľký podiel.

6.2.3 Analýza výsledkov experimentov

Podobne ako v predchádzajúcej experimentálnej úlohe sú v tejto časti tiež popísané a interpretované výsledky vykonaných experimentov. Tieto výsledné údaje sú získané najmä z tabuliek poradia odľahlých množín, tabuliek poradia odľahlých bodov, tabuliek poradia odľahlých podsekvencií a grafov s vyznačenými anomáliami, ktoré sú výsledkom každého experimentu vo webovej aplikácii. Názvy súborov, ktoré zodpovedajú týmto tabuľkám a grafom, možno nájsť v prílohe C.

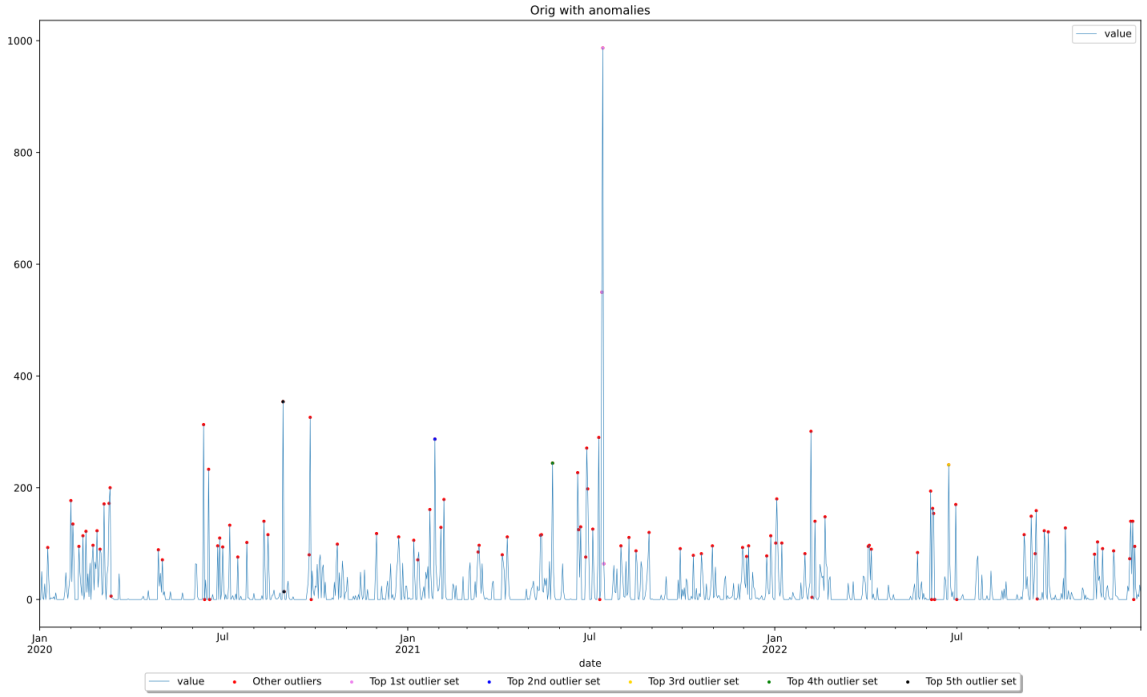
DeepLog

Detekčný algoritmus *DeepLog* vo svojom prvom experimente detekoval ako najvýznamnejšiu anomáliu neočakávaný odľahlý bod. Tento bod predstavuje hodnotu, ktorá sa len veľmi málo približuje globálnemu minimu v zvyšných dátach po dekompozícii. Táto hodnota však predstavuje pomerne značný pokles od predchádzajúcej hodnoty. Aj keď nejde o najvýraznejšiu zmenu hodnoty v dátach, môže to byť dôvodom najvyššieho dosiahnutého skoré pri detekcii. Treťou najvýznamnejšou anomáliou je odľahlá podsekvencia štyroch hodnôt, ktorej najvyššia hodnota predstavuje aj globálne maximum v dátach po dekompozícii. Aj v tomto prípade ide hodnoty, ktoré sa vyskytujú po prudkej zmene hodnoty.



Obr. 6.8: Výsledok detekcie algoritmu *DeepLog* na prvej dátovej sade

Ani v experimente na druhej dátovej sade, ktorej dáta majú iný charakter, sa nedá tvrdiť žeby algoritmus detekoval na prvých miestach významností anomálie, ktoré by zodpovedali najvyšším hodnotám v dátach. Takéto správanie by v dátach s podobným charakterom mohlo dávať zmysel. Dôvodom absencie takéhoto správania však môže byť príliš veľký podiel detekovaných odľahlých hodnôt, ktorý sa pri algoritme *DeepLog* nedá nastaviť prostredníctvom konfiguračných parametrov. Tým sú niektoré vysoké hodnoty súčasťou detekovaných odľahlých podsekvencií, čím dosahujú nižšie priemerné skóre. Na základe tohto faktu je možné povedať, že v dátach s podobným charakterom detekuje algoritmus za významnejšie anomálie hodnoty, ktoré patria k najvyšším hodnotám dát. Za najvýznamnejšiu anomáliu bola zvolená odľahlá podsekvencia, ktorej najvyššia hodnota je jednoznačným globálnym maximom vstupných dát.

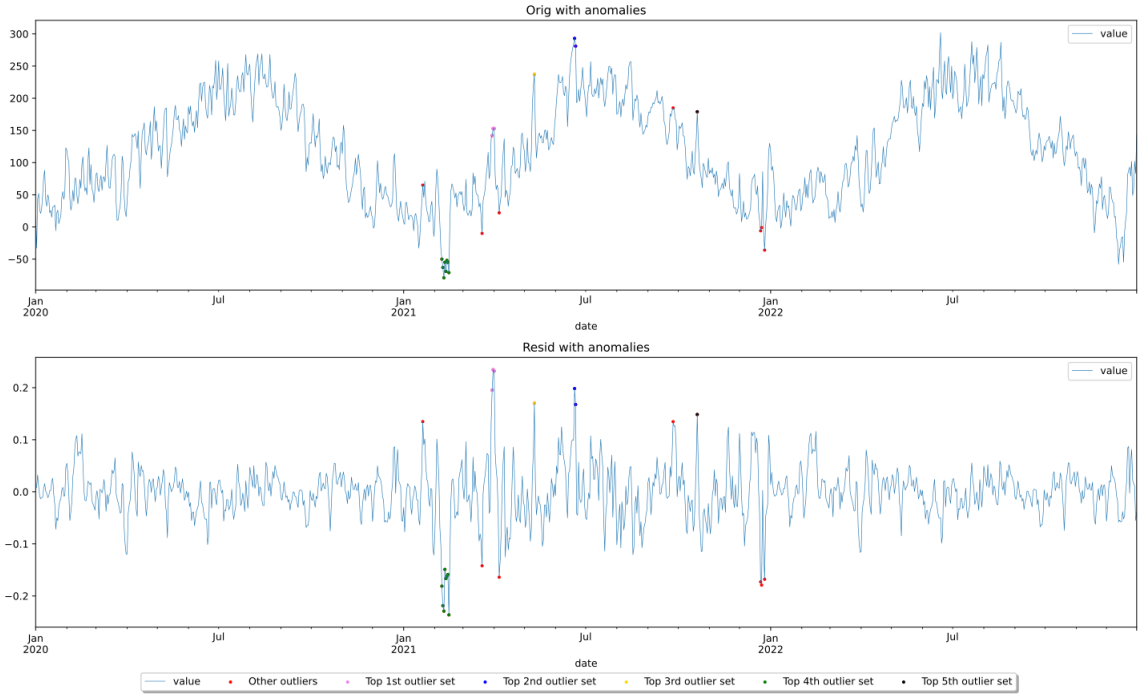


Obr. 6.9: Výsledok detekcie algoritmu *DeepLog* na druhej dátovej sade

Na základe dosiahnutých výsledkov možno povedať, že tento algoritmus dokáže spoľahlivo a konzistentne detekovať anomálie v oboch typoch testovaných dát. Z týchto anomálií však za najvýznamnejšie nie sú nutne považované tie, u ktorých je zmena hodnoty v dátach jedna z najradikálnejších. Rovnako je to s anomáliami, u ktorých nie sú hodnoty blízke ku globálnym extrémom vstupných dát. Dôkazom je najvýznamnejšia anomália prvého experimentu.

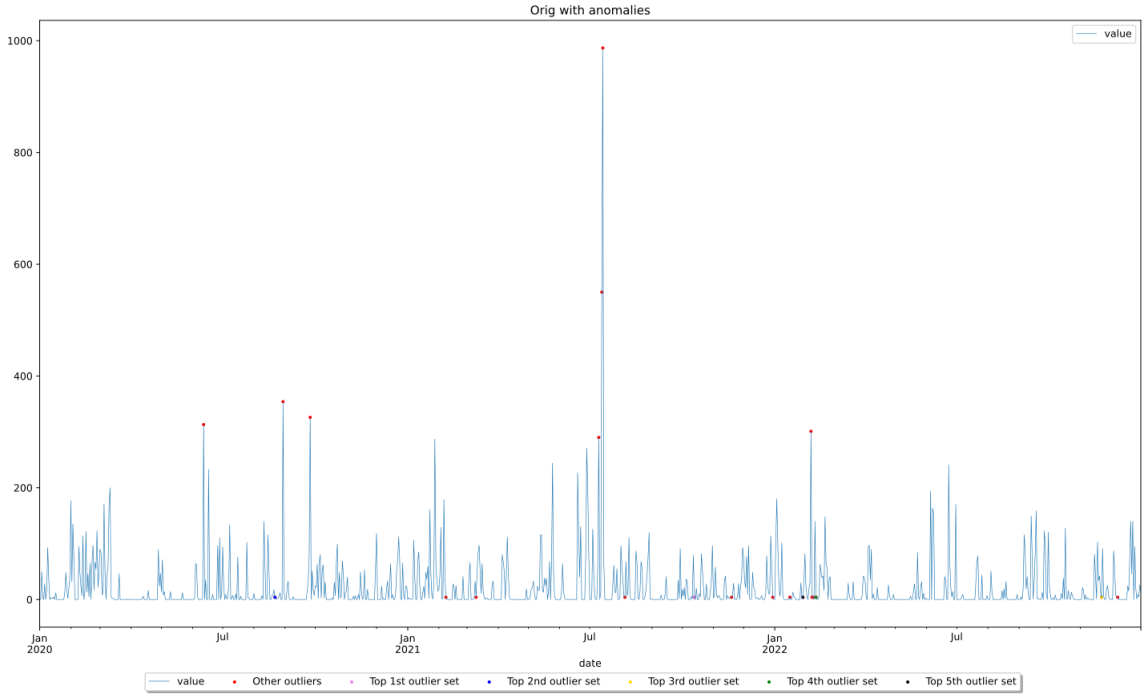
PyodLOF

Oveľa očakávanejšie výsledky na prvej dátovej sade dosiahol algoritmus *PyodLOF*. Pri pohľade na výsledný graf je možné vidieť, že najvýznamnejšie anomálie zodpovedajú globálne najvyšším a najnižším hodnotám dát zvyšného komponentu po dekompozícii. Najvýznamnejšou anomáliou bola identifikovaná odľahlá podsekvencia, skladajúca sa z troch hodnôt na prelome marca a apríla 2021. Táto podsekvencia obsahuje hodnotu globálneho maxima vstupného časového radu. Medzi najvýznamnejších päť anomálií sa dostala aj podsekvencia, ktorá pôsobí odľahlo aj v dátach pred dekompozíciou. Táto odľahlá podsekvencia, ktorá popisuje prudké ochladenie vo februári 2021, obsahuje osem hodnôt.



Obr. 6.10: Výsledok detekcie algoritmu *PyodLOF* na prvej dátovej sade

Na dátach s výraznými skokovitými nárastmi nad nulovou hodnotou, ktoré poskytuje druhá dátová sada, sú výsledky tohto algoritmu prekvapivé. Síce boli globálne najvyššie hodnoty dát zvolené za odľahlé, vôbec nepatria k najvýznamnejším detekovaným anomáliám. Dvoj-hodnotová odľahlá podsekvencia, ktorá zahŕňa hodnotu globálneho maxima, dosiahla v poradí všetkých osemnástich odľahlých množín až štrnástu priečku. Naopak za prvých päť najodľahlejších množín boli detekované odľahlé body s hodnotami blízkymi nule. Keďže sa však tieto body vyskytujú v okoliach dát, kde je zvýšená frekvencia nenulových nameraných zrážok, nemožno hovoriť o nesprávnom chovaní algoritmu. Každopádne ide o neštandardné chovanie detekcie v porovnaní s algoritmom *DeepLog*.

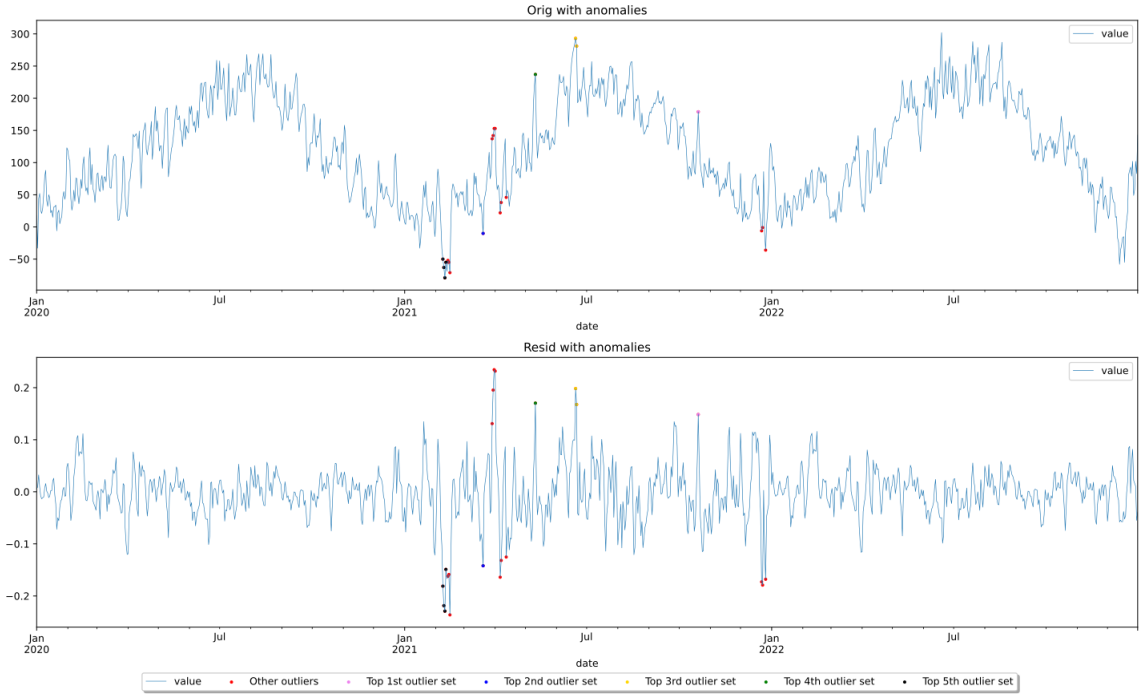


Obr. 6.11: Výsledok detekcie algoritmu *PyodLOF* na druhej dátovej sade

Analyzované výsledky algoritmu *PyodLOF* možno označiť za rôznorodé. U dát s plynulým i sezónnym vývojom hodnoty zodpovedali najvýznamnejšie anomálie hodnotám blízkym globálnemu minimu a maximu s najvyššími lokálnymi fluktuáciami. Naopak u dát so skokovitými nárastmi a poklesmi boli za najvýznamnejšie anomálie označené hodnoty blízke nulovým hodnotám s nižšími fluktuáciami.

PyodABOD

Algoritmus *PyodABOD* vyprodukoval na prvej dátovej sade podobné výsledky ako predchádzajúci algoritmus. Zmenou je len poradie jednotlivých najvýznamnejších anomálií. Odľahlá podsekvencia, ktorá zahŕňa globálne maximum dát, sa v tomto experimente ocitla až na šiestom mieste. Ako najvýznamnejšia anomália bol detekovaný odľahlý bod, ktorý symbolizuje prudké oteplenie v októbri 2021. Táto hodnota patrí medzi najvyššie hodnoty vstupných dát. Podobné správanie sa dá zachytiť pri najnižších globálnych hodnotách. Aj v nich je totiž ako významnejšia anomália považovaný odľahlý bod, ktorého hodnota nie je globálnym minimom dát.



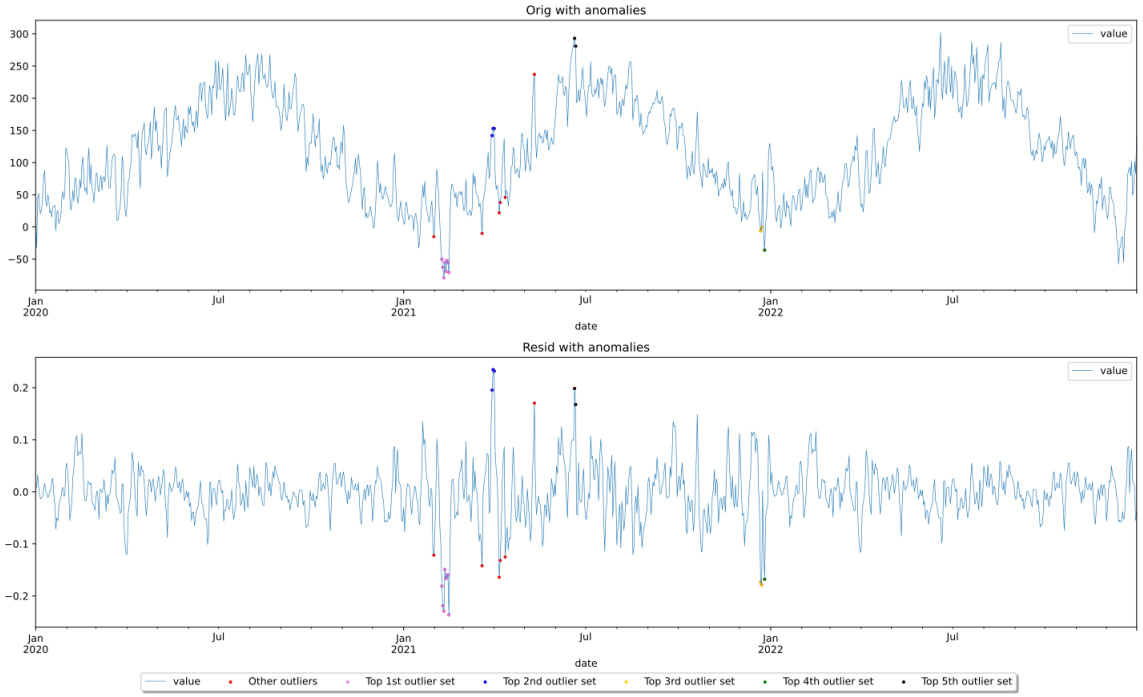
Obr. 6.12: Výsledok detekcie algoritmu *PyodABOD* na prvej dátovej sade

Zaujímavý je výsledok tohto algoritmu na druhej dátovej sade. V nej totiž *PyodABOD* nedetekoval žiadne anomálie. Vo výslednej tabuľke dátovej sady s označením odľahlosti možno vidieť, že veľkému množstvu hodnôt chýba pridelené skóre. To robí z tohto algoritmu nevhodného kandidáta pre detekciu anomálií v dátach s podobným charakterom.

Dosiahnuté výsledky algoritmu *PyodABOD* možno tiež označiť za rôznorodé. Výsledky na prvej dátovej sade sú podobné algoritmu *PyodLOF*, aj keď v tomto prípade neboli ako najvýznamnejšie anomálie detekované najextrémnejšie hodnoty vstupných dát. Rôznorodosť výsledkov tkvie v nedetekovaní anomálií na druhej dátovej sade, čo robí z algoritmu nevhodný nástroj pre detekciu anomálií v dátach so skokovitými nárastmi a poklesmi.

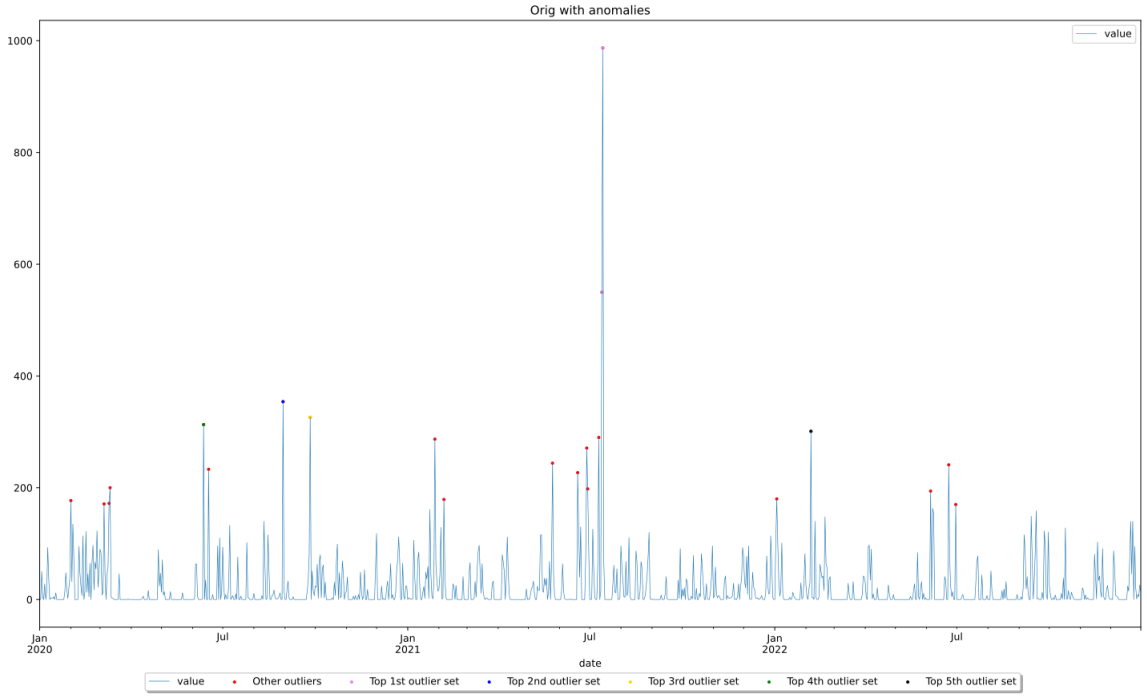
PyodAE

Algoritmus *PyodAE* je prvým algoritmom, ktorý na prvej dátovej sade detekoval ako dve najvýznamnejšie anomálie odľahlé množiny, obsahujúce obe globálne extrémy dát. Za najvýznamnejšiu odľahlú množinu bolo zvolené prudké ochladenie vo februári 2012. Ide o odľahlú podsekvenciu obsahujúcu osem hodnôt, ktorá pôsobí ako jedna z najodľahlejších aj v dátach pred dekompozíciou. Prudké oteplenie na prelome marca a apríla 2021 bolo zvolené za druhú najvýznamnejšiu anomáliu, ktorá zahŕňa globálne maximum vstupného časového radu. Ďalšie najvýznamnejšie anomálie zodpovedajú hodnotám, ktoré sa svojou veľkosťou blížia ku spomínaným globálnym extrémom dát.



Obr. 6.13: Výsledok detekcie algoritmu *PyodAE* na prvej dátovej sade

Podobné výsledky dosiahol tento algoritmus tiež v druhom experimente. V druhej dátovej sade algoritmus detekoval za anomálie najvyššie hodnoty. Navyše poradie najvýznamnejších anomálií zodpovedá poradiu hodnôt podľa ich veľkosti. Najvýznamnejšia odľahlá množina je ako jediná odľahlou podsekvenciou. Ostatnými anomáliami sú odľahlé body.

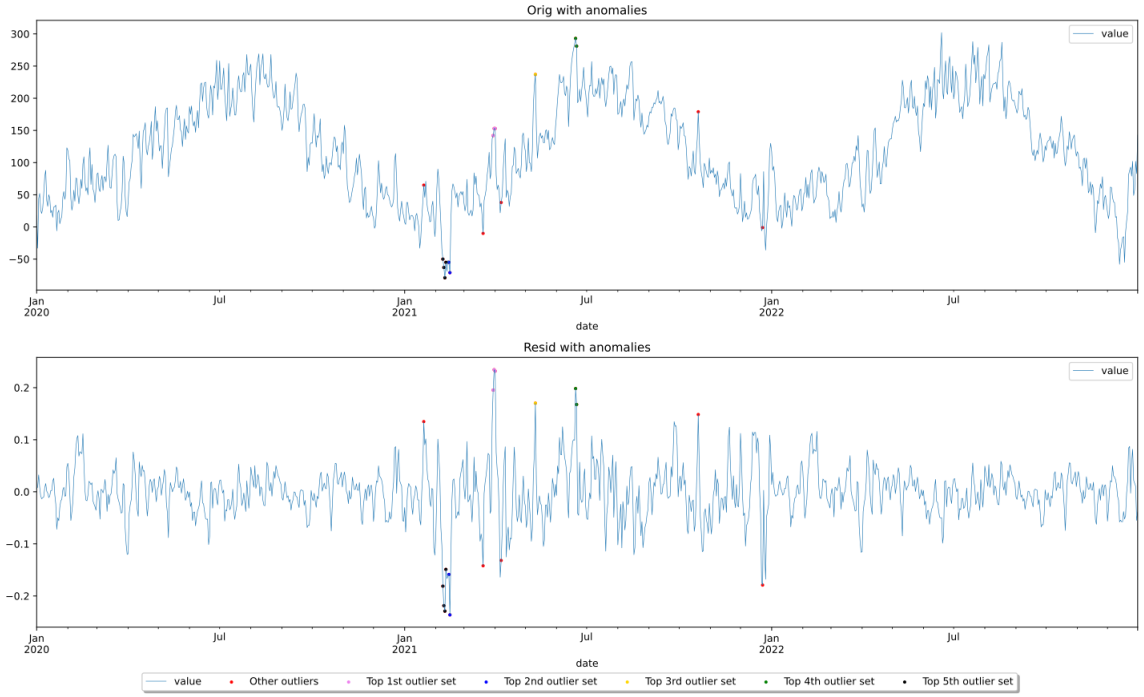


Obr. 6.14: Výsledok detekcie algoritmu *PyodAE* na druhej dátovej sade

Najdôležitejším záverom dosiahnutých výsledkov algoritmu *PyodAE* je jeho konzistentnosť výsledkov na dátach s rôznym charakterom. V oboch prípadoch detekoval ako najvýznamnejšie tie odľahlé množiny, ktoré sú rovné alebo blízke extrémom vstupných dát.

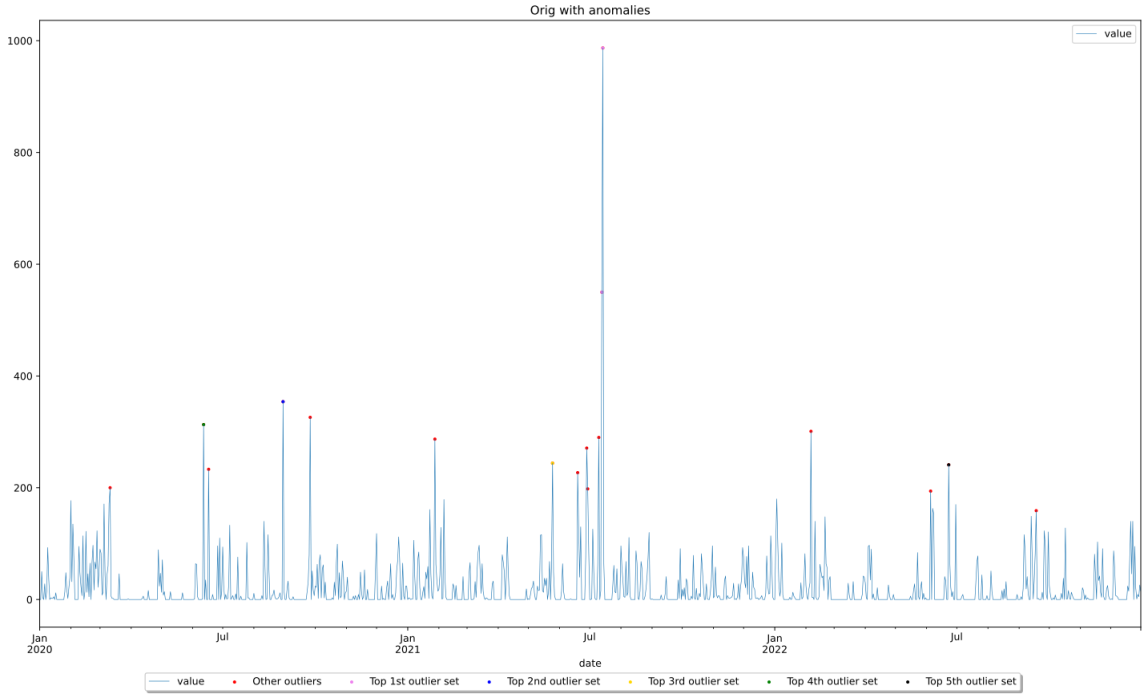
PyodKNN

V prvom experimente algoritmu *PyodKNN* sú detekované anomálie a ich poradie významnosti veľmi podobné výsledkom algoritmu *PyodLOF*. Spomínané poradie významností anomálií sa líši len minimálne. Drobnou zmenou v tomto prípade je, že prudké ochladenie vo februári 2021 nie je detekované ako jedna odľahlá podsekvencia, ale ako dve odľahlé podsekvencie. Týmto podsekvenciám patrí druhá a piata priečka z pohľadu významnosti. Na druhej strane najvýznamnejšou detekovanou anomáliou je prudké oteplenie na prelome marca a apríla 2021. O ostatných detekovaných anomáliách možno aj v tomto prípade tvrdiť, že sú blízke globálnym extrémom.



Obr. 6.15: Výsledok detekcie algoritmu *PyodKNN* na prvej dátovej sade

Tento algoritmus si poradil aj s dátami s výraznými skokovitými nárastmi a poklesmi. Prvé dve detekované anomálie zodpovedajú trom najvyšším hodnotám dát. Aj keď všetky ostatné detekované odľahlé hodnoty zodpovedajú najvyšším hodnotám vstupných dát, neplatí žeby poradie významnosti anomálií odzrkadľovalo poradie veľkostí týchto hodnôt.

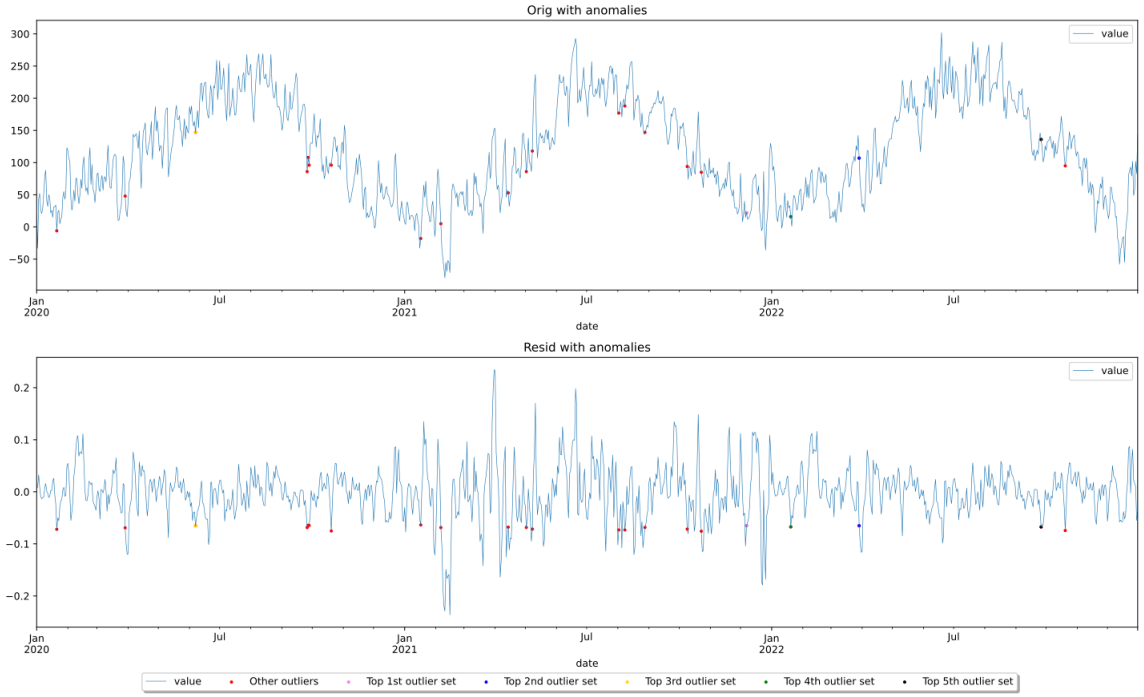


Obr. 6.16: Výsledok detekcie algoritmu *PyodKNN* na druhej dátovej sade

Z analýzy výsledkov oboch experimentov vyplýva záver, že algoritmus *PyodKNN* poskytuje konzistentnú detekciu anomálií na dátach s rôznym charakterom. Algoritmus detekuje ako odľahlé tie hodnoty, ktoré zodpovedajú alebo sú blízke globálnym extrémom. Napriek tomu poradie významností jednotlivých anomálií nezodpovedá poradiu veľkostí príslušných hodnôt.

PyodSoGaal

Algoritmus *PyodSoGaal* vyprodukoval veľmi neobvyklé detekčné výsledky. Za anomálie v prvom experimente zvolil len hodnoty, ktoré sa nachádzajú v okoliach lokálnych miním dát. To zodpovedá len záporným hodnotám v dátach po dekompozícii. Zároveň sa o veľkej časti detekovaných anomálií dá povedať, že v ich okoliach sa nachádzajú odľahlejšie hodnoty, ktoré ale za anomálne identifikované neboli. Na základe týchto výsledkov možno tvrdiť, že daný algoritmus nie je vhodný pre detekciu anomálií v dátach s podobným charakterom.



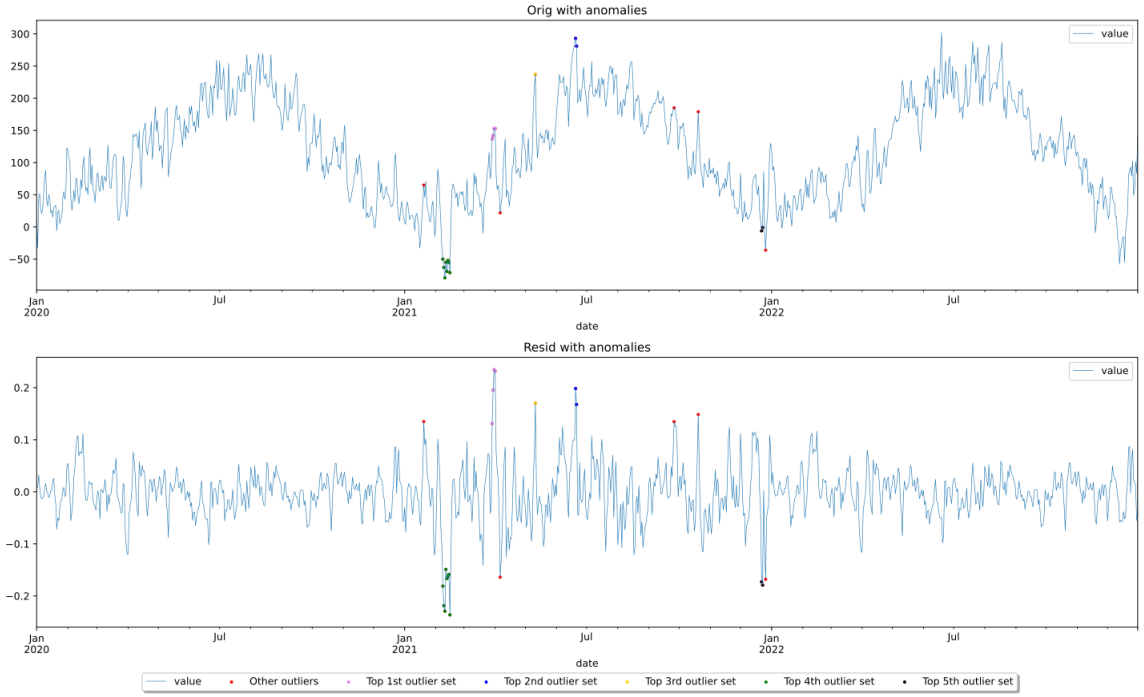
Obr. 6.17: Výsledok detekcie algoritmu *PyodSoGaal* na prvej dátovej sade

Lepšie výsledky nepriniesol ani druhý experiment vykonaný na tomto algoritme. V tomto prípade nedetekoval ani jednú odľahlú hodnotu. Podobný výsledok na druhej testovacej dátovej sade dosiahol tiež algoritmus *PyodABOD*. V porovnaní s ním sa však algoritmu *PyodSoGaal* nestalo, aby hodnoty dátovej sady vo výslednej tabuľke neobsahovali algoritmom priradené skóre. Na základe týchto výsledkov možno učiniť záver, že daný algoritmus nie je vhodný pre detekciu anomálií v dátach s takýmto charakterom.

Výsledky oboch experimentov algoritmu *PyodSoGaal* sa dajú označiť za neuspokojivé. Zatiaľ čo vo vstupných dátach s plynulým a sezónnym vývojom vrátil len záporné anomálie, v dátach s výraznými nárastmi a poklesmi nevrátil žiadnu odľahlú hodnotu. To robí z analyzovaného algoritmu nevhodným pre detekciu anomálií v jednorozmerných časových radoch.

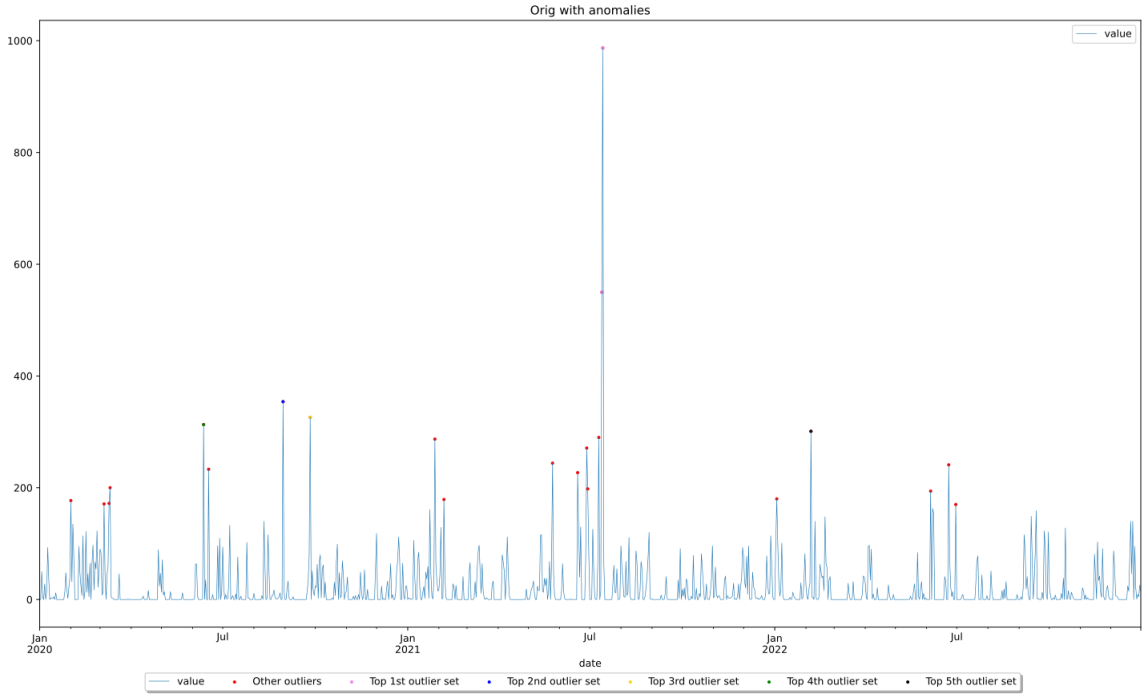
PyodIsolationForest

Výsledky algoritmu *PyodIsolationForest* sú aj v tomto prípade z pohľadu poradia významnosti veľmi podobné výsledkom algoritmu *PyodLOF*. Jedno z mála zmien je, že najvýznamnejšia odľahlá množina obsahuje o jednu hodnotu viac. Druhou zmenou je zvolenie piatej najvýznamnejšej anomálie, ktorou je v tomto prípade prudký pokles teploty v decembri 2021. Ostatné anomálie zodpovedajú hodnotám najbližším ku globálnym extrémom vstupných dát.



Obr. 6.18: Výsledok detekcie algoritmu *PyodIsolationForest* na prvej dátovej sade

V druhom experimente algoritmu sa nezopakovala podobnosť výsledkov s algoritmom *PyodLOF*. Poradie najvýznamnejších anomálií totiž zodpovedá poradiu príslušných hodnôt podľa ich veľkosti. Všetky ostatné odľahlé hodnoty podobne zodpovedajú najvyšším hodnotám vstupnej dátovej sady. Rozoberanými výsledkami na druhej dátovej sade sa algoritmus podobá algoritmu *PyodAE*.



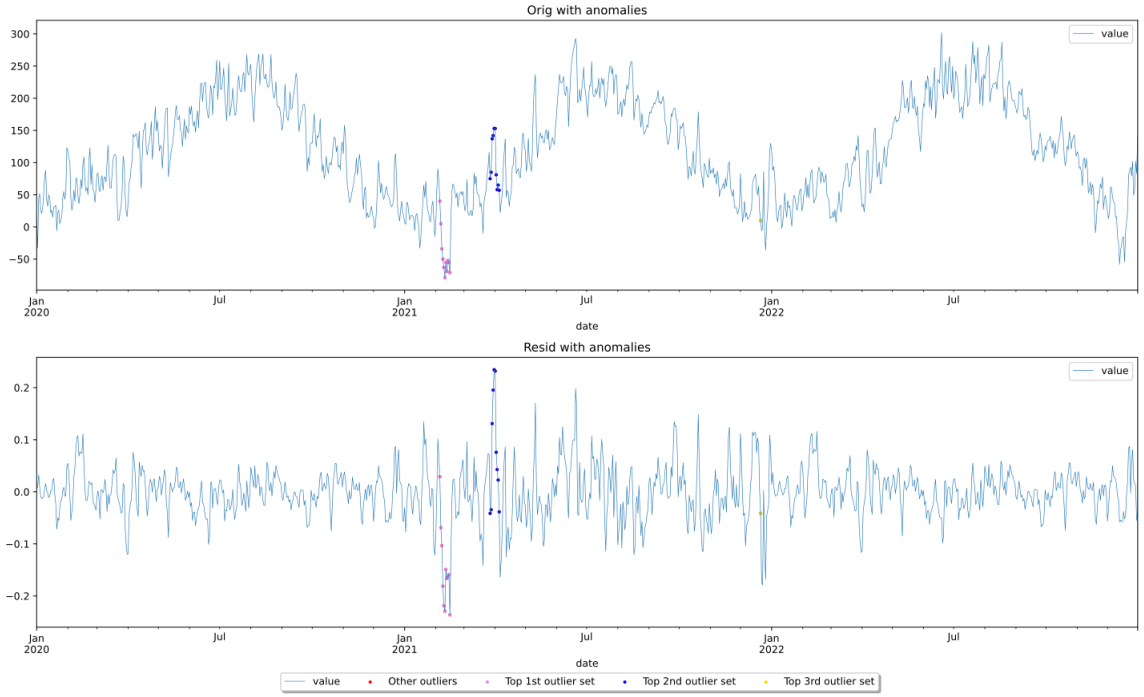
Obr. 6.19: Výsledok detekcie algoritmu *PyodIsolationForest* na druhej dátovej sade

Záverom vykonaných experimentov na algoritme *PyodIsolationForest* je konzistentnosť detekcie anomálií tohto algoritmu na dátach s rôznym charakterom. Algoritmus detekuje ako najvýznamnejšie anomálie hodnoty, ktoré zodpovedajú alebo sa blížia ku globálnym extrémom dát.

KDiscordODetecet

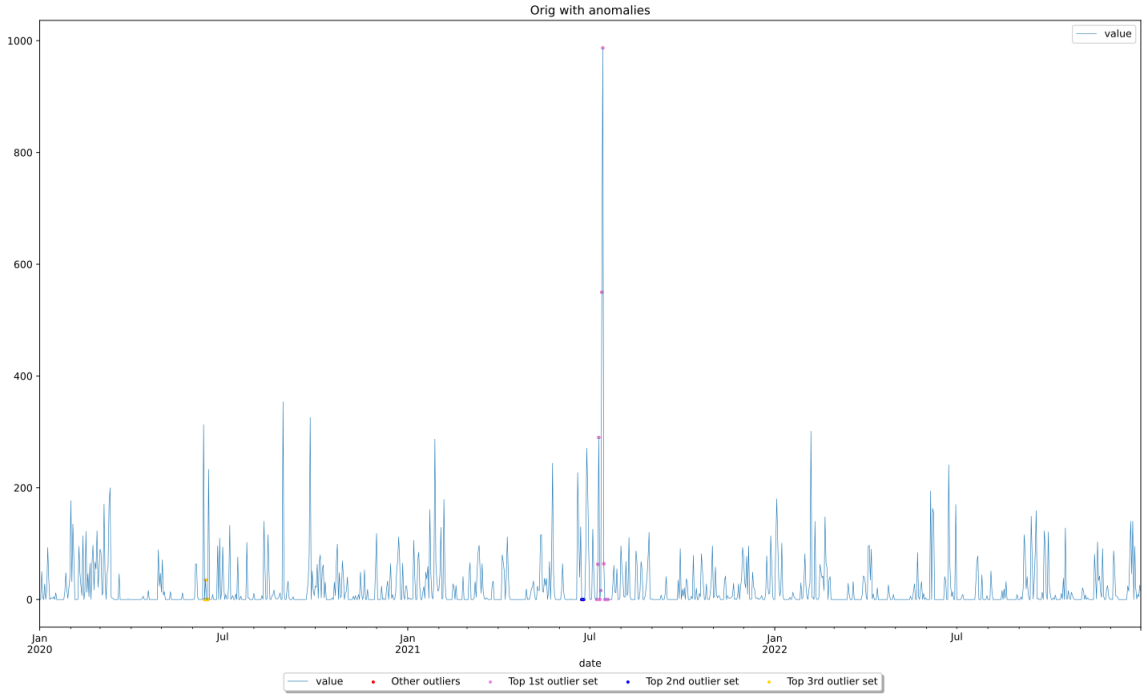
Algoritmom *KDiscordODetecet* sa táto experimentálna úloha dostáva k experimentom na algoritmoch, ktoré detekujú prevažne odľahlé podsekvencie.

V prvom experimente identifikoval algoritmus *KDiscordODetecet* tri odľahlé množiny, z ktorých dve sú odľahlé podsekvencie a jedna odľahlý bod. Najvýznamnejšou anomáliou je prudké ochladenie vo februári 2021. Ide o odľahlú podsekvenciu, skladajúcu sa z jedenástich hodnôt. Na druhej priečke z pohľadu významnosti bola detekovaná odľahlá podsekvencia desiatich hodnôt, ktorá popisuje markantné oteplenie na prelome marca a apríla 2021. Ako už bolo spomenuté, obe tieto anomálie obsahujú oba globálne extrémy vstupných dát. Treťou detekovanou anomáliou je odľahlý bod, ktorý síce odľahlo nepôsobí, no je súčasťou prudkej zmeny teploty. Dá sa predpokladať, že zvýšením hodnoty konfiguračného parametra *contamination*, by tento odľahlý bod s jeho okolitými hodnotami tvoril ďalšiu odľahlú podsekvenciu.



Obr. 6.20: Výsledok detekcie algoritmu *KDiscordODetecet* na prvej dátovej sade

V druhom experimente tento algoritmus identifikoval len tri odľahlé podsekvencie. Anomáliou s najvyšším skóre je dvanásť-bodová podsekvencia, ktorá zahŕňa globálne maximum. Z tohto dôvodu nejde o prekvapivý výsledok. Zaujímavé sú však zvyšné dve odľahlé podsekvencie. Obe sa totiž skladajú z hodnôt, ktoré sa rovnajú alebo sú blízke nule. Napriek tomu ide o podsekvencie, o ktorých možno tvrdiť, že sú skutočne anomálne. Dôvodom je, že v okoliach týchto podsekvencií je zvýšená frekvencia vyšších nenulových hodnôt.

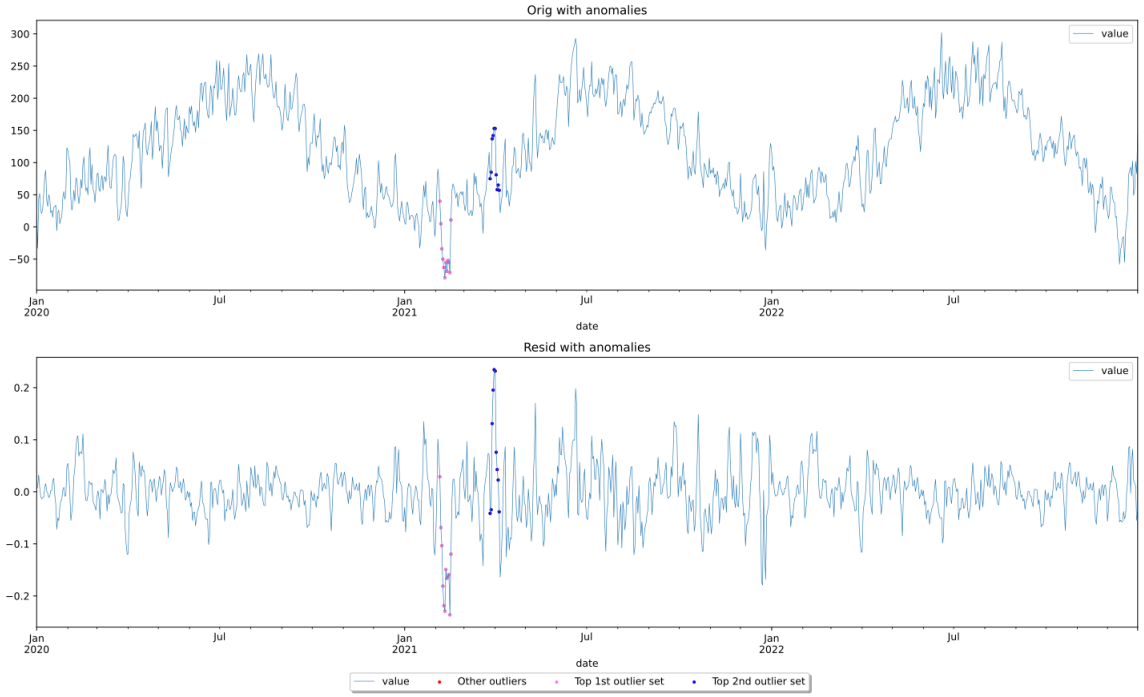


Obr. 6.21: Výsledok detekcie algoritmu *KDiscordODetecet* na druhej dátovej sade

Algoritmus *KDiscordODetecet* možno na základe výsledkov označiť za spoľahlivý a konzistentný detektor odľahlých podsekvencií. To platí o jeho použití v dátach s rôznym charakterom.

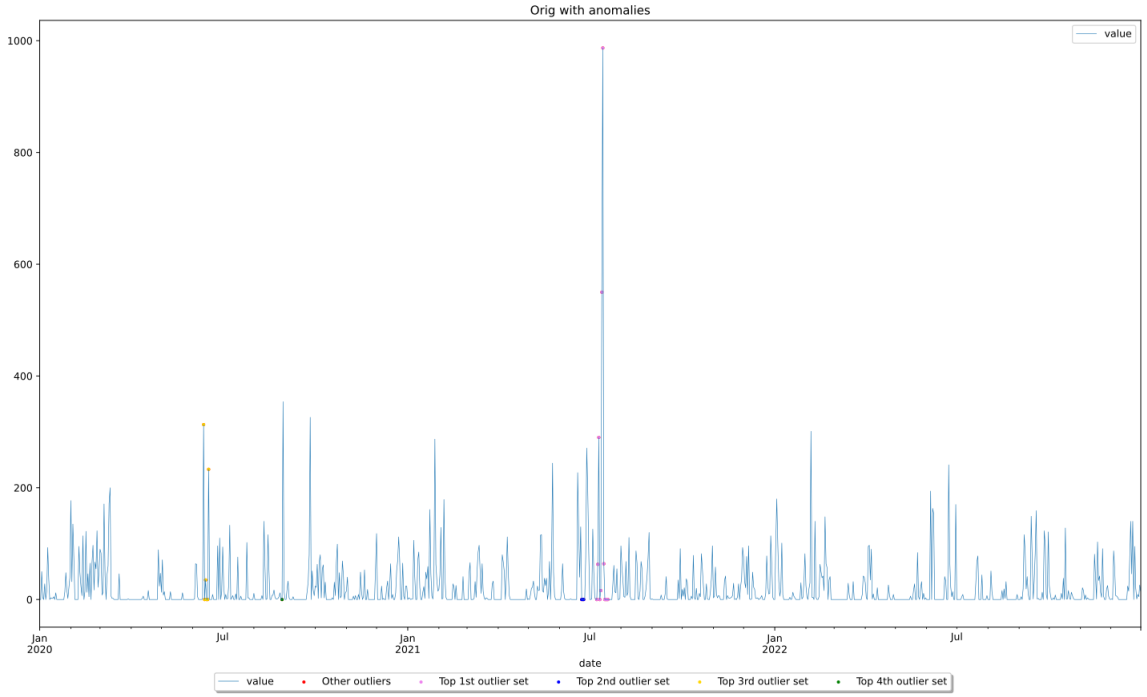
PCAODetect

Algoritmus *PCAODetect* vyprodukoval v experimente na prvej dátovej sade takmer totožné výsledky ako predchádzajúci algorimtus. Jedinou zmenou je absencia tretej najvýznamnejšej odľahlej množiny. Tento algoritmus detekoval dve odľahlé podsekvencie, ktorých poradie podľa významnosti je totožné práve s predchádzajúcim algoritmom. Zmenou v týchto anomáliách je, že najvýznamnejšia odľahlá podsekvencia obsahuje o jednu odľahlú hodnotu viac.



Obr. 6.22: Výsledok detekcie algoritmu *PCAODetect* na prvej dátovej sade

Takmer totožné výsledky v porovnaní s algoritmom *KDiscordODetecet* dosiahol tento algoritmus tiež vo svojom druhom experimente. Jednou malou zmenou je, že tretia najvýznamnejšia odľahlá množina obsahuje o dve hodnoty viac. Ďalšou zmenou je detekcia aj štvrtej anomálie. Síce ide o odľahlý bod, charakterom je podobný druhej i tretej najvýznamnejšej anomálii. Charakterom sa v tomto prípade myslí veľkosť danej hodnoty a okolitých hodnôt.

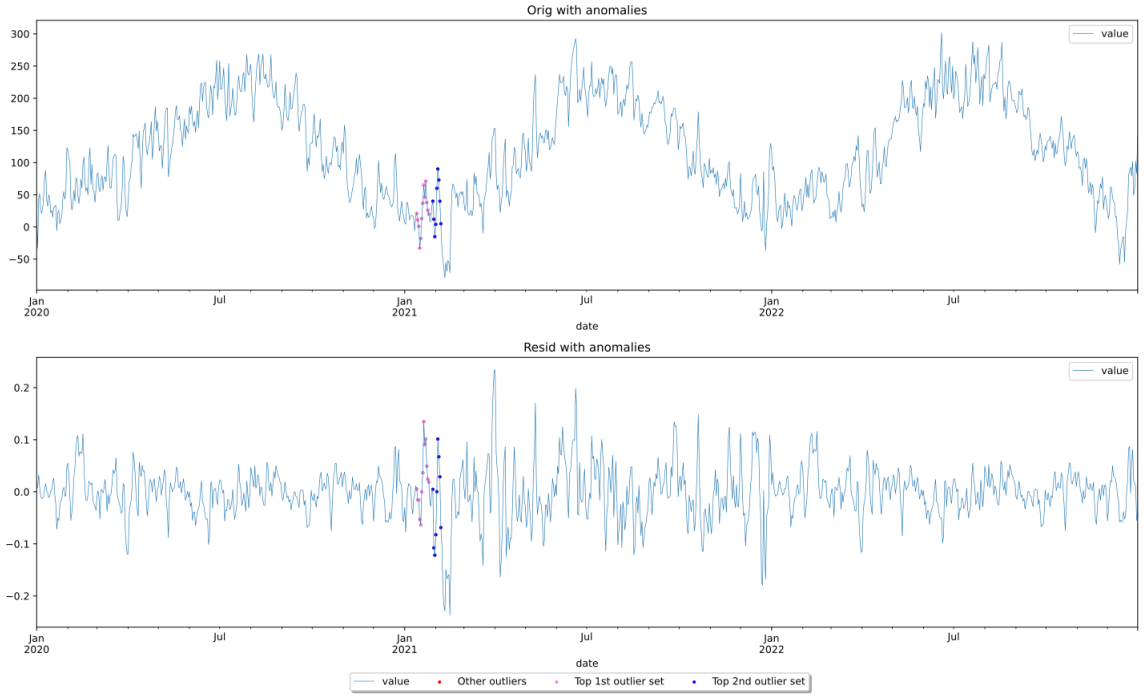


Obr. 6.23: Výsledok detekcie algoritmu *PCAODetect* na druhej dátovej sade

Rovnako ako v prípade algoritmu *KDiscordODetecet* je možné u algoritmu *PCAODetect* učiniť záver, že dokáže spoľahlivo detekovať odľahlé podsekvencie v dátach s rôznym charakterom.

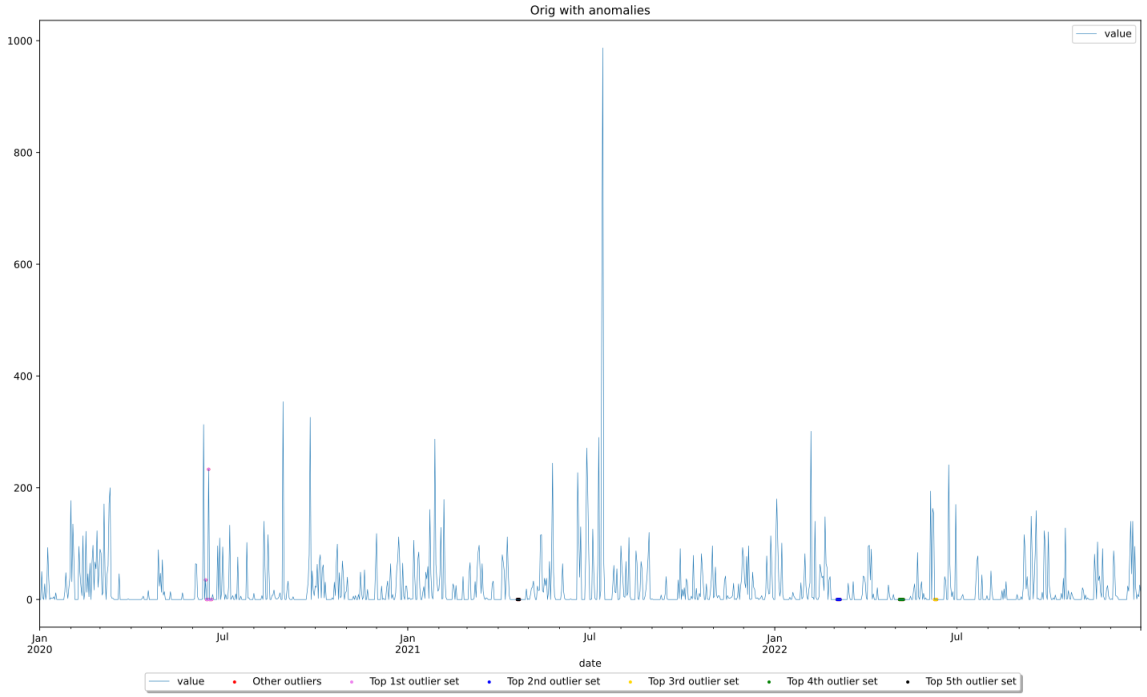
LSTMODetect

Prvý experiment algoritmu *LSTMODetect* priniesol neočakávané výsledky. Algoritmus totiž detekoval ako odľahlé dve podsekvencie hodnôt, ktoré pri pohľade na graf vstupných dát nepôsobia príliš odľahlo. Nie je možné hovoriť ani o nejakých nezvyčajných vzoroch ktoré by sa na iných miestach v dátach nevyskytovali. Obe detekované anomálie majú spoločné, že sa skladajú z poklesu hodnoty a jej bezprostredného nárastu. Na základe týchto poznatkov sú výsledky tohto experimentu označené za nedostatočné.



Obr. 6.24: Výsledok detekcie algoritmu *LSTMODetect* na prvej dátovej sade

Neočakávané výsledky priniesol aj experiment vykonaný na druhej dátovej sade. V nej sa algoritmu *LSTMODetect* podarilo detekovať odľahlú podsekvenciu, ktorú identifikovali aj predchádzajúce dva algoritmy. Napriek tomu sú ostatné detekované anomálie tvorené hodnotami rovné nule. O týchto anomáliach sa nedá úplne povedať, že by ich blízke okolie, s výnimkou tretej najvýznamnejšej anomálie, boli hodnoty so zvýšenou frekvenciou nenulových hodnôt. Zároveň za anomáliu v tomto experimente nebol identifikovaný prudký nárast zrážok z júla 2021, ktorý sa dá považovať za jednoznačnú anomáliu vstupných dát. Na základe analýzy tohto experimentu možno považovať tento algoritmus za nedostatočný detektor anomálií v časových radoch s podobným charakterom.



Obr. 6.25: Výsledok detekcie algoritmu *LSTMODetect* na druhej dátovej sade

S ohľadom na analýzu výsledkov oboch experimentov možno dodať, že algoritmus *LSTMODetect* nie je vhodný pre detekciu odľahlých hodnôt v dátach s podobným charakterom, ako je charakter testovacích dátových sád.

6.2.4 Záver

Vybrané algoritmy boli v tejto experimentálnej úlohe otestované a porovnané na dvoch dátových sadách s dátami rôzneho charakteru.

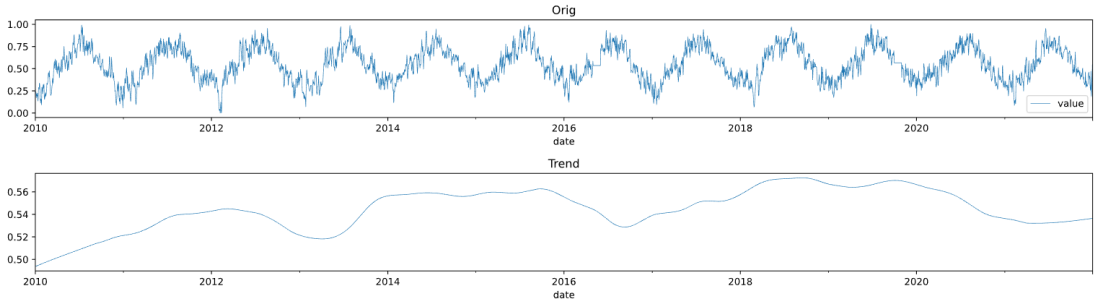
Analýza výsledkov jednotlivých experimentov preukázala, že veľká väčšina algoritmov zvládne detekovať anomálie rôznych typov na oboch testovaných dátových sadách. V tomto prípade sa výsledky algoritmov na totožných dátach líšili hlavne vo významnosti jednotlivých anomálií, teda ich poradia na základe dosiahnutého skóre v príslušných experimentoch. Ďalším aspektom, v ktorom sa totožné anomálie v závislosti od algoritmu líšili, bol počet hodnôt z ktorých sa dané anomálie skladali. Ani v tomto prípade však nešlo o veľké rozdiely.

Medzi testovanými algoritmami sa však našli aj také, ktorých výsledky boli označené za nedostatočné. Ide najmä o algoritmy *PyodSoGaal* a *LSTMODetest*, ktorých výsledky boli nedostatočné na oboch dátových sadách. Dané algoritmy teda nie sú odporúčané pre detekciu odľahlých hodnôt v jednorozmerných časových radoch. Poslednú kategóriu algoritmov tvorí algoritmus *PyodABOD*. Ten priniesol neuspokojivé výsledky len na dátach s výraznými skokovitými nárastmi a poklesmi. Na druhej strane tento algoritmus možno odporučiť na detekciu anomálií v dátach so sezónnym a plynulým vývojom hodnoty.

6.3 Praktické využitie experimentov

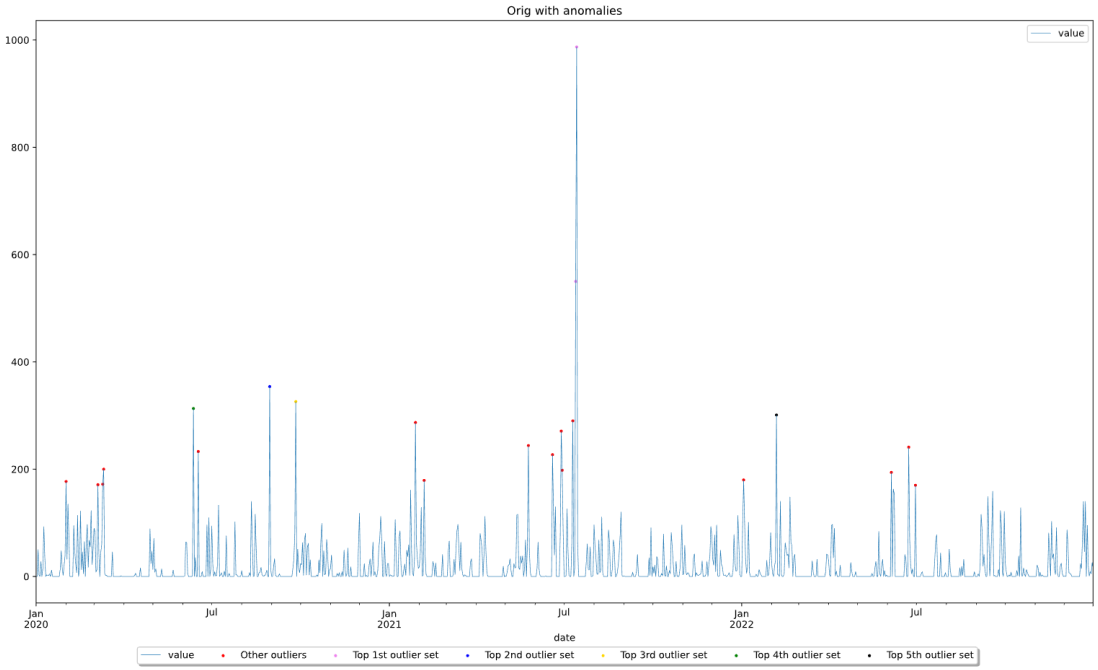
V nasledujúcom texte je poskytnutých niekoľko príkladov praktického využitia experimentov detekcie anomálií v časových radoch s použitím implementovaného webového nástroja.

Prvým príkladom je využitie výsledku dekompozície časového radu. Na obrázku 6.26 možno v druhom grafe vidieť trend vývoja priemernej teploty vzduchu v meste Praha. Na danom grafe je možné vidieť, že najvyššiu hodnotu trendu si počas celého roka udržal rok 2018. To znamená že rok 2018 bol najteplejším z uvedených rokov. Túto informáciu potvrdzuje hneď niekoľko zdrojov. [5]



Obr. 6.26: Výsledok dekompozície časového radu priemernej teploty v meste Praha

Druhý uvedený príklad je založený na samotnej detekcii odľahlých hodnôt. Na obrázku 6.27 je uvedený výsledný graf z detekcie anomálii vykonanej algoritmom *PyodIsolationForest*. Anomálie sú detekované na dátach denných dopadov zrážok v nemeckom meste Aachen. V grafe možno vidieť, že v júli 2021 je detekovaná najvýznamnejšia anomália. Ide o odľahlú podsekvenciu dvoch hodnôt. Tieto hodnoty zodpovedajú dňom, kedy v rôznych častiach Nemecka vyčíňal ničivý orkán, počas ktorého boli prítomné silné búrky. [6]



Obr. 6.27: Výsledok detekcie anomálií dopadnutých denných zrážok v Aachene

Kapitola 7

Záver

V prvej časti tejto diplomovej práce bola predstavená základná teória k odboru získavania znalostí z databáz. V rámci nej bola uvedená definícia tohto odboru, definícia jeho procesu, či stručné vysvetlenie jeho najrozšírenejších typov úloh.

Následne bola vysvetlená teória odboru získavania znalostí z databáz so špecializáciou na riešený problém, ktorým je detekcia anomálií v temporálnych dátach. Súčasťou tejto sekcie je vysvetlenie základných pojmov ale aj vlastnosti techník pre detekciu anomálií v časových radoch.

Obsahom ďalšej kapitoly práce boli popísané vybrané dátové sady, ktoré sú neskôr použité v experimentoch práce. Súčasťou tohto obsahu bol tiež popis dostupných technológií, vhodných pre riešenie daného problému. V závere bola vybraná technológia a popis jej detekčných algoritmov, ktoré sú neskôr implementované a využité v experimentálnej časti práce.

Jedným z výsledkov tejto práce je implementovaná webová aplikácia pre detekciu anomálií v jednorozmerných časových radoch. V ďalšej kapitole je teda stručne popísaný jej návrh a implementácia. Záver kapitoly patrí predstaveniu finálnej funkčnosti implementovaného nástroja.

V poslednej časti textu sú popísané najdôležitejšie výsledky tejto práce. Ide o experimentálnu časť práce, ktorá porovnáva jednotlivé implementované algoritmy. V rámci tejto experimentálnej časti boli vykonané dve experimentálne úlohy. Prvá experimentálna úloha porovnáva detekčné algoritmy na základe typov anomálií, ktoré dokážu detekovať. Na druhej strane druhá experimentálna úloha porovnáva algoritmy na základe najvýznamnejších detekovaných anomálií. Záver poslednej časti práce patrí príkladom praktického využitia detekcie anomálií v časových radoch pomocou implementovaného nástroja.

Ďalším pokračovaním tejto práce by mohlo byť rozšírenie implementovanej aplikácie pre detekciu anomálií vo viacrozmerných časových radoch. Keďže detekčné algoritmy vybraného nástroja dokážu podľa dokumentácie identifikovať anomálie aj v spomínaných viacrozmerných časových radoch, ide o vhodný námet pre ďalšie porovnanie detekčných algoritmov.

Literatúra

[1] *Introduction — Luminaire* [online]. [cit. 2023-01-17]. Dostupné z: https://zillow.github.io/luminaire/.

[2] *Kats / Kats* [online]. [cit. 2023-01-17]. Dostupné z: https://facebookresearch.github.io/Kats/.

[3] *Orion — Orion 0.4.1.dev0 documentation* [online]. [cit. 2023-01-17]. Dostupné z: https://sintel.dev/Orion/.

[4] *Welcome to TODS's documentation! — TODS 0.0.1 documentation* [online]. [cit. 2023-01-17]. Dostupné z: https://tods-doc.github.io/.

[5] *Uplynulý rok 2018 byl nejteplejší od roku 1961, srážek však celkově ubylo* [online], 2. jan 2019. Dostupné na: https://www.idnes.cz/zpravy/domaci/rok-2018-nejteplejsi-od-1961.A190102_163605_domaci_zaz.

[6] *Orkán na severu Německa si vyžádal tři oběti. Další tři lidé zahynuli v Polsku* [online], 17. feb 2022. Dostupné na: https://www.denik.cz/staty-eu/nemecko-vitr-zeleznice-20220217.html.

[7] Aggarwal, C. C. *Linear Models for Outlier Detection*. Cham: Springer International Publishing, 2017. 65–110 s. ISBN 978-3-319-47578-3. Dostupné z: https://doi.org/10.1007/978-3-319-47578-3_3.

[8] Angiulli, F. a Pizzuti, C. Fast Outlier Detection in High Dimensional Spaces. In: Elomaa, T., Mannila, H. a Toivonen, H., ed. *Principles of Data Mining and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, s. 15–27. ISBN 978-3-540-45681-0.

[9] Bartík, V. *1. Úvod, základní pojmy databázových technologií* [Fakultná prednáška predmetu ZZN]. FIT VUT Brno.

[10] Blázquez García, A., Conde, A., Mori, U. a Lozano, J. A. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. apr 2021, zv. 54, č. 3. DOI: 10.1145/3444690. ISSN 0360-0300.

[11] Breunig, M. M., Kriegel, H.-P., Ng, R. T. a Sander, J. LOF: identifying density-based local outliers. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2000, s. 93–104. SIGMOD '00. DOI: 10.1145/342009.335388. ISBN 1581132174. Dostupné z: https://doi.org/10.1145/342009.335388.

[12] Du, M., Li, F., Zheng, G. a Srikumar, V. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2017, s. 1285–1298. CCS '17. DOI: 10.1145/3133956.3134015. ISBN 9781450349468. Dostupné z: https://doi.org/10.1145/3133956.3134015.

[13] Dunham, M. *Data Mining: Introductory And Advanced Topics*. Pearson Education, 2006. ISBN 9788177587852.

[14] Graves, A. *Long Short-Term Memory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. 37–45 s. ISBN 978-3-642-24797-2. Dostupné z: https://doi.org/10.1007/978-3-642-24797-2_4.

[15] Gullo, F. From Patterns in Data to Knowledge Discovery: What Data Mining Can Do. *Physics Procedia*. 2015, zv. 62, s. 18–22. DOI: https://doi.org/10.1016/j.phpro.2015.02.005. ISSN 1875-3892. 3rd International Conference Frontiers in Diagnostic Technologies, ICFDT3 2013, 25-27 November 2013, Laboratori Nazionali di Frascati, Italy.

[16] HAN, J., KAMBER, M. a PEI, J. *Data mining: concepts and techniques*. 3rd ed. Morgan Kaufmann, 2012. ISBN 978-0-12-381479-1.

[17] Hyndman, R. J. a Athanasopoulos, G. *Forecasting: Principles and Practice*. 2021. Dostupné z: https://otexts.com/fpp3/.

[18] Kriegel, H.-P., Schubert, M. a Zimek, A. Angle-based outlier detection in high-dimensional data. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2008, s. 444–452. KDD '08. DOI: 10.1145/1401890.1401946. ISBN 9781605581934. Dostupné z: https://doi.org/10.1145/1401890.1401946.

[19] Liu, F. T., Ting, K. M. a Zhou, Z.-H. Isolation Forest. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, s. 413–422. DOI: 10.1109/ICDM.2008.17.

[20] Liu, Y., Li, X., Ma, S., Song, Y. a Cui, L. Generative Adversarial Active Learning for Unsupervised Outlier Detection. *ArXiv preprint arXiv:1910.07169*. 2019. Dostupné z: https://arxiv.org/abs/1910.07169.

[21] Mamoulis, N. *Temporal Data Mining*. Boston, MA: Springer US, 2009. 2948–2952 s. ISBN 978-0-387-39940-9. Dostupné z: https://doi.org/10.1007/978-0-387-39940-9_393.

[22] Medico, R. Rob-med/awesome-TS-anomaly-detection. Zenodo. Aug 2020. DOI: 10.5281/zenodo.3972944. Dostupné z: https://github.com/rob-med/awesome-TS-anomaly-detection?tab=readme-ov-file.

[23] Rai, P. a Shubha, S. A Survey of Clustering Techniques. *International Journal of Computer Applications*. Október 2010, zv. 7. DOI: 10.5120/1326-1808.

[24] Vaisman, A. a Zimányi, E. *Data warehouse systems: design and implementation.* 2014. ISBN 978-3-642-54654-9.

[25] Witten, I. H., Frank, E. a Hall, M. A. Chapter 7 - Data Transformations. In: Witten, I. H., Frank, E. a Hall, M. A., ed. *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition).* Third Edition. Boston: Morgan Kaufmann, 2011, s. 305–349. The Morgan Kaufmann Series in Data Management Systems. DOI: https://doi.org/10.1016/B978-0-12-374856-0.00007-9. ISBN 978-0-12-374856-0.

[26] Wu, H.-S. A survey of research on anomaly detection for time series. In: *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP).* 2016, s. 426–431. DOI: 10.1109/ICCWAMTIP.2016.8079887.

[27] Zendulka, J., Bartík, V., Lukáš, R. a Rudolfová, I. *Získávání znalostí z databází (ZZN): Studijní opora* [Fakultné skripty predmetu ZZN]. FIT VUT Brno.

Príloha A

Obsah priloženého pamäťového média

V rámci odovzdania bolo k tejto práci priložené aj pamäťové médium, ktorého obsah je nasledovný:

- `dip-xondri05` – koreňový adresár,
  - `aplikacia` – adresár zdrojové súbory implementovanej aplikácie spolu s použitými frameworkmi a knižnicami (obsah popísaný v prílohe B),
  - `experimenty` – adresár obsahujúci vykonané experimenty,
    - `experiment-task-1` – adresár pre experimenty prvej experimentálnej úlohy,
      - `1_contamination-0.03` – adresár pre experimenty vykonané s nastavením konfiguračného parametra *contamintaion* na hodnote 0.03,
      - `2_contamination-0.10` – adresár pre experimenty vykonané s nastavením konfiguračného parametra *contamintaion* na hodnote 0.10,
    - `experiment-task-2` – adresár pre experimenty druhej experimentálnej úlohy,
      - `1_prague-meanTemperature` – adresár pre experimenty vykonané na prvej dátovej sade,
      - `2_aachen-precipitation` – adresár pre experimenty vykonané na druhej dátovej sade,
  - `technicka-sprava-zdroj` – adresár obsahujúci LaTeX zdrojové texty technickej správy (prekladané v nástroji *Overleaf*),
  - `README.md` – dokumentácia,
  - `technicka-sprava.pdf` – text technickej správy vo formáte *PDF*.

Príloha B

Súborová štruktúra aplikácie

V tejto prílohe je vypísaná súborová štruktúra implementácie webovej aplikácie, ktorá sa nachádza v adresári `aplikacia`. Uvedené sú len dôležité adresáre a súbory:

```
.
|-- app-build-run.sh  ->  Skript pre inšt. a spust. aplikácie v Linuxe.
|-- db.sql
|-- docker-compose.yml
|-- Dockerfile-db
|-- Dockerfile-webapp
|-- libs
|    |-- tods  ->  Adresár Python balíčka TODS s jeho implementáciou.
|-- web-app  ->  Koreňový adresár frameworku CodeIgniter.
     |-- application
     |    |-- config  ->  Konfiguračné súbory frameworku CodeIgniter.
     |    |-- controllers  ->  Implementované kontroléri webovej aplikácie.
     |    |    |-- Experiments.php
     |    |    |-- Pages.php
     |    |    |-- Upload.php
     |    |-- models  ->  Implementované modely webovej aplikácie.
     |    |    |-- ExperimentsMod.php
     |    |    |-- UploadMod.php
     |    |-- views  ->  Implementované pohľady (views) webovej aplikácie.
     |         |-- about.php
     |         |-- experimentDetail.php
     |         |-- experimentDetailActions.php
     |         |-- experiments.php
     |         |-- experimentsTableContent.php
     |         |-- experimentsTableHeader.php
     |         |-- fileTableContent.php
     |         |-- fileTableHeader.php
     |         |-- footer.php
     |         |-- header.php
     |         |-- home.php
     |         |-- image.php
```

```
|         |-- prepare2.php
|         |-- prepare.php
|         |-- sectionEnd.php
|         |-- sectionStart.php
|         |-- uploadDetail.php
|         |-- uploadDetailTable.php
|         |-- upload.php
|-- css  ->  Štýlovacie súbory CSS pre pohľady.
|    |-- bootstrap.min.css
|    |-- custom.css
|-- results  ->  Adresár pre výstupné súbory experimentov.
|-- scripts  ->  Implementované skripty jazyka Python.
|    |-- detection_app.py
|-- system  ->  Systémový adresár frameworku CodeIgniter.
|-- uploads  ->  Adresár pre vstupné dátové súbory používateľa.
```

Príloha C

Súborová štruktúra experimentu

V tejto prílohe je popísaná súborová štruktúra jedného vykonaného experimentu. Každý experiment je definovaný adresárom, ktorý pozostáva z času vykonania, názvu súboru dátovej sady a názvu detekčného algoritmu, ktorý bol použitý. Príkladom názvu takého adresára je `2024-05-05__12-58-22__aachen-meanTemperature.csv__PyodABOD`. V nasledujúcom texte je zoznam názvov súborov, ktoré môžu byť obsiahnuté v takomto experimentálnom adresári. V zátvorkách sú uvedené tiež názvy, zodpovedajúce názvom vo webovej aplikácii. V prípade, že zátvorka s takýmto názvom absentuje, príslušný súbor sa vo webovej aplikácii nezobrazuje.

Experimentálne súbory sú:

- `0_info.txt` – vstupné údaje experimentu od používateľa,
- `1_loaded_data.csv` – tabuľka načítaných dát dátovej sady,
- `1_loaded_data.svg` – graf načítaných dát dátovej sady,
- `2_cleaned_data.csv` – tabuľka dát po čistení,
- `2_cleaned_data.svg` – graf dát po čistení,
- `3_filtered_data.csv` (*Cleaned and filtered data – Table*) – tabuľka dát po filtrácii,
- `3_filtered_data.svg` (*Cleaned and filtered data – Chart*) – graf dát po filtrácii,
- `4_decomp_data.csv` (*Seasonal/Trend decomposition – Table*) – tabuľka dát zvyšku po dekompozícii,
- `4_decomp_data.svg` (*Seasonal/Trend decomposition – Chart*) – súbor grafov komponentov dekompozície,
- `5_algorithm_params.txt` – vstupné konfiguračné parametre detekčného algoritmu,
- `6_result_table.csv` (*Detection results – Table*) – výsledná tabuľka dát dátovej sady s určením odľahlosti pre každú hodnotu,
- `7_result_chart.svg` (*Detection results – Chart*) – výsledný graf dát dátovej sady s vyznačenými odľahlými hodnotami,

- `8_result_chart_wide.svg` (*Detection results – Chart wide*) – výsledný graf dát dátovej sady s vyznačenými odľahlými hodnotami v širšom grafickom formáte,
- `9_result_table_outlier_values.csv` (*Detection results – Outlier values table*) – výsledná tabuľka so zoznamom odľahlých hodnôt, ktorým sú priradené aj odľahlé množiny, do ktorých boli zaradené,
- `10_result_table_top_outlier_sets.csv` (*Detection results – Top outlier sets table*) – tabuľka poradia odľahlých množín podľa významnosti resp. dosiahnutého skóre,
- `11_result_table_top_outlier_points.csv` (*Detection results – Top outlier points table*) – tabuľka poradia odľahlých bodov podľa významnosti resp. dosiahnutého skóre,
- `12_result_table_top_outlier_subsequences.csv` (*Detection results – Top outlier subsequences table*) – tabuľka poradia odľahlých podsekvencií podľa významnosti resp. dosiahnutého skóre,
- `13_statistics.csv` (*Detection results – Statistic*) – vysledná tabuľka rôznych štatistík experimentu.

Príloha D

Návod pre spustenie aplikácie

Pre spustenie aplikácie je potrebné mať na pracovnom stroji nainštalovaný nástroj *Docker Compose*. Ak je tento nástroj nainštalovaný spustenie aplikácie je jednoduché. V termináli operačného systému je potrebné na ceste adresára `aplikacia`, ktorý je súčasťou odovzdaných súborov práce, spustiť príkaz: `docker-compose up -d --build`. Následne by malo byť možné aplikáciu otvoriť v internetovom prehliadači prostredníctvom odkazu `http://localhost:28000/xondri05-dip-app/`

Týmto by mali byť na pozadí operačného systému spustené dva virtuálne kontajnery (kontajner pre webovú aplikáciu a kontajner pre databázový systém) na aplikačných portoch `28000` a `28001` hostiteľského stroja. V prípade, že práve tieto čísla portov nevyhovujú, je ich možné zmeniť v súbore `aplikacia/docker-compose.yml` v sekciách s názvom `ports`. Kontajner webovej aplikácie je zároveň nastavený tak, že je vytvorené prepojenie medzi obsahom adresára `aplikacia/web-app/results` a zodpovedajúcimi súbormi vo virtuálnom kontajneri. Vďaka tomu je možné na hostiteľskom stroji v adresári `aplikacia/web-app/results` priamo prehľadávať súbory vykonaných experimentov webovej aplikácie.

Aplikáciu je následne možné vypnúť príkazom: `docker-compose down -v`. Tento príkaz je tiež nutné spustiť v termináli operačného systému hostiteľského stroja na ceste adresára `aplikacia`.