VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA METOD PRO DETEKCI ODLEHLÝCH HODNOT

ANALYSIS OF OUTLIER DETECTION METHODS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DOMINIK LABAŠ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IVANA BURGETOVÁ, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Labaš Dominik, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Informační systémy
Název: **Analýza metod pro detekci odlehlých hodnot**
Analysis of Outlier Detection Methods
Kategorie: Data mining
Zadání:

1. Prostudujte různé metody pro detekci odlehlých hodnot.
2. Po dohodě s vedoucí připravte vhodné datové sady pro testování metod pro detekci odlehlých hodnot.
3. Navrhněte aplikaci, která umožní srovnání vlastností vybraných metod pro detekci odlehlých hodnot.
4. Po dohodě s vedoucí navrženou aplikaci implementujte.
5. Otestujte vybrané metody na zvolených datových sadách.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1
- Aggarwal, Charu C.: Outlier analysis. New York: Springer, 2013, xv, 446 p., ISBN 978-1-4614-6395-5.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Burgetová Ivana, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 22. října 2020

Abstrakt

Téma tejto práce je analýza metód pre detekciu odľahlých hodnôt. Na začiatok je poskytnutý popis odľahlých hodnôt a rôznych metód pre ich detekciu. Následne popisuje zvolené dátové sady, určené k testovaniu metód detekcie odľahlých hodnôt. Je predstavený návrh aplikácie, určenej k analýze popísaných metód. Sú predložené technológie, ktoré poskytujú modely pre detekciu odľahlých hodnôt. Implementácia je následne bližšie popísaná. Následne sú predložené výsledky z experimentov, ktoré predstavujú hlavnú časť tejto práce. Výsledky práce sú zhodnotené a jednotlivé modely sú navzájom porovnané. Na záver je predložený spôsob urýchlenia detekcie odľahlých hodnôt.

Abstract

The topic of this thesis is analysis of methods for detection of outliers. Firstly, a description of outliers and various methods for their detection is provided. Then a description of selected data sets for testing of methods for detection of outliers is given. Next, an application design for the analysis of the described methods is presented. Then, technologies are presented, which provide models for described methods of detection of outliers. The implementation is then described in more detail. Subsequently, the results of experiments are presented, which represent the main part of this thesis. The results are evaluated and the individual models are compared with each other. Lastly, a method for accelerating outlier detection is demonstrated.

Kľúčové slová

odľahlé hodnoty, štatistika, data mining, analýza, porovnanie

Keywords

outliers, statistics, data mining, analysis, comparison

Citácia

LABAŠ, Dominik. *Analýza metod pro detekci odlehlých hodnot.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivana Burgetová, Ph.D.

Analýza metod pro detekci odlehlých hodnot

Prehlásenie

Prehlasujem, že som túto semestrálnu prácu vypracoval samostatne pod vedením pani Ing. Ivany Burgetovej, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Dominik Labaš
17. mája 2021

Poďakovanie

Chcel by som poďakovať pani Ing. Ivany Burgetovej, Ph.D., za jej profesionálnu pomoc a vedenie pri vypracovaní tejto práce.

Obsah

Kapitola 1

Úvod

Detekcia odľahlých hodnôt je jednou z veľmi dôležitých častí oboru štatistiky. Jedná sa o hodnoty, ktoré sa od skupiny dát odčleňujú tak, že sa javia ako keby do danej dátovej sady nepatrili. Jednou z najdôležitejších vlastností odľahlých hodnôt je však to, že nie vždy sú nechcenou časťou dát. Niekedy sa môže jednať o nechcené hodnoty, no vo veľkej časti prípadov reprezentujú odľahlé hodnoty výskyt javu, ktorý je pre nás významne zaujímavý. Pri vyvíjaní nových liekov sú odľahlé hodnoty práve tie, ktoré sú účinné. Pri kontrolovaní bezpečnosti bankových účtov sú odľahlé hodnoty práve tie, ktoré vyžadujú najväčšiu pozornosť. Pri testoch v nemocnici budú chorí pacienti taktiež vstupovať ako odľahlé hodnoty. A tieto dôvody sú práve tie, ktoré robia detekciu odľahlých hodnôt tak významnou. A práve tieto dôvody sú tie, ktoré dávajú význam detekčné metódy odľahlých hodnôt analyzovať a takisto predstavujú dôvod pre vznik tejto práce.

Téma tejto práce je teda analýza metód odhaľovania odľahlých hodnôt. Jej cieľom je preštudovať dostupné metódy pre detekciu odľahlých hodnôt a následne ich experimentálne otestovať. Pre testovanie sú zvolené dátové sady, ktoré sú tvorené reálnymi dátami, ktoré sú určené k experimentom. Samotné experimenty budú prebiehať pomocou aplikácie, ktorá bola navrhnutá a implementovaná počas vypracovania tejto práce. Na záver je predložené zhodnotenie experimentov, ktoré reprezentujú výsledok celej práce.

Tieto body sú v tomto dokumente postupne vypracované a jeho štruktúra je nasledovná. Na úvod v kapitole 1 je predložená motivácia k zhotoveniu tejto práce. V kapitole 2 sa nachádza popis odľahlých hodnôt, spôsob ich rozlišovania a približuje spôsob pre ich následné spracovanie. Následne sa tu nachádza predstavenie jednotlivých spôsobov odhaľovania odľahlých hodnôt. Uvedené sú príklady a popis metód, ktoré sú následne implementované a analyzované. Kapitola 3 popisuje zvolené dátové sady, ktoré boli vybrané pre testovanie metód odhaľovania odľahlých hodnôt. Tieto dáta sú nevyhnutnou súčasťou experimentovania a analýzy. Následne sa v kapitole 4 nachádza návrh aplikácie, ktorá sa tomuto problému venuje. Kapitola 5 popisuje jazyky a knižnice, ktoré boli v práci využité. Kapitola 6 obsahuje podrobnejšie informácie o implementácii metód a programe určenom pre ich testovanie. Priebeh testovania je popísaný v kapitole 7, kde sú pomocou experimentovania vytvorené testy a ich výsledky sú zobrazené pomocou tabuliek a grafov. Záver v kapitole 8 obsahuje zhrnutie dosiahnutých výsledkov a výsledné zhodnotenie práce.

1.1 Motivácia

Štatistika je veda, založená na využívaní empirických dát [13]. To znamená, že v prvom kroku získame dáta pozorovaním alebo experimentovaním. Tie sú naďalej brané ako náš vstup určený pre spracovanie s účelom vytvorenia výstupu. Požadovaným výstupom sú čo najpresnejšie informácie popisujúce charakteristiky našich vstupných dát.

Problém nastáva v prípade, kedy sa v našich dátach vyskytuje chyba. Napríklad na vstupe sme obdržali sadu pozorovaných hodnôt a chceme získať priemernú hodnotu. Náš vstup vyzerá následovne: 10, 11, 9, 10, 11, 100, 9, 12, 10, 8. Po výpočte priemeru získame výsledok 19. Z dát je síce zrejmé, že hodnoty sa pri našom pozorovaní pohybovali okolo hodnoty 10. Ako je teda možné, že sa náš výsledok toľko líši od reálnej hodnoty? Je to z toho dôvodu, že sa v dátach nachádza chybná hodnota 100. Jedná sa o chybu v dátach, ktorá môže vzniknúť pri chybnom meraní, prenose dát, alebo je to chyba človeka pri práci s dátami. Chyby v reálnych dátach sú bežné a dáta sú často nekvalitné a nekvalitné dáta vedú k nekvalitnému výsledku.

Preto sa pred samotným spracovaním musia dáta často upraviť do čo najkvalitnejšej formy. V štatistike predspracovanie dát obsahuje niekoľko úkonov, ako odhaľovanie chýbajúcich hodnôt, normalizácia, formátovanie alebo odstránenie duplicitných hodnôt. Táto príprava dát môže byť často náročná a má výrazný podiel na zložitosti projektu. Niekto by mohol namietať, že je ľahké chyby odhaliť na prvý pohľad. Predstavme si však, že namiesto 10 záznamov máme na vstupe 10 000 záznamov. V reálnom živote sú dáta taktiež častokrát mnoho-dimenzionálne, čo zložitosť odhaľovania chybných hodnôt mnohokrát zväčšuje. Preto je vhodné si zvoliť metódy úpravy dát tak, aby sme vo výsledku vynaložili čo najmenej času. Táto skutočnosť nás spolu s porekadlom ’čas sú peniaze’ nabádajú k ich analýze. Jedným z krokov prípravy dát je aj detekcia odľahlých hodnôt a práve analýze tejto časti sa táto práca venuje.

Detekcia odľahlých hodnôt je jednou z dôležitých oblastí v obore dolovaní dát [4]. Ako už bolo spomenuté využitie detekcie odľahlých hodnôt je rozšírené v mnohých oblastiach, kde nás zaujímajú práve odľahlé hodnoty ako je bankovníctvo a detekcia podvodov, zdravotníctvo, sledovanie dopravy alebo vojenské sledovanie.

Kapitola 2

Odľahlé hodnoty a metódy ich detekcie

Táto kapitola predstavuje teoretický podklad pre zhotovenie tejto práce. V sekcii 2.1 detailnejšie popisuje odľahlú hodnotu, ako odľahlé hodnoty rozlišujeme, odhaľujeme a ako s nimi pracujeme. Sekcia 2.2 následne popisuje dostupné metódy detekcie odľahlých hodnôt, ich ohodnotenie a využitie.

2.1 Čo je odľahlá hodnota

Čo je to odľahlá hodnota? Odľahlá hodnota v štatistike predstavuje dátové body, ktoré sa viditeľne líšia od ostatných hodnôt v dátach [9]. Je to hodnota, ktorá vybočuje z očakávaného vzoru dát. Jej existencie môže pri spracovaní dát výrazne zmeniť výsledky. Odľahlé hodnoty sa delia na základe niekoľko faktorov [14]. Môžu to byť jednorozmerné alebo viacrozmerné odľahlé hodnoty. Ďalší spôsob delenia je na bodové, kontextové alebo regionálne odľahlé hodnoty. Tretím spôsobom popisujeme globálne a lokálne odľahlé hodnoty. Toto rozdelenie sa môže navzájom prekrývať a teda jedna odľahlá hodnota môže byť naraz viacrozmerná, globálna a aj bodová. Jednotlivé typy odľahlých hodnôt sú popísané v nasledujúcich sekciách.

2.1.1 Univariátne a multivariátne hodnoty

Delenie, ktoré je popísane v tejto sekcii je založené na tvare jednotlivých hodnôt. Univariátne a multivariátne alebo aj jednorozmerné a viacrozmerné odľahlé hodnoty reprezentujú hodnoty delené na základe ich počtu rôznych atribútov.

V prípade jednorozmerných hodnôt sa hodnota skladá len z jedného atribútu. Odľahlá hodnota je teda podozrivé pozorovanie tohto atribútu. Príklad jednorozmerných hodnôt sa nachádza v tabuľke 2.1.

Viacrozmerné hodnoty sa skladajú z viacerých atribútov. Príklad takýchto hodnôt sa nachádza na obrázku 3.1, kde sú hodnoty skladajúce sa z dvoch atribútov. V takomto prípade môže nastať situácia, kde odľahlá hodnota popisuje normálne správanie na základe jedného atribútu, ale jej správanie vybočuje podľa iného atribútu. V tomto prípade sa stáva výzvou množstvo konfigurácií v rámci viacrozmernej množiny pozorovaných hodnôt. Tento typ hodnôt je aj tým, ktorým sa táto práca z najväčšej časti venuje.

2.1.2 Bodové, kontextové a regionálne hodnoty

Toto delenie je na rozdiel od predošlého založené na základe jeho chovania v rámci konkrétnej dátovej sade.

Bodové odľahlé hodnoty, ako aj ich názov popisuje, sú pozorovania, ktoré predstavujú jeden dátový bod. Ten sa vyznačujú jedinečným a podozrivým správaním v porovnaní so svojím okolím alebo k celému súboru údajov. Príklad bodových hodnôt sa nachádza na obrázku 3.1.

Kontextové odľahlé hodnoty, sú také hodnoty, ktoré popisujú správne pozorovanie v jednom kontexte, ale v inom reprezentujú odľahlú hodnotu. Napríklad pri pozorovaní počtu návštevníkov v obchodnom dome bude odľahlou hodnotou pozorovanie, ktoré informuje o návštevníkoch mimo otváraciu dobu daného obchodu. Ďalším príkladom môže byť bežná hodnota 0°C v zime, ktorá sa stáva odľahlou ak je nameraná v strede leta.

Regionálne odľahlé hodnoty reprezentujú podmnožinu, ktorá má spoločné vlastnosti. Dátové body v tejto podmnožine sa javia ako správne hodnoty vo vzťahu jeden k druhému v rámci danej podmnožiny. No vzhľadom k zvyšnému súboru dát ich vlastnosti predstavujú odľahlú hodnotu. Takýto stav môže napríklad nastať pri nesprávnom nakonfigurovaní meracieho prístroja, pomocou ktorého bola uskutočnená sada meraní. Pri zbere dát sa potom vyskytne skupina meraní, ktorá popisuje odlišné chovanie od zbytku dát.

2.1.3 Globálne a lokálne hodnoty

Finálne delenie, predstavené v tejto sekcii je založené na chovaní hodnôt v rámci podmnožín dátovej sady.

Globálne odľahlé hodnoty popisujú chovanie odlišné od celej zvyšnej sady dát. Často popisujú špecifické chovanie, ktoré sa veľmi výrazne odchyľuje od normálneho chovania. Tieto odľahlé hodnoty sú preto aj jednými z najľahšie oddeliteľných hodnôt.

Lokálne odľahlé hodnoty nie sú výrazne odlučiteľné v porovnaní s chovaním celej sady dát. Jedná sa teda o odľahlé hodnoty, ktoré nie je jednoduché odhaliť. Tieto hodnoty však javia podozrivé správanie v rámci podmnožiny dátovej sady. K ich odhaleniu je nutné na rozdiel od ich chovania v rámci dátovej sady pozorovať susedské vzťahy jednotlivých hodnôt. Tým dosiahneme, že sa vyhľadávanie obmedzí len na podmnožinu susedov v rámci dátovej sady.

2.1.4 Ako odhaliť odľahlú hodnotu

Na to, aby bolo možné začať hľadať odľahlé hodnoty je potrebné získať informácie o danej sade pozorovaní. Takéto informácie môžu byť napríklad priemer alebo medián danej sady. V tabuľke 2.1 sú zobrazené niektoré základné informácie, ktoré je možné z danej sady hodnôt získať. Nie všetky informácie sú rovnako citlivé na výskyt odľahlých hodnôt. Jedným príkladom je priemer, ktorý, ako je možné vidieť v tabuľke, je veľmi citlivý na výskyt odľahlej hodnoty. Na rozdiel od priemeru medián zostáva rovnaký nezávisle na tom, či sa v dátach vyskytuje odľahlá hodnota alebo nie. Vo výsledku je ku správnym hodnotám hodnota mediánu bližšia, ako hodnota priemeru. Označme teda rozdiel medzi jednotlivými hodnotami dátovej sady a mediánom, prípadne rozdiel medzi jednotlivými hodnotami dátovej sady a priemerom, ako skóre daných dátových bodov. Následne absolútna hodnota tohto skóre predstavuje ako veľké je riziko, že daný dátový bod je odľahlá hodnota. Odľahlé hodnoty sú teda práve tie, ktoré majú najväčšie skóre. Vo výsledku je skóre vypočítané pomocou me-

8, 9, 9, 10, 10, 10, 11, 11, 12, 100	8, 9, 9, 10, 10, 10, 11, 11, 12
Priemer: 19.0	Priemer: 10.0
Medián: 10	Medián : 10
Modus : 10	Modus : 10
Rozsah: 92	Rozsah: 4
Minimum: 8	Minimum: 8
Maximum: 100	Maximum: 12
Suma: 190	Suma: 90
IQR: 1.75	IQR: 2
Štandardná odchýlka: 28.4839	Štandardná odchýlka: 1.2247

Tabuľka 2.1: Tabuľka ukazujúca sadu dát obsahujúcu odľahlú hodnotu v ľavej polovici a sadu dát bez odľahlej hodnoty v pravej polovici. Ďalej sú uvedené niektoré štatistiky, ktoré poukazujú na vlastnosti daných dát.

diánu presnejšie ako skóre získané pomocou priemeru. Preto je možné povedať, že medián je na odhaľovanie odľahlých hodnôt výhodnejší ako priemer.

Ak máme vybranú informáciu o normálnom chovaní dát je potrebné určiť, ako veľmi sa od nej ostatné hodnoty môžu odchýliť pre to, aby boli stále považované za správne pozorovania. Napríklad by bolo možné využiť štandardnú odchýlku. To by znamenalo, že v príklade z tabuľky 2.1 sú správne hodnoty v rozmedzí od -18.5 do 38.5 pre ľavú polovicu a v rozmedzí od 8.8 do 11.2 pre pravú polovicu. Nielen je viditeľné, že štandardná odchýlka je veľmi citlivá na odľahlé hodnoty, ale dokonca v pravej polovici označí hodnoty 8 a 12 ako odľahlé.

Je teda dôležité nájsť, čo najpresnejší spôsob detekcie. V predchádzajúcej časti bolo ukázané, že medián nieje ľahko ovplyvniteľný odľahlými hodnotami. Ak zoberieme medián z ľavej a pravej polovice, získame 25. (Q_1) respektíve 75. (Q_3) percentil, taktiež známe ako prvý a tretí kvartál. Ich rozdiel sa nazýva medzi-kvartálny rozsah (IQR). Teda $IQR = Q_3 - Q_1$. Pravidlo medzi-kvartálneho rozsahu tvrdí, že správne hodnoty sa vyskytujú v rozsahu od spodnej hranice získanej ako $Q_1 - (1.5 \cdot IQR)$ po hornú hranicu $Q_3 + (1.5 \cdot IQR)$. Týmto spôsobom sme získali jeden z používaných spôsobov detekcie odľahlých hodnôt, ktorý je často využívaný pre jednorozmerné dáta [12].

2.1.5 Čo s odľahlými hodnotami?

Odľahlé hodnoty nieje potrebné len odhaliť, ale aj správne spracovať. Ako už bolo spomenuté nie vždy predstavujú odľahlé hodnoty nežiadané chyby. Práve naopak niekedy sú odľahlé hodnoty tým najzaujímavejším pozorovaním. Napríklad pri sledovaní hladiny rieky je riziko záplavy práve v prípade, kedy sa v dátach vyskytne príliš veľká hodnota a tú je možné odhaliť ako odľahlú.

Pri výskyte odľahlej hodnoty je teda v prvom rade určiť jej príčinu. Ak sa jedná o chybné hodnoty, ktoré sú nežiadanou zložkou dát sú k dispozícii dva rôzne smery úpravy. Prvým spôsobom je odľahlé hodnoty z dát odstrániť a pokračovať v práci s redukovanou sadou pozorovaní. Táto možnosť je rýchla a jednoduchá. Na druhú stranu sa jedná o veľmi tvrdý krok, ktorý pri veľkej populácii odľahlých hodnôt môže dátovú sadu zmenšiť na toľko, že by ju znehodnotil. Pri rozhodovaní, či je vhodné hodnoty odstrániť je taktiež miera falošne označených odľahlých hodnôt. Falošne označené hodnoty sú pri hľadaní odľahlých hodnôt

bežné, pretože je výhodnejšie označiť správnu hodnotu za odľahlú, ako odľahlú hodnotu za správnu. Teda pri ich odstránení odľahlých hodnôt prichádza aj k strate tých správnych.

Druhým smerom je odľahlé hodnoty opraviť. Táto voľba je ale zložitejšia na vykonanie. Oprava je možná pomocou nahradením hodnôt na základe správnych blízkych hodnôt. Ďalšou možnosťou je vykonať nové merania, čo zväčša predstavuje nákladný krok. K dispozícii sú aj vzorce a funkcie pre opravu dát, no ich vykonanie môže do dát vložiť šum. Jednou z metód je aj interpolácia susedných bodov, pri čom môže dôjsť k nadmernému vyhladení dát.

V prípade, že sa jedná o odľahlé hodnoty, ktoré nás zaujímajú, sa odhalené hodnoty naďalej spracúvajú podľa toho o aké dáta sa jedná.

2.2 Dostupné metódy detekcie odľahlých hodnôt

Pre detekciu odľahlých hodnôt je dostupné množstvo odlišných metód. Ich spoločnou vlastnosťou je, že vždy vytvárajú model popisujúci normálne chovanie dát. Techniky detekcie odľahlých metód sú v tejto práci popísané v 5 hlavných skupinách. Prvá skupina modelov je založená na základe distribúcie dát, túto skupinu reprezentujeme pomocou pravdepodobnostných a štatistických modelov. Ďalšia skupina využíva rozklad alebo projekciu dát a je reprezentovaná lineárnymi modelami. Treťou skupinou sú metódy založené na vzdialenosti a hustote. Štvrtá skupina sa zaoberá vysoko-dimenzionálnymi dátami. Posledná časť je venovaná kombinácii viacerých metód. Táto sekcia bola vytvorená za pomoci knihy Outlier Analysis [1].

2.2.1 Základný model pre detekciu odľahlých hodnôt

V podstate všetky metódy detekcie odľahlých hodnôt sa snažia k jednotlivým dátovým bodom priradiť skóre, vďaka ktorému je možné určiť, ktoré dátové body sú odľahlé. Pre to aby bolo možné skóre vypočítať, je najprv potrebné vytvoriť model normálnych vzorov v dátach. Tieto modely sú generatívne modely, napríklad ako zmesový Gaussov model alebo model založený na regresii [11]. Skóre je následne vypočítané ako odchýlka od vytvoreného modelu. Pre detekciu odľahlých hodnôt je potrebné si zvoliť správny model. A to z toho dôvodu, že každý model podáva rôzny výkon na základe použitých dát.

Pravdepodobnostné modely sú prvé modely pre detekciu odľahlých hodnôt. Pri využití pravdepodobnostných modelov je skóre určené ako pravdepodobnosť, že dátový bod patrí do generatívneho modelu.

2.2.2 Lineárny model

Skóre v lineárnom modelovaní je zvyšková vzdialenosť dátového bodu od reprezentácie dát v nižšej dimenzii. Lineárne modely využívajú fakt, že v dátach sa medzi atribútami vyskytuje určitá závislosť. To umožňuje na základe jedného atribútu predpovedať iný. Hľadanie týchto závislostí sa označuje ako regresné modelovanie a jedná sa o analýzu vzájomných vzťahov. Hlavným predpokladom lineárnych modelov je ten, že dáta sa môžu byť zobrazené do nižšej dimenzie. Odľahlé hodnoty sú teda dátové body, ktoré nieje možné takto zobraziť. Teda hľadáme také pod-dimenzie, v ktorých sa odľahlé hodnoty chovajú výrazne inak ako ostatné hodnoty. Atribút y je teda možné modelovať ako lineárnu funkciu závislú od d premenných ako:

$$y = \sum_{i=1}^{d} w_i \cdot x_i + w_{d+1}, \tag{2.1}$$

kde d predstavuje pôvodný počet dimenzií a hodnoty $x_1 ... x_d$ predstavujú pôvodné atribúty dátového bodu. Koeficienty $w_1 ... w_{d+1}$ sú získané postupným učením sa z dát. Cieľom je nájdenie takého pod-priestoru, kde každý atribút je takzvaná hlavná zložka. Hlavná zložka je atribút, pre ktorý neexistuje ďalší atribút, medzi ktorými existuje závislosť.

Analýza hlavných komponentov

Metóda lineárnych modelov je využitá pri modelovaní pomocou analýzy hlavných komponentov (PCA). Táto metóda postupne hľadá dimenziu *d - 1* a skóre jednotlivých bodov je vypočítané ako zvyšná ortogonálna vzdialenosť. Takto je možné pomocou metódy PCA postupne nájsť k-rozmernú dimenziu kde $k < d$. Pre učenie sa využíva funkcia druhej mocninovej straty.

Metóda podporných vektorov

Jedným zo spôsobov hľadania cieleného priestoru je takzvaná metóda využívajúca One-Class Support Vector Machines. Tie využívajú funkciu označenú ako kernel, pre transformáciu dát z jedného priestoru do iného. Dátový bod je tu reprezentovaný vektorom $\overline{x}$. Tento vektor je transformovaný pomocou funkcie $\phi()$ na nový vektor označený ako $\phi(\overline{x})$. Pomocou tejto funkcie dokážu One-Class Support Vector Machines zostrojiť nad-rovinu, ktorá oddelí odľahlé hodnoty od zvyšných dát. Táto nad-rovina je reprezentovaná ako:

$$\overline{w} \cdot \phi(\overline{x}) + b = 0, \tag{2.2}$$

kde $\overline{w}$ je koeficient, ktorý patrí novej dimenzii. Pre učenie nových hodnôt sa využíva funkcia pre strojové učenie hinge loss. Hodnota b reprezentuje posun. Na obrázku 2.1 je zobrazená aplikácia tento metódy na dáta, kde bola chybovosť dát určená ako 50%. Výsledkom je teda rozdelenie dát na dve polovice. Graf na ľavej strane používa lineárny kernel, graf na pravej strane používa ako kernel radiálnu bázovú funkciu (rbf).

Neurónové siete

Neurónové siete sú pre lineárne modelovanie ako stvorené. Jedná sa o sieť rozdelenú do vrstiev, ktoré sa skladajú z neurónov. Hodnota neurónu sa vypočíta ako súčin dvoch vektorov:

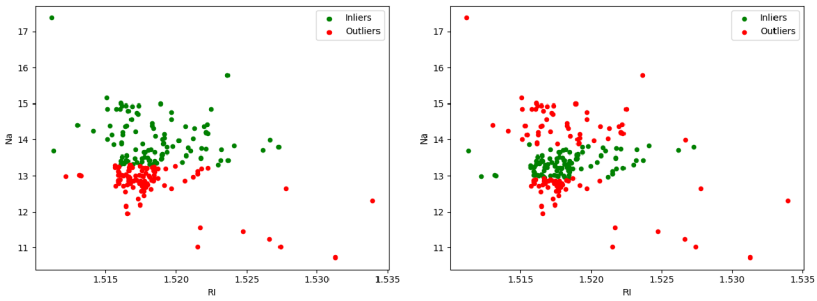
$$f(x) = \overline{x} \cdot \overline{w} + b \tag{2.3}$$

Vektor $\overline{x}$ reprezentuje vstupnú vrstvu, teda dátový bod. Vektor $\overline{w}$ predstavuje jednotlivé váhy a b predstavuje posun.

Jedným zo spôsobov ako využiť neurónové siete pri detekcii odľahlých hodnôt je dátové body neurónovou sieťou transformovať a následne hľadať odľahlé hodnoty výpočtom chyby pri rekonštrukcii.

2.2.3 Model založený na vzdialenosti a hustote

V modelovaní založenom na blízkosti je skóre vzdialenosť dátového bodu a k-najbližších susedov alebo hodnota lokálnej hustoty. Teda sa predpokladá, že odľahlé hodnoty sa nachá-



Obr. 2.1: Príklad využitia One-Class Support Vector Machines, ktoré očakávajú, že 50% dát sú odľahlé hodnoty.

dzajú vo veľkej vzdialenosti od väčšiny dát. Tieto techniky definujú odľahlú hodnotu ako bod, ktorého okolie je riedko osídlené. Najčastejšie sa využívajú tri techniky založené na vzdialenosti, hustote alebo zhlukovaniu.

Podľa vzdialenosti

Pomocou vzdialenosti dátového bodu s jeho susedom sa definuje ich blízkosť. Dátové body s veľkými vzdialenosťami k najbližším susedom sú definované ako odľahlé hodnoty. Algoritmy založené na vzdialenosti zvyčajne vykonávajú analýzu s oveľa podrobnejšou granularitou ako metódy hustoty a zhlukovania. Ich nevýhodou je zvyčajne vysoká výpočtová zložitosť.

Najjednoduchšou metódou je zistiť vzdialenosť dátového bodu s k najbližšími susedmi a následne tieto hodnoty spočítať. Takýmto spôsobom vypočítame skóre pre každý bod a body s najväčším skóre sú odľahlé hodnoty.

Ďalšie možnosti sú vypočítať skóre ako priemernú vzdialenosť ku k najbližším susedom. Alebo jednotlivé vzdialenosti so susedmi zoradiť a skóre určiť ako medián.

Podľa hustoty

Počet dátových bodov nachádzajúcich sa v špecifickom priestore okolia jedného dátového bodu je využitý na určenie miestnej hustoty. Táto hustota je následne prevedená na skóre, podľa ktorého určíme, ktoré dátové body sú odľahlými hodnotami. Výhoda týchto algoritmov je ich schopnosť odhaliť lokálne odľahlé hodnoty.

Prvým spôsobom detekcie odľahlých hodnôt je *Local Outlier Factor* (LOF). Ten meria lokálnu odchýlku dátového bodu s jeho susedmi. Táto metóda vykonáva dva kroky. V prvom kroku sa určí hustota pre dátový bod na základe jeho vzdialenosti s k-najbližšími susedmi. Následne sa porovnávajú hustoty priradené k jednotlivým dátovým bodom. Týmto spôsobom sa určia oblasti s podobnou hustotou. Dátové body v oblastiach s nízkou hustotou sú teda označené ako odľahlé.

Vzdialenosť dátového bodu **X** od k-tého najbližšieho suseda je určená ako $D^k(X)$. Vzdialenosť

$$R_k(X,Y) = max(vzdialenost(X,Y), (D^k(Y)) \tag{2.4}$$

popisuje takzvanú vzdialenosť dosiahnuteľnosti bodu X od bodu Y. Táto vzdialenosť predstavuje reálnu vzdialenosť dvoch dátových bodov *vzdialenosť(X, Y)* alebo vzdialenosť $D^k(Y)$ v prípade, že bod X patrí medzi k-najbližších susedov bodu Y. Následne je možné určiť priemernú dosiahnuteľnú vzdialenosť bodu X ako priemer všetkých vzdialeností $R_k(X,Y)$, kde body Y reprezentujú k-najbližších bodov bodu X. Táto vzdialenosť je označená ako $AR_k(X)$.

Local Outlier Factor bodu X označený ako $LOF_k(X)$ je priemer pomeru vzdialeností $AR_k(X)$ s hodnotami $AR_k(Y)$ k-najbližších bodov. teda:

$$LOF_k(X) = Priemer_{Y \in L_k(X)} \frac{AR_k(X)}{AR_k(Y)}, \tag{2.5}$$

kde $L_k(X)$ je množina bodov Y, ktoré patria medzi k-najbližších bodov bodu X.

Ďalšou možnosťou je metóda LOCI, ktorá definuje hustotu $M(X,\epsilon)$. Táto hustota predstavuje počet dátových bodov nachádzajúcich sa v okolí dátového bodu X s polomerom ϵ.

Zhlukovanie

Samotné zhlukovanie nieje metóda určená pre detekciu odľahlých hodnôt. Jedná sa o metódu, ktorá jednotlivé dátové body zoskupuje do zhluku s podobnými vlastnosťami. Detekcia odľahlých hodnôt v tejto metóde môže byť označená ako jej vedľajší produkt. Ak po rozdelení dátových bodov exitujú dátové body, ktoré neboli priradené do zhlukov sú označené ako odľahlé. Skóre odľahlých hodnôt sa počíta na základe toho, či patria do zhluku, vzdialenosti od najbližšieho zhluku a jeho veľkosti. Rozdiel medzi zhlukovaním a výpočtom hustoty je ten, že zhlukovanie rozdeľuje a ohodnocuje dátové body. Metódy na základe hustoty na druhú stranu rozdeľujú a ohodnocujú samotný priestor. Jednou z metód zhlukovania, ktorá podáva dobré výsledky je napríklad metóda DBSCAN [6].

2.2.4 Detekcia odľahlých hodnôt vo vysoko-dimenzionálnych dátach

V reálnych situáciách môžu dáta obsahovať stovky atribútov v jednom dátovom bode. So stúpajúcim počtom dimenzií, veľká skupina predstavených modelov začína strácať efektivitu. Pri vysokej dimenzionalite dáta sa od seba viac a viac odďaľujú a odľahlé hodnoty začínajú byť maskované vysokým počtom nerelevantných atribútov. Napríklad metódy založené na vzdialenosti berú ohľad na každú dimenziu. No v realite niektoré dimenzie nemusia byť vôbec dôležité. Identifikácia relevantných dimenzií je extrémne ťažký problém. Ale pre hľadanie odľahlých hodnôt je dôležité tieto relevantné dimenzie zvoliť. Odľahlé hodnoty sa v niektorých dimenziách môžu javiť ako správne. Preto je voľba zvolenia si jednej relevantnej dimenzie nedostatočná. Teda obecne je dôležité kombinovanie viacerých výsledkov z rôznych pod-priestorov v dátovom sete. Z toho sa dá detekcia odľahlých hodnôt vo vysoko-dimenzionálnych dátach popísať ako kombinovanie viacerých modelov. Existuje niekoľko metód vyberania pod-priestorov. Prvým spôsobom je nezaujatý, kde sa vyberajú dimenzie náhodne z pôvodnej dátovej sady. Jedná sa o veľmi jednoduchý spôsob, no často dosahuje dobrých výsledkov. Tento spôsob je reprezentovaný metódou feature bagging. Ďal-

šia varianta je vyberať podľa vzácnosti distribúcie dátových bodov. Poslednou možnosť je rozhodovanie na základe agregácie štatistických údajov.

Metóda feature bagging rozdelí dáta do podmnožín atribútov, na ktoré sú aplikované metódy ako LOF alebo k-najbližších susedov. Atribúty pre podmnožiny sú vyberané náhodne a ich počet môže byť upresnený podľa celkového počtu atribútov. Výsledkom je potom zjednotenie jednotlivých podmnožín. Toto výsledné skóre je získané napríklad spriemerovaním alebo výberom maximálnych hodnôt pre každý dátový bod.

Ďalšou metódou detekcie sú izolačné lesy. Izolačný les je kombinácia izolačných stromov. V izolačnom strome sú dáta rekurzívne delené rezmi paralelnými s osami v náhodne vybranom bode a náhodne vybranom atribúte. Toto delenie prebieha až do stavu, kde sú dátové body z dátovej sady rozdelené do samostatných uzlov. Vo výsledku sa odľahlé hodnoty nachádzajú n listoch s najmenšou hĺbkou v stromovej štruktúre. To z toho dôvodu, pretože sa odľahlé hodnoty nachádzajú v riedko osídlených častiach. Taktiež je možné dĺžku jednotlivých vetví pre každý list spriemerovať a vypočítať tak mieru normality rozhodovacej funkcie.

2.2.5 Kombinovanie modelov

Kombinovanie modelov je využívané k spresneniu výsledkov pri detekcii odľahlých hodnôt. Patria sem už spomínané techniky ako feature bagging a izolačné lesy. Kombinovacie metódy využívajú výsledky z rôznych modelov k vytvoreniu jednotného výstupu. Myšlienka tejto metódy je tá, že niektoré metódy fungujú efektívnejšie na určité typy dát. Kombinácia modelov je však často schopná pracovať robustnejšie kvôli jej schopnosti kombinovať výstupy viacerých algoritmov.

Prvá kombinačná metóda na zjednotené skóre odľahlých hodnôt je ich spriemerovanie skrz výsledky niekoľkých modelov. Ďalšia metóda predstavuje výber maxima z výsledkov modelov. Taktiež je možné vyberať medián, alebo na základe väčšiny hlasov, kde odľahlá hodnota musí byť označená väčšinou modelov.

Metóda spriemerovania maxím je metóda, kde výsledky z rôznych modelov rozdelíme do niekoľkých skupín. V rámci každej skupiny vyberieme pre každý dátový bod maximálne skóre, ktoré následne spriemerujeme cez všetky skupiny do jednotného výsledku.

2.2.6 Ohodnotenie kvality modelu

V predchádzajúcich sekciách boli popísané metódy pre detekciu odľahlých hodnôt. Cieľom týchto metód je identifikovať odľahlé hodnoty čo najpresnejšie. To znamená identifikovať všetky odľahlé hodnoty pričom označiť čo najmenší počet správnych hodnôt ako odľahlé. Pre získanie presnosti je teda potrebné vedieť, ktoré dátové body sú skutočne odľahlými, čo nie je bežne dostupné. Z toho dôvodu sa často metódy najskôr testujú na umelo vytvorených dátových sadách, do ktorých sa odľahlé hodnoty umelo pridávajú. ich príklad je zobrazený na obrázku 3.1. Po otestovaní sú tieto metódy aplikované na reálne dátové sady.

Väčšinou sa presnosť udáva pomocou miery pravých pozitívnych označení a falošných pozitívnych označení. Teda počet odľahlých hodnôt označených za odľahlé a počet správnych hodnôt označených za odľahlé. Výkonnosť metód je možné zobraziť pomocou krivky ROC (receiver operating characteristic) [3]. Tá je zobrazená ako graf ukazujúci pomer medzi skutočnými označenými odľahlými hodnotami a falošnými. Výkon modelu pomocou krivky ROC je následne možné určiť ako plochu nachádzajúcu sa pod danou krivkou. Ďalší spôsob znázornenia výkonnosti metódy je využitím P@n (Precision @ n) [5], ktoré je vypočítané ako:

$$P@n = tp/(tp + fp), \tag{2.6}$$

kde *tp* predstavuje pravé pozitívne označenia a *fp* predstavuje falošne pozitívne označené hodnoty. Táto presnosť predstavuje schopnosť modelu označiť odľahlú hodnotu ako odľahlú.

Ďalším dôležitým kritériom modelov je čas využitý modelmi pre detekciu odľahlých hodnôt. Dôležitosť tohto kritéria neustále stúpa z dôvodu vylepšovania systémov pre zber dát, ktoré vyžadujú spracovanie dát nielen so stúpajúcim počtom atribútov ale aj počtom jednotlivých záznamov.

Na záver je možné sledovať veľkosť modelom využitej pamäte pri vykonávaní detekcie.

2.2.7 Práca s modelom

V sekcii 2.1.4 sme popísali metódu výpočtu dolnej a hornej hranice pomocou IQR. Vytvorenie modelu je v tomto prípade teda deklarovanie existencie dolnej a hornej hranice, popísanie spôsobu ich výpočtu a detekcie odľahlej hodnoty s ich využitím. Trénovanie modelu reprezentuje samotný výpočet dolnej a hornej hranice. Aplikovanie modelu je následné určenie, či sa daná hodnota nachádza v rozsahu medzi dolnou a hornou hranicou.

Vytvorenie a využitie modelu prebieha v troch krokoch. V prvom je vytvorený model, kde je potrebné špecifikovať jeho parametre. V druhom kroku nastáva trénovanie modelu, kde je potrebné modelu poskytnúť trénovaciu sadu dát. Následne je možné model využiť na detekciu odľahlých hodnôt. Detekcia odľahlých hodnôt je možná na dátach, ktoré majú rovnaký počet dimenzií ako trénovacie dáta. Taktiež je možné odľahlé hodnoty vyhľadať priamo v trénovacej dátovej sade.

Kapitola 3

Dátové sady

Táto kapitola popisuje dáta určené k testovaniu metód pre detekciu odľahlých hodnôt. V sekcii 3.1 sú predstavené sady obsahujúce reálne dáta a spôsob ich čítania. Sekcia 3.2 popisuje spôsob pre umelé generovanie dát spolu s dôvodom, prečo je táto možnosť pre túto prácu vhodná.

3.1 Reálne dáta

Pre experimentovanie boli vybraté dátové sady z úložiska datasetov ODDS [10]. Tieto dáta sú taktiež k dispozícii v dátovom sklade pre strojové učenie UCI[2]. Pre experimenty boli tieto dátové sady vybraté preto, že tieto dátové sady obsahujú informácie o odľahlých hodnotách, ktoré sa v nich nachádzajú. Dátové sady z úložiska ODDS sú k dispozícii upravené tak, že ich dátové body boli rozdelené do skupín, z ktorých bol počet výskytov niektorých zhlukov znížený a označený ako odľahlý. Týmto spôsobom je teda možné odľahlé hodnoty simulovať. K daným sadám sú uvedené ich názvy, počet záznamov, ktoré reprezentujú dátové body. Ďalej je uvedený počet dimenzií a krátky popis obsahu dátovej sady. Ak dátová sada obsahuje záznam s identifikačným číslom, nieje tento atribút zahrnutý v počte dimenzií z toho dôvodu, že táto informácia je k detekcii odľahlých hodnôt nevyužiteľná a mohla by znehodnocovať konečný výsledok. Taktiež je uvedený počet odľahlých hodnôt nachádzajúci sa v dátach. Zoznam vybratých dátových sád je následovný:

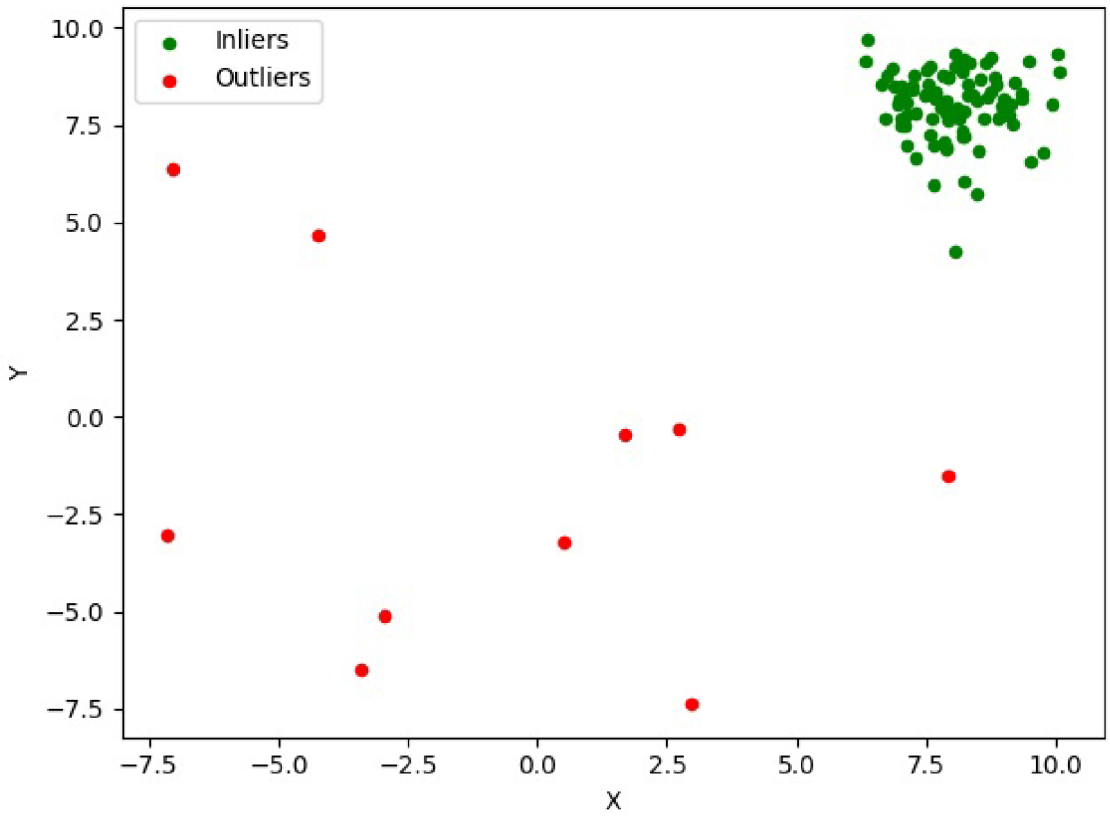
- Názov: Glass Identification, počet záznamov: 214, počet dimenzií: 9, počet odľahlých hodnôt: 9 (4.2%). Jedná sa o štúdiu klasifikácie typov skiel pre kriminologické vyšetrovanie. Obsahuje informácie ohľadom typu skla pre jeho identifikáciu, ktorá by mohla byť využitá ako dôkaz. Pôvodné dáta obsahovali 11 hodnôt, kde prvý stĺpec reprezentujúci index bol odstránený a posledný stĺpec dát obsahoval označenie skla. Toto označenie bolo reprezentované celočíselnou hodnotou v rozsahu 1–7. Pomocou tejto hodnoty boli dátové body rozdelené medzi správne a odľahlé. na záver bol tento stĺpec odstránený a dátové body pozostávajú len z 9 reálnych čísel.
- Názov: Breast Cancer Wisconsin (Diagnostic), počet záznamov: 278, počet dimenzií: 30, počet odľahlých hodnôt: 21 (5.6%). Dataset obsahujúci záznamy prípadov rakoviny prsníka. Dátový set zo zborníka ODDS obsahuje upravenú sadu, kde bol zámerne znížený počet zhubných výskytov na 21 prípadov. Tieto sú považované za odľahlé hodnoty. Všetky hodnoty v dátových bodoch sú reprezentované kladnými desatinnými číslami v rozsahu 0–1.

- Názov: Cardiotocography, počet záznamov: 1831, počet dimenzií: 21, počet odľahlých hodnôt: 176 (9.6%). Dátový set, ktorý obsahuje záznamy srdečného tepu plodu a počtu kontrakcií. Taktiež je dostupný dátový set upravený pre detekciu odľahlých hodnôt. Vlastnosti dátových bodov sú reprezentované reálnymi číslami.
- Názov: Arrhythmia, počet záznamov: 452, počet dimenzií: 274, počet odľahlých hodnôt: 66 (15%). Dátový set určený pre detekciu prítomnosti srdcovej arytmie a jej klasifikáciu. Dátové body obsahujú zmes reálnych čísel, celočíselných hodnôt a pravdivostných hodnôt reprezentovaných celočíselnou hodnotou 1 alebo 0.
- Názov: Statlog (Shuttle), počet záznamov: 49097, počet dimenzií: 9, počet odľahlých hodnôt: 3511 (7%). Dátový set obsahujúci dáta zoradené v časovej postupnosti. Dátový bod obsahuje 9 celočíselných hodnôt.
- Názov: Statlog (Landsat Satellite), počet záznamov: 6435, počet dimenzií: 36, počet odľahlých hodnôt: 2036 (32%). Tento dátový set obsahuje informácie o satelitných snímkach. Všetky hodnoty dátových bodov sú reprezentované celými číslami.
- Názov: Optical Recognition of Handwritten Digits, počet záznamov: 5216, počet dimenzií: 64, počet odľahlých hodnôt: 150 (3%). Dátový set, ktorý obsahuje informácie na základe ručne písaných číslic. Všetky hodnoty nachádzajúce sa v týchto dátach sú reprezentované kladnými celými číslami.
- Názov: MNIST, počet záznamov: 7603, počet dimenzií: 100, počet odľahlých hodnôt: 700 (9.2%). Jedná sa o databázu ručne písaných číslic, ktorá obsahuje vyše 70 000 záznamov. Z týchto záznamov je vybratá podmnožina, ku ktorej sú priradené atribúty, ktoré sú vhodné pre detekciu odľahlých hodnôt. Dátové body sa skladajú prevažne z reálnych čísiel doplnených o 8 výskytov pravdivostných hodnôt 1 alebo 0.
- Názov: Covertype, počet záznamov: 286048, počet dimenzií: 10, počet odľahlých hodnôt: 2747 (0.9%). Dátový set určený k rozpoznávaniu lesov. Tento dátový set má najväčší počet záznamov z všetkých vybratých datasetov. Dátové body sú zložené z kladných celočíselných hodnôt. Hodnoty sa pohybujú v rozsahu 1–7000, čo reprezentuje najvýraznejšie zmeny zo všetkých vybraných dátových sád.
- Názov: Pen-Based Recognition of Handwritten Digits, počet záznamov: 6870, počet dimenzií: 16, počet odľahlých hodnôt: 156 (2.27%). Dátové body v tejto sade sú zložené z kladných desatinných čísel pohybujúcich sa v rozsahu 0–1.
- Názov: Speech, počet záznamov: 3686, počet dimenzií: 400, počet odľahlých hodnôt: 61 (1.65%). Tento dátový set bol zaznamenaný skupinou spracovania reči na VUT v Brne, čo je hlavný dôvod jeho zvolenia. Jednotlivé hodnoty dátových bodov sú reprezentované reálnymi číslami.

V súboroch sú dáta uložené v textovom formáte, kde každý riadok predstavuje jeden dátový bod. Každý dátový bod obsahuje niekoľko hodnôt, ktoré reprezentujú jednotlivé atribúty dátového bodu. Tieto atribúty sú od seba oddelené čiarkou. Dáta zo súboru s koncovkou *.csv*, prípadne *.data* sú čítané pomocou funkcie `read_csv()` z knižnice pandas[8]. Tieto dáta sú načítané do dátovej štruktúry `DataFrame`. V prípade dát s koncovkou *.mat* sú dáta načítané pomocou funkcie `loadmat()` z knižnice scipy.io[15]. V tomto prípade dáta obsahujú aj vyhodnotenie, či je daný dátový bod odľahlá hodnota alebo nie. Teda hodnota 1 označuje odľahlý bod, naopak hodnota 0 značí správny bod.

3.2 Generovanie dát

Do projektu je zahrnutá možnosť generovania dát pomocou funkcie `generate_data()` z knižnice pyOD[16]. Táto možnosť je využitá v tom prípade, kedy sú pre experimenty vyžadované dáta o presne stanovených dimenziách. Generovanie dát taktiež umožní vytvorenie sekvencie dátových sád s postupne sa zväčšujúcim počtom dimenzií pre lepšie vytváranie grafov a tabuliek pri experimentovaní. Dáta sú generované pomocou viacrozmerného Gaussoveho rozloženia, odľahlé hodnoty sú do dát pridané pomocou rovnomerného rozloženia. Funkcia `generate_data_clusters()` generuje dáta, ktoré sa rozdelia do viacerých skupín. Takéto dáta modelujú problém nízkej hustoty dát a taktiež globálne odľahlé hodnoty. Pri generovaní dát je možné meniť počet vygenerovaných dátových bodov a ich dimenzionalitu reprezentovanú počtom vlastností. Počet odľahlých hodnôt vo vygenerovaných dátach je určený ako percentuálny zlomok z celkového počtu dátových bodov. Dáta je naďalej možné rozdeliť do ľubovoľného počtu zhlukov.



Obr. 3.1: Príklad zobrazujúci vygenerované dáta pomocou funkcie `generate_data()` z knižnice pyOD[16].

Kapitola 4

Návrh aplikácie

Táto kapitola popisuje návrh aplikácie pre analýzu metód detekcie odľahlých hodnôt. Hlavným cieľom aplikácie je porovnanie jednotlivých metód detekcie medzi sebou. Výsledky aplikácie sú preto hlavne zamerané na štatistiky využitých modelov tak, aby boli ľahko porovnateľné. Samotné ohodnotenie odľahlých hodnôt je možné uložiť, ale jedná sa len o vedľajší produkt aplikácie. Sekcia 4.1 obsahuje analýzu požiadaviek pomocou neformálnej špecifikácie a popisuje roľu užívateľa. V Sekcii 4.2 je zobrazený návrh architektúry aplikácie. Tá popisuje jednotlivé moduly a ich činnosti. Priebeh programu je priblížený v sekcii 4.3.

4.1 Analýza požiadaviek

Táto sekcia obsahuje analýzu požiadaviek na výslednú aplikáciu. Na začiatok je uvedená neformálna špecifikácia. Nato je popísaná rola používateľa spolu s jeho prípadom použitia.

4.1.1 Neformálna špecifikácia

Cieľom aplikácie je umožniť užívateľovi experimentovať s vybranými metódami pre detekciu odľahlých hodnôt. Taktiež umožňuje analyzovať ich vlastnosti a metódy navzájom porovnať. Vstupom aplikácie sú informácie o metódach detekcie, ako ich identifikácia a využité parametre. Na vstupe aplikácie môže byť viac metód, ktoré budú analyzované v jednom behu. Ďalej na vstupe očakávame dátovú sadu, v ktorej majú byť odľahlé hodnoty vyhľadané. Vstupné dáta sú uložené v súbore vo forme textu, kde každý nový riadok predstavuje jeden záznam. Užívateľ môže využiť generátor dát, kde je potrebné uviesť požadované parametre. Testy môžu obsahovať viac ako jednu dátovú sadu, kde sa každá uvedená metóda detekcie aplikuje na každú dátovú sadu. Posledná vstupná informácia, je špecifikácia formy výstupu. Tu je určené, ktoré štatistiky sa budú monitorovať a ako sa zobrazia, prípadne uložia. Výstup aplikácie sú tabuľky hodnôt, alebo grafy s výsledkami testov.

Z dôvodu rozsiahleho požadovaného množstva informácií pre beh programu je vstup programu zvolený ako konfiguračný súbor. Tento súbor obsahuje vyššie spomenuté informácie vo formáte JSON.

Ďalšou možnosťou je spustenie grafickej verzie programu pre vytváranie konfiguračných súborov. Po spustení grafickej verzie bude užívateľom ponúknutý zoznam implementovaných metód, z ktorých je možné si vyberať. Po vybratí metódy sa užívateľovi zobrazia parametre, ktoré daná metóda využíva. Ďalej sa tu nachádza možnosť pridať súbor s dátovými sadami, ktoré sa ukladajú do zoznamu. Po zvolení metód a dát je potrebné upresniť výstup programu.Zo zobrazenej ponuky užívateľ určí aké informácie majú byť sledované,

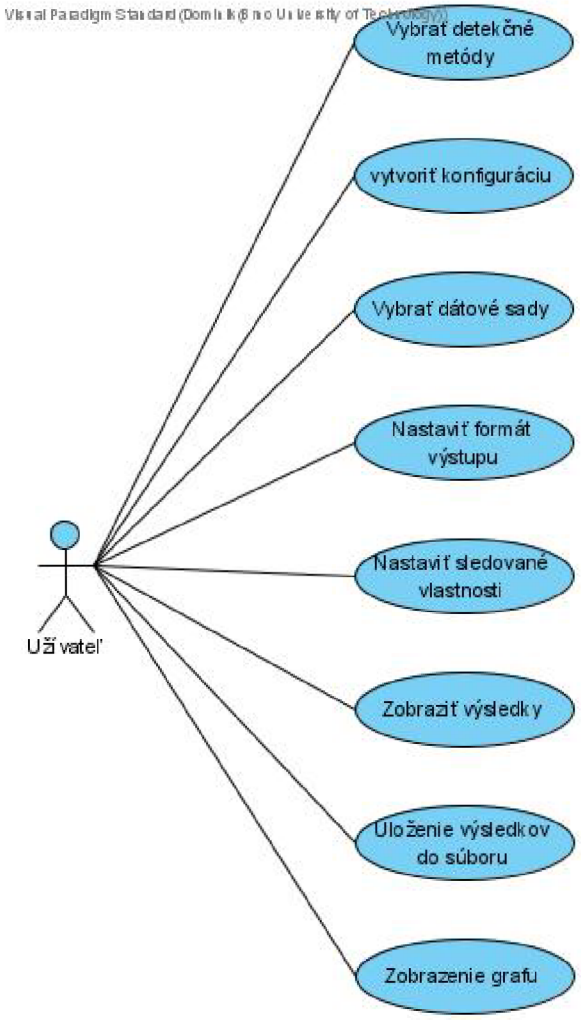
ako rýchlosť alebo presnosť jednotlivých metód. Na koniec užívateľ zadá názov súboru a dá príkaz k jeho uloženiu. Aplikácia taktiež umožňuje zobraziť existujúci konfiguračný súbor pre jeho úpravu.

4.1.2 Rola používateľa

Pri práci s aplikáciou existuje len jedna roľa, ktorou je užívateľ. Aplikácia užívateľovi ponúka pre prácu tri základné úkony:

- Spustiť analýzu na základe konfiguračného súboru.
- Vytvoriť nový konfiguračný súbor pre analýzu.
- Zobraziť a upravovať existujúce konfiguračné súbory.

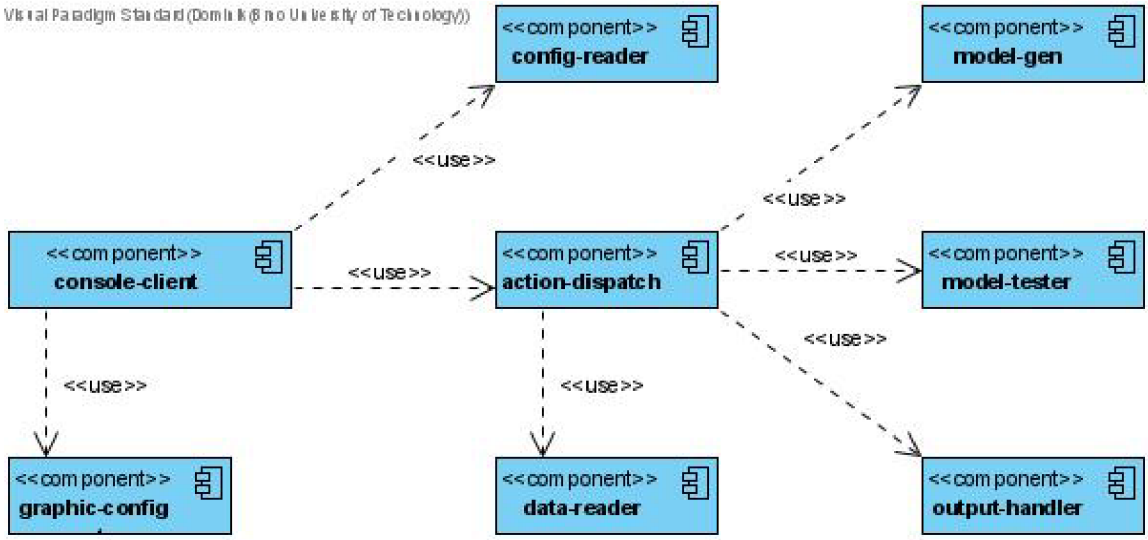
Tieto body sú ďalej rozdelené na dielčie činnosti a zobrazené na diagrame prípadov užitia 4.1.



Obr. 4.1: Diagram prípadu použitia

4.2 Architektúra modulov

Táto sekcia popisuje architektúru programu. Diagram zobrazený na obrázku 4.2 zobrazuje rozdelenie do niekoľkých modulov, ktoré sa podieľajú na behu programu. V následujúcej časti budú jednotlivé komponenty popísané.



Obr. 4.2: Diagram komponent

4.2.1 Console-client

Console-client je prvý modul, s ktorým príde užívateľ do kontaktu. Tento modul sa stará o spustenie a ukončenie samotného programu, spracúva argumenty a rozhoduje o spôsobe spustenia. Tento modul využíva tri ďalšie moduly *graphic-configuration*, *config-reader* a *action-dispatcher*. Ak na základe argumentov užívateľ uviedol, že chce prejsť do grafického režimu je volaný modul *graphic-configuration*. Ak je potrebné otvorenie alebo uloženie konfiguračného súboru je využitý modul *config-reader*. V prípade, že užívateľ chce vykonať analýzu, tento modul predá štruktúru s detailami z konfiguračného súboru modulu *action-dispatcher*. Ak sa pri behu programu vyskytne chyba, modul *console-client* informuje užívateľa o typu chyby. Následne pokračuje v práci tak, že užívateľ môže spúšťať ďalšie testy. Program je ukončený ak užívateľ dá pokyn k zavretiu okna.

4.2.2 Graphic-configuration

Tento modul je spolu s modulom *console-client* jediný, ktorý komunikuje so samotným užívateľom. Predstavuje grafické rozhranie určené k zobrazeniu a tvorby konfiguračných súborov. Na základe komunikácie s užívateľom vytvorí dátovú štruktúru reprezentujúcu konfiguračný súbor, ktorý po dokončení práce predáva spätne modulu *console-client*. Tento modul môže vyžadovať spoluprácu s modulom *config-reader* v prípade, že užívateľ chce zobraziť a upravovať už existujúci konfiguračný súbor. Po uložení konfiguračného súboru prejde do východzieho stavu, kde užívateľ môže pokračovať v úprave alebo vytváraní nových konfiguračných súborov.

4.2.3 Config-reader

Modul *config-reader* je zodpovedný za prácu s konfiguračnými súbormi. Ten konfiguračné súbory na základe cesty otvorí, prečíta a vytvorí dátovú štruktúru, s ktorou je následne možné pracovať v ostatných moduloch. Taktiež je tento modul zodpovedný za prevod medzi dátovou štruktúrou, s ktorou pracuje program a formátom JSON. Tento modul je využívaný aj v prípade ukladania konfiguračných súborov.

4.2.4 Action-dispatcher

Tento modul riadi priebeh samotnej analýzy. Na vstupe obdrží štruktúru s informáciami z konfiguračného súboru. Jeho úlohou je správne rozčleniť tieto informácie a predať ich správnym modulom. Taktiež zabezpečuje zber výsledkov analýzy pre ich finálne spracovanie. Informácie o metódach detekcie postupne využíva na vytvorenie modelov pomocou modulu *model-gen*. Pre získanie dátovej sady využíva modul *data-reader*. Po vytvorení modelov a načítaní dátových sád príde na radu samotná analýza metód, ktorú zabezpečuje *model-tester*. Modul *model-tester* taktiež zabezpečuje samotnú detekciu odľahlých hodnôt. Výsledky analýzy sú následne predané modulu *output-handler* pre ich odovzdanie užívateľovi. Po skončení analýzy predá riadenie späť modulu *console-client*.

4.2.5 Model-gen

Vytvorenie modelov zabezpečuje modul *model-gen* s pomocou knižníc popísaných v kapitole 5. Na vstupe obdrží informácie o metóde ako je identifikácia pomocou názvu a parametre, ktoré funkcie využívajú pre ich vytvorenie. Modul taktiež sleduje čas potrebný na vytvorenie modelu pre výslednú analýzu.

4.2.6 Data-reader

Dátové sady pre analýzu sú načítané pomocou modulu *data-reader*. Ten na základe názvu načíta dáta zo súboru. Dátové sady sú uložené v priečinku *data* pre ľahší prístup najmä v grafickom režime. Modul následne označí jednotlivé atribúty. Modul je taktiež zodpovedný za informácie ako počet dátových bodov a atribútov. Na záver získané dáta spolu s ich štatistikami vráti modulu *action-dispatch*.

4.2.7 Model-tester

Vytvorené modely sú testované na dátových sadách pomocou modulu *model-tester*. Ten na vstupe obdrží model a dátovú sadu určené k analýze. Na vstupe sa taktiež očakávajú informácie o typu požadovanej analýzy. Na výstupe sa nachádzajú informácie o čase trénovania a detekcie ako aj pole s výsledným ohodnotením odľahlých hodnôt. Tento modul tiež obsahuje funkcie k výpočtu úspešnosti jednotlivých modelov, ktoré úspešnosť hodnotia pomocou predošlého výsledku modulu, ktorý je porovnaný s pravdivostnou tabuľkou.

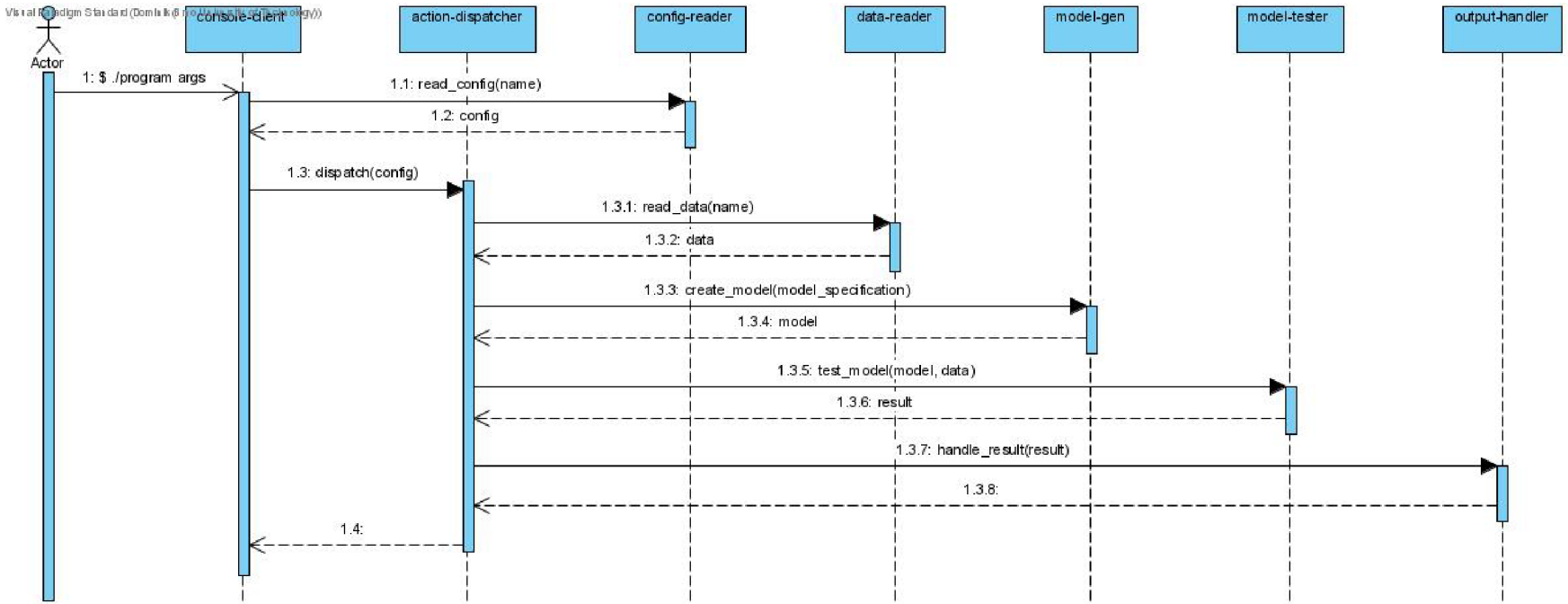
4.2.8 Output-handler

Modul *output-handler* zabezpečuje zobrazovanie a ukladanie výsledkov analýzy. Výsledky sú ukladané podľa názvu konfiguračného súboru. Výsledky taktiež obsahujú informácie o využitých modeloch, dátových sadách ako aj informácie o dátume a čase ich vykonania.

4.3 Priebeh analýzy v programe

Postup vykonania analýzy je popísaný pomocou diagramu sekvencie 4.3. V tomto diagrame popisujeme spustenie programu s jedným konfiguračným súborom. Analýza prebieha postupne podľa nasledujúceho zoznamu krokov:

1. Užívateľ spustí program s názvom konfiguračného súboru v argumente.
2. Modul console-client volá config-reader s názvom konfiguračného súboru.
3. Modul config-reader otvorí konfiguračný súbor a načíta jeho obsah.
4. Modul config-reader prevedie JSON štruktúru na internú štruktúru programu a vráti ju modulu console-client.
5. Moful console-client volá modul action-dispatcher a predá mu štruktúru s podrobnosťami testu.
6. Action-dispatcher volá modul data-reader a predá mu názov súboru s dátami.
7. Data-reader otvorí a načíta zo súboru dátovú sadu a predá ju modulu action-dispatcher pre spracovanie.
8. Action-dispatcher spracuje informácie prevzaté od data-reader modulu a následne volá modul model-gen, ktorému predá špecifikáciu o požadovanom modeli.
9. Model-gen podľa názvu zvolí funkciu, pomocou ktorej vytvorí model a predá ho modulu action-dispatcher.
10. Action-dispatcher spracuje informácie z model-gen modulu a volá modul model-tester.
11. Model-tester obdrží vytvorený model a dáta, podľa ktorých model natrénuje a identifikuje odľahlé hodnoty v dátach. Modul sleduje čas a pamäť využitú testovaným modelom. Následne modulu action-dispatcher predá zaznamenané hodnoty a skóre jednotlivých dátových bodov.
12. Modul action dispatcher predá výsledky modulu output handler.
13. Output-handler vytvorí dátovú štruktúru obsahujúcu výsledky experimentov. Táto štruktúra je následne využitá pri zobrazení výsledkov užívateľovi.
14. Model action-dipatcher skontroluje, či boli všetky testy vykonané. Následne vracia kontrolu modulu console-client.
15. Console-client skontroluje vykonanie všetkých konfiguračných súborov a ukončí beh programu.



Obr. 4.3: Diagram sekvencie

Kapitola 5

Použité technológie

Táto kapitola popisuje programovacie jazyky, nástroje a knižnice využité pri vytváraní programu. Táto kapitola je rozdelená do dvoch sekcií. V sekcii 5.1 sú popísané technológie využité pre vytvorenie programu ako programovací jazyk, nástroj pre vytvorenie grafického užívateľského rozhrania a knižnice pre prácu s viac dimenzionálnymi dátami. Následne v sekcii 5.2 sú uvedené knižnice, ktoré zabezpečujú implementáciu modelov pre detekciu odľahlých hodnôt z kapitoly 2.

5.1 Technológie pre vytváranie aplikácie

Pre implementáciu aplikácie bol zvolený programovací jazyk python[1]. Grafické užívateľské rozhranie je vytvorené pomocou sady nástrojov Tkinter[2]. Pre prácu s dátovými štruktúrami sú využité knižnice numpy[3] a pandas[4]. Zobrazovanie grafov je riešené pomocou rozhrania pyplot z knižnice matplotlib[5]. V tejto sekcii sa nachádza ich krátky popis. Pre zobrazovanie výsledkov bola využitá knižnica pandastable[6].

5.1.1 Python

Jedná sa o dynamicky typovaný, interpretovaný programovací jazyk. Jeho tvorcom je Guido van Rossum. Podporuje procedurálne, objektovo orientované a funkcionálne programovanie. Je vhodný pre vytváranie skriptov a je využívaný v obore dátovej vedy. Veľkou výhodou jazyka python je predovšetkým veľké množstvo dostupných knižníc. Okrem dostupných knižníc je jedným z hlavných dôvodov pre jeho využitie v tejto práci to, že uľahčuje prácu so zložitými dátovými štruktúrami.

5.1.2 Tkinter

Sada nástrojov Tkinter je štandardná knižnica pre tvorbu grafického užívateľského rozhrania v jazyku python. Jedná sa o objektovo orientovanú vrstvu nad jazykom Tcl, ktorý slúži pre ovládanie knižnice Tk. Tkinter je najrozšírenejšou a najviac používanou knižnicou pre

[1] https://www.python.org
[2] https://wiki.python.org/moin/TkInter
[3] https://numpy.org
[4] https://pandas.pydata.org
[5] https://matplotlib.org
[6] https://github.com/dmnfarrell/pandastable

tvorbu GUI medzi programátormi jazyka pytohn. Knižnica ponúka sadu vizuálnych elementov ako sú okná, tlačidlá, dialógy a iné. Hlavnou výhodou je jeho ľahké použitie a možnosť rozšírení vo forme widgetov.

5.1.3 Numpy

NumPy je knižnica pre jazyk python používaná na prácu s n-dimenzionálnymi poľami. Predstavuje jednu zo základných knižníc v oblasti vedy a analytiky. Jeho výhodou je ľahké použitie, prepracovaná dokumentácia ako aj množstvo dostupných návodov a príkladov. Táto knižnica je taktiež využívaná v knižniciach pandas, scipy a matplotlib, ktoré sú v práci taktiež využívané.

5.1.4 Pandas

Jedná sa o knižnicu pre jazyk python určenú k analýze a manipulácii s dátami. Táto knižnica sa v práci využíva na čítanie dátových sád a preto je možné povedať že je jednou zo základných kameňov celej práce. Hlavným dôvodom jej využitia je objekt `DataFrame`, ktorý sa využíva na reprezentáciu dát. Knižnica pandas taktiež ponúka prepracované indexovanie a delenie dátových štruktúr. Taktiež ponúka sadu funkcií nad dátovou štruktúrou `DataFrame`. Jednou z funkcií je aj funkcia `quantile()`, ktorá vypočíta požadovaný kvantil. Táto funkcia je využitá pri výpočte IQR.

5.1.5 Matplotlib

Matplotlib je rozšírenou knižnicou pre vizualizáciu v jazyku python. Poskytuje objektovo orientované API pre vkladanie grafov do aplikácií. Je taktiež odporúčaným nástrojom pri práci s nástrojmi pre tvorbu grafických rozhraní ako je napríklad Tkinter.

5.1.6 Scipy

Knižnica scipy je využívaným nástrojom pre vedecké výpočty. Jedná sa o nadstavbu využívajúcu dátovú štruktúru poskytovanú knižnicou numpy. Ponúka sadu algoritmov a funkcií pre vedecké výpočty. Tie sú dostupné v balíkoch ako sú stats a io. Balík stats predstavuje knižnicu štatistických funkcií. Medzi tie patrí aj výpočet z-skóra, ktorý je využitý v tejto práci. Balík io ponúka funkcie pre čítanie a zapisovanie dát do súborov s rôznymi formátmi. Tento balík je v práci využitý pre čítanie dát zo súborov vo formáte .mat, kde je využitá funkcia `loadmat()`.

5.1.7 Pandastable

Knižnica pandastable dopĺňa knižnicu Tkinter o možnosť zobrazenia interaktívnej tabuľky. K tomu využíva triedu `DataFrame` z knižnice pandas. Interaktívna tabuľka ponúka užívateľovi možnosti ako odstraňovanie riadkov a stĺpcov, jednoduché označovanie buniek pomocou myši, kopírovanie a ukladanie obsahu tabuľky, ako aj jeho formátovanie. Taktiež je možné dáta triediť pomocou hodnôt v stĺpcoch a riadkoch, tabuľku ukladať ako aj importovať a exportovať jej textovú podobu. Jednou z vhodných možností je aj vykresľovanie interaktívnych grafov.

5.2 Technológie pre vytváranie modelov

V tejto sekcii sa nachádza popis knižníc, ktoré sa zaoberajú implementáciou metód pre detekciu odľahlých hodnôt z kapitoly 2. Tieto knižnice sú najviac využívané modulom *model-tester* k testovaniu jednotlivých modelov pre detekciu. Jednou z predných knižníc venujúcim sa detekcii odľahlých hodnôt sú knižnice pyOD[7] alebo Pyodds[8].

5.2.1 Sklearn

Knižnica sklearn[9] je jendným zo základných nástrojov pre analýzu dát. Obsahuje rôzne algoritmy pre zhlukovanie, klasifikáciu a regresiu dát. Je určený k spolupráci s knižnicami ako sú numpy, scipy alebo matplotlib. Táto knižnica ponúka štyri základné spôsoby pre detekciu odľahlých hodnôt. Tými sú menovite *EllipticEnvelope*, *LocalOutlierFactor*, *OneClassSVM* a *IsolationForest*.

5.2.2 PyOD

PyOD je sada nástrojov v jazyku python určená k detekcii odľahlých hodnôt vo viacdimenzionálnych dátach. Knižnica pyOD ponúka viac ako 30 algoritmov pre detekciu odľahlých hodnôt[10]. Taktiež ponúka sadu doplnkových funkcií ako sú funkcie pre generovanie dát, vyhodnocovanie úspešnosti daných metód a zobrazovaniu výsledkov.

5.2.3 PyNomaly

Knižnica PyNomaly[11] implementuje metódu `LocalOutlierProbability`, ktorá vypočítava faktor lokálnych odľahlých hodnôt. Táto metóda je založená na hustote rozloženia dátových bodov v dátovej sade.

5.2.4 Pyodds

Knižnica pyodds[7] predstavuje systém pre detekciu odľahlých hodnôt. Táto knižnica ponúka viac ako 10 rôznych metód pre detekciu odľahlých hodnôt. Na rozdiel od ostatných knižníc je forma vytvárania modelov implementovaná pomocou funkcie `algorithm_selection()`, ktorá rozhoduje na základe uvedeného tokenu reprezentujúci názov algoritmu. Knižnica ponúka na výber z nasledovného zoznamu tokenov: *iforest, lof, ocsvm, robustcovariance, staticautoencoder, luminol, cblof, knn, hbos, sod, pca, dagmm, autoencoder, lstm_ad, lstm_ed*.

5.2.5 Isotree

Isotree[12] predstavuje viacvláknovú implementáciu metódy pomocou rozšírených izolačných lesov. Táto knižnica užívateľom sľubuje vysokú rýchlosť detekcie odľahlých hodnôt a preto je aj v práci využitá.

[7] https://pyod.readthedocs.io/en/latest/
[8] http://pyodds.com/index.html
[9] https://scikit-learn.org/stable/
[10] https://pyod.readthedocs.io/en/latest/
[11] https://github.com/vc1492a/PyNomaly
[12] https://github.com/david-cortes/isotree

5.2.6 Zoznam dostupných metód

Táto sekcia uvádza zoznam metód, ktoré boli vybrané z dostupných metód vo vyššie spomenutých knižniciach. Taktiež je zrejmé, že použitím rôznych knižníc sa v metódach niektoré metódy vyskytnú duplicitne. V zozname metód sú metódy rozdelené do skupín podľa druhov uvedených v kapitole 2. Tento zoznam taktiež predstavuje metódy, ktoré sú dostupné v programe implementovanom v kapitole 6. Z dôvodu, že niektoré metódy sú implementované duplicitne, budú v programe rozdelené podľa jednotlivých knižníc.

1. Pravdepodobnostné modely:
   - ABOD (Angle-base Outlier Detection) - model implementovaný knižnicou pyOD
   - COPOD (Copula Based Outlier Detector) - model implementovaný knižnicou pyOD
   - SOS (Stochastic Outlier Selection) - model implementovaný knižnicou pyOD
   - zscore - výpočet z-skóre implementovaný knižnicou scipy
2. Lineárne modely:
   - PCA (Principal Component Analysis) - model implementovaný knižnicou pyOD a pyodds
   - MCD (Minimum Covariance Determinant) - model implementovaný knižnicou pyOD
   - OCSVM (One-Class Support Vector Machines) - model implementovaný knižnicou pyOD, pyodds a sklearn
   - LMDD (Linear Model Deviation-base outlier detection) - model implementovaný knižnicou pyOD
   - robustcovariance - model implementovaný knižnicou pyodds
   - EllipticEnvelope - model implementovaný knižnicou sklearn
3. Modely založené na vzdialenosti:
   - CBLOF (Clustering Based Local Outlier Factor) - model implementovaný knižnicou pyOD
   - COF (Connectivity-Based Outlier Factor) - model implementovaný knižnicou pyOD
   - HBOS (Histogram-based Outlier Detection) - model implementovaný knižnicou pyOD a pyodds
   - KNN (k-Nearest Neighbors Detector) - model implementovaný knižnicou pyOD a pyodds
   - LOF (Local Outlier Factor) - model implementovaný knižnicou pyOD, pyodds, sklearn a pyNomaly
   - DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - metóda zhlukovania z knižnice sklearn
   - OPTICS (Ordering Points To Identify the Clustering Structure) - metóda zhlukovania z knižnice sklearn

4. Modely založené na neurónových sietiach:

    - AutoEncoder - model implementovaný knižnicou pyOD a pyodds
    - VAE (Variational Auto Encoder) - model implementovaný knižnicou pyOD
    - MO_GAAL (Multiple-Objective Generative Adversarial Active Learning) - model implementovaný knižnicou pyOD
    - SO_GAAL (Single-Objective GAAL) - model implementovaný knižnicou pyOD
    - dagmm (Deep Autoencoding Gaussian Mixture Model) - model implementovaný knižnicou pyodds
    - lstm_ad (Long Short Term Memory Anomaly Detection) - model implementovaný knižnicou pyodds
    - lstm_ed (LSTM Encoder-Decoder) - model implementovaný knižnicou pyodds

5. Kombinačné modely:

    - IForest - model implementovaný knižnicou pyOD, sklearn, outliertree a isotree
    - FeatureBagging - model implementovaný knižnicou pyOD
    - LODA (Lightweight on-line detector of anomalies) - model implementovaný knižnicou pyOD
    - Combination - sada nástrojov z knižnice pyOD, ktoré slúžia na spojenie skóre z výstupov rôznych modelov

Kapitola 6

Implementácia

Táto kapitola je zameraná na implementáciu programu určeného pre analýzu a experimentovanie. Kapitola 4 popísala architektúru hlavných komponent programu, táto kapitola ju rozširuje a ponúka viac technických detailov. Kapitola je rozdelená do úsekov, popisujúcich rôzne časti programu, prípadne detailnejšie popisuje zaujímavé časti riešenia a funkcionality. Sekcia 6.1 sa venuje komunikácii medzi užívateľom a programom, ktoré prebieha pomocou grafického užívateľského rozhrania, ponúka informácie o jeho dizajne a ovládaní programu. Sekcia 6.2 sa venuje logickej časti, ktorá vykonáva detekciu odľahlých hodnôt a analýzu modelov, ktoré sú k detekcii využité.

Program je napísaný v jazyku Python, pričom sú využité knižnice z kapitoly 5. Pre implementáciu bolo využité integrované vývojové prostredie PyCharm, vyvíjané spoločnosťou JetBrains.

6.1 Grafické užívateľské rozhranie

Sekcia grafické užívateľské rozhranie je rozdelená do štyroch pod-sekcií. Tieto sekcie sa venujú štyrom krokom pri práci s programom, presnejšie sú to časti pre prácu s konfiguračnými súbormi, vytváranie zoznamov pre modely k experimentu, pre dátové sady určené k experimentu a zobrazenie výsledkov experimentu. Sekcia 6.1.1 sa venuje vytváraniu, ukladaniu a načítaniu konfiguračných súborov, ako aj spúšťaniu samotných experimentov. Nasledujúca sekcia 6.1.2 ponúka možnosť vytvorenia zoznamu modelov a ich úprave. Výberu dátových sád určených k experimentom sa venuje sekcia 6.1.3. V sekcii 6.1.4 sa nachádza popis okna pre zobrazenie výsledkov experimentov a ich ukladaniu.

Grafické užívateľské rozhranie je implementované pomocou knižnice tkinter. Táto knižnica využíva objektovo orientované programovanie pre vytvorenie výsledného užívateľského rozhrania. Vďaka tomu je možné, že každá časť rozhrania je reprezentovaná objektom. Implementácia grafického užívateľského rozhrania sa nachádza v skripte *graphic.py*. Hlavné okno aplikácie je následne reprezentované pomocou triedy `TkinterApp`. Toto okno je následne rozdelené pomocou rámcov z knižnice tkinter na hornú a spodnú časť. Rámce sú vytvorené pomocou objektov typu `Frame`. Rámec vo vrchnej časti má vždy zobrazený rovnaký obsah, ktorý obsahuje štyri tlačidlá. Pomocou týchto tlačidiel si používateľ môže zobraziť požadovaný obsah, ako je možné vidieť na obrázku C.1. Tlačidlá sú implementované pomocou objektu `ttk.Button()`. Pri vytváraní okna sú vytvorené inštancie tried `StartPage, MethodPage, DataPage, ResultPage`. Tieto objekty reprezentujú obsah spodného rámca,

zobrazujúceho obsah aktuálne vybratej stránky. Pohyb medzi stránkami je implementovaný pomocou metódy `show_frame()`.

6.1.1 Práca s konfiguračnými súbormi

Ako aj názov napovedá trieda `StartPage` predstavuje obsah úvodnej obrazovky. Trieda vytvára rámec, ktorý je zobrazený v hlavnom okne pri každom spustení programu. Dizajn úvodnej stránky je zobrazený na obrázku C.1. Táto stránka slúži pre načítanie, vytváranie a ukladanie konfiguračných súborov, ako aj k ich spúšťaniu. Informácie o konkrétnom konfiguračnom súbore sú uložené v štruktúre typu dictionary, ktorá je uložená v premennej `configuration`. Vytváranie nového konfiguračného súboru je implementované metódou `new()`. Ukladanie konfiguračného súboru je zaistené metódou `save()`, ktorá využíva funkciu `json.dump()`. Načítanie konfiguračného súboru je zaistené metódou `load()` a funkciou `json.load()`. Pre spustenie experimentu z konfiguračného súboru využíva funkciu `action_dispatch.action_sequence()`. Táto funkcia je spúšťaná v novom vlákne pomocou funkcie *threading.Thread().start()* preto, aby nedochádzalo k zamrznutiu okna. Parametre získané od používateľa sú ukladané v objektoch typu *tk.StringVar()*.

6.1.2 Zostavenie zoznamu modelov

Stránka pre vytváraní modelov pre detekciu odľahlých hodnôt je implementovaná pomocou triedy *MethodPage*. Príklad stránky je zobrazený na obrázku C.2.

Táto stránka zobrazuje zoznam metód a polí ich parametrov. Navigácia medzi dostupnými metódami prebieha pomocou výberu knižnice funkciou `change_library()` a výberu metódy pomocou funkcie `change_method()`. Nová metóda je vytvorená pomocou funkcie `create_method()`. Ukladanie metód je zaistené funkciami `save_new()` a `save()`. Zoznam metód je uložený v premennej `configuration`.

Každá metóda vyžaduje zobrazenie rôznych tlačidiel a textových okien. Zobrazenie týchto informácií je riešené v skripte *methods_buttons.py*. Tento skript implementuje funkciu `add_buttons()`, ktorá vytvorí potrebné objekty. Pre vytváranie tlačidiel implementuje a využíva funkcie `add_button()` a `add_choices_button()`. Vytváranie informácií potrebných pre uloženie modelu je riešené v skripte *methods.py*, kde sa nachádza zoznam knižníc a nimi implementovaných metód. Taktiež je tu implementovaná funkcia `new_method()`, ktorá vráti vytvorenú štruktúru požadovanej metódy. Vytvorená štruktúra obsahuje prednastavené štandardné hodnoty.

6.1.3 Zostavenie zoznamu dátových sád

DataPage je stránka, ktorá slúži na výber dátových sád pre experimenty. Je tu taktiež umiestnený generátor dátových sád. Ukážka tejto stránky je zobrazená na obrázku C.3.

Táto trieda implementuje metódy `add()` a `remove()`, ktoré slúžia pre pridávanie a odstraňovanie dátových sád do vytvoreného zoznamu. Taktiež implementuje metódu `refresh()`, ktorá znovu vykreslí okno a abecedne zoradí jednotlivé dátové sety.

Implementácia generovania dát sa nachádza v skripte *data_generator.py*, kde je implementovaná funkcia `generate_data()`. Táto funkcia využíva dve metódy z knižnice *pyod* a to funkciu `generate_data()` pri generovaní dát s jedným zhlukom. V prípade, že používateľ zadá viac ako jeden zhluk je využíva funkcia `generate_data_clusters()`. Vygenerované dáta sú ukladané pomocou funkcie `to_csv()`.

6.1.4 Zobrazenie výsledkov

Po úspešnom prebehnutí experimentu je možné výsledky zobraziť vo forme tabuľky v aplikácii na stránke *ResultPage*. Tabuľka s výsledkami sa nachádza v pravej časti stránky ako je možné vidieť aj na obrázku C.4.

Tabuľka s výsledkami je reprezentovaná ako objekt `Table` z knižnice pandastable. Vykreslenie tabuľky prebieha pomocou metódy `show()`, prípadne jej aktualizácia pomocou metódy `show_redraw()`. Obsah tabuľky je reprezentovaný pomocou objektu `DataFrame` z knižnice pandas. Pre ukladanie výsledkov do súboru sa využíva skript *output_handler.py*. Táto knižnica implementuje funkcie pre zápis do súboru. Zápis prebieha funkciami `to_excel()`, `to_csv()` a `to_latex()`. Implementuje funkciu `output_print_labels()`, ktorá z tabuľky vyberá správne stĺpce pre uloženie. Taktiež implementuje funkciu `new_keys()`, ktorá slúži k úprave názvu dátových sád.

6.2 Backend

Väčšina backend logiky je tvorená systémom pre realizáciu experimentu. Táto logika je popísaná v sekcii 6.2.1. Následne v sekcii 6.2.2 je popísaný spôsob ukladania konfigurácie a konfiguračného súboru. Sekcia 6.2.3 popisuje spôsob načítania a spracovania dát, ktoré program využíva.

6.2.1 Testovanie modelov

Testovanie modelov začína v okamihu stlačenia tlačidla *Start*. Objekt typu `Button`, ktorý odpovedá tomuto tlačidlu je naviazaný na metódu `start()` nachádzajúcej sa v triede `StartPage`. Priebeh experimentu je riadený funkciou `action_sequence()`, ktorá sa nachádza v skripte *action_dispatch.py*.

V prvom kroku sú načítané dáta pomocou funkcie `load_data()`. Aplikácie následne pomocou načítaných dát vypočíta ako veľkú časť dát tvoria odľahlé hodnoty a následne vytvára výslednú tabuľku, do ktorej uloží informácie o počte dátových bodov, počte dimenzií a odľahlých hodnôt. Následne pomocou cyklu program pre každú zvolenú metódu detekcie, ktorá je určená k testovaniu, volá funkciu `generate_model()`, ktorá vytvorí model pre danú metódu. Následne sú na každom vytvorenom modeli testované všetky dátové sady. Detekcia odľahlých hodnôt prebieha pomocou funkcie `test_model()`. Po vykonaní detekcie sú z návratových hodnôt vypočítané hodnoty pre krivku ROC pomocou funkcie `roc_auc_score()` a presnosť pomocou funkcie `precision_score()`. Výsledky sú uložené do výslednej tabuľky, ktorá je na záver uložená do premennej `result`. Táto premenná je následne využitá pri zobrazení finálnych výsledkov.

Implementácia funkcie `generate_model()` sa nachádza v skripte *model_gen.py*. Táto funkcia sa stará o to, aby boli volané správne metódy z knižníc, ktoré dané modely implementujú. Taktiež využíva dáta z konfigurácie k tomu, aby metódam predala požadované parametre. Funkcia ako návratovú hodnotu vracia vytvorený model.

Funkcia `test_model()` zo skriptu *model_tester.py* zodpovedá za spustenie detekcie odľahlých hodnôt a zaznamenávaní využitých zdrojov. Na vstupe očakáva model vytvorený funkciou `generate_model()` a jednu sadu dát. Funkcia pripraví dáta pre trénovanie a následne volá funkciu `fit()` nad obdržaním modelom. Následne vykoná predikciu využitím funkcie `predict()`, ktorá vracia pravdivostné hodnoty a funkcie `decision_function()`, ktorá vypočíta skóre jednotlivých dátových bodov. V prípade, že sa jedná o model, ktorý

neimplementuje funkciu `decision_function()` je výsledné skóre nulové. Funkcia taktiež zaznamenáva údaje o využitom čase pomocou funkcie `time()` z knižnice time. Informácie o využitej pamäti sú získané pomocou funkcie `get_traced_memory()` z knižnice tracemalloc.

6.2.2 Konfigurácia

Súbory s konfiguráciami experimentov sú načítané a ukladané pomocou metód `load()` a `save()` z triedy `StartPage`. Konfiguračné súbory sú uložené vo formáte JSON. Pre samotné ukladanie a načítanie sa využíva knižnica JSON, z ktorej sú využité funkcie `json.load()` a `json.dump()`. V programe je konfigurácia uložená pomocou premennej `configuration`. Premenná `configuration` je typu dictionary. Uložený konfiguračný súbor vo formáte JSON vyzerá nasledovne:

```
{
   "name":"Name",
   "trials":"1",
   "train_fraction":"100",
   "methods":[
      {
         "library":"pyod",
         "name":"IForest",
         "save_as":"IForest",
         "n_estimators":"100",
         "contamination":"0.0",
         "max_features":"1.0",
         "n_jobs":"1",
         "verbose":"0"
      },
      {
         "library":"pyod",
         "name":"KNN",
         "save_as":"KNN",
         "contamination":"0.0",
         "n_neighbors":"5",
         "method":"largest",
         "radius":"1.0",
         "leaf_size":"30",
         "n_jobs":"1"
      }
   ],
   "dataSets":[
      "arrhythmia.mat",
      "cardio.mat",
      "glass.mat",
      "mnist.mat",
      "wbc.mat"
   ]
}
```

6.2.3 Práca s dátami

Dátové sady pre experimenty sú načítané pomocou implementácie zo skriptu *data_reader.py*, ktorý obsahuje funkciu `load_data()`. Tu sa využívajú funkcie `loadmat()` a `read_csv()` pre načítanie zo súboru. Dáta sú uložené ako pole implementované knižnicou numpy, ktoré je možné získať pomocou funkcie `to_numpy()`.

Generovanie dát prebieha pomocou funkcie `generate_data()` zo skriptu *data_generator.py* tu sú vygenerované dáta uložené ako objekt `DataFrame` pomocou triedy `pandas.DataFrame()`. K vygenerovaným dátam je pripojené ohodnotenie, ktoré označuje odľahlé hodnoty pomocou funkcie `join()`. Výsledný objekt je uložený do súboru pomocou funkcie `to_csv()`.

Skript *output_handler.py* je zodpovedný za ukladanie výsledku. Implementuje funkcie pre ukladanie výsledkov experimentov do súborov typu *xlsx, tex* a *csv*. K tomu využíva implementované funkciu `cut_index()` pre vytváranie výrezov z premennej *result* obsahujúcej výsledky experimentu.

Kapitola 7

Experimenty

Táto kapitola sa venuje experimentom, pomocou ktorých sú metódy pre detekciu odľahlých hodnôt analyzované. Na úvod je predstavené existujúce vyhodnotenie z knižnice pyOD[16]. Následne v sekcii 7.2 je popísaný spôsob tvorby metód a ich nastavenia. V sekcii 7.3 sú metódy testované na základe rozdelenia v sekcii 5.2.6. Metódy, ktoré dosahujú najlepšie výsledky v rámci svojich skupín, sú následne využité v sekcii 7.4, kde sú medzi sebou porovnané a vyhodnotené. Sekcia 7.6 navrhuje a prezentuje spôsob urýchlenia tvorby modelu pre detekciu odľahlých hodnôt.

Experimenty boli vykonané na operačnom systéme Microsoft Windows 10 Education. Program bol spúšťaný na stolnom počítači s procesorom Intel i5-9400F @ 2.90GHz, dostupná pamäť RAM 16 GB. Implementácia a spúšťanie programu bolo vykonané pomocou programu PyCharm s verziou 2021.1.1.

Pre vyhodnotenie modelov sa využívajú hodnoty *Precision @ rank n* a plochy pod krivkou ROC, ktorých hodnoty sa pohybujú v rozmedzí 0–1. Výsledky taktiež udávajú čas v sekundách, potrebný pre tvorbu modelu a detekciu odľahlých hodnôt s jeho využitím. Ďalej je uvedená pamäť využitá pri priebehu tvorby a detekcie modelu v MB.

Súhrn informácií o dátach využitých pri testovaní modelov, popísaných v kapitole 3, môžeme vidieť v tabuľke 7.1.

Data	#samples	#dimensions	%outliers
arrhythmia	452	274	14,6
cardio	1831	21	9,61
cover	286048	10	0,065
glass	214	9	4,21
mnist	7603	100	2,47
optdigits	5216	64	2,88
pendigits	6870	16	2,27
satellite	6435	36	3,79
shuttle	49097	9	0,37
speech	3686	400	1,65
wbc	378	30	5,56

Tabuľka 7.1: Tabuľka zobrazujúca informácie o dátach určených pre experimentovanie. Zobrazuje názov dátovej sady, počet dátových bodov, počet dimenzií a percentuálnu časť reprezentujúcu odľahlé hodnoty.

7.1 Existujúce riešenia

Dokumentácia knižnice pyOD ponúka referenčné výsledky implementovaných modelov[1]. Výsledky testovania boli vypočítané ako priemer z 10 behov. Dátové sety boli rozdelené do dvoch častí, z ktorých 60% dát bolo využitých na trénovanie a 40% na testovanie modelov. Toto rozdelenie bolo náhodne zvolené pre každý z behov. Referenčné výsledky boli vykonané na rozsiahlej skupine dátových sád. Dátové sady využité v tejto práci, ktoré sa prekrývajú s dátovými sadami využitých v referenčných výsledkov sú: arrhythmia, cardio, glass, mnist, optdigits, pendigits, satellite, shuttle a wbc. Dátové sady cover a speech v referenčných výsledkoch neboli využité.

Data	ABOD	CBLOF	FB	HBOS	Iforest	KNN	LOF	MCD	OCSVM	PCA
arrhythmia	0.3808	0.4539	0.4230	0.5111	0.4961	0.4464	0.4334	0.3995	0.4614	0.4613
cardio	0.2374	0.5876	0.1690	0.4476	0.5041	0.3323	0.1541	0.4317	0.5011	0.6090
glass	0.1702	0.0726	0.1476	0.0000	0.0726	0.0726	0.1476	0.0000	0.1726	0.0726
mnist	0.3555	0.3348	0.3299	0.1188	0.3135	0.4204	0.3343	0.3462	0.3962	0.3846
optdigits	0.0060	0.0000	0.0244	0.2194	0.0301	0.0000	0.0234	0.0000	0.0000	0.0000
pendigits	0.0812	0.2768	0.0658	0.2979	0.3422	0.0984	0.0653	0.0893	0.3287	0.3187
satellite	0.3902	0.4152	0.3902	0.5690	0.5676	0.4994	0.3893	0.6845	0.5346	0.4784
shuttle	0.1977	0.2943	0.0695	0.9551	0.9546	0.2184	0.1424	0.7506	0.9542	0.9501
wbc	0.3060	0.5055	0.5188	0.5817	0.5088	0.4952	0.5188	0.4577	0.5125	0.4767

Tabuľka 7.2: Referenčné výsledky knižnice PyOD zobrazujúce P@n.

Name	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
arrhythmia	0,3125	0,4576	0,4423	0,5000	0,4924	0,4918	0,4500	0,4939	0,3030	0,4242
cardio	0,2139	0,4824	0,1671	0,4716	0,4855	0,3481	0,1879	0,4358	0,0045	0,6080
glass	0,1000	0,1111	0,2056	0,0000	0,1111	0,1111	0,2500	0,0333	0,0000	0,1111
mnist	0,2921	0,4495	0,3739	0,1915	0,3606	0,5455	0,3714	0,3207	0,7926	0,5745
optdigits	0,0691	0,0067	0,1167	0,1867	0,0227	0,0431	0,1229	0,0000	0,0533	0,0000
pendigits	0,0714	0,3224	0,0693	0,3205	0,3282	0,0833	0,0709	0,1026	0,1346	0,3269
satellite	0,5092	0,9910	0,4769	0,9508	0,9410	0,8053	0,4933	0,9754	0,8893	1,0000
shuttle	0,0594	0,4022	0,2338	1,0000	0,9896	0,4737	0,3878	0,5956		0,7377
wbc	0,3077	0,5714	0,4398	0,6190	0,5524	0,4737	0,5000	0,4286	0,0000	0,5714

Tabuľka 7.3: Výsledky experimentu zobrazujúci P@n.

V tabuľkách 7.2 a 7.4 sú ukázané referenčné výsledky obsiahnuté v dokumentácii knižnice pyOD. Cieľom experimentu v tejto sekcii je tento test zreprodukovať pomocou implementovaného programu. Modely sú vytvorené s využitím predvoleného nastavenia a dáta sú využité celé ako pre trénovanie, tak aj pre testovanie. Výsledky vykonaného experimentu sú zobrazené v tabuľkách 7.3 a 7.5. Experimentu odpovedá konfiguračný súbor *PyOD_bechmarks.json*. Z výsledkov experimentu a referenčných výsledkov je možné vidieť rozdiely, ktoré sa vyskytli z dôvodu využitia celej dátovej sady pre trénovanie a aj pre testovanie. Výsledky sa môžu líšiť aj z toho dôvodu, že niektoré metódy pre detekciu využívajú náhodne generované čísla. Pri experimentoch sa tiež vyskytla závažná chyba, kde model OCSVM prestal pracovať pri využití dát shuttle. Táto chyba nastala z toho dôvodu, že knižnica pyOD využíva funkcie z knižnice scipy, ktoré sú zastaralé a s využitím aktuálnej verzie knižnice scipy sa vyskytla chyba kompatibility. Preto je v experimente využitý model OCSVM s iným nastavením ako predvolené. Aj napriek týmto rozdielom je v tabuľkách možné pozorovať podobné správanie modelov. Vo výsledku boli v experimentoch dosiahnuté lepšie výsledky, ktoré nastali z dôvodu využitia celej dátovej sady pri trénovaní modelov.

[1] https://pyod.readthedocs.io/en/latest/benchmark.html

Data	ABOD	CBLOF	FB	HBOS	Iforest	KNN	LOF	MCD	OCSVM	PCA
arrhythmia	0.3667	0.2123	0.5651	0.1383	0.2669	0.1075	0.0743	1.65	0.0473	0.0596
cardio	0.3824	0.1255	0.7741	0.0053	0.2672	0.2249	0.0993	0.5418	0.0883	0.0035
glass	0.0352	0.0359	0.0317	0.0022	0.1724	0.0173	0.0025	0.0325	0.0010	0.0011
mnist	7.92	1.1339	48.2750	0.0480	1.14	7.31	6.01	4.48	5.0203	0.1569
optdigits	2.79	0.4977	14.2399	0.0303	0.7783	2.1205	1.99	1.99	1.18	0.0519
pendigits	1.39	0.2847	3.85	0.0090	0.5879	0.8659	0.5936	2.09	0.9666	0.0062
satellite	1.70	0.4269	7.66	0.0161	0.6449	1.78	0.9868	2.16	1.97	0.0245
shuttle	14.3611	1.24	59.2131	0.0953	3.06	9.58	11.1500	12.1449	44.6830	0.0378
wbc	0.1014	0.0691	0.0771	0.0063	0.2030	0.0287	0.0078	0.0864	0.0062	0.0035

Tabuľka 7.4: Referenčné výsledky z dokumentácie knižnice PyOD, zobrazujúce čas využitý modelmi.

Name	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
arrhythmia	0,5315	0,3095	1,2722	0,3844	0,6917	0,2130	0,1665	1,3420	0,0272	0,1390
cardio	0,9880	0,1826	1,5013	0,0105	0,6657	0,3733	0,2001	0,9311	0,0668	0,0080
glass	0,0919	0,0654	0,0559	0,0047	0,5721	0,0247	0,0052	0,0539	0,0028	0,0028
mnist	18,9364	1,0732	113,8263	0,0742	2,6861	17,5499	16,4055	7,4804	4,6435	0,1901
optdigits	6,6324	0,3499	32,5742	0,0445	1,6426	4,9733	4,8810	3,2234	0,5379	0,0642
pendigits	3,5440	0,2377	5,8808	0,0146	1,1251	1,3096	0,9396	3,2711	0,2845	0,0113
satellite	4,3479	0,3330	13,1368	0,0266	1,2457	2,1950	1,8169	4,6457	0,4990	0,0364
shuttle	26,5093	0,4176	48,9879	0,0241	4,5513	10,6315	9,2169	17,9411		0,0389
wbc	0,1737	0,0908	0,1498	0,0122	0,6127	0,0532	0,0181	0,1472	0,0058	0,0062

Tabuľka 7.5: Výsledky experimentu zobrazujúci čas využitý modelmi.

7.2 Tvorba a nastavenie modelov

Táto sekcia sa venuje tvorbe experimentov, na úvod je ukázané prečo je vhodné využitie opakovaných zaznamenávaní. Následne sa venuje metódam, ktoré sú duplicitne implementované v rôznych knižniciach, kde je na niektorých príkladoch ukázaný ich rôzny výkon. Na záver je na vybraných modeloch ukázané, ako sa ich výkon mení na základe nastavenia ich vstupných parametrov.

7.2.1 Počet opakovaní v experimentoch

Prvým dôvodom, prečo je vhodné testy opakovať je presnosť výsledkov pri sledovaní času. To že čas sa pri dvoch rôznych sledovaní toho istého modelu bude líšiť je očakávané a preto priemer viacero sledovaní spresní dosiahnuté výsledky. Ďalším dôvodom je to, že niektoré modely využívajú generátor náhodných čísiel. Jednou z metód, ktorá využíva náhodnosť je metóda izolovaných lesov 2.2.4.

Preto bol navrhnutý experiment *Single_trials.json*, ktorý zaznamená výsledky troch behov vybraných metód a vypíše ich oddelene. Výsledky tohto experimentu sú zobrazené v tabuľke 7.6. Je možné vidieť, že čas využitý modelmi sa mení s každou iteráciou. Presnosť modelov ABOD a LOF zostáva rovnaká. Presnosť metód IForest a EllipticEnvelope sa s každou iteráciou menia, z čoho je možné usúdiť, že pre detekciu odľahlých hodnôt využívajú náhodne generované čísla. Využitá pamäť sa mení len u metódy IForest, pretože táto metóda delí vytvorený strom pomocou náhodne vygenerovaných čísel.

7.2.2 Duplicitne implementované metódy

V sekcii 7.1 bol objavený problém, pri ktorom jeden z modelov nepracoval správne. To nám naznačuje dôvod, prečo je vhodné mať k dispozícii viac implementácií jednej metódy. V

Model	ROC	P@n	time	used memory
ABOD	0,6407	0,0714	3,4628	1,7063
ABOD	0,6407	0,0714	3,4692	1,7063
ABOD	0,6407	0,0714	3,5003	1,7063
IForest	0,9446	0,3974	0,2226	2,6432
IForest	0,9062	0,3590	0,2135	2,6457
IForest	0,9487	0,3141	0,2175	2,6542
lof	0,5206	0,0709	0,9516	5,6930
lof	0,5206	0,0709	0,9609	5,6930
lof	0,5206	0,0709	0,9574	5,6930
EllipticEnvelope	0,1692	0,1026	3,5950	7,0003
EllipticEnvelope	0,1693	0,1026	3,6003	7,0003
EllipticEnvelope	0,1687	0,1026	2,9745	7,0003

Tabuľka 7.6: Tabuľka zobrazujúca namerané hodnoty jednotlivých behov. Zodpovedá konfiguračnému súboru *Single_trials.json*.

tejto sekcii sa experimenty venujú práve tým metódam, ktoré sú implementované vo viacerých knižniciach. V tabuľke 7.7 sú zobrazené výsledky pre duplicitne implementovanú metódu OCSVM. Najlepšie výsledky dosiahla metóda OneClassSVM, ktorá je implementovaná knižnicou sklearn. V ďalších experimentoch budú využité metódy, ktoré dosahujú najlepšie výsledky.

Model	arrhythmia	cardio	satellite	shuttle
ROC				
ocsvm	0,5022	0,0648	0,4926	0,2753
OneClassSVM	0,2106	0,1108	0,4137	0,0117
OCSVM	0,5247	0,8895	0,4929	
P@n				
ocsvm	0,1491	0,1923	0,3150	0,1133
OneClassSVM	0,4444	0,3415	0,5988	0,3539
OCSVM	0,1667	0,3523	0,2787	
time				
ocsvm	0,1202	0,1272	4,9129	135,9621
OneClassSVM	0,0379	0,0603	1,1474	38,0158
OCSVM	0,1800	0,0788	18,8177	
memory				
ocsvm	0,9914	0,3082	1,8539	3,5356
OneClassSVM	0,9914	0,3082	1,8539	3,5356
OCSVM	0,9915	0,3084	1,8541	

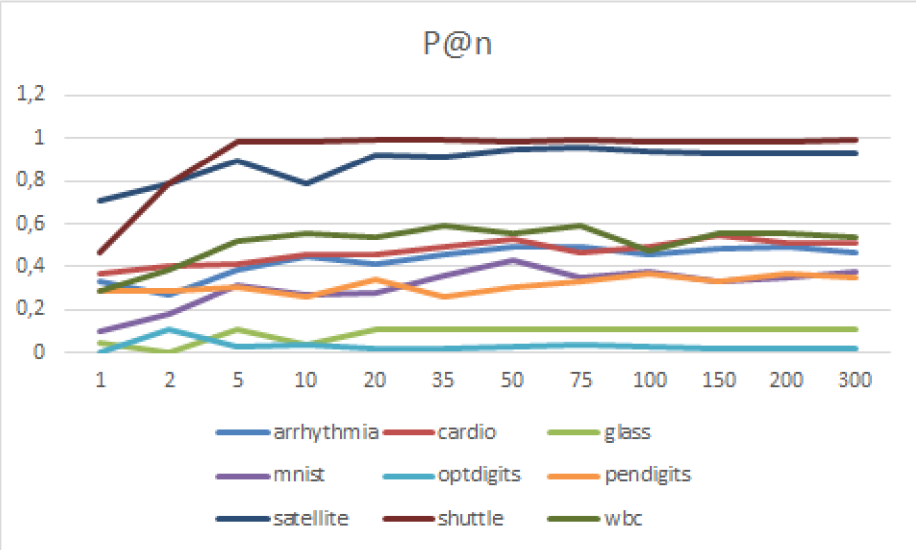
Tabuľka 7.7: Tabuľka zobrazujúca výkon rôznych implementácií modelu OCSVM. Označenie ocsvm zodpovedá knižnici pyodds, OneClassSVM knižnicy scipy a OCSVM knižnici pyOD. Zodpovedá konfiguračnému súboru *OCSVM.json*.

7.2.3 Nastavenie modelu a výsledné hodnoty

Táto sekcia je zameraná na metódy, ktorých presnosť je možné meniť pomocou ich nastavení.

Model	ROC	P@n	time	memory
KNN-3-largest	0,6474	0,7544	2,2397	1,0335
KNN-5-largest	0,6701	0,8053	2,4637	1,6666
KNN-10-largest	0,6931	0,8462	2,5473	3,2432
KNN-3-mean	0,6407	0,8824	2,4130	1,0359
KNN-5-mean	0,6559	0,8793	2,5073	1,6665
KNN-10-mean	0,6763	0,8058	2,7489	3,2431
KNN-3-median	0,6295	0,7533	2,8450	1,0356
KNN-5-median	0,6474	0,7544	2,9125	1,6666
KNN-10-median	0,6741	0,8299	3,1316	3,2431

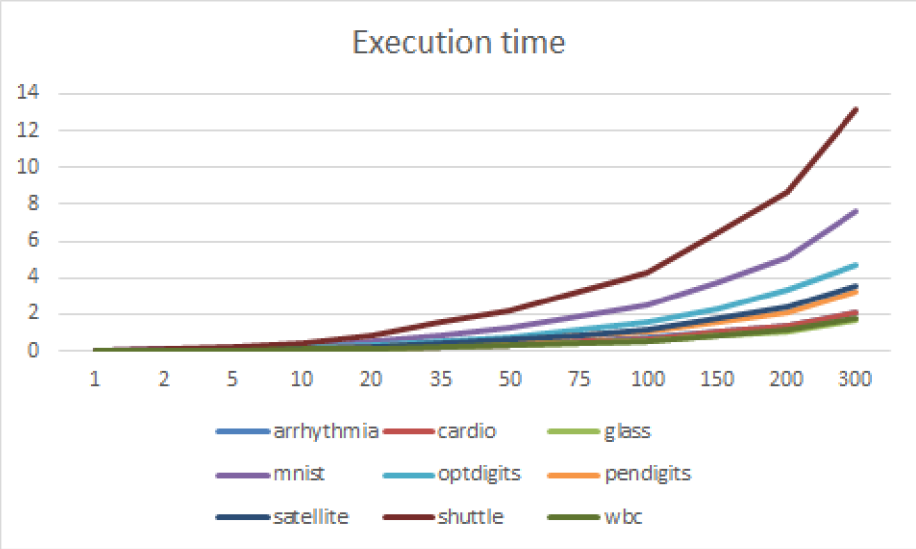
Tabuľka 7.8: Tabuľka zobrazujúca výkon modelu kNN z knižnice pyOD. Zodpovedá konfiguračnému súboru *KNN.json*.



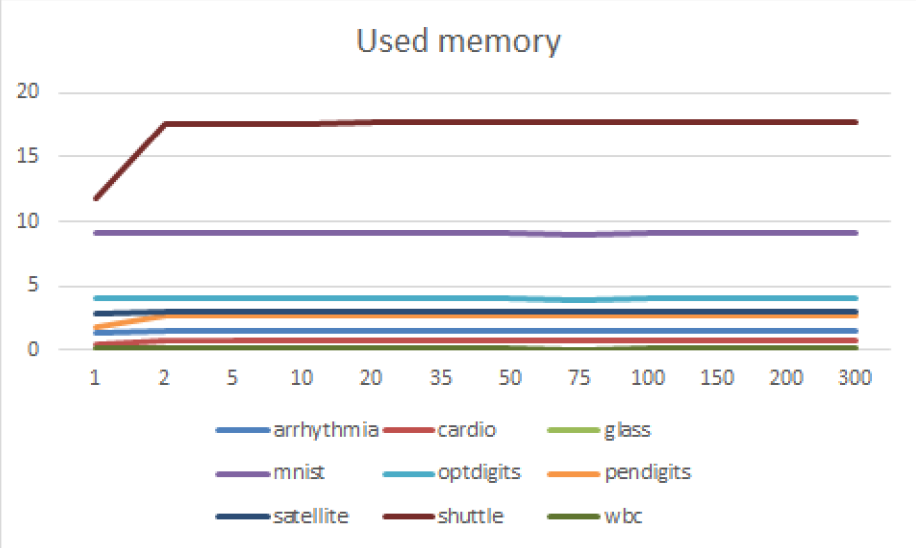
Obr. 7.1: Graf zobrazujúci zmenu presnosti P@n pri detekcii odľahlých hodnôt pri zmene počtu odhadov s využitím modelu IForest.

Takýto model je napríklad kNN, kde je možné nastaviť počet susedov, ktoré má model sledovať alebo metódu pre výpočet vzdialenosti. V experimente z konfiguračného súbora *KNN.json* boli využité modely, ktoré sledovali susedov s počtom 3, 5 a 10. Pre experiment bola využitá dátová sada satellite. Výsledky experimentu sú zobrazené v tabuľke 7.8. Pri využití maximálnej vzdialenosti sa pri prechode z 3 na 10 susedov presnosť zlepšila z 75% na 85%. Naopak pri využití priemernej vzdialenosti sa presnosť znížila z 88% na 80%. Pri zvýšení počtu susedov taktiež stúpa čas potrebný pre detekciu, ktorého zvýšenie z 3 na 10 susedov je okolo 0.3 sekundy pre všetky metódy výpočtu vzdialenosti. Jednou z najvýraznejších zmien je však pamäť využitá modelmi, ktorá sa 3-násobne zväčšila.

Ďalším príkladom je počet odhadov pri využití izolačných lesov. Tento parameter metódy IForest určuje, koľko pokusov náhodného rozdelenia vytvoreného stromu bude vykonaných pre každú iteráciu metódy. Pre tento prípad bol pripravený konfiguračný súbor *Iforest_estimators.json*, ktorý experimentuje s rôznym počtom odhadov. Výsledky zobrazujúce úspešnosť modelov sú zobrazené na grafoch 7.1 a 7.2. Jedným zo zaujímavých vlastností modelu IForest je ten, že pri detekcii odľahlých hodnôt sa so zvyšujúcim počtom odhadov



Obr. 7.2: Graf zobrazujúci zmenu času potrebného pre detekciu pri zmene počtu odhadov pri využití modelu IForest.



Obr. 7.3: Graf zobrazujúci využitú pamäť pri detekcii odľahlých hodnôt pri zmene počtu odhadov s využitím modelu IForest.

nezvyšuje potrebná pamäť, túto skutočnosť možno pozorovať v grafe 7.3. Z experimentov je možné odvodiť, že pri metóde IForest sa presnosť od počtu odhadov vyšších ako 20 výrazne nezvyšuje. Potrebný čas pre detekciu následne od počtu odhadov vyšších ako 20 výrazne rastie. Z toho je možné odvodiť, že najvýhodnejší počet odhadov je pre naše dátové sady v rozsahu 5–20.

7.3 Testovanie na základe spôsobu detekcie

Táto sekcia sa venuje porovnaniu metód detekcie v rámci ich rozdelenia do spoločných skupín podľa kapitoly 2. Toto rozdelenie je taktiež k dispozícii vo forme zoznamu v sekcii 5.2.6. Pre experimenty sú vybrané modely, ktorých analýza sa zaoberá tromi pohľadmi. Prvá informácia uvádza presnosť modelu pomocou výsledkov P@n. Následne uvádzame čas, ktorý modely využijú pri detekcii odľahlých hodnôt. Posledná informácia popisuje modelmi využitú pamäť. Uvedené výsledky sú najlepšie, ktoré boli počas experimentovania dosiahnuté.

K experimentom v tejto kapitole boli z kapitoly 3 vybrané nasledovné dátové sety: glass, wbc, cardio, pendigits, satellite a speech. Tieto dátové sady boli vyberané podľa ich veľkostí (viz. tabuľka 7.1).

7.3.1 Pravdepodobnostné modely

Táto sekcia sa venuje pravdepodobnostným modelom, pre experiment boli vybrané modely ABOD, COPOD, SOS a z-score. Konfigurácia experimentu sa nachádza v súbore *Probabilistic.json*.

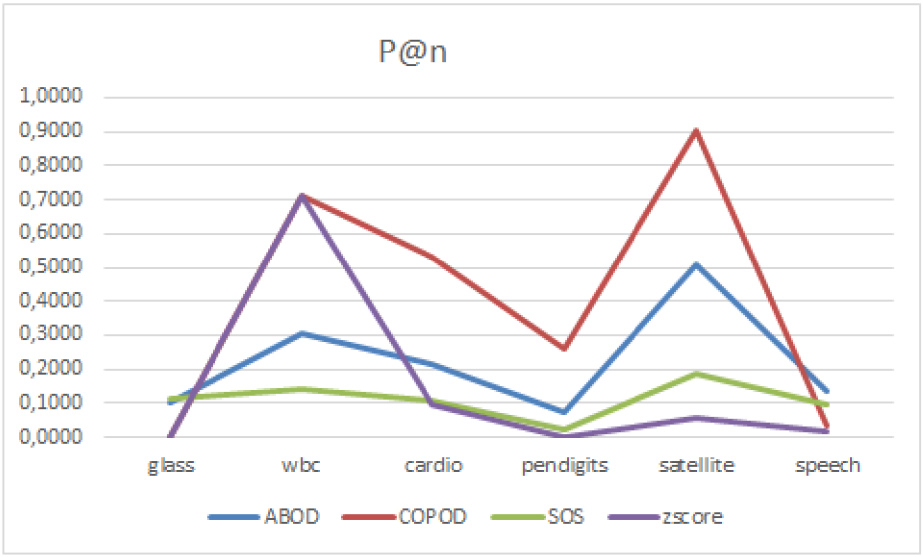
Data	ABOD	COPOD	SOS	zscore
glass	0,1000	0,0000	0,1111	0,0000
wbc	0,3077	0,7143	0,1429	0,7143
cardio	0,2139	0,5284	0,1080	0,0966
pendigits	0,0714	0,2628	0,0256	0,0000
satellite	0,5087	0,9057	0,1885	0,0545
speech	0,1348	0,0328	0,0984	0,0164

Tabuľka 7.9: Tabuľka zobrazujúca presnosť P@n pravdepodobnostných modelov.

Pri tomto experimente sa ukázalo, že využitie z-skóre je najrýchlejší spôsob detekcie, no zároveň je aj najmenej presný. Z grafu 7.4 alebo tabuľky 7.9 je viditeľné, že najpresnejším modelom v skupine pravdepodobnostných modelov je model COPOD. Všetky testované

Data	ABOD	COPOD	SOS	zscore
glass	0,0893	0,0197	0,2844	0,0006
wbc	0,1691	0,0605	0,5709	0,0004
cardio	0,9309	0,0774	3,6724	0,0008
pendigits	3,5475	0,1949	32,6685	0,0012
satellite	4,3077	0,3597	24,9961	0,0032
speech	19,1005	2,6980	10,6119	0,0187

Tabuľka 7.10: Tabuľka zobrazujúca čas využitý jednotlivými modelmi.



Obr. 7.4: Graf zobrazujúci presnosti P@n pri detekcii odľahlých hodnôt pomocou pravdepodobnostných modelov.

Data	ABOD	COPOD	SOS	zscore
glass	0,0436	0,1274	1,5318	0,0488
wbc	0,1386	0,7322	4,6386	0,2494
cardio	0,5299	2,4673	107,3483	0,6889
pendigits	1,7063	7,0413	1510,3671	1,8526
satellite	2,6280	14,8326	1325,1615	3,7987
speech	12,2401	94,3745	434,8374	23,6780

Tabuľka 7.11: Tabuľka zobrazujúca pamäť využitú vybranými pravdepodobnostnými modelmi.

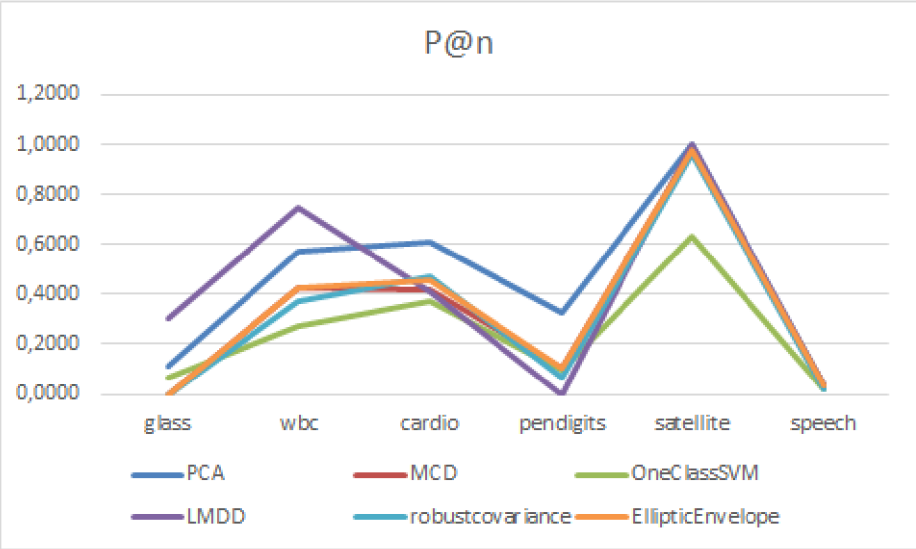
modely si nedokázali poradiť s dátovými sadami glass a speech. Tento model je taktiež druhým najrýchlejším modelom tejto skupiny (viz. tabuľka 7.10). Jedným z prekvapivých výsledkov je pamäť využitá modelom SOS, ktorého spotreba modelu dosahuje v niektorých prípadoch viac ako 1GB (viz. tabuľka 7.11). Jedna zo zaujímavých vlastností modela SOS je tá, že na jeho rýchlosť a využitú pamäť má počet dátových bodov väčšiu záťaž ako počet dimenzií. Spotrebovaný čas a pamäť ostatných modelov experimentu je viacej závislá od počtu dimenzií. Najvýkonnejší model ABOD v týchto prípadoch využíva pamäť pohybujúcu sa okolo 10MB.

7.3.2 Lineárne modely

V tejto sekcii boli pri experimentovaní využité modely PCA, MCD, OneClassSVM, LMDD, robustcovariance a EllipticEnvelope. Experiment zodpovedá konfiguračnému súboru *Linear_Models.json*.

Data	PCA	MCD	OneClassSVM	LMDD	robustcovariance	EllipticEnvelope
glass	0,1111	0,0000	0,0625	0,3000	0,0000	0,0000
wbc	0,5714	0,4286	0,2727	0,7500	0,3684	0,4286
cardio	0,6080	0,4148	0,3686	0,4135	0,4699	0,4545
pendigits	0,3269	0,1026	0,0958	0,0000	0,0670	0,1026
satellite	1,0000	0,9754	0,6363	1,0000	0,9658	0,9754
speech	0,0328	0,0328	0,0182	0,0351	0,0190	0,0328

Tabuľka 7.12: Tabuľka zobrazujúca presnosť lineárnych modelov.



Obr. 7.5: Graf zobrazujúci presnosti P@n pri detekcii odľahlých hodnôt pomocou lineárnych modelov. Využité hodnoty sú zobrazené v tabuľke 7.12.

Z grafu 7.5 alebo tabuľky 7.12 je možné vidieť, že model PCA je z testovaných modelov najpresnejší. Model PCA je nielen najpresnejší ale aj najrýchlejší. Model OneClassSVM z knižnice sklearn využíva najmenej pamäte. Modely MCD, robustcovariance a EllipticEnvelope využívajú najviac pamäte pri dátach s vysokým počtom dimenzií. To je možné vidieť v tabuľke 7.14 pri dátach speech, ktoré majú 400 dimenzií. Ich využitá pamäť je približne

Data	PCA	MCD	OneClassSVM	LMDD	robustcovariance	EllipticEnvelope
glass	0,0035	0,0536	0,0020	1,0147	0,0540	0,0530
wbc	0,0067	0,1389	0,0050	0,5716	0,1356	0,1376
cardio	0,0085	0,8556	0,0479	5,5167	0,8540	0,8923
pendigits	0,0128	2,8951	0,5897	46,3863	2,7671	3,3070
satellite	0,0354	4,3848	0,8251	127,7845	4,6093	4,3393
speech	1,3880	42,1179	2,5757	547,4504	46,9284	40,8660

Tabuľka 7.13: Tabuľka zobrazujúca čas využitý vybranými lineárnymi modelmi.

Data	PCA	MCD	OneClassSVM	LMDD	robustcovariance	EllipticEnvelope
glass	0,0606	0,1095	0,0160	1,4479	0,1400	0,1421
wbc	0,3818	0,7076	0,0913	0,1977	0,7096	0,7099
cardio	1,1996	2,7205	0,3082	0,6722	2,7279	2,7341
pendigits	3,3610	6,9721	0,8800	1,9568	6,9985	6,9993
satellite	7,3018	11,5940	1,8539	3,8925	11,6211	11,6196
speech	52,2397	490,7906	11,7958	23,6993	490,8076	490,8057

Tabuľka 7.14: Tabuľka zobrazujúca pamäť využitú vybranými lineárnymi modelmi.

490MB. Ich presnosť a rýchlosť sú taktiež podobné a môžeme z toho vyvodiť, že ich spôsob detekcie je veľmi podobný. Najpomalším modelom je model LMDD, ktorý pri dátach speech potreboval k detekcii takmer 10 minút. Z tabuľky 7.13 je taktiež možné odvodiť, že rýchlosť lineárnych modelov je viacej závislá na počte dimenzií ako na počte samotných dátových bodov.

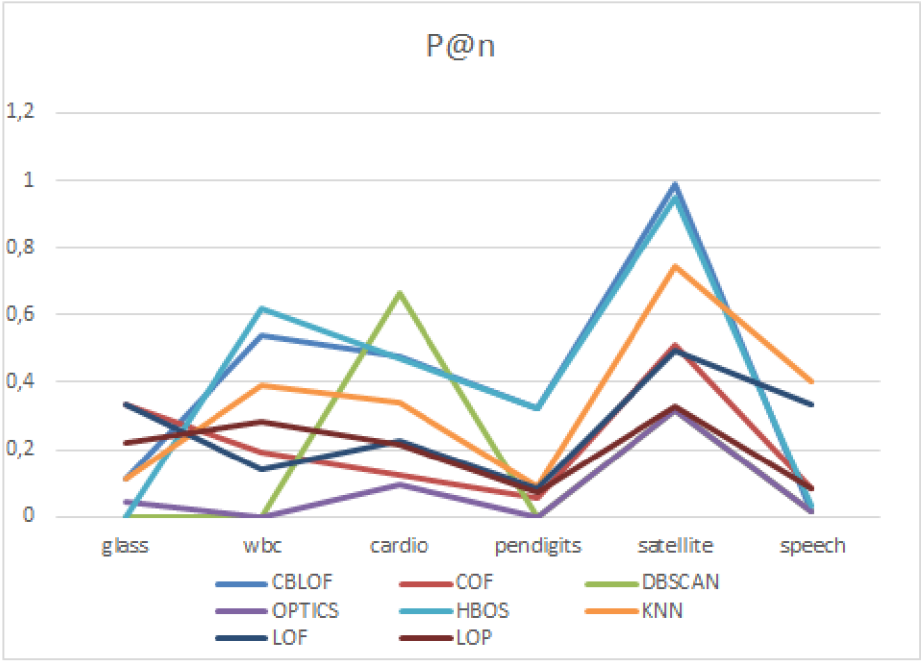
7.3.3 Modely založené na vzdialenosti

Experiment zodpovedajúci tejto sekcii využíva modely CBLOF, COF, DBSCAN, OPTICS, HBOS, KNN, LOF a LOP. Experiment zodpovedá súboru *Proximity_Based.json*.

Modely s najlepším výkonom v presnosti sú CBLOF a HBOS, čo je zobrazené v grafe 7.6 alebo v tabuľke 7.15. V tomto experimente boli zahrnuté aj metódy pre zhlukovanie, ktorých presnosť je z vybraných modelov najmenšia. Jedinou výnimkou je metóda DBSCAN, ktorá bola pri dátach cardio z modelov najpresnejšia. Čas využitý modelmi je zobrazený v tabuľke 7.16. Model HBOS je najrýchlejší, tento model podáva veľmi dobré výsledky s dátami s vysokým počtom dimenzií. Druhý najrýchlejší je model CBLOF, ktorý v porovnaní s modelom HBOS spomaľuje pri vysoko-dimenzionálnych dátach. Pri pohľade na využitý čas sú modely COF a LOP najpomalšie. Využitú pamäť modelmi je možné vidieť v tabuľke 7.17. Najmenej pamäte využil model KNN, model COF na druhú stranu využil pamäte najviac. Spotreba pamäte modelu HBOS je nižšia ako modelu CBLOF. Oba modely HBOS

Data	CBLOF	COF	DBSCAN	OPTICS	HBOS	KNN	LOF	LOP
glass	0,1111	0,3333	0,0000	0,0439	0,0000	0,1111	0,3333	0,2222
wbc	0,5380	0,1904	0,0000	0,0000	0,6190	0,3888	0,1428	0,2857
cardio	0,4767	0,1250	0,6666	0,0957	0,4711	0,3402	0,2258	0,2159
pendigits	0,3217	0,0576	0,0000	0,0000	0,3205	0,0888	0,0833	0,0705
satellite	0,9909	0,5082	0,3163	0,3163	0,9502	0,7452	0,4949	0,3300
speech	0,0163	0,0819	0,0165	0,0165	0,0327	0,4000	0,3333	0,0819

Tabuľka 7.15: Tabuľka zobrazujúca presnosť modelov založených na vzdialenosti.



Obr. 7.6: Graf zobrazujúci presnosti P@n pri detekcii odľahlých hodnôt pomocou modelov založených na vzdialenosti. Hodnoty využité v grafe je možné vidieť v tabuľke 7.15.

Data	CBLOF	COF	DBSCAN	OPTICS	HBOS	KNN	LOF	LOP
glass	0,0638	0,1560	0,0057	0,1759	0,3062	0,0427	0,0046	0,3341
wbc	0,0886	0,4757	0,0055	0,3915	0,0126	0,0847	0,0158	1,0335
cardio	0,1548	10,5775	0,5970	3,1973	0,0112	0,5229	0,1817	24,9195
pendigits	0,2283	169,2276	0,6385	20,9389	0,0149	1,9144	0,6305	348,4862
satellite	0,3403	180,3563	0,7364	1,4445	0,0282	2,8609	1,5591	307,6862
speech	1,4508	185,7353	39,5190	16,2210	0,2390	18,5711	17,0304	104,8106

Tabuľka 7.16: Tabuľka zobrazujúca čas využitý vybranými modelmi založených na vzdialenosti.

a CBLOF podávajú veľmi dobrý výkon, pri čom výkon modelu HBOS je lepší. V rámci testovania tvoria dáta speech, ktoré obsahujú 400 dimenzií veľký problém pre modely nielen s rýchlosťou, ale hlavne s presnosťou. Prekvapivé je, že modely KNN a LOF pri týchto dátach podávajú značne lepšie výkony ako ostatné modely.

7.3.4 Kombinačné modely

Táto sekcia sa zaoberá experimentovaní s kombinačnými modelmi. Sú využité modely IForest, FeatureBagging a LODA. Experiment zodpovedá konfiguračnému súboru *Ensembles.json*.

Ako je zobrazené v grafe 7.7 a v tabuľke 7.18, presnosť modelov IForest a LODA je lepšia, ako presnosť modelu FeatureBaggign. Model LODA je taktiež z využitých modelov najrýchlejší a využíva najmenej pamäte. Model FeatureBaggign je najpomalší a využíva najviac pamäte. Využitá pamäť modelu IForest je porovnateľná s modelom FeatureBaggign, jeho rýchlosť je avšak znateľne lepšia, ako je možné vidieť v tabuľke 7.19. Pamäť využitú modelmi je možné vidieť v tabuľke 7.20.

Data	CBLOF	COF	DBSCAN	OPTICS	HBOS	KNN	LOF	LOP
glass	0,0476	6,6246	0,0917	0,1325	0,5625	0,0439	0,0438	0,0099
wbc	0,2600	2,2832	0,3172	0,3512	0,1569	0,0787	0,1373	0,0382
cardio	0,9073	53,6264	1,2214	1,0947	0,3738	0,3833	0,5170	0,1835
pendigits	2,6244	755,0956	4,0068	3,1374	0,9456	1,4422	1,6532	0,6870
satellite	5,1989	662,4962	5,2635	5,0064	1,9195	1,3505	2,5783	0,6430
speech	34,5280	217,3562	27,4132	27,2692	11,8614	0,7723	12,2124	21,7492

Tabuľka 7.17: Tabuľka zobrazujúca pamäť využitú vybranými modelmi založenými na vzdialenosti.

Data	IForest	FB	LODA
glass	0,1111	0,2222	0,0000
wbc	0,5238	0,4444	0,6190
cardio	0,4147	0,1527	0,4375
pendigits	0,3974	0,0692	0,2948
satellite	0,9344	0,4864	1,0000
speech	0,0327	0,0833	0,0327

Tabuľka 7.18: Tabuľka zobrazujúca presnosť P@n kombinačných modelov.

Data	IForest	FB	LODA
glass	0,5704	0,0560	0,0314
wbc	0,5941	0,1529	0,0502
cardio	0,7087	1,5658	0,0625
pendigits	1,0983	6,5080	0,1001
satellite	1,2464	12,3146	0,0971
speech	4,2403	108,3728	0,1498

Tabuľka 7.19: Tabuľka zobrazujúca čas využitý kombinačnými modelmi.

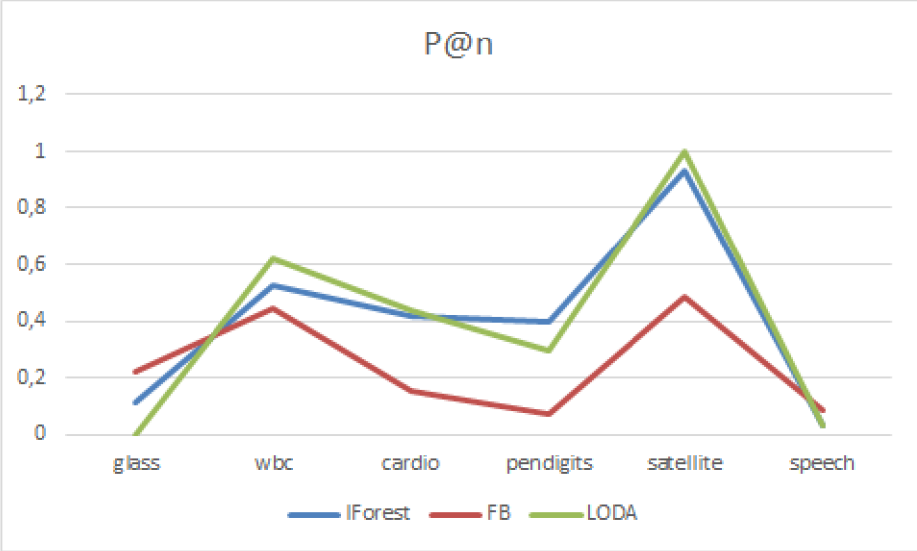
7.3.5 Modely založené na neurónových sietiach

Experiment pre modely využívajúce neurónové siete je navrhnutý v konfiguračnom súbore *Neural_Networks.json*. Jednou z hlavných nevýhod použitia neurónových sietí je ten, že je potrebné počet neurónov upravovať podľa vstupných dát. Z toho dôvodu metódy porovnávame len pomocou dátovej sady speech, ktorá predstavovala najťažšiu dátovú sadu pre predošlé modely. V tomto experimente boli využité modely AutoEncoder, MO_GAAL, SO_GAAL a VAE.

Výsledky experimentu sú zobrazené v tabuľke 7.21. Presnosť týchto modelov je takmer zhodná. Najmenej pamäte využil model SO_GAAL. Najrýchlejšie modely sú AutoEncoder a VAE. Pri využití neurónových sietí je taktiež poukázané na to, že ich trénovací čas je oveľa vyšší ako čas potrebný pre výpočet označenia odľahlých hodnôt, alebo skóre jednotlivých dátových bodov. Z využitých modelov je model MO_GAAL najpomalší, pričom je doba jeho trénovanie výrazne vyššia od ostatných modelov. najviac pamäte využil model VAE.

7.4 Celkové porovnanie

Táto sekcia sa venuje modelom, ktoré v predošlých sekciách dosahovali najlepšie výsledky. Experimenty prebiehajú na všetkých dátových sadách z kapitoly 3. Počet opakovaní merania



Obr. 7.7: Graf zobrazujúci presnosti P@n z tabuľky 7.18, pri detekcii odľahlých hodnôt pomocou kombinačných modelov.

Data	IForest	FB	LODA
glass	0,0788	0,2347	0,0089
wbc	0,2051	0,4976	0,0125
cardio	0,7505	1,8092	0,0626
pendigits	2,6651	5,9930	0,2339
satellite	3,0159	7,0720	0,2191
speech	17,7521	21,2392	0,1273

Tabuľka 7.20: Tabuľka zobrazujúca pamäť využitú kombinačnými modelmi.

je 10. Cieľom experimentu je zhodnotiť a porovnať výkonnosť jednotlivých skupín metód detekcie.

Úspešnosť detekcie odľahlých hodnôt modelov využitých v tejto sekcii je u všetkých modelov veľmi podobná. Zobrazenie úspešnosti modelov je možné vidieť na grafe 7.8 alebo v tabuľke 7.23. Výsledky zobrazujúce plochu pod krivkov ROC sú zobrazené v tabuľke 7.22. Ich rozdiel je často pozorovateľný len pri špecifických dátach, ako je napríklad úspešnosť modelu PCA na dátach mnist. Viditeľný rozdiel sa taktiež vyskytol u dát shuttle, kde modely HBOS, COPOD a IForest dosahujú takmer 100% úspešnosť. Výsledky z grafu 7.8 taktiež poukazujú na fakt, že vo väčšine prípadov je presnosť modelov závislá hlavne od vstupných dát.

Z grafu 7.9 alebo tabuľky 7.24 je zrejmé, že rýchlosť modelov je vo väčšine prípadov podobná. Rozdiel nastáva len pri dátach, ktoré dosahujú vysoké hodnoty u počtu dátových bodov alebo dimenzií. Model CBLOF a COPOD podávajú slabší výkon v rýchlosti od ostatných modelov v prípade veľkého počtu dátových bodov. Model COPOD taktiež ukazuje pomalý výkon pri dátach s vysokým počtom dimenzií.

Pri pohľade na využitú pamäť z grafu 7.10 alebo tabuľky 7.25 je najvýkonnejší model LODA. Najväčšiu spotrebu pamäte naopak javí model COPOD.

Data	roc	prec	fit time	predict time	memory
AE	0,4690	0,0327	11,8082	0,1154	29,4882
MO_GAAL	0,5091	0,0327	270,8101	0,0720	29,2173
SO_GAAL	0,4496	0,0338	32,3424	0,0709	0,0241
VAE	0,4692	0,0327	12,9905	0,5338	206,4876

Tabuľka 7.21: Tabuľka zobrazujúca výsledky experimentu s modelmi využívajúcich neurónové siete.

Data	COPOD	PCA	CBLOF	HBOS	IForest	LODA
arrhythmia	0,7998	0,7748	0,8175	0,8101	0,7393	0,7517
cardio	0,9219	0,9500	0,8031	0,8377	0,8606	0,8572
cover	0,8841	0,9345	0,6305	0,6498	0,8239	0,8674
glass	0,6450	0,6027	0,7909	0,6705	0,7061	0,4583
mnist	0,7739	0,8498	0,8123	0,6868	0,7064	0,7419
optdigits	0,6824	0,5137	0,8826	0,8702	0,6647	0,5276
pendigits	0,9048	0,9355	0,9718	0,9283	0,8849	0,9317
satellite	0,6335	0,6012	0,7312	0,7659	0,6872	0,6154
shuttle	0,9945	0,9899	0,9924	0,9793	0,9862	0,6608
speech	0,4911	0,4692	0,4708	0,4757	0,4747	0,4761
wbc	0,9636	0,9349	0,9296	0,9581	0,9252	0,9299

Tabuľka 7.22: Tabuľka zobrazujúca plochu pod krivkov ROC.

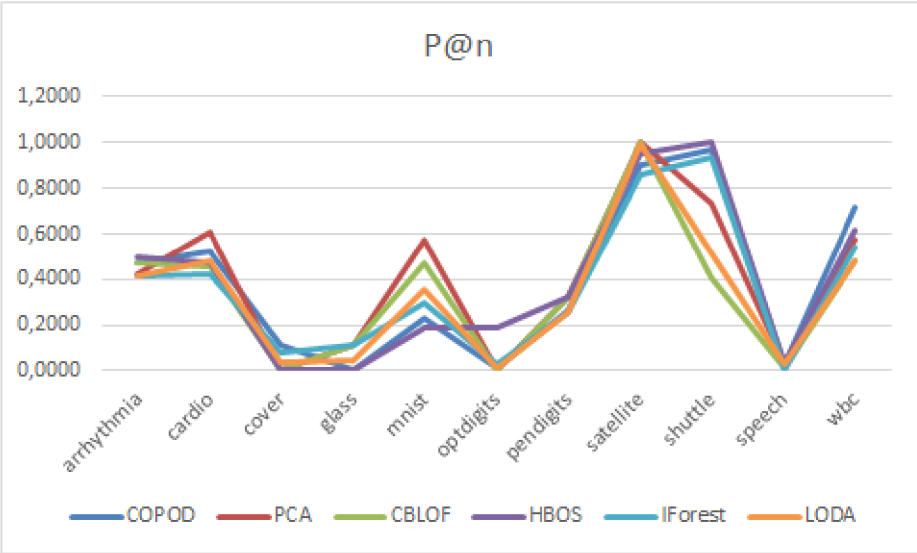
7.4.1 Spájanie výsledkov viacerých modelov

Táto sekcia sa venuje technike, ktorá spája výsledné skóre z viacerých modelov. V experimente boli využité modely z predchádzajúcej sekcie. Pri experimente boli využité metódy kombinovanie pomocou priemeru, výberu maximálnej hodnoty a mediánu.

Graf 7.11 zobrazuje presnosť modelov, ktoré boli využité pri kombinovaní doplnené o výsledky získané samotným kombinovaním. Všetky tri metódy dosahujú takmer rovnakú presnosť. Prekvapivý výsledok je však pri využití dát satellite, kde všetky využité modely pre kombináciu dosahujú takmer 100% úspešnosť. Kombinované výsledky však dosahujú len 50% úspešnosť, čo je veľmi veľký pokles oproti pôvodným hodnotám.

Data	COPOD	PCA	CBLOF	HBOS	IForest	LODA
arrhythmia	0,4697	0,4242	0,4697	0,5000	0,4182	0,4118
cardio	0,5284	0,6080	0,4563	0,4716	0,4215	0,4839
cover	0,1123	0,0000	0,0000	0,0000	0,0828	0,0342
glass	0,0000	0,1111	0,1111	0,0000	0,1125	0,0422
mnist	0,2287	0,5745	0,4702	0,1915	0,2963	0,3563
optdigits	0,0133	0,0000	0,0073	0,1867	0,0300	0,0093
pendigits	0,2628	0,3269	0,3218	0,3205	0,2522	0,2538
satellite	0,9057	1,0000	1,0000	0,9508	0,8581	0,9942
shuttle	0,9672	0,7377	0,4027	1,0000	0,9375	0,5137
speech	0,0328	0,0328	0,0164	0,0328	0,0082	0,0264
wbc	0,7143	0,5714	0,4857	0,6190	0,5429	0,4855

Tabuľka 7.23: Tabuľka zobrazujúca presnosť P@n.



Obr. 7.8: Graf zobrazujúci presnosti P@n pri detekcii odľahlých hodnôt z tabuľky 7.23.

Data	COPOD	PCA	CBLOF	HBOS	IForest	LODA
arrhythmia	0,4343	0,1370	0,3010	0,3409	0,0786	0,0587
cardio	0,0771	0,0081	0,1541	0,0106	0,0713	0,0619
cover	7,2937	0,2880	9,7828	0,1974	2,8242	2,3204
glass	0,0190	0,0028	0,0615	0,0049	0,0577	0,0313
mnist	0,7902	0,1879	0,9642	0,0728	0,3051	0,1195
optdigits	0,4145	0,0631	0,3281	0,0432	0,1722	0,0915
pendigits	0,1969	0,0112	0,2287	0,0139	0,1084	0,1003
satellite	0,3557	0,0337	0,3066	0,0264	0,1318	0,0939
shuttle	0,7138	0,0389	0,4405	0,0235	0,4545	0,2942
speech	2,6003	1,3681	1,3545	0,2299	0,6057	0,1384
wbc	0,0580	0,0061	0,0863	0,0125	0,0604	0,0496

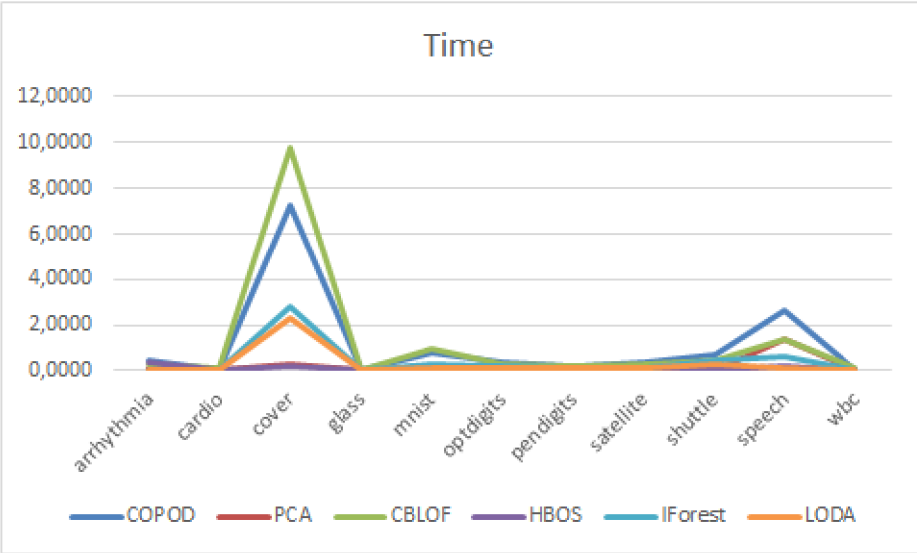
Tabuľka 7.24: Tabuľka zobrazujúca čas využitý modelmi.

Graf 7.12 zobrazuje čas využitý modelmi. Z grafu je možné vidieť, že čas kombinovaných metód je rovný súčte časov všetkých dielčich modelov. Ak vo výsledku zoberieme fakt, že úspešnosť výsledkov nebola vyššia, je výkonnosť týchto metód stratová.

V grafe 7.13, ktorý zobrazuje využitú pamäť je ukázané, že využitá pamäť kombinačných metód je približne rovná maximálnej využitej pamäte z dielčích metód. Tento fakt zodpovedá predpokladu z faktu, že výsledky jednotlivých dielčich modelov sú získavané jednotlivo a nie paralelne.

7.5 Experimenty pohľadom na dáta

Táto sekcia sa venuje tomu, ako je výstup experimentov ovplyvnený formou vstupných dát. Pri experimentovaní bol využitý generátor dát. Vygenerované dáta sa venujú trom oblastiam. V prvom prípade je menený počet vygenerovaných dátových bodov s pevným počtom 10 dimenzií a správne dáta sú rozdelené do 3 zhlukov. Druhý experiment sa venuje zmene dimenzií. Počet vygenerovaných dátových bodov je 2000 a správne dáta sú rozdelené



Obr. 7.9: Graf zobrazujúci využitý čas modelmi z tabuľky 7.24.

Data	COPOD	PCA	CBLOF	HBOS	IForest	LODA
arrhythmia	7,9371	5,7712	2,8048	1,0570	1,6504	0,0159
cardio	2,4673	1,2003	0,9132	0,3738	0,9472	0,0626
cover	183,0771	84,6749	68,6963	25,1728	119,8598	9,1540
glass	0,1274	0,0621	0,0479	0,0315	0,0890	0,0089
mnist	48,6656	24,4745	16,7635	6,1486	10,7569	0,2589
optdigits	21,3711	10,6924	7,6174	2,7368	4,9606	0,1777
pendigits	7,0413	3,3617	2,6243	0,9456	3,1992	0,2339
satellite	14,8326	7,3019	5,1823	1,9195	4,2432	0,2191
shuttle	28,2863	12,9656	11,1720	3,9283	20,0096	1,6696
speech	94,3745	52,2397	34,8582	11,8614	19,3741	0,1273
wbc	0,7322	0,3828	0,2641	0,1569	0,2349	0,0126

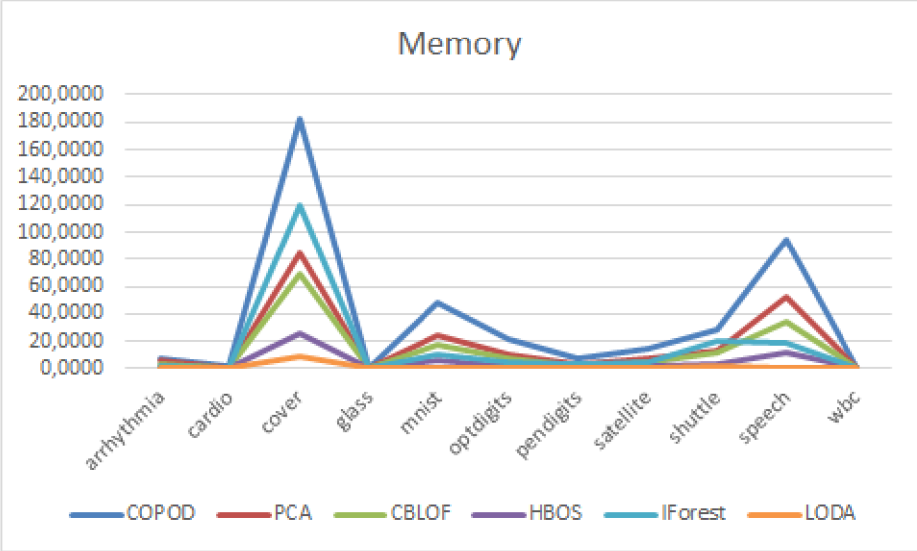
Tabuľka 7.25: Tabuľka zobrazujúca využitú pamäť.

do 3 zhlukov. Tretí experiment sa venuje rozdeleniu medzi rôzny počet zhlukov. Počet vygenerovaných dátových bodov je 2000 a počet dimenzií dát je 10.

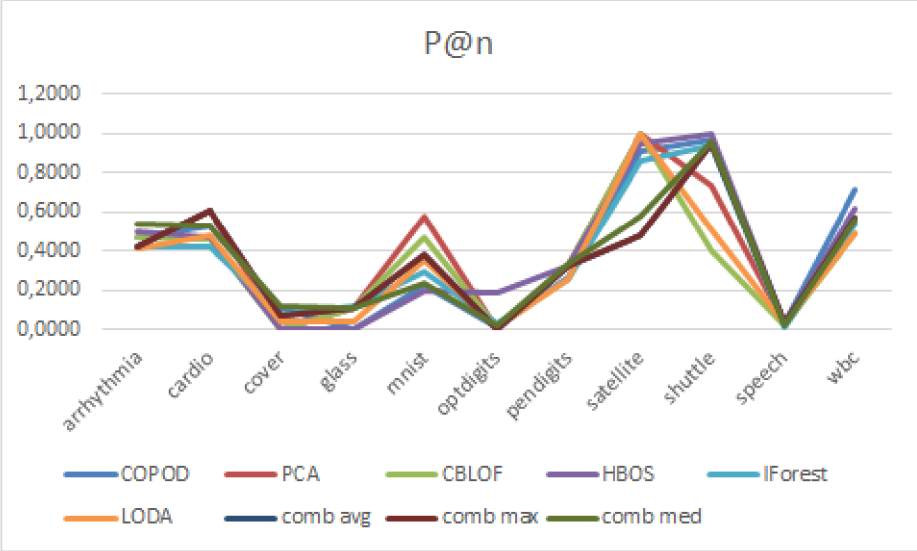
Pri zmene počtu vygenerovaných bodov je presnosť modelov mierne rastúca, čo je možné vidieť v grafe 7.14. Najpresnejšie modely sú CBLOF, IForest, HBOS a LODA. Závislosť medzi časom a počtom dátových bodov je možné vidieť v grafe 7.15. Najlepší výkon pri pohľade na čas podávajú modely PCA, HBOS a LODA.

Presnosti modelov zmene počtu dimenzií je možné vidieť v grafe 7.16. Najpresnejšie modely sú CBLOF a LODA. Závislosť medzi časom a počtom dimenzií je možné vidieť v grafe 7.17. Najlepší výkon pri pohľade na čas podáva model LODA, ktorý využíva takmer rovnaký čas nezávisle od počtu dimenzií.

Graf 7.18 zobrazuje presnosti modelov pri zmene počtu zhlukov v dátach. Najpresnejší je model LODA. Zvyšné modely ukazujú tendenciu strácať výkon pri zvyšujúcom sa počte dimenzií. Závislosť medzi časom a počtom zhlukov v dátach je možné vidieť v grafe 7.19. Najlepší výkon pri pohľade na čas podávajú modely PCA, HBOS a LODA. Modely podávajú približne rovnaký časový výkon pri ľubovolnom počte zhlukov.



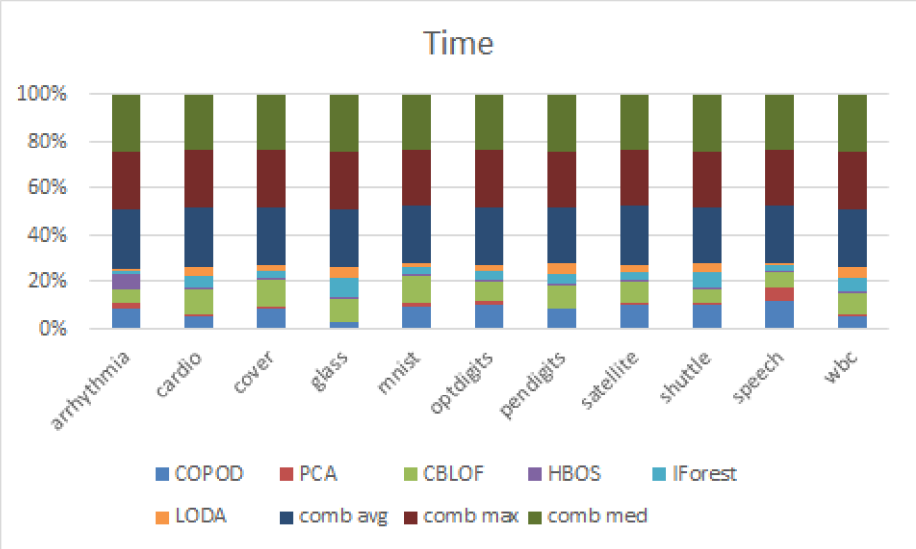
Obr. 7.10: Graf zobrazujúci využitú pamäť z tabuľky 7.25.



Obr. 7.11: Graf zobrazujúci presnosti P@n pri detekcii odľahlých hodnôt z predošlej sekcie s pridanými výsledkami kombinačných metód.

Modely podávajú veľmi dobré výsledky v prípade, kedy sú využité vygenerované dáta. No pri využití reálnych dátových sád boli v niektorých sád presnosti modelov veľmi nízke. V grafe 7.8 je možné vidieť to, že všetky modely majú tendenciu dosahovať podobné výsledky pri rovnakých vstupných dátach.

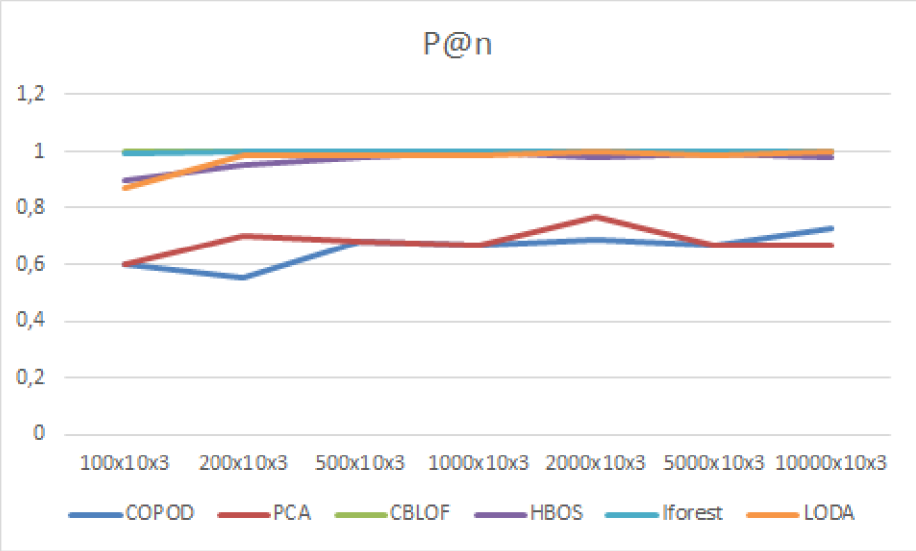
Reálne dátové sady vieme rozdeliť do troch skupín. Prvou skupinou sú dátové sady, ktoré dosahujú takmer 100% úspešnosti. Tieto dátové sady sú satellite a shuttle. Z kapitoly 3 vieme, že tieto dátové sady sa skladajú z hodnôt rovnakého typu, ktorým sú celé čísla. Druhou skupinou sú dátové sady, ktoré dosahujú presnosť okolo 50%. Tieto dátové sady sú arrhythmia, cardio, mnist, pendigits a wbc. Tieto dátové sady obsahujú reálne čísla, ktoré sú často doplnené o ďalší typ informácií, ako sú napríklad pravdivostné hodnoty. Tu je možné,



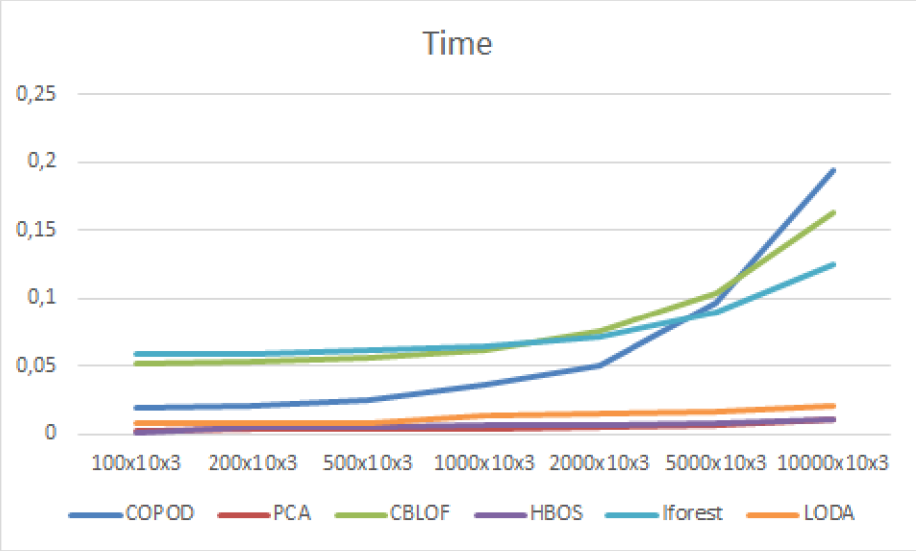
Obr. 7.12: Graf zobrazujúci využitý čas modelmi z predošlej sekcie s pridanými výsledkami kombinačných metód.

Obr. 7.13: Graf zobrazujúci využitú pamäť z predošlej sekcie s pridanými výsledkami kombinačných metód.

že sa niektoré informácie strácajú z dôvodu rôznych typov dát. V iných prípadoch sa jedná o desatinné hodnoty v rozsahu 0–1, z čoho je možné odhadnúť, že sa jedná o percentá vyjadrené desatinnými číslami. V tomto prípade sa odľahlé hodnoty môžu nachádzať v tesnej blízkosti správnych, čo prekáža ich odhaleniu. Tretia skupina sú dátové sady, kde je presnosť modelov takmer 0%. Tieto dátové sady sú cover, glass, optdigits a speech. Nakoľko sa jedná o reálne dátové sady je možné, že jednotlivé skupiny dát sú príliš podobné. V tomto prípade môžeme očakávať, že modely nedokážu nájsť zhluky dát, ktoré by dáta oddelili od odľahlých.



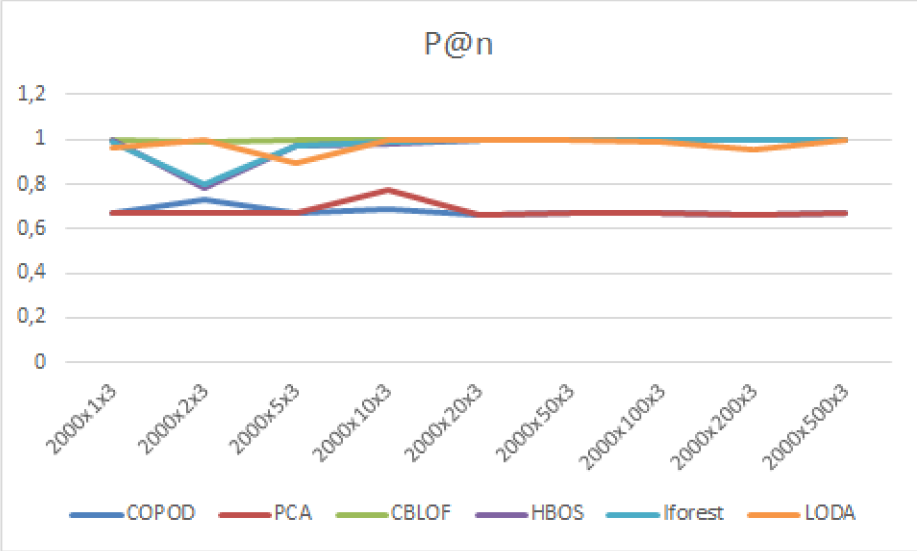
Obr. 7.14: Graf zobrazujúci presnosť modelov pri zmene počtu dátových bodov vo vygenerovaných dátach.



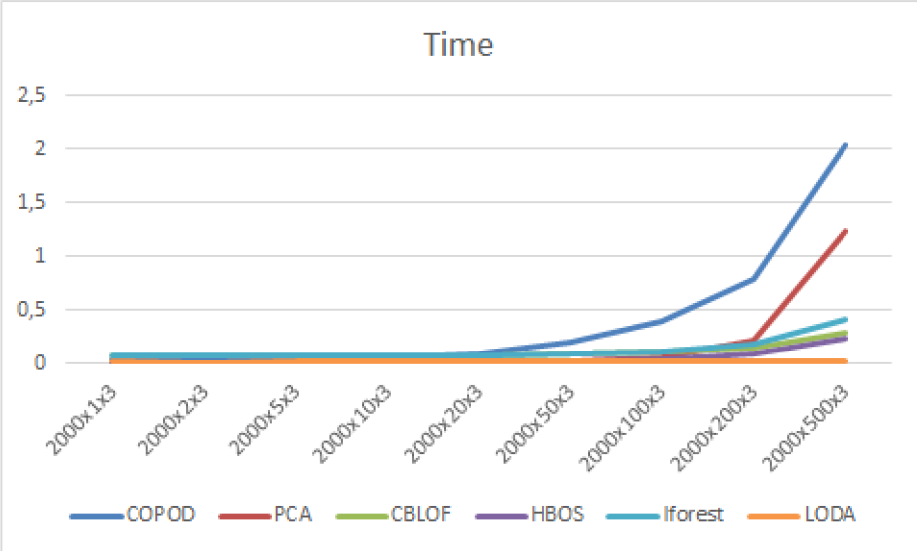
Obr. 7.15: Graf zobrazujúci čas využitý modelmi pri zmene počtu dátových bodov vo vygenerovaných dátach.

7.6 Urýchlenie znížením veľkosti dátovej sady pre trénovanie modelu

V tejto sekcii je navrhnuté urýchlenie modelov pomocou zníženia veľkosti dátovej sady využitej pri trénovaní modelov funkciou `fit()`. V tejto sekcii sa zameriame na najdlhšie trvajúce výsledky, ktoré využívajú najväčšiu dátovú sadu cover. V experimente sú využité modely zo sekcie 7.4. V grafe 7.20 je zobrazený pomer trvania trénovania funkciou `fit()` a ohodnotením dát funkciou `predict()` pri využití celej dátovej sady. Z grafu je možné sledovať, že trénovanie modelov je vo väčšine prípadov časovo oveľa náročnejšie. Výsledky



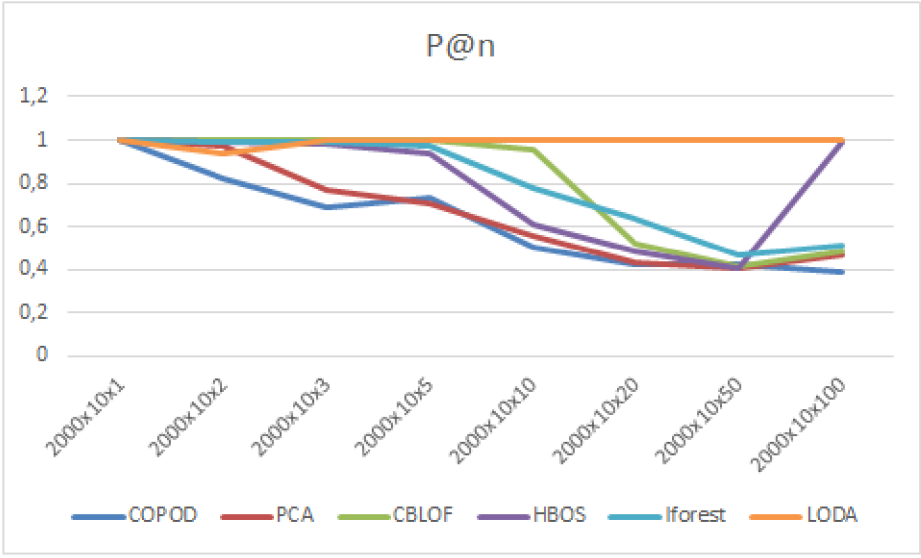
Obr. 7.16: Graf zobrazujúci presnosť modelov pri zmene počtu dimenzií vo vygenerovaných dátach.



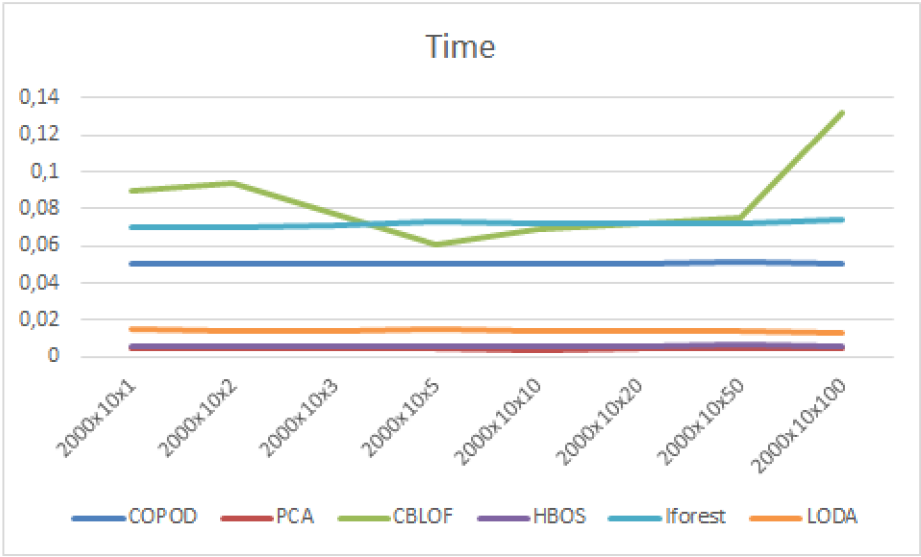
Obr. 7.17: Graf zobrazujúci čas využitý modelmi pri zmene počtu dimenzií vo vygenerovaných dátach.

experimentu sú zobrazené v grafe 7.21, kde je zobrazený pomer časov potrebných pre vykonanie týchto dvoch funkcií v prípade, kedy sa na trénovanie použije len 20% z celkových dát. Na ohodnotenie funkciou `predict()` je vždy využitých 100% dát.

V grafe 7.22 je zobrazený pomer trvania metód `fit()` pri nevyužití a využití navrhovaného zrýchlenia. Graf jednoznačne zobrazuje značné urýchlenie modelov. Toto urýchlenie je taktiež znázornené v tabuľke 7.26. Z tabuľky je následne možné vyčítať, že zrýchlenie je vo väčšine prípadov približne 5-násobné.



Obr. 7.18: Graf zobrazujúci presnosť modelov pri zmene počtu zhlukov vo vygenerovaných dátach.

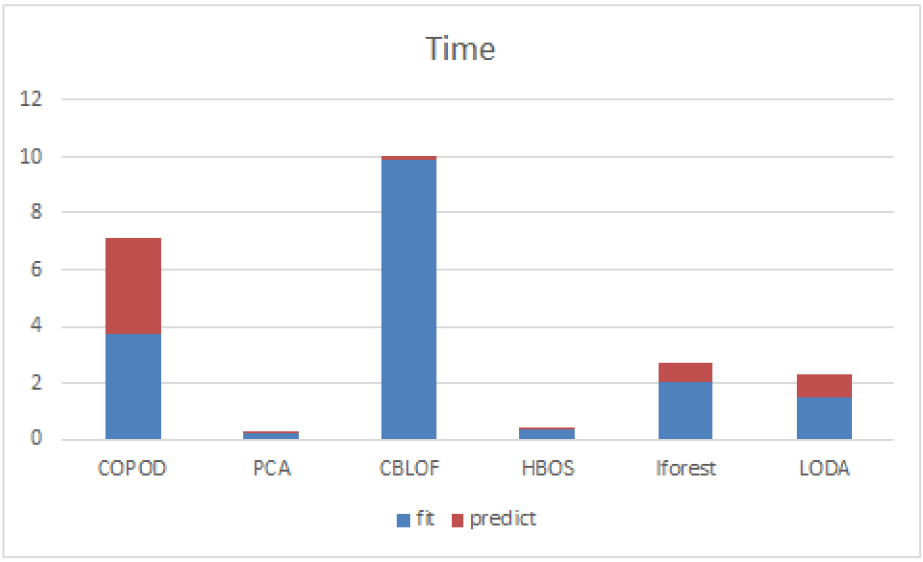


Obr. 7.19: Graf zobrazujúci čas využitý modelmi pri zmene počtu zhlukov vo vygenerovaných dátach.

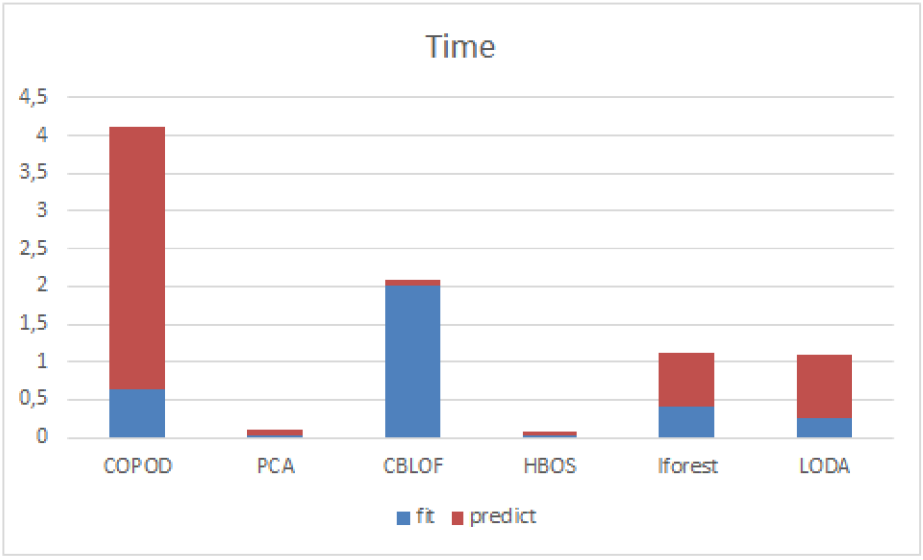
7.6.1 Presnosť modelov pri zrýchlení

Táto sekcia sleduje presnosť modelov pri ich urýchlení navrhnutým riešením. Pri experimente využijeme dátové sady, ktoré dosahovali najlepšie výsledky v presnosti. Tieto dátové sady sú satellite a shuttle. Dáta určené pre metódu `fit()` sú vyberané náhodne z celej dátovej sady.

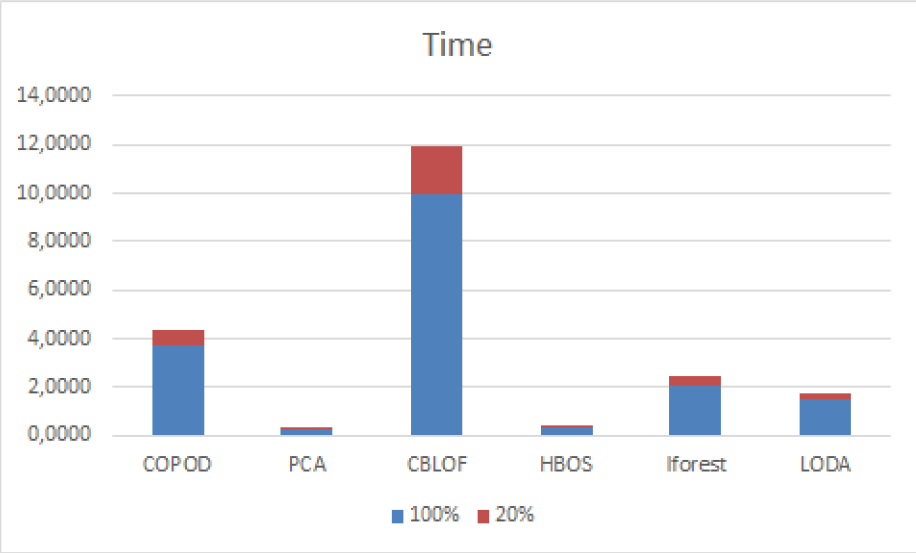
Z grafu 7.23 je možné odvodiť, že presnosť zrýchlenej detekcie nieje ovplyvnená zmenšením dátovej sady až do bodu, kde využívame len 1% z celkových dát. Pri využití 0.2% dát sa vo výsledkoch vyskytuje pokles v presnosti využitých metód.



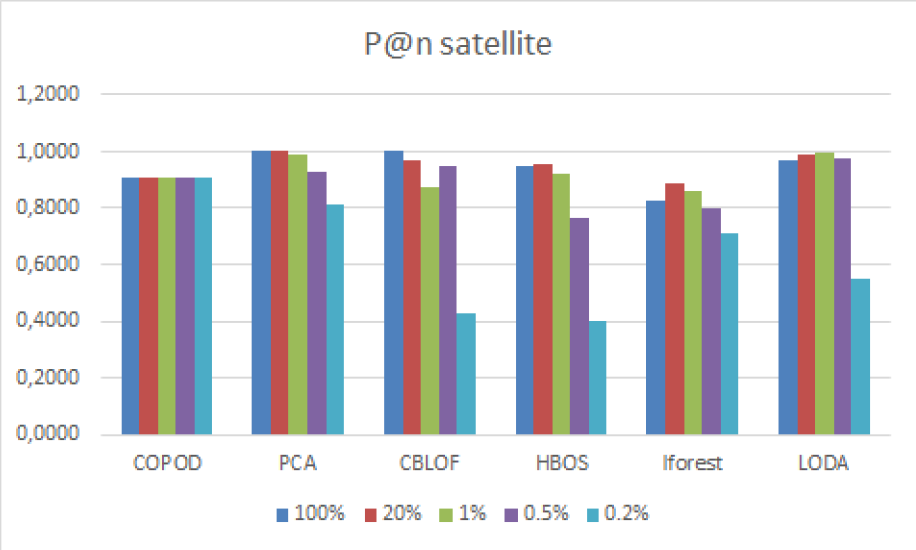
Obr. 7.20: Graf zobrazujúci pomer spotreby času pri učení modelu a predikcii s využitím 100% dostupných dát.



Obr. 7.21: Graf zobrazujúci pomer spotreby času pri učení modelu s s využitím 20% dát a predikcii s využitím 100% dát.



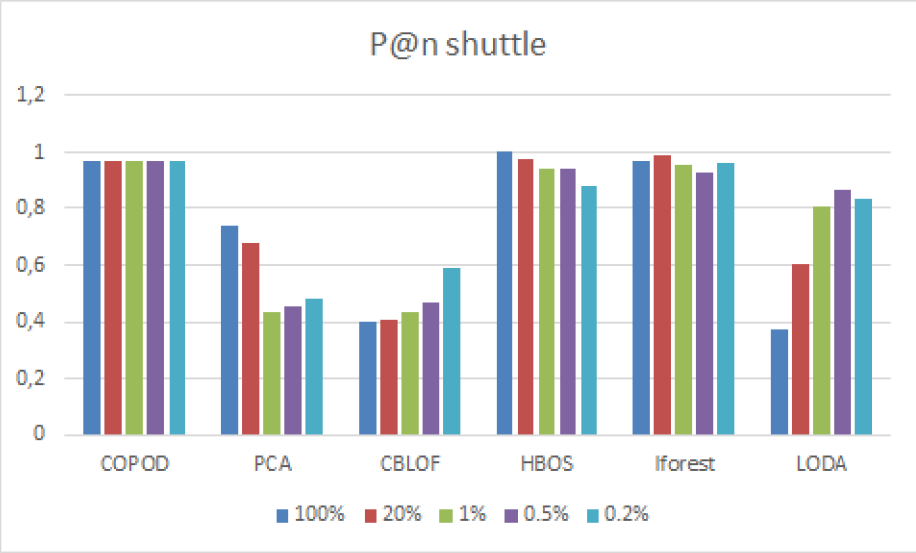
Obr. 7.22: Graf zobrazujúci pomer spotreby času pri učení modelov s využitím celej a zmenšenej dátovej sady.



Obr. 7.23: Graf zobrazujúci presnosti modelov pri zmene veľkosti využitých dát pri trénovaní modelov. Dáta sú z dátovej sady vybrané náhodne, využíva sa dátová sada satellite.

model	100%	20%	acceleration
COPOD	3,7246	0,6346	5,8690
PCA	0,2147	0,0385	5,5744
CBLOF	9,9300	2,0041	4,9549
HBOS	0,3240	0,0288	11,2427
Iforest	2,0254	0,4192	4,8317
LODA	1,4883	0,2635	5,6478

Tabuľka 7.26: Tabuľka zobrazuje čas trénovania modelu pri využití 100% a 20% dát. V stĺpci acceleration je vypočítané n-násobné zrýchlenie modelov.



Obr. 7.24: Graf zobrazujúci presnosti modelov pri zmene veľkosti využitých dát pri trénovaní modelov. Dáta sú z dátovej sady vybrané náhodne, využíva sa dátová sada shuttle.

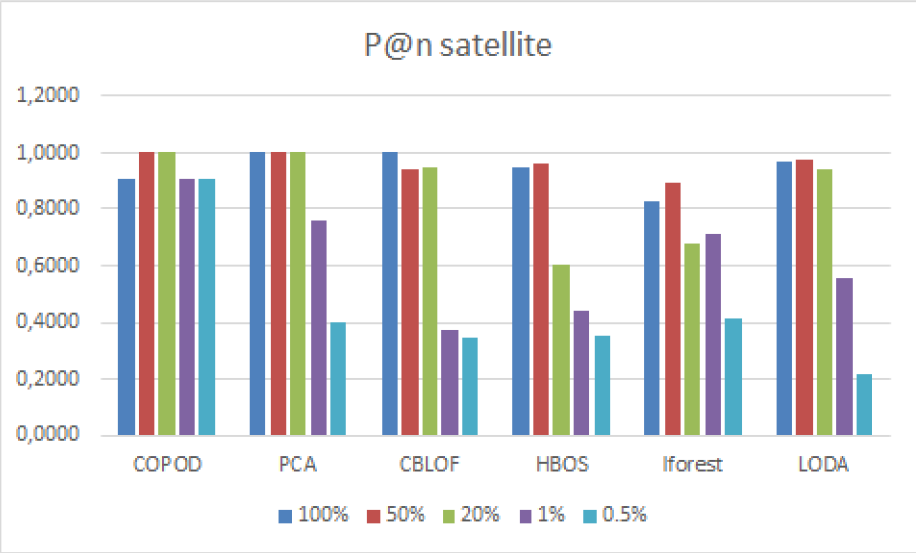
V tabuľke 7.1 je možné vidieť, že dátová sada shuttle obsahuje väčší počet dát ako dátová sada satellite. Preto je možné predpokladať, že presnosť metód bude stabilná pri menšom zlomku dát ako pri predchádzajúcom prípade. Tento predpoklad je potvrdený v grafe 7.24.

7.6.2 Vynechanie výberu náhodných dátových bodov

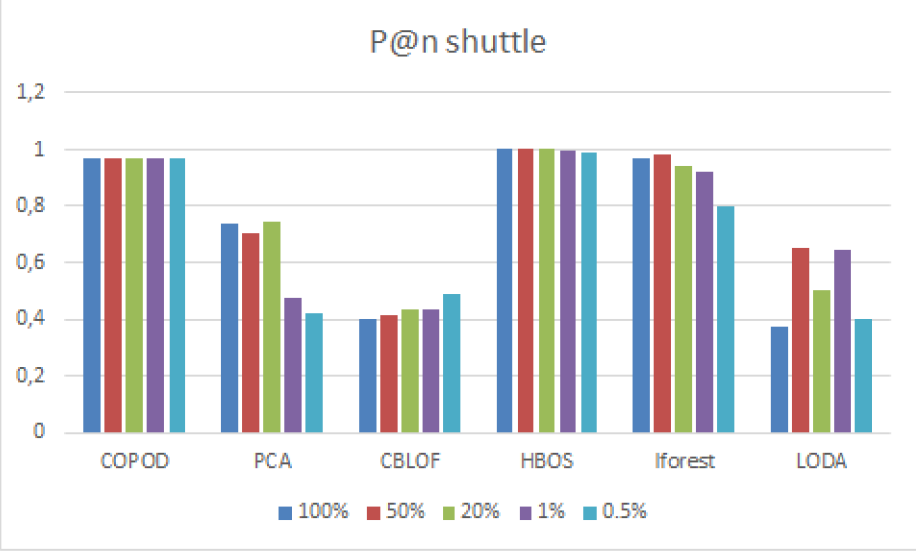
V tejto sekcii opakujeme experiment pre sledovanie presnosti zrýchlenia, no dátové body pre metódu `fit()` vyberáme zo začiatku dátovej sady. Toto riešenie pri využitých dátových sadách predkladá metóde len pravdivé dátové body, nakoľko sú dáta v súboroch vopred zoradené.

V prípade využitia dátovej sady satellite sa presnosť metód znížila výraznejšie ako pri využití náhodných dátových bodov. Túto skutočnosť je možné vidieť v grafe 7.25.

V rámci dátovej sady shuttle je v grafe 7.26 viditeľné, že presnosť modelu sa so zmenšením počtu dát výrazne nemení.



Obr. 7.25: Graf zobrazujúci presnosti modelov pri zmene veľkosti využitých dát pri trénovaní modelov. Dáta sú vybrané zo začiatku dátovej sady, využíva sa dátová sada satellite.



Obr. 7.26: Graf zobrazujúci presnosti modelov pri zmene veľkosti využitých dát pri trénovaní modelov. Dáta sú vybrané zo začiatku dátovej sady, využíva sa dátová sada shuttle.

Kapitola 8

Záver

Témou tejto práce bola analýza metód pre detekciou odľahlých hodnôt. Na začiatku tejto práce boli predstavené odľahlé hodnoty a následne boli popísané rôzne metódy ich detekcie. Ďalej boli uvedené vybraté dátové sady spolu s možnosťou ich generovania. Po poskytnutí teoretickej časti bol predstavený návrh aplikácie využívajúci predstavené možnosti a dátové sady pre ich analyzovanie. Následne boli predložené technológie určené k implementácii samotnej aplikácie. Ďalej bol poskytnutý popis samotnej implementácie. Pomocou implementovanej aplikácie boli vytvorené konfiguračné súbory, ktoré obsahujú navrhnuté experimenty. Výsledky experimentov boli v práci následne zdokumentované a zhodnotené.

Experimentálna časť sa venovala rozsiahlej skupine rôznych metód detekcie odľahlých hodnôt. Vo výsledku sa analýza skôr priblížila k porovnávaniu jednotlivých metód a vytvorených modelov. Rozsiahle množstvo vybraných metód taktiež spôsobila, že metódy neboli analyzované príliš detailne. Ponúka sa teda možnosť rozšírenia tejto práce, kde bude kladený väčší dôraz na dátové sety a chovanie jednotlivých modelov.

Ako rozšírenie riešenia bolo taktiež navrhnuté a aplikované urýchlenie priebehu detekcie. Navrhnuté riešenie bolo veľmi efektívne a rapídne znížilo čas potrebný pre prípravu modelu.

Pozitívnym dôsledkom experimentálnej časti je taktiež množstvo skúseností, získané s prácou s knižnicami pre detekciu odľahlých hodnôt. Najlepšie výsledky dosahovali metódy z knižnice sklearn, táto knižnica je odporúčaná. Nevýhodou tejto knižnice je ale malé množstvo dostupných metód. Metódy z knižnica pyOD taktiež dosahovali dobré výsledky, táto knižnica taktiež ponúka množstvo metód a preto je druhou odporúčanou knižnicou.

Pri analýze boli medzi sebou porovnané vybrané modely a najlepšie výsledky dosahovali modely COPOD, PCA, CBLOF, HBOS, IForest a LODA. Taktiež bolo ukázané, že modely nedosahujú rovnaké výsledky na všetkých dátových sadách. Tento fakt zdôrazňuje to, že detekcie odľahlých hodnôt nie je jednoduchá a vyžaduje porozumenie nielen modelom, ale aj samotným dátam, v ktorých chceme odľahlé hodnoty vyhľadať.

Literatúra

[1] Aggarwal, C. C. *Outlier analysis.* New York: Springer, 2013. ISBN 978-1-4614-6395-5.

[2] Dua, D. a Graff, C. *UCI Machine Learning Repository.* 2017. Dostupné z: http://archive.ics.uci.edu/ml.

[3] Fawcett, T. An introduction to ROC analysis. *Pattern recognition letters.* Elsevier. 2006, zv. 27, č. 8, s. 861–874.

[4] Han, J. *Data Mining: Concepts and Techniques.* 3rd ed.. 2011. The Morgan Kaufmann Series in Data Management Systems Ser. ISBN 9780123814807.

[5] Järvelin, K. a Kekäläinen, J. IR evaluation methods for retrieving highly relevant documents. In: ACM New York, NY, USA. *ACM SIGIR Forum.* 2017, sv. 51, č. 2, s. 243–250.

[6] Kumar, K. M. a Reddy, A. R. M. A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method. *Pattern Recognition.* Elsevier. 2016, zv. 58, s. 39–48.

[7] Li, Y., Zha, D., Zou, N. a Hu, X. PyODDS: An End-to-End Outlier Detection System. 2019.

[8] McKinney Wes. Data Structures for Statistical Computing in Python. In: Walt Stéfan van der a Millman Jarrod, ed. *Proceedings of the 9th Python in Science Conference.* 2010, s. 56 – 61. DOI: 10.25080/Majora-92bf1922-00a.

[9] Ord, K. Outliers in statistical data : V. Barnett and T. Lewis, 1994, 3rd edition, (John Wiley & Sons, Chichester), 584 pp., [UK pound]55.00, ISBN 0-471-93094-6. *International Journal of Forecasting.* March 1996, zv. 12, č. 1, s. 175–176. Dostupné z: https://ideas.repec.org/a/eee/intfor/v12y1996i1p175-176.html.

[10] Rayana, S. *ODDS Library.* 2016. Dostupné z: http://odds.cs.stonybrook.edu.

[11] Reynolds, D. A. Gaussian Mixture Models. *Encyclopedia of biometrics.* Berlin, Springer. 2009, zv. 741, s. 659–663.

[12] Upton, G. a Cook, I. *Understanding statistics.* Oxford University Press, 1996.

[13] Upton, G. a Cook, I. *Statistics.* Oxford University Press, 2008. DOI: 10.1093/acref/9780199541454.013.1566. Dostupné z: https://www.oxfordreference.com/view/10.1093/acref/9780199541454.001.0001/acref-9780199541454-e-1566.

[14] V, M. *Study of outlier detection methods In data mining*. 2016. Dizertačná práca. Annamalai University. Dostupné z: http://hdl.handle.net/10603/154074.

[15] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, zv. 17, s. 261–272. DOI: 10.1038/s41592-019-0686-2.

[16] Zhao, Y., Nasrullah, Z. a Li, Z. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*. 2019, zv. 20, č. 96, s. 1–7. Dostupné z: http://jmlr.org/papers/v20/19-011.html.

Príloha A

Využité knižnice pri implementácii programu

Táto sekcia naväzuje na kapitolu 5 a uvádza všetky knižnice ktoré boli potrebné pri implementácii programu. Niektoré knižnice využité pri implementácii sú závislé od ďalších knižníc, ktoré bolo pri implementácii potrebné a bolo dôležité ich pre správnu prácu interpreta nainštalovať. Zoznam všetkých knižníc je nasledujúci: Click, Cython, Keras, Keras-Applications, Keras-Preprocessing, Markdown, Pillow, PyNomaly, PyYAML, Werkzeug, absl-py, astor, astunparse, atomicwrites, attrs, cached-property, cachetools, certifi, chardet, colorama, combo, cycler, dataclasses, et-xmlfile, future, gast, google-auth, google-auth-oauthlib, google-pasta, grpcio, h5py, hdbscan, idna, importlib-metadata, iniconfig, isotree, joblib, kiwisolver, llvmlite, luminol, matplotlib, numba, numexpr, numpy, oauthlib, openpyxl, opt-einsum, outliertree, packaging, pandas, pandastable, patsy, pip, pluggy, protobuf, py, pyasn1, pyasn1-modules, pyod, pyodds, pyparsing, pytest, python-dateutil, python-utils, pytz, requests, requests-oauthlib, rsa, scikit-learn, scipy, seaborn, setuptools, six, sklearn, statsmodels, suod, taos, tb-nightly, tensorboard, tensorboard-plugin-wit, tensorflow, tensorflow-gpu, tensorflow-gpu-estimator, termcolor, tf-estimator-nightly, threadpoolctl, toml, torch, torchaudio, torchvision, tqdm, typing-extensions, urllib3, wheel, wrapt, xgboost, xlrd, zipp.

Príloha B

Obsah pamäťového média

Táto príloha popisuje obsah pamäťového média priloženého k výslednej práci.

```
Root
├── analyzer
│   ├── data
│   ├── output
│   ├── action_dispatch.py
│   ├── config.py
│   ├── data_generator.py
│   ├── data_reader.py
│   ├── ensembles.py
│   ├── graphic.py
│   ├── main.py
│   ├── methods.py
│   ├── methods_buttons.py
│   ├── methods_other_tester.py
│   ├── model_gen.py
│   ├── model_tester.py
│   ├── output_handler.py
│   ├── Readme
│   ├── GNU
│   └── configurations
│       ├── Combination.json
│       ├── Ensembles.json
│       ├── General.json
│       ├── Iforest_estimators.json
│       ├── KNN.json
│       ├── Linear_Models.json
│       ├── Neural_Networks.json
│       ├── OCSVM.json
│       ├── Probabilistic.json
│       ├── Proximity_Based.json
│       ├── PyOD_bechmarks.json
│       └── Single_trials.json
├── thesis.pdf
└── source
```

Príloha C

Manuál

Táto príloha obsahuje manuál k programu určenému k analýze. Pri implementácii boli využité knižnice, ktoré je možné vidieť v prílohe A. Spustenie programu prebieha pomocou príkazu:

```
python main.py
```

K spusteniu je taktiež odporúčané využiť vývojové prostredie, ako je IDE pyCharm, ktoré bolo využité pri implementácii. Nasledujúce sekcie popisujú možnosti práce s programom. Sekcie sú rozdelené podľa zobrazeného obsahu okna.

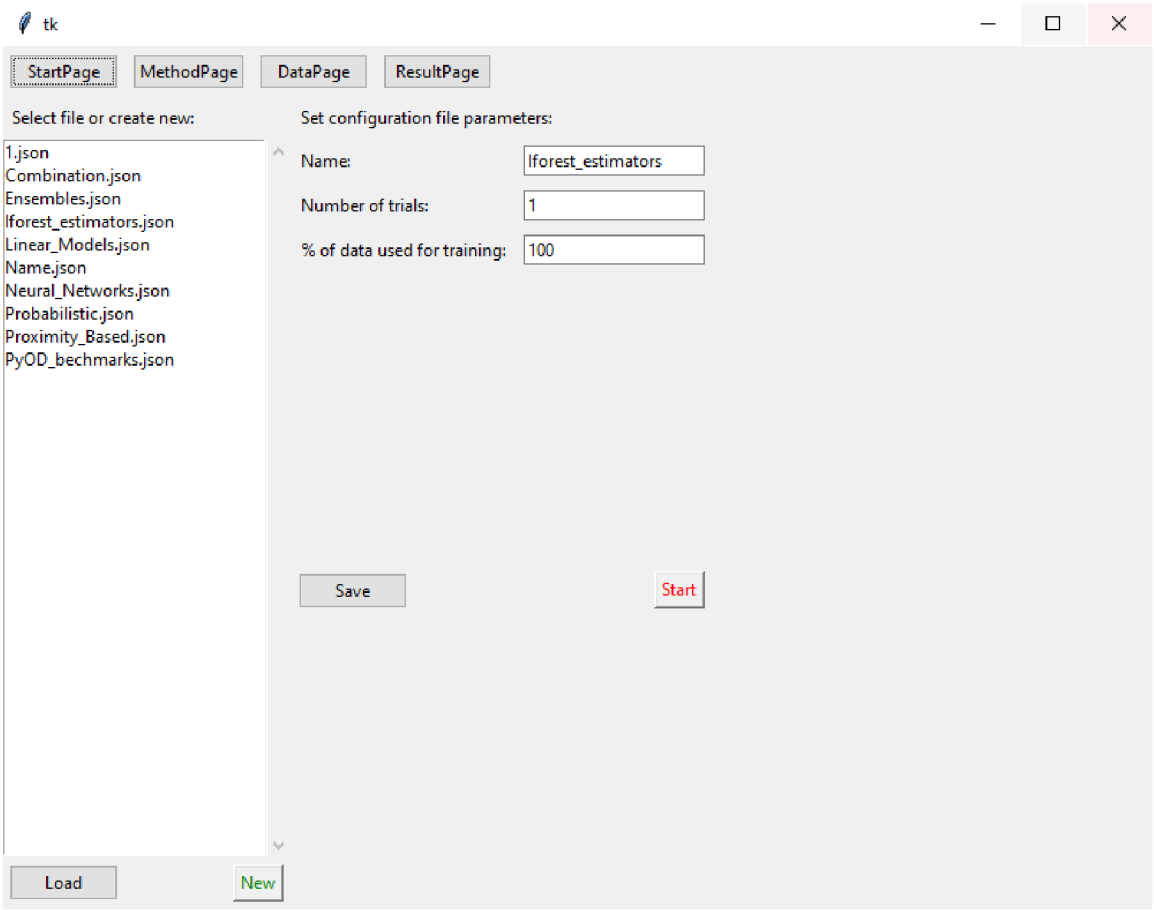
C.1 Práca s konfiguračnými súbormi

Dizajn úvodnej stránky je zobrazený na obrázku C.1. Táto stránka slúži pre načítanie, vytváranie a ukladanie konfiguračných súborov, ako aj k ich spúšťaniu. K jej zobrazeniu slúži tlačidlo *StartPage* nachádzajúce sa na hornej lište okna.

Na ľavej strane obrazovky sa nachádza zoznam s dostupnými a uloženými konfiguračnými súbormi. Konfiguračné súbory je možné načítať buď to pomocou dvojkliku alebo pomocou tlačidla *Load* nachádzajúceho sa v ľavom rohu pod zoznamom. Pod zoznamom sa taktiež nachádza zelene označené tlačidlo *New*, pomocou ktorého užívateľ môže aplikáciu informovať o tom, že vytvára nový konfiguračný súbor. Pri spustení aplikácie je užívateľské rozhranie automaticky pripravené pre vytváranie nového konfiguračného súboru, teda tlačidlo *New* nieje potrebné stláčať.

Pravá strana obrazovky slúži k zadávaniu informácií o konfiguračnom súbore a jeho spustení. Informácie, ktoré používateľ môže zadať sú názov konfiguračného súboru, počet jeho spustení pre presnejšie odvodenie výsledku a percentuálne určenie veľkosti dát, ktoré majú byť použité pre trénovanie metód. Taktiež sa tu nachádzajú tlačidlá *Save* slúžiace k zadaniu pokynu pre uloženie konfiguračného súboru. Červene označené tlačidlo *Start* slúži k spusteniu experimentu. Toto tlačidlo zostáva zobrazené ako stlačené počas celého priebehu experimentu a po jeho úspešnom dokončení sa vráti do úvodnej podoby.

Obsah a množstvo možností pre úpravu údajov nachádzajúcich sa na úvodnej stránky je väčší ako pri aplikáciách pre širšiu skupinu používateľov. Takéto vypracovanie úvodnej strany bolo možné z toho dôvodu, že pri používaní aplikácie sa predpokladá, že používateľ je o ovládaní vopred informovaný a má predošlé znalosti o metódach pre detekciu odľahlých hodnôt.



Obr. C.1: Ukážka úvodného okna aplikácie.

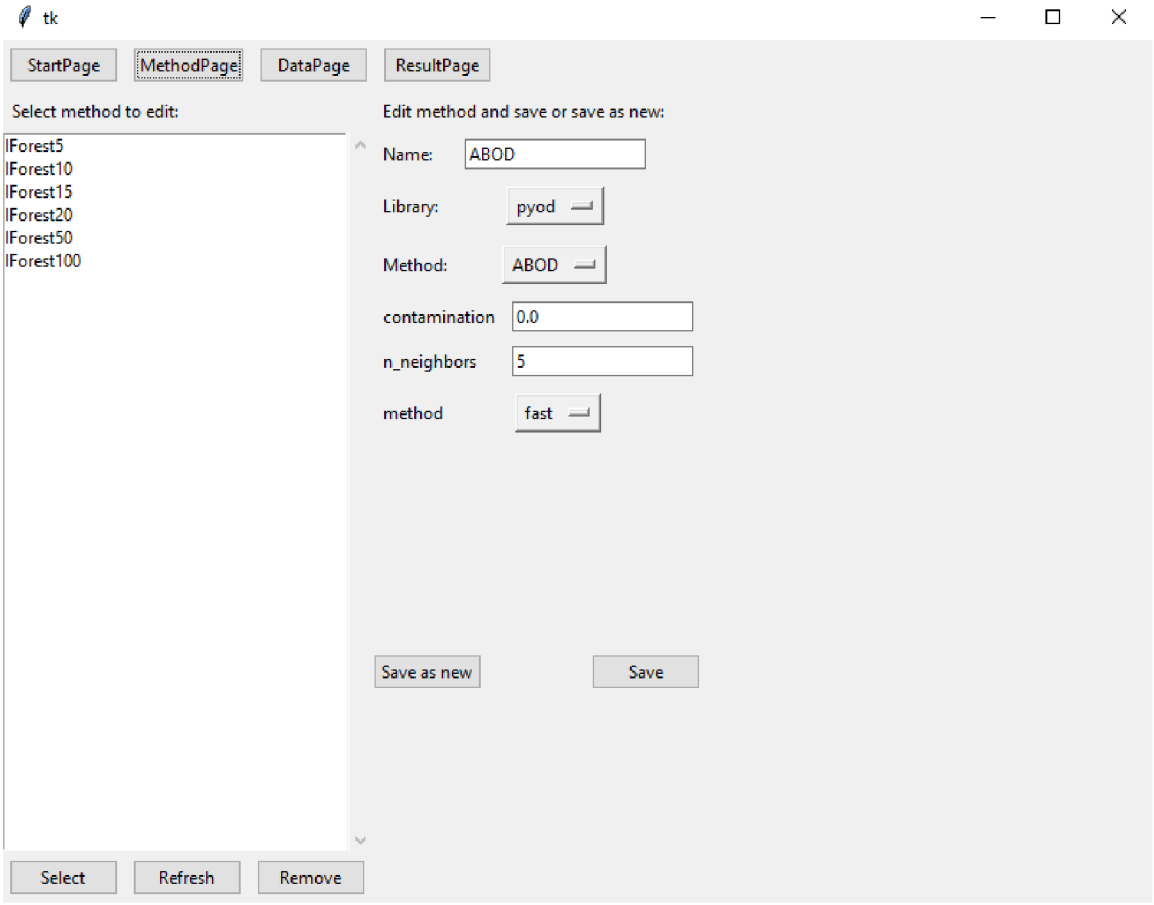
C.2 Zostavenie zoznamu modelov

Stránka pre vytváraní modelov pre detekciu odľahlých hodnôt je implementovaná pomocou triedy *MethodPage*. Príklad stránky je zobrazený na obrázku C.2. K jej zobrazeniu slúži tlačidlo *MethodPage* nachádzajúce sa na hornej lište okna.

Navrhnuté modely sú uložené v zozname nachádzajúcom sa na ľavej strane stránky. Modely sú uložené pod používateľom zvolenom názve v poradí, ako sú do zoznamu pridávané. Pod zoznamom sa nachádzajú tlačidlá *Select*, *Refresh* a *Remove*. Tlačidlo *Select* slúži k vyberaniu modelu zo zoznamu, model je taktiež možné vybrať pomocou dvojkliku priamo v zozname. Informácie o modeli sa následne zobrazia v pravej časti stránky, kedy je možné vybraný model ľubovoľne upravovať. Pomocou tlačidla *Refresh* sa stránka obnoví. Toto tlačidlo umožňuje zotavenie z možného výskytu chyby alebo špatného zobrazenia zoznamu. Model je zo zoznamu možné odstrániť pomocou tlačidla *Remove*.

V pravej časti sa zobrazujú informácie o aktuálne vytváranom, prípadne upravovanom modeli. Vytváranie modelu prebieha pomocou tlačidla *Save as new*. V prípade, že používateľ vybral model zo zoznamu môže ho prepísať pomocou tlačidla *Save*. V prípade, že žiadny model nieje upravovaný, toto tlačidlo model uloží ako nový. Do textového pola označeného ako *Name* sa vkladá názov označenia modelu. V menu *Library* je možné vyberať medzi knižnicami, ktoré implementujú metódy pre vytváranie samotných modelov. Tieto metódy sa potom zobrazia v menu *Method*. V strede stránky sa zobrazujú textové polia a výberové

tlačidlá podľa toho, aký model je práve vybraný. A to z toho dôvodu, že každý model požaduje od užívateľa rôzne parametre.



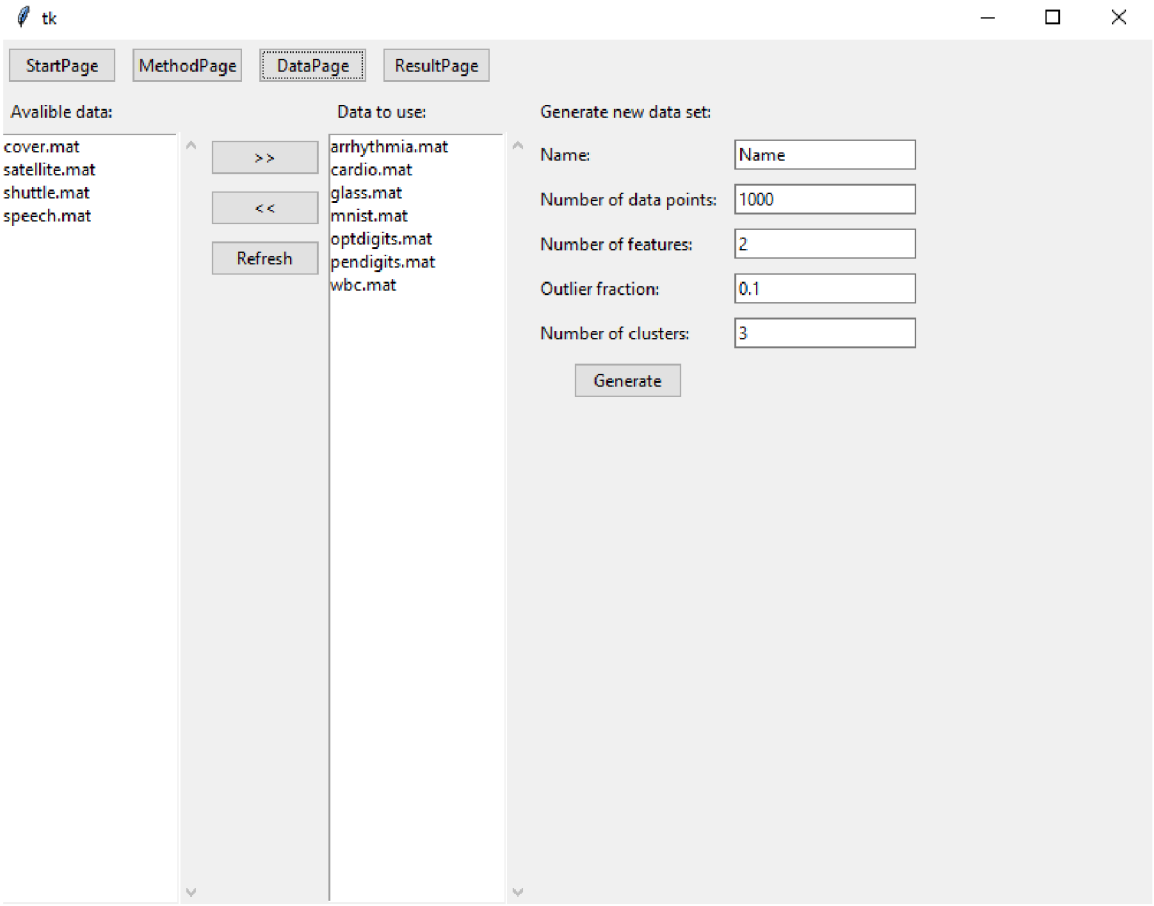
Obr. C.2: Ukážka okna aplikácie pre vytváranie zoznamu metód.

C.3 Zostavenie zoznamu dátových sád

DataPage je stránka, ktorá slúži na výber dátových sád pre experimenty. Je tu taktiež umiestnený generátor dátových sád. Ukážka tejto stránky je zobrazená na obrázku C.3. K jej zobrazeniu slúži tlačidlo *DataPage* nachádzajúce sa na hornej lište okna.

Ľavá strana stránky slúži pre výber dátových sád. Nachádzajú sa tu dva zoznamy, z ktorých ľavý zobrazuje dostupné dátové sady a pravý zobrazuje dátové sady, ktoré sú už do experimentu zahrnuté. Pri vybratí dátovej sady je jej názov z aktuálneho zoznamu odstránený a dátová sada je pridaná do druhého zoznamu. Tento postup je rovnaký pre obidva zoznamy. Presunutie dátovej sady je možné buď dvojitým kliknutím na jeho názov v zozname alebo využitie tlačidiel » a «. Tlačidlá naznačujú smer pohybu, teda tlačidlo » pridáva dátovú sadu z dostupných dátových sád do vybraných. Tlačidlo « naopak odstráni vopred vybratú dátovú sadu a vloží ju do dostupných. K dispozícii je taktiež tlačidlo *Refresh*, ktoré slúži na nové vykreslenie stránky. Toto tlačidlo taktiež zoradí dátové sady podľa abecedy.

Pravá strana stránky obsahuje textové polia, ktoré reprezentujú vstup nastavenia generátoru dátových sád. Pole označené ako *Name* slúži k pomenovaniu vygenerovaného súboru obsahujúceho dáta. Následne je možné vložiť počet vygenerovaných dátových bodov, počet vlastností dátových bodov, taktiež reprezentujúci počet dimenzií dát. Textové pole *Outlier fraction* predstavuje zlomok dát, ktoré budú generované ako odľahlé hodnoty. Posledné textové pole reprezentuje počet zhlukov, ktoré sa v dátach budú vyskytovať. Súbor s vygenrovanými dátami je uložený do zložky *data*.



Obr. C.3: Ukážka okna aplikácie pre zostavenie zoznamu dátových sád určených pre experiment.

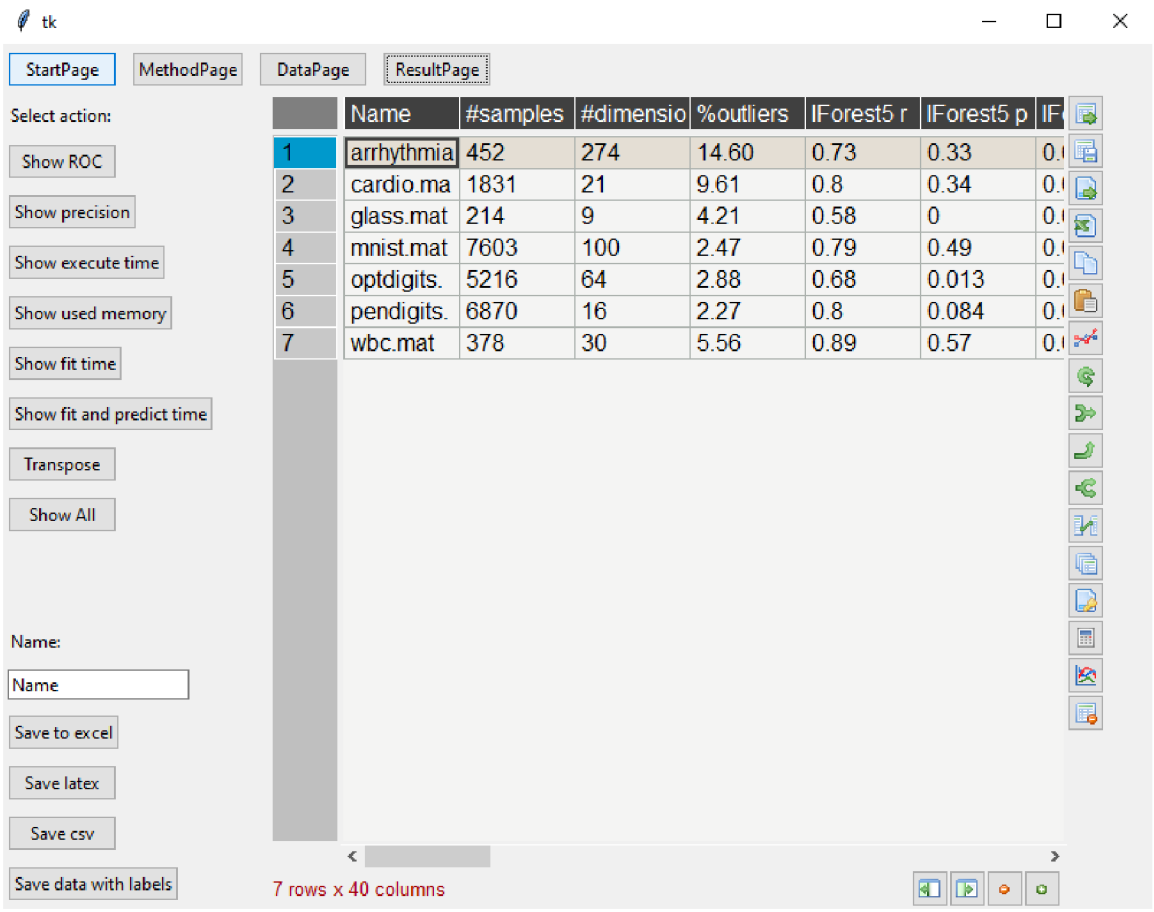
C.4 Zobrazenie výsledkov

Po úspešnom prebehnutí experimentu je možné výsledky zobraziť vo forme tabuľky v aplikácii na stránke *ResultPage*. Tabuľka s výsledkami sa nachádza v pravej časti stránky ako je možné vidieť aj na obrázku C.4. K jej zobrazeniu slúži tlačidlo *ResultPage* nachádzajúce sa na hornej lište okna.

Vykreslená tabuľka je interaktívna a používateľ môže označovať, mazať a pridávať stĺpce a riadky. Tabuľka taktiež ponúka možnosť vykreslenia grafov. Stĺpce môžu byť využité na indexovanie riadkov pre lepšie vykresľovanie grafov.

V ľavej časti sa nachádza skupina tlačidiel *Button*, ktoré používateľovi uľahčia zobrazovanie obsahu tabuľky. Tlačidlo *Show All* zobrazí všetky informácie získané z experimentu. Pretože často krát je potrebné sa zamerať len na jednu konkrétnu informáciu, sú k dispozícii tlačidlá, ktoré ju z tabuľky vystrihnú a zobrazia. Tlačidlo *Show ROC* zobrazí hodnotu reprezentujúcu obsah nachádzajúci sa pod krivkou ROC. Tlačidlo *Show precision* zobrazí vypočítanú presnosť P@n. Čas potrebný pre detekciu odľahlých hodnôt je zobrazený tlačidlom *Show execute time*, v prípade, kedy chceme vidieť čas, ktorý je potrebný pre trénovanie a detekciu oddelene sú k dispozícii tlačidlá *Show fit time* a *Show fit and predict time*. Tlačidlo *Show used memory* zobrazí pamäť využitú pri detekcii. Na záver je tu taktiež tlačidlo *Transpose*, ktoré transponuje tabuľku a stĺpce zobrazí na riadkoch.

Možnosti pre ukladanie výsledkov sa nachádzajú v ľavom spodnom rohu okna. Do textového pola užívateľ zadá názov výsledného súboru a tlačidlom zvolí typ uloženého súboru. Tlačidlo *Save to excel* výslednú tabuľku uloží do súboru *.xlsx*, ktorý je možné otvoriť pomocou aplikácie Microsoft Excel. Táto možnosť je zahrnutá pre uľahčenie práce s výsledkami a pridaním ďalšej možnosti pre tvorbu špecifických grafov. Tlačidlo *Save latex* uloží tabuľku vo formáte *.tex*, táto možnosť uľahčí vkladanie výsledkov do textového editoru LaTeX. Poslednou možnosťou je tabuľku uložiť vo formáte *.csv*. Implementovaná je taktiež možnosť uloženia dát spolu s pravdivostným označením odľahlých hodnôt, ktoré boli získané pri behu experimentu. Toto uloženie sa spustí pomocou tlačidla *Save data with labels*. Výsledný súbor s dátami a ich označením je vo formáte *csv*. Súbory s výsledkami sú uložené do zložky *output*.



Obr. C.4: Ukážka okna aplikácie zobrazujúceho výsledok vykonaného experimentu.