ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

Bc. MARIÁN KAŠUBA

Aplikácia klasifikačných algoritmov na medicínske údaje

Vedúci práce: Ing. Ján Rabčan PhD.

Registračné číslo: 1923/2021

Ministerské číslo práce: 28360020222089

Žilina, 2022

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ

PRÁCA

ŠTUDIJNÝ ODBOR: INFORMATIKA

Bc. MARIÁN KAŠUBA

Aplikácia klasifikačných algoritmov na medicínske údaje

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Katedra Informatiky
Žilina, 2022

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program: Biomedicínska informatika

Meno a priezvisko
Marián Kašuba

Osobné číslo
559852

Názov práce v slovenskom aj anglickom jazyku

Aplikácia klasifikačných algoritmov na medicínske údaje

Application of classification algorithms to medical data

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:
Cieľom práce je implementácia vybraných klasifikačných algoritmov a porovnanie ich vlastností pri spracovaní medicínskych dát. Pomocou implementovaných algoritmov budú následne vykonané experimenty na vopred zvolených medicínskych dátach.

Obsah: Navrhovaný postup:

- Analýza súčastného stavu
- Popis vybraných algoritmov
- Implementačný návrh a implementácia algoritmov
- Porovnanie algoritmov

Meno a pracovisko vedúceho DP: Ing. Ján Rabčan, PhD., KI, ŽU
Meno a pracovisko tútora DP:

- 7 MAR. 2022

vedúci katedry
(dátum a podpis

Zadanie zaregistrované dňa - 3 MAR. 2022 pod číslom 1923/2021 podpis

Čestné vyhlásenie

Čestne vyhlasujem, že som bakalársku prácu s názvom Aplikácia klasifikačných algoritmov na medicínske údaje vypracoval samostatne s použitím literatúry, ktorej zoznam som uviedol na príslušnom mieste.

Žilina 5.4.2022 .....

Bc. Marián Kašuba

Poďakovanie

Chcel by som poďakovať môjmu vedúcemu práce Ing. Jánovi Rabčanovi PhD. za jeho odbornú pomoc a ochotu pri písaní mojej diplomovej práce.

ABSTRAKT V ŠTÁTNOM JAZYKU

KAŠUBA, Marián: *Aplikácia klasifikačných algoritmov na medicínske údaje*. Diplomová práca. – Žilinská univerzita v Žiline. Fakulta riadenie a informatiky; Katedra Informatiky. – Vedúci: Ing. Ján Rabčan, PhD. – Stupeň odbornej kvalifikácie: Inžinier v študijnom programe Biomedicínska informatika, Žilina: FRI UNIZA, 2022. Počet strán 75.

Cieľom diplomovej práce bolo obohatenie knižnice WEKA o varianty, ktoré rozširujú základný klasifikačný algoritmus na hľadanie K-najbližších susedov. Varianty, ktoré boli implementované pri určitých dátových množinách zvyšujú celkovú presnosť klasifikácie.

Na úvod bola potrebná teoretická analýza hĺbkovej analýzy dát, klasifikačných algoritmov a ich výhod, nevýhod alebo prínosov v oblasti medicíny. Po získaní teoretických znalostí základného algoritmu KNN sme mohli preskúmať možnosti ako tento algoritmus vylepšiť. Zlepšenie bolo možné nielen z časového hľadiska prostredníctvom pokročilých údajových štruktúr Ball-Tree alebo Kd-Tree, ale aj z hľadiska presnosti prostredníctvom variant weighted KNN, fuzzy KNN alebo KNN na princípe harmonického stredu.

Následne bola vykonaná implementácia štruktúr a variant v súlade s pravidlami knižnice WEKA, čo umožňuje jednoduché prepojenie knižnice s rozšírením.

Nakoniec sme vykonali porovnanie implementovaných štruktúr a variant prostredníctvom experimentov a dokázali sme význam implementácie štruktúr a variant.

Kľúčové slová: Hĺbková analýza dát, KNN, WEKA

ABSTRACT

KAŠUBA, Marián: Application of classification algorithms to medical data. Diploma thesis. – University of Zilina. Faculty of Management Science and Informatics; Department of Informatics. – Supervisor: Ing. Ján Rabčan PhD. – Qualification degree: Engineer in Biomedical Informatics, Žilina: FRI UNIZA, 2022. Number of pages 75.

The goal of the diploma thesis was to enhance the WEKA library with variants that extend the basic classification algorithm for searching the k nearest neighbors. Variants that have been implemented for certain datasets increase the overall accuracy of the classification.

However, a theoretical analysis of in-depth data analysis, classification algorithms, and their advantages, disadvantages, or benefits in the field of medicine was needed at the beginning. After gaining theoretical knowledge of the basic KNN algorithm, we were able to explore ways to improve this algorithm. The improvement was possible not only in terms of time through advanced data structures Ball-Tree or Kd-Tree but also in terms of accuracy through variants such as weighted KNN, fuzzy KNN, or KNN on the principle of harmonic center.

Subsequently, the structures and variants were implemented in accordance with the rules of the WEKA library, which allows easy connection of the library with the extension.

Finally, we compared the implemented structures and variants by experiments, and we demonstrated the importance of implementing structures and variants.

Key words: Data mining, KNN, WEKA

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Vďaka bakalárskemu štúdiu informatiky na Fakulte riadenia a informatiky na Žilinskej Univerzite v Žiline som nadobudol znalosti z oblasti informatiky. Keďže pochádzam z rodiny, v ktorej sú štyria príbuzný lekári, rozhodol som sa, že si vyberiem inžinierske štúdium so zameraním práve na biomedicínsku informatiku. Záverečnou skúškou štúdia je diplomová práca, a preto som si vybral tému, ktorá by moju študentskú cestu, čo najviac odzrkadľovala. Vybraná diplomová práca preto zahŕňa oblasť informatiky a oblasť medicíny. Informatická časť predstavuje implementáciu algoritmov, pokročilých údajových štruktúr, vytvorenie implementačného návrhu a prácu s knižnicou. Medicínska časť predstavuje údaje, ktoré boli spracované. Každá práca, či už diplomová alebo bakalárska, prípadne výskum musí mať presne definovaný cieľ. Cieľ diplomovej práce je možné rozdeliť do nasledujúcich 4 fáz.

1. **Fáza analýzy súčasného stavu**

Počas tejto fázy, ktorá predstavuje teoretickú časť diplomovej práce, sme analyzovali hĺbkovú analýzu dát zo širšieho hľadiska. Cieľom bolo teda popísať hĺbkovú analýza dát ako pojem, jej princíp a proces fungovania. Následne sme analyzovali medicínske dáta, techniky a algoritmy využívané v hĺbkovej analýze dát. Väčší dôraz bol venovaný klasifikácií a klasifikačným algoritmov, pričom vybraný algoritmus bol popísaný spoločne so štúdiou, ktorá ho využila v sfére medicíny.

2. **Fáza popisu vybraného algoritmu, jeho modifikácií, či variant.**

Algoritmus, ktorý je predmetom skúmania a implementácie sa nazýva algoritmus K-najbližšieho suseda alebo skrátené KNN. Algoritmus bol modifikovaný prostredníctvom pokročilých údajových štruktúr Kd-Tree a Ball-Tree. Jeho modifikácia spočívala aj v implementácií rôznych variant. V druhej fáze bolo teda potrebné porozumieť, analyzovať a na záver spracovať samotný algoritmus, štruktúry a varianty.

3. **Fáza implementácie**

Algoritmy, ktoré boli popísané v druhej fáze boli implementované na základe konvencií knižnice WEKA ako prípadné rozšírenie do budúcnosti, vzhľadom na to, že WEKA nepodporuje určité popísané varianty. Z praktickej časti cieľ diplomovej práce spočíval práve v implementácií algoritmov štruktúr a variant.

4. Experimenty a vyhodnotenie

V poslednom fáze prebehli rôzne testy a experimenty na určených dátach za účelom porovnania implementovaných algoritmov.

Vzhľadom na to, že cieľ diplomovej práce bol definovaný, bolo potrebné tieto ciele splniť prostredníctvom kapitol na základe, ktorých bola práca rozdelená. **Prvá kapitola** predstavuje takzvaný teoretický úvod a spracováva prvú fázu nami definovaného cieľa. Popisuje medicínske dáta techniky a algoritmy hĺbkovej analýzy dát v sfére medicíny, príklady súčasného využitia a následné výhody a nevýhody. **Druhá kapitola** zadefinuje vlastnosti, popisuje slovné postupy a objasňuje prostredníctvom pseudokódu údajové štruktúry a varianty implementované v diplomovej práci. **Tretia kapitola** popisuje spôsoby, akým je možné validovať a ohodnotiť klasifikačné algoritmy. Medzi tieto spôsoby radíme maticu zámen a krížovú validáciu. **Štvrtá kapitola** v úvode popisuje knižnicu WEKA. Následne definuje implementačný návrh a samotnú implementáciu. **Piata kapitola** porovnáva implementované algoritmy z hľadiska rýchlosti a presnosti.

Vďaka implementácii praktickej časti diplomovej práce je knižnica WEKA obohatená o varianty algoritmu KNN. Medzi tieto varianty radíme variant fuzzy KNN a variant fungujúci na princípe harmonického stredu, ktorý sa nazýva HMDKNN.

1 Úvod do hĺbkovej analýzy dát v medicíne

V posledných rokoch sa nielen v oblasti informačných technológií stále viac a viac stretávame s pojmom hĺbková analýza dát. Hĺbková analýza dát sa v súčasnej dobe dostáva do popredia aj v iných oblastiach. Medzi tieto oblasti radíme vzdelávanie, analýzu finančného trhu, odhaľovanie podvodov, finančné bankovníctvo, zdravotnú starostlivosť, bio informatiku a iné. Napriek tomu, že hĺbková analýza dát sa využíva v mnohých odvetviach, diplomová práca skúma a analyzuje len oblasť, ktorá súvisí s medicínou [1].

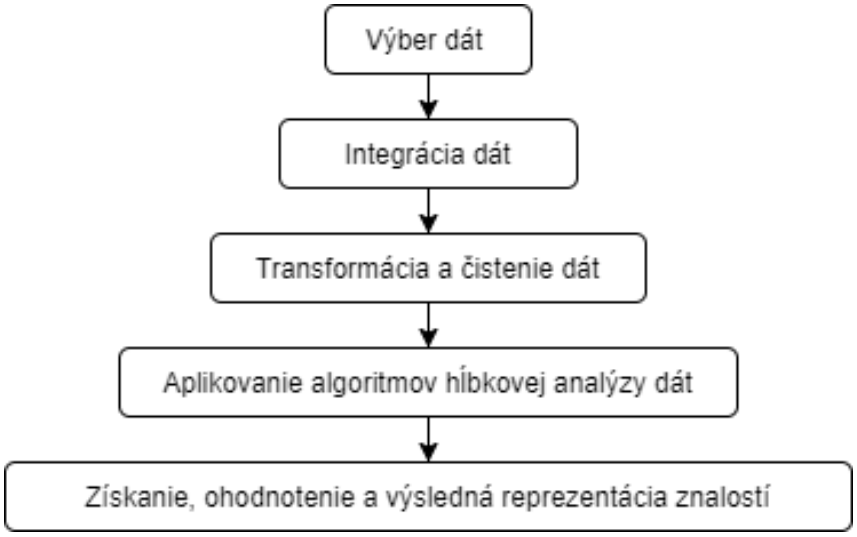
Je potrebné si uvedomiť, že pojem hĺbková analýza dát sa z časti líši, vzhľadom na odvetvie, v ktorom sa používa. Jedna z najčastejších, respektíve najznámejších definícií popisuje hĺbkovú analýzu dát ako proces spracovania veľkého množstva dát. Cieľom tohto spracovania je identifikácia vzorov a vytvorenie vzťahov za účelom riešenia problémov. Tento proces sa vykonáva prostredníctvom analýzy údajov [2].

Aby sme mohli vôbec vykonávať hĺbkovú analýzu dát je potrebné zabezpečiť dve veci, ktoré vyplývajú už zo samotnej definície [2].

- Dáta samotné, pričom je potrebné pracovať s väčším množstvom dát.
- Výpočtový výkon, ktorý bude schopný pracovať s údajmi.

Je zrejmé, že dáta, ich obsah, štruktúra a vlastnosti sú pri hĺbkovej analýze dát veľmi dôležité. Vzhľadom na túto skutočnosť platí, že čím viac sú dáta organizované, tým je ľahšie získať z nich užitočné informácie [2].

Hĺbková analýza dát ako proces je reprezentovaná v niekoľkých krokoch. Ako prvý krok je výber dát. Výber dát vykonáme podľa požiadaviek našej štúdie alebo výskumu. Pri výbere dát je možné využiť viaceré zdroje, z čoho logicky vyplýva ďalší krok integrácia dát. Pod integráciou dát si predstavíme skombinovanie dát z viacerých zdrojov do jedného. Nastane filtrovanie, čistenie a transformovanie dát. Keďže dáta samotné nedokážu vykonať požadované úlohy, je potrebné vytvoriť model a aplikovať algoritmy za účelom získania znalostí. Získané znalosti na záver prebehnú procesom ohodnotenia. Dosiahnuté výsledky odprezentujeme. Popísaný postup bol pre lepšiu predstavu a porozumenie graficky znázornený obrázkom 1 [3].



Obr. 1 Hĺbková analýza dát ako proces

Pod slovom model v oblasti hĺbkovej analýzy dát sú prezentované dva druhy modelov. Prediktívny model a deskriptívny model. Prediktívny model funguje na princípe strojového učenia s učiteľom za cieľom predikcie. Predmetom predikcie sú neznáme alebo budúce hodnoty, ktoré budú predikované na základe známej množiny dát, ktorej je predikovaná hodnota známa. Deskriptívny model funguje na princípe strojového učenia bez učiteľa za cieľom nájdenia vzorov a závislosti na skúmanej množine dát. Strojové učenie je schopnosť programu adekvátne reagovať na rôzne vstupné hodnoty, bez toho, aby bol naprogramovaný program práve na tieto vstupné hodnoty. Program vychádza len zo znalostí, ktoré sa naučil, z čoho aj vyplýva jeho názov strojové učenie [4].

1.1 Medicínske dáta

Dáta, ktoré sú predmetom spracovania pri hĺbkovej analýze dát vo sfére medicíny sa nazývajú medicínske dáta. Pod medicínskymi dátami si môžeme predstaviť akýkoľvek typ dát, ktorý nesie medicínsku informáciu. Existujú rôzne medicínske štandardy ako napríklad nasledovné štandardy [5].

- **CPT kódy** popisujú ambulantné služby a postupy na účely sledovania liečby a fakturácie.
- **HCPCS kódy** sú rozšírením CPT kódov určené pre všetky druhy služieb s prepojením na medicínsku starostlivosť.
- **LOINC kódy** označujú laboratórne objednávky a výsledky.

- **DICOM** predstavuje medzinárodný komunikačný protokol a formát súborov na výmenu lekárskych snímok a súvisiacich údajov medzi hardvérovými a softvérovými nástrojmi. Ako príklad nástrojov je možné uviesť magnetickú rezonanciu a počítačovú tomografiu.

Spôsoby akým môžu byť dáta reprezentované je rôzny a je možné ho rozdeliť do nasledujúcich kategórií [6].

- **Numerické dáta** predstavujú číselné hodnoty. V medicíne sa ako číselné hodnoty udávajú predovšetkým tlak, pulz, teplota a fyzikálne vlastnosti pacienta.
- **Textové údaje** môžu predstavovať rôzne informácie o pacientovi ako je krvná skupina alebo rh faktor pacienta. Prostredníctvom textu sa udávajú aj hodnoty skupiny látok, ktoré nazývame nádorové markery alebo aj onkomarkery.
- **Dokumenty** sú neštruktúrované súbory obsahujúce medicínske informácie. V medicíne je hĺbková analýza dát využívaná ako spôsob extrahovania informácií z veľkého množstva dokumentov, vzhľadom na to, že ich spracovanie klasickými metódami by bolo neefektívne a časovo náročné.
- **Obrázky** sú v súčasnosti stále viac a viac preferovaný nástroj v medicíne. Ich hlavná výhoda spočíva v podporovaní rozhodovania pri diagnostike.
- **Grafy** ako prezentácia dát sú vo väčšej miere využívané v aplikáciách s medicínskou tematikou. Príkladom môže byť grafická reprezentácia chemických zlúčenín, ktoré sa následne môžu využiť na výrobu liekov.

Pri práci s dátami môžu nastať určité nejasnosti o samotných dátach. V medicíne sú dáta získavané a zhromažďované hlavne prostredníctvom elektronických lekárskych záznamoch, ktoré sú predovšetkým využívané na analytické účely. Takto zhromaždené údaje obsahujú veľké množstvo problémov. Údaje sú často nesprávne alebo neúplne. Opravovanie, neautomatické filtrovanie, extrahovanie, integrovanie a čistenie dát by bolo neefektívne a časovo priam nezvládnuteľné. Nesprávne alebo neúplne dáta nie sú jediným problémom, s ktorým sa pri práci s dátami môžeme stretnúť. Príliš komplexné dáta môžu byť pre lekárov alebo výskumníkov nezrozumiteľné. Následné analýzy, ktoré by viedli k výsledkom alebo diagnózam sa tak stávajú priam nereálne. Aj práve na základe týchto informácií proces hĺbkovej analýzy dát pozostáva z činnosti čistenia a integrácie dát [6].

1.2 Techniky a algoritmy

V hĺbkovej analýze dát sú aplikované rôzne techniky, metódy a algoritmy za cieľom vyriešenia rôznych typov problémov. Medzi tieto techniky a metódy radíme zhlukovanie, asociáciu, regresiu a klasifikáciu. V diplomovej práci sme sa venovali len technike klasifikácia.

1.2.1 Klasifikácia

Je jednou z najpopulárnejších techník hĺbkovej analýzy dát. Princíp tejto techniky spočíva v rozdelení vzorky údajov alebo dát do cieľových tried. Klasifikácia sa delí na binárnu a viacúrovňovú. V rámci binárnej metódy klasifikácie je možné brať do úvahy len dve cieľové triedy. Na druhej strane viacúrovňová klasifikácia je schopná rozdeliť vstupné údaje do viacerých tried [7], [8].

Súbor údajov je rozdelený na trénovaciu množinu dát a testovaciu množinu dát. Dáta sú v trénovacej a testovacej množine popísané pomocou atribútov. Atribúty teda prezentujú určité vlastnosti objektov. Hodnoty, ktoré môžu tieto atribúty nadobúdať sú buď binárne alebo nominálne. Binárne hodnoty sú reprezentované len dvomi stavmi. Nominálne môžu nadobúdať viacero hodnôt, pričom ich rozdeľujeme na diskrétne a spojité [7], [8].

Klasifikácia je proces pozostávajúci z dvoch fáz. Výsledkom prvej etapy nazývanej aj tréningová fáza je vytvorenie klasifikačného modelu. Ako názov tejto fázy napovedá, na vybudovanie modelu sa použije tréningová množina dát. Cieľom druhej fázy, nazývanej aj klasifikačná fáza, je klasifikácia množiny dát. Klasifikácia sa robí na základe modelu, ktorý bol výsledkom prvej fázy. Pod pojmom klasifikácia si môžeme predstaviť priradenie objektov do cieľových tried. Proces klasifikácia využije na tvorbu modelu takzvaný klasifikačný algoritmus. Medzi klasifikačné algoritmy patria stroje s podpornými vektormi, metóda K-najbližšieho suseda, rozhodovacie stromy, náhodný les, umelá neurónová sieť a Naive Bayes [7], [8].

Metóda K-najbližšieho suseda

Princíp metódy K-najbližšieho suseda, známej pod skratkou KNN, vychádza z predpokladu, že objekty s podobnými vlastnosťami existujú v tesnej blízkosti. Cieľom algoritmu je zaradenie nového objektu na základe podobnosti takzvaných susedov do existujúcej kategórie. Výhodou je pomerne jednoduchá implementácia. Na druhej strane jeho princíp spôsobuje pomalý proces spracovania pri veľkom množstve dát [9].

Vďaka svojej jednoduchosti metóda K-najbližšieho suseda je využívaná v mnohých medicínskych štúdiách a výskumoch. Ako príklad je možné uviesť prácu od autorov Shuheng Zhang a spol [10]. Ich štúdia skúmala možnosti a výsledky použitia algoritmu KNN na klasifikáciu metastáz lymfatických uzlín pri rakovine žalúdka. Výsledky následne porovnávali s klasickými metódami prostredníctvom helikálnej a multidetektorovej počítačovej tomografie. Ich cieľom bolo využiť metódu K-najbližšieho suseda na rozlíšenie metastáz v lymfatických uzlinách od metastáz v nelymfatických uzlinách. Výsledkom bola 96,33% celková presnosť pri diagnostikovaní metastáz v lymfatických uzlinách. Helikálna počítačová tomografia diagnostikovala s presnosťou 75,27% a multideteková počítačová tomografia s presnosťou 82,09%. Štúdia poukázala na využitie algoritmu KNN ako optimálnu voľbu [10].

Mnohé klasifikačné algoritmy sa používajú spoločne za cieľom dosiahnutia lepších výsledkov. Taktiež sa vytvárajú rôzne varianty za cieľom zvýšiť ich rýchlosť, prípadne zmierniť dopad ich nevýhod. Niektoré algoritmy sa využívajú aj ako súčasť iných metodík hĺbkovej analýzy dát. Využitiu klasifikačných algoritmov v súčasnosti v sfére medicíny sme sa venovali v nasledovnej kapitole 1.3.

1.3 Súčasné využitie hĺbkovej analýzy dát v medicíne

V súčasnej dobe existuje veľké množstvo dát v medicínskej sfére. Vzhľadom na aktuálne dianie vo svete súvisiace s pandémiou spôsobenou vírusom Covid-19 sa oblasť spracovania dát v medicíne bezprostredne dotýka takmer každého občana, nielen na území Slovenskej republiky, ale aj celého sveta. Dáta v oblasti medicíny sa ale netýkajú len súčasnej pandemickej situácií, ale sú získavané a spracovávané aj v iných oblastiach medicíny, zdravotnej starostlivosti a bio informatiky.

V roku 2021 dáta získané z oblasti medicíny produkovali 30% zo všetkých dát na svete. Analytici predpovedajú, že v roku 2025 sa toto percento ešte zvýši na 36%. Schopnosť porozumieť, rozoznať dôležitosť a klasifikovať dáta bude preto ešte dôležitejšia. Vďaka efektívnej implementácie hĺbkovej analýze dát je možné v medicíne získať mnohé výhody [11].

Zvýšená presnosť diagnostiky

Hĺbková analýza dát v medicíne pomáha lekárom vykonať diagnózy, ktoré sú presnejšie. Takto vykonané diagnózy sú založené na dôkazoch a ich získanie prebieha

v krátkom časovom rámci. Softvér s podporou hĺbkovej analýzy dát dokáže spracovať obrovské množstvo údajov v priebehu niekoľkých sekúnd. Stále je ale potrebné, aby ku konečnému rozhodnutiu dospel skúsený lekár. Informácie ako sú röntgenové snímky, snímky vyhodnotené magnetickou rezonanciou alebo krvné testy prechádzajú veľmi rýchlou analýzou a klasifikáciou, ktorá je kľúčová a pomáha pri včasnej detekcii nádorov a iných abnormalít. Rýchlosť získania informácií a presnosť informácií môžu znamenať rozdiel pri liečbe zložitých stavov s nejednoznačnými príznakmi [11].

Detekcia škodlivých liekov

Neodmysliteľnou súčasťou medicíny sú lieky. Je potrebné si ale uvedomiť, že niektoré lieky môžu byť menej účinné alebo spôsobiť nežiadúce vedľajšie účinky, ak ich užívanie nastáva v kombinácii s iným liekom alebo určitým druhom potravín. V súčasnosti prebieha neustály vývoj nových a nových liekov. Aj vďaka hĺbkovej analýze dát lekári dokážu sledovať chemické zloženie liekov a analyzovať výskumné a klinické údaje. Analýza môže viesť k odhaleniu kombinácii liekov, ktoré by mohli mať doteraz nepoznané nežiadúce účinky [11].

Odhalenie podvodov

Podvodné praktiky sa vyskytujú vo všetkých druhov poistenia. Veľmi dôležitou výhodou využitia hĺbkovej analýzy dát je schopnosť odhaliť podvody vrátane zdravotného poistenia. Podvody v zdravotnom poistení predstavujú úmyselné obchádzanie pravidiel za cieľom získania benefitu v podobe zníženia zdravotných výdavkov. Rozsah toho problému je natoľko kritický, že sa stáva jedným z prioritných problémov v zdravotníctve. Metódy a techniky bez využitia hĺbkovej analýzy dát sú časovo veľmi náročné a nie príliš efektívne. Na základe veľkého množstva dokumentov sa vytvoria pravidlá na detekciu určitých nezrovnalostí. Následne sa prostredníctvom pokročilých analýz a efektívneho využitia algoritmov identifikujú nezrovnalosti v poistných dokumentov. Systém tieto nezrovnalosti podľa určitých pravidiel vyhodnotí a označí príslušný dokument varovným signálom s potencionálnou hrozbou podvodu. Výsledkom je odhalenie pravdepodobných poistných podvodoch. Identifikované poistenia môžu prejsť podrobným preskúmaním [11].

1.3.1 Prediktívna analýza

Dáta, ktoré sú získané a spracované sú užitočným nástrojom, ktorý vie doktor uplatniť pri určovaní diagnózy. Účelom týchto dát je vytvorenie určitých možností ako sa vyhnúť zdravotným ťažkostiam pacienta. Vďaka tomu pacientovi môže byť poskytnutá lepšia zdravotná starostlivosť a čo je najdôležitejšie môže byť poskytnutá včas. Vzhľadom

na tieto skutočnosti hĺbková analýza dát je v súčasnosti využívaná pri mnohých ochoreniach, čo zahŕňa aj nasledovné:

Ochorenie diabetes mellitus

Diabetes mellitus alebo cukrovka je jedným z najrýchlejšie rastúcim chronickým ochorením, ktoré v súčasnosti postihuje milióny ľudí naprieč celým svetom. Jeho diagnostika a správna liečba sú preto kľúčové. Hĺbková analýza dát založená na analýze údajov o cukrovke pri včasnej detekcii môže viesť k predpovedi ochoreniam ako je hypoglykémia respektíve hyperglykémia. Hlavná úloha nespočíva v hľadaní alebo dokazovaní dát ktoré sú k dispozícii. Primárny účel spočíva v predikovaní informácií, ktoré nie sú známe. Viaceré techniky boli a stále sú vyvíjané za účelom detekcie, predikcie a klasifikácie cukrovky [12].

Rakovina prsníka

Hlavná príčina úmrtia žien v rozvinutých krajinách je rakovina prsníka. Vzhľadom na túto nepríjemnú skutočnosť sa v medicíne snažia identifikovať možnosti ako túto bilanciu zmeniť. Najefektívnejší spôsob ako znížiť úmrtnosť na rakovinu prsníka je odhaliť toto ochorenie čím skôr. Lekári musia definovať diagnózu včas, čo si vyžaduje presný a spoľahlivý diagnostický postup. Účelom diagnózy je rozlíšenie nádorov prsníka do dvoch skupín. Benígny nádor, nazývaný aj nezhubný nádor, nie je rakovinový nádor a nepredstavuje pre pacienta ohrozenie na živote v takom rozsahu malígny nádor. Malígny nádor, nazývaný aj zhubný nádor, je rakovinový nádor, ktorý si vyžaduje aj chirurgické odstránenie a je pre pacienta nebezpečnejší. Cieľom predpovedí je priradenie pacientov do jednej z týchto dvoch skupín nádorov. Predpovedanie výsledku choroby je jednou z najnáročnejších úloh pri vývoji aplikácií s využitím hĺbkovej analýzy dát. Systémy pracujúce na tejto problematike zbierajú extrémne veľké objemy dát. Predmetom týchto dát sú predovšetkým vek, rodinná anamnéza a genetická predispozícia. Výsledkom je predpoveď výskytu choroby na základe poskytnutých údajov. Získané znalosti sú potom prezentované lekárom, ktorý ich využívajú pri diagnóze [13].

Srdcové ochorenia

Diagnóza srdcového ochorenia je náročná úloha. V prípade automatizovanej predikcii stavu srdca pacienta by bola liečba jednoduchšia a efektívnejšia. Liečba je zvyčajne založená na príznakoch, symptómov a fyzickom vyšetrení pacienta. Ochoreniam srdca sa v určitej miere dá predísť aj na strane pacienta. Existuje totiž niekoľko určitých faktorov, ktoré mieru rizika srdcového ochorenia zvyšujú. Medzi tieto faktory môžeme zaradiť

predovšetkým fajčenie, stres, obezitu, vysokú hladinu cholesterolu v tele a nedostatok športovej aktivity. Zníženie výskytu srdcového ochorenia u pacienta je možne vďaka efektívnym rozhodnutiam v správny časový okamih. Rozhodnutia vyplývajú z údajov, ktoré sú získané z pokročilých techník hĺbkovej analýzy dát. Existujú mnohé štúdie, ktorých účelom je práve predikcia srdcového ochorenia. Cieľom je vytvorenie systému, ktorý by túto predikciu vykonal s čo najväčšou pravdepodobnosťou [14].

Ako príklad je možné uviesť systém využívajúci neurónovú sieť na predpovedanie úrovne rizika srdcových chorôb, ktorý využíva 15 medicínskych parametrov. Medzi tieto parametre radíme napríklad vek, pohlavie, krvný tlak a cholesterol. Výsledkom spomínaného systému je pravdepodobnosť, že pacienti dostanú srdcové ochorenie [14].

1.4 Nevýhody hĺbkovej analýzy dát

V predchádzajúcich kapitolách sme uviedli využitie hĺbkovej analýzy dát v medicíne len z pozitívneho hľadiska. Vzhľadom na to, že pre možnosť využitia hĺbkovej analýzy dát sú potrebné dáta, spája sa ich využitie s určitými nevýhodami a rizikami [15].

Náklady

Jedna z nevýhod je aj vysoká finančná náročnosť. Počas procesu je potrebné dáta s ktorými sa pracuje uchovať a udržiavať ich [15].

Bezpečnosť

Bezpečnostný únik dát je obrovským problémom pri využívaní hĺbkovej analýzy dát. Ak zabezpečenie, ktoré je poskytnuté nie je dostatočné, neoprávnené osoby môžu získať prístup k citlivým dátam. V prípade získania týchto dát dochádza totiž k porušeniu všeobecného nariadenia o ochrane osobných údajov. Medzi medicínske údaje radíme aj dátumy narodenia, čísla zdravotného poistenia. Krádež týchto údajov pre postihnutú osobu môže znamenať stratu finančných prostriedkov, prípadne sa môže stať obeťou zločinu krádeže identity [15].

Presnosť

Ako sme spomínali hĺbková analýza dát sa využíva predovšetkým na predikciu rôznych ochorení. Táto predikcia sa ale spája s rizikom, vzhľadom na to, že nie je 100%, a preto sa môže stať, že zdraví pacient bude diagnostikovaný ako chorý pacient. To môže viesť nielen k psychickým ťažkostiam, ale aj k zdravotným problémom pacienta z rôznych liekov a liečebných procedúr ako napríklad chemoterapia [15].

Morálka

Počas celosvetovej pandémie sa lekári stretávali s otázkou, ktorý pacient bude hospitalizovaný a ktorý pacient nebude. Lekári sa stretávajú s podobným rozhodovaním aj pri transplantácií orgánov. Zaradenie pacienta do čakacej listiny na transplantáciu môže byť vykonané aj prostredníctvom hĺbkovej analýze dát na základe životosprávy pacienta. Výsledky poskytnuté hĺbkovou analýzou dát môžu rozhodovať medzi životom a smrťou pacienta na úkor iného pacienta [16].

1.5 Nástroje hĺbkovej analýzy dát

Vzhľadom na nekončiace využitie hĺbkovej analýzy dát existujú mnohé voľno dostupné programy, ktoré slúžia ako nástroj pre prácu s hĺbkovou analýzou dát. Je potrebné si uvedomiť, že tieto nástroje sú určené predovšetkým pre programátorov pracujúcich v tomto odvetví. Vzhľadom na to, že súčasťou diplomovej práce je aj práca s knižnicou WEKA, tento nástroj bude detailnejšie popísaný ako súčasť implementačného návrhu. Medzi tieto programy radíme nasledovné nástroje [17].

- **KNIME** je komplexný analytický nástroj založený na jazyku Java, ktorý integruje, transformuje, analyzuje a spracováva dáta.
- **R** je programovací jazyk a zároveň prostredie primárne určený pre štatistické výpočty zahŕňajúci rôzne knižnice na prácu s hĺbkovou analýzou dát.
- **RapidMiner** je komplexný analytický nástroj založený na jazyku Java so zameraním na prediktívnu a obchodnú analýzu.
- **Scikit-Learn** je knižnica strojového učenia pre programovací jazyk Python so zameraním na hlbokú analýzu dát a analýzu údajov.
- **Spark** je rýchly a všeobecný nástroj na spracovanie údajov vo veľkom rozsahu, ktoré sú príliš veľké na to, aby mohli byť spracované v rámci jedného počítača.
- **Matlab** je programovacie prostredie, ktoré prostredníctvom nástrojov uľahčuje prácu s hlbokou analýzou dát. Pred spracováva údaje, vytvára modely a nasadzuje modely do IT systémov [18].

2 Popis implementovaných klasifikačných algoritmov

Táto kapitola sa venuje podrobnému popisu implementovaných klasifikačných algoritmov. Medzi klasifikačné algoritmy, ktoré boli implementované, radíme metódu K-najbližšieho suseda a jeho variácie. Na začiatku je potrebné popísať samotný základ algoritmu KNN bez dodatočných zmien. Zmeny boli implementované za cieľom zvýšiť efektivitu, respektíve presnosť samotného algoritmu pri určitých dátových množinách (datasetoch).

2.1 KNN

Metódu K-najbližšieho suseda je možné pochopiť ako odzrkadlenie človeka v spoločnosti. Počas celého života sme ovplyvnení rodičmi, priateľmi, kolegami a aj miestom kde žijeme. Je zrejmé, že ľudia, ktorí bývajú v našej blízkosti a trávia s nami väčšinu času majú na naše rozhodovanie a názory väčší vplyv ako ľudia, s ktorými sme sa stretli len raz v živote. Ľudia s rovnakými záujmami a názormi sa následne stretávajú, prípadne spájajú do určitých skupín alebo klubov. Na rovnakom princípe pracuje aj algoritmus KNN.

Algoritmus sa častokrát označuje aj ako lenivý, čiže „lazy“. Tento prívlastok, ale nevznikol na základe jeho jednoduchosti z hľadiska porozumenia, ale kvôli tomu, že nevykonáva takzvaný tréning. Počas etapy trénovania sa totiž vykonáva len zber dát. Ako lenivý algoritmus nevykonáva žiadne matematické výpočty, predlohy pre predikcie ani nič podobné. Výsledkom fázy trénovania je štruktúra obsahujúca všetky dáta z trénovacej množiny. Celý matematický proces kalkulácie sa následne vykonáva počas testovacej fázy [9].

Dáta, ktoré sa nachádzajú v tesnej blízkosti majú viac spoločných vlastností ako dáta, ktorých vzájomná vzdialenosť je vysoká. Ako aj pri ostatných klasifikačných algoritmov aj pri KNN sa dáta delia na dve množiny, tréningovú a testovaciu množinu. Cieľom KNN je správne predpovedať triedu pre testovacie dáta. Dáta sa skladajú z takzvaných inštancií, pričom je možné ich zapísať nasledovne [9].

$$x_{11}, x_{12}, x_{13} \cdots x_{1j}, y_n^i$$
$$\vdots$$
$$x_{n1}, x_{n2}, x_{n3} \cdots x_{nj}, y_n^m$$

Dáta si môžeme predstaviť ako tabuľku skladajúcu sa z *n* riadkoch (inštancií) a *j* stĺpcov, kde *j* je počet atribútov danej inštancie. Jeden z atribútov predstavuje cieľovú triedu inštancie. Cieľová trieda môže nadobúdať rôzne typy hodnôt y^i. Premenná *m* vyjadruje práve hodnoty ktoré môže cieľová trieda nadobúdať. Ak vezmeme príklad z medicínskeho hľadiska premenná *m* by predstavovala dve hodnoty {0 – chorý pacient, 1 – zdravý pacient}. Pričom platí, že viacero pacientov môže byť chorých a viacero pacientov môže byť zdravých [9].

Sú rôzne spôsoby ako je možne nájsť K-najbližších susedov pre novo pridanú inštanciu. Jedna z týchto možností sa nazýva aj metóda „hrubou silou“. Pri tejto metóde vykonáme výpočet vzdialeností medzi každou jednou inštanciu v trénovacej množine a novo pridanou inštanciou. Vzdialenosti usporiadame vzostupne. Na prvých pozíciách sa preto budú nachádzať najbližšie inštancie takzvaný susedia a na posledných tie najviac vzdialené. Počet najbližších susedov, ktorý je udávaný premennou *k* predstavuje *k* prvých inštancií.

Každá inštancia v testovacej množine obsahuje informáciu o cieľovej triede. Na základe týchto informácií prebehne takzvaný proces hlasovania. Hlavný princíp procesu spočíva v tom, že pre každú jednu cieľovú triedu zistíme výskyt tejto triedy spomedzi K-najbližších susedov. Trieda, ktorá získala najväčší počet hlasov sa stane výslednou triedou.

Aby sme mohli identifikovať výslednú triedu pre novú inštanciu je potrebné implementovať algoritmus. Vstupom je premenná *k* a novo pridaná inštancia x'. V algoritme KNN pre každú inštanciu platí, že je jej priradená práve jedna výstupná trieda y^i. Premenná *x* predstavuje inštancie a premenná *j* udáva počet atribútov. Pseudokód je možné popísať nasledovne.

Pseudokód [19]**:**

- Pre každú novú inštanciu x' vykonaj nasledovné.
    - Vypočítaj vzdialenosť $d(x, x')$ medzi novou inštanciou a ostatnými inštanciami.
        - Na výpočet je možné použiť ľubovoľnú metriku vzdialeností. Medzi najznámejšie, respektíve najpoužívanejšie metriky sa radí:
            a. Euklidovská vzdialenosť.

$$d(x, x') = \sqrt{\sum_{i=1}^{j}(x_i - x'_i)^2}$$

b. Manhattanovskú vzdialenosť.

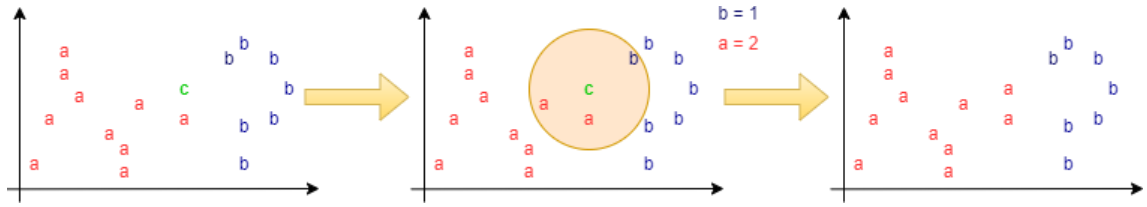
$$d(x, x') = \sum_{i=1}^{j}|x_i - x'_i|$$

- Vyber podmnožinu, ktorú označíme ako D_z s k prípadmi najmenšej vzdialenosti.
- Urči cieľovú (výslednú) triedu y^* pre novo pridanú inštanciu x' prostredníctvom hlasovania definovaného nasledovným vzťahom. Premenná v predstavuje cieľovú triedu konkrétnej inštancie $x \in D_z$, čiže cieľovú triedu najbližšieho suseda. Premenná y_i predstavuje možné hodnoty cieľovej triedy.

$$y^* = arg\,max_v \sum_{(x,x')\mathrm{g}D_z} I(v = y^i); \qquad I(áno) = 1;\ I(nie) = 0$$

Analýza algoritmu na príklade v 2D priestore

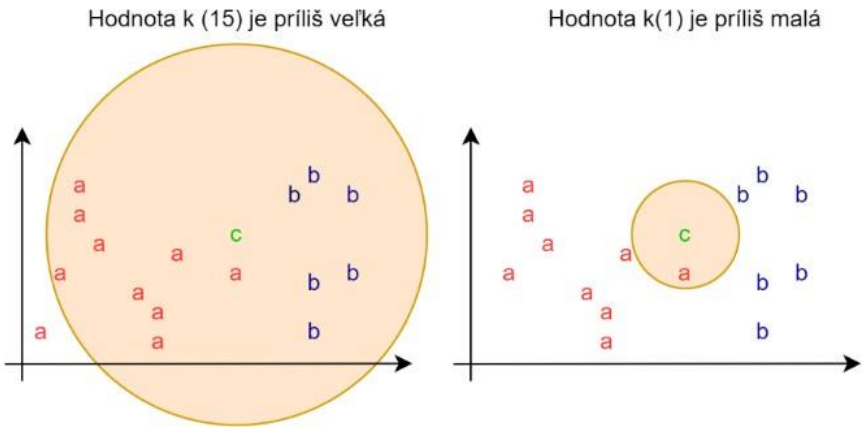
Nech máme definovaný 2d priestor, k = 3 a cieľové triedy a, b, pričom je potrebné klasifikovať novú inštanciu c, pre ktorú je cieľová trieda neznáma. Prostredníctvom euklidovskej vzdialenosti zistíme troch (k) najbližších susedov. Vyberieme tú triedu, ktorá má medzi najbližšími susedmi najväčšie zastúpenie. Údaj c priradíme do tejto triedy a ukončíme algoritmus. Popísaný postup je znázornený na obrázku 2 [20].



Obr. 2 Algoritmus priradenia výstupnej triedy pre nový vstupný údaj

Ako je možné vidieť na obrázku 2, dôležitosť výberu hodnoty parametru k je skutočne kľúčová. V prípade ak vyberieme v dvojdimenzionálnom priestore párnu hodnotu, môže nastať situácia takzvanej remízy. Počet najbližších susedov bude pre každú jednu

cieľovú triedu rovnaký a preto nebude možné vykonať výber. Hodnota *k* taktiež nesmie nadobúdať ani násobok počtu tried z rovnakého dôvodu. V prípade, že sa splnia obidve podmienky je potrebné sa taktiež vyvarovať nasledovným problémom. Malá hodnota *k* môže obsahovať náhodné prvky a veľká hodnota by mohla obsiahnuť veľkú časť nesprávne definovanej cieľovej triedy. Obrázok 3 znázorňuje spomínaný problém [20].



Obr. 3 Grafické znázornenie nesprávne zvolenej hodnoty *k*

Na základe týchto problémov sa vynára otázka ako teda vybrať optimálnu hodnotu *k*. Nanešťastie v súčasnosti nie sú žiadne definované pravidlá na základe ktorých by sme dokázali určiť túto hodnotu. Ako odporúčanie je možné zvoliť hodnotu *k* ako odmocninu z počtu inštancií v skúmanom datasete. Nesmieme pri tom zabudnúť na spomínané pravidlá o násobnosti a o párnej hodnote [20].

Časová zložitosť algoritmu

Ku každému úkonu je napísaná aj časová zložitosť s akou sa daný príkaz vykoná. Pričom platí, že nie sú zapísané konštantné zložitosti. Ako príklad je možne uviesť výpočet euklidovskej vzdialenosti ktorej zložitosť by predstavovala počet atribútov inštancií alebo výber k najbližších susedov. Pre každú jednu inštanciu je potrebné vykonať nasledovné [21].

- Vypočítaj vzdialenosti pre každý bod (N)
- Zoraď tieto vzdialenosti (N * log N – N^2)
- Celkový čas O(N^3)

Metóda hrubou silou sa javí na základe vypočítanej zložitosti javí ako extrémne časovo náročná. Aby sa nemuseli pri hľadaní K-najbližších susedov prechádzať všetky

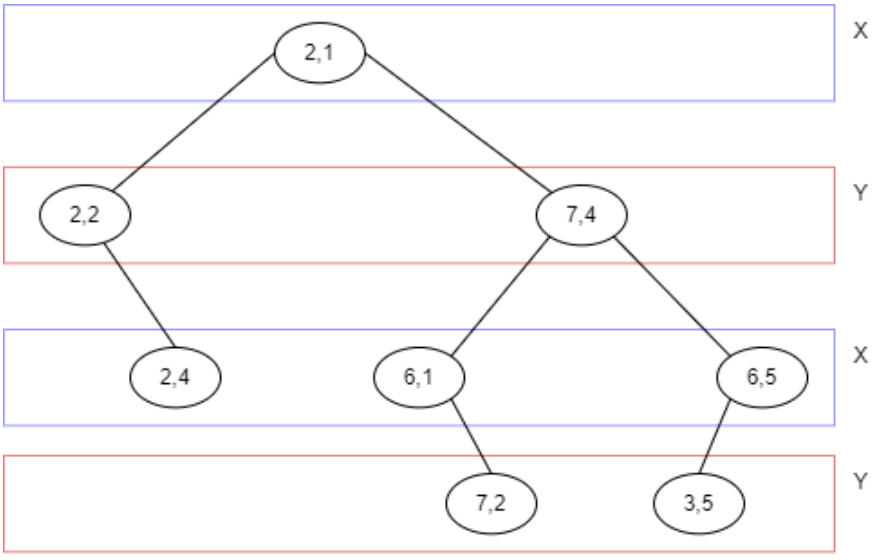
inštancie pre každú novo pridanú inštanciu je možné využiť pokročilé údajové štruktúry. Jedna z týchto štruktúr je aj Kd-Tree, ktorá je popísaná v nasledujúcej kapitole.

2.2 Kd-Tree

K-dimenzionálny strom alebo skrátene Kd-Tree je modifikácia klasického binárneho vyhľadávacieho stromu, v ktorej každý vrchol predstavuje bod v K-dimenzionálnom priestore. Medzi jeho základnú vlastnosť patrí takzvaná úroveň vrcholu. Rozsah úrovne je od nuly po počet dimenzií. Úroveň definuje ktorá dimenzia je v danom vrchole smerodajná. V údajovej štruktúre Kd-Tree totiž platí, že kľúče, ktoré sú menšie alebo rovné ako hodnoty kľúčov v danom vrchole sa nachádzajú iba v ľavom podstrome. Kľúče, ktoré sa nachádzajú v pravom podstrome majú väčšiu hodnotu ako otec daného vrcholu. Je potrebné si uvedomiť, že táto vlastnosť platí pre danú dimenziu, ktorá je práve určená úrovňou. Keďže sa kľúče v ľavom podstrome môžu aj rovnať, je zrejmé, že štruktúra Kd-Tree narozdiel od klasického binárneho stromu podporuje aj duplicitu kľúčov.

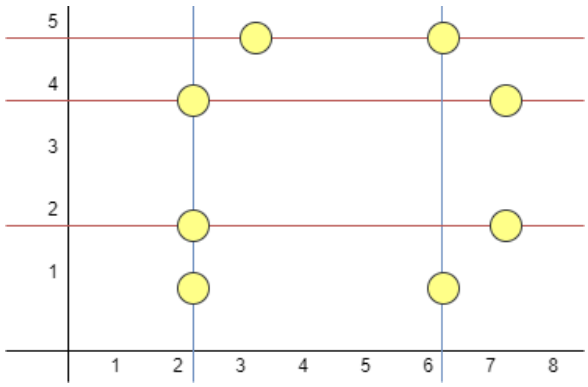
Príklad použitia

Princíp pokročilej údajovej štruktúry Kd-Tree si znázorníme prostredníctvom nasledujúceho datasetu { 2,1,0; 7,4,0; 6,1,0; 6,5,1; 2,2,0; 2,4,0; 3,5,1; 7,2,1 }. Posledná hodnota predstavuje výslednú triedu danej inštancie. Bod v priestore je teda určený hodnotami atribútov na prvých dvoch miestach. Hodnoty 0,1 môžu predstavovať čokoľvek, keďže sa pohybujeme v medicínskej sfére, môžeme si ich predstaviť nasledujúco {0 – chorý pacient, 1 – zdravý pacient}. Obrázok 4 popisuje príklad použitia v stromovej štruktúre.



Obr. 4 Stromová štruktúra Kd-Tree

Obrázok 5 znázorňuje uloženie dát, ktoré boli súčasťou príkladu použitia v 2d priestore, pri použití pokročilej údajovej štruktúre Kd-Tree.



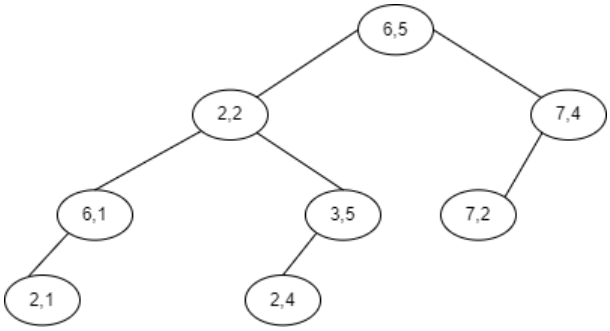
Obr. 5 Kd-Tree v 2d priestore

Konštrukcia Kd-Tree

Pri vytváraní klasifikátora máme prístup k celej testovacej množine inštancií. Vzhľadom na túto skutočnosť je postup konštrukcie nasledovný. V prvom rade zistíme inštanciu, ktorá predstavuje medián z celej testovacej množiny pre danú dimenziu. Následne rozdelíme inštancie. Inštancie rozdeľujeme na základe popísaných pravidiel v Kd-Tree. To znamená, že ak hodnota inštancie pre aktuálnu dimenziu je menšia alebo rovná ako je hodnota mediánu pre danú dimenziu, inštanciu pridám do ľavého podstromu. V opačnom prípade inštanciu pridám do pravého podstromu. Inštanciu, ktorú sme vybrali ako medián, ďalej už nerozdeľujeme. Výber mediánu a rozdelenie množiny inštancií vykonávame pre každý novovzniknutý vrchol. Algoritmus končí keď každú inštanciu priradíme ako vrchol stromu. Z hľadiska časovej zložitosti je značná nevýhoda hľadania mediánu O(N log N). Medián sa ale hľadá len pri operácií konštrukcia stromu [21].

Príklad použitia

Obrázok 6 znázorňuje výsledok vykonania operácie, pričom bol použitý rovnaký dataset ako pri popisovaní implementácie Kd-Tree.



Obr. 6 Grafické znázornenie inštancií v Kd-Tree

K-najbližší susedia

Aby sme mohli nájsť K-najbližších susedov pre novú inštanciu už nie je potrebné prehľadávať každú jednu inštanciu ako sme to robili pri metóde hrubou silou. V Kd-Tree totiž inštancie nie sú náhodne umiestnené a ich rozloženie spĺňa určité pravidla, ktoré sme zadefinovali pri operácií konštrukcia Kd-Tree. V prípade, že vieme s určitosťou povedať, že v ľavom alebo v pravom podstrome sa nenachádza žiadna inštancia, ktorej vzdialenosť by bola menšia ako je vzdialenosť doteraz nájdenej, najvzdialenejšej inštancie, nemusíme inštancie v danom podstrome prehľadávať a spracovávať. Zadefinovanie práve tohto prípadu vychádza z neskôr popísaného slovného popisu operácie nájdenia K-najbližších susedov.

Algoritmus s využitím Kd-Tree je možné popísať podľa nasledovného slovného popisu. Pri implementácii algoritmu sa použila aj základná údajová štruktúra zásobník. Zásobník slúži na ukladanie už spracovaných vrcholov. Pre uchovanie K-najbližších inštancií sa využila základná údajová štruktúra prioritný front. Pričom platí, že na prvej pozícií sa nachádza inštancia, ktorá je od novo pridanej inštancie najvzdialenejšia. Takáto modifikácia prioritného frontu sa aj nazýva max-first-queue-priority. Počas hľadania K-najbližších susedov je potrebné spracovať vrcholy takzvaným traverzovaním. Pod pojmom traverzovanie stromu rozumieme prechádzanie stromu, kde ak je hodnota inštancie pre danú dimenziu menšia alebo rovná ako v práve spracovávanom vrchole zadefinujem, že práve spracovávaný vrchol je ľavý syn daného vrchola. V opačnom prípade je tento vrchol pravý syn daného vrchola. Popísaný postup opakujem pokiaľ práve spracovávaný vrchol je list.

Slovný popis [21]:

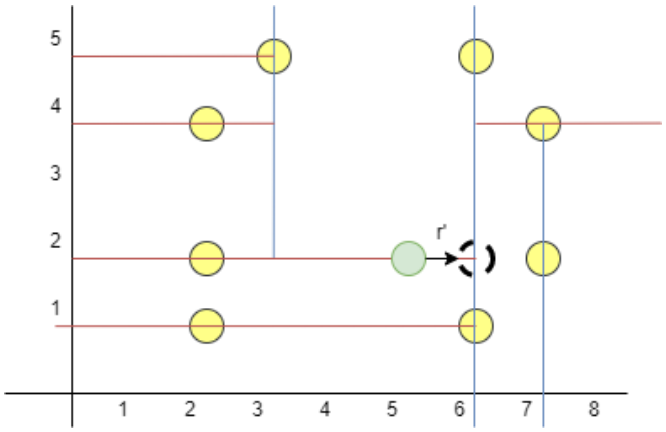
Pre každú novo-pridanú inštanciu vykonaj nasledovné:

- Vytvor pole vzdialeností o veľkosti *k* a inicializuj ho na maximálne hodnoty.
- Traverzuj stromom od koreňa do listu, kde by sa nová inštancia mala nachádzať.
    - Uchovaj prechádzané vrcholy do zásobníka.
- Pri každom novom vrchole vypočítaj vzdialenosť inštancie od inštancie v danom vrchole.
- Ak si našiel menšiu vzdialenosť ako je najväčšia vzdialenosť v prioritnom fronte, uchovaj túto vzdialenosť a príslušnú inštanciu na základe popísaných pravidiel.

- Ak si v liste, vyber vrchol zo zásobníka.
    - Zisti, či je v nepreskúmanom podstrome možnosť nájdenia lepšej vzdialenosti.
        - Ak možnosť existuje, pokračuj prechádzaním daného podstromu.
        - Ak možnosť neexistuje, vyber vrchol zo zásobníka.
- Algoritmus končí keď, je zásobník prázdny.

Otázka, ktorá vyplýva zo slovného popisu je, ako zistíme, či je v nepreskúmanej sekcii možnosť nájdenia lepšej vzdialenosti. Odpoveď spočíva vo vypočítaní vzdialenosti medzi novo pridanou inštanciou a najbližšou možnou inštanciou z danej sekcie. Tu ale vyplýva ďalšia otázka. Ako je definovaná najbližšia možná inštancia z danej sekcie? Pre nájdenie odpovede je potrebné si uvedomiť, že vrcholy predstavujú medián pre danú dimenziu. To znamená, že vrchol predstavujúci medián rozdeľuje inštancie na dve sekcie pre danú dimenziu, prostredníctvom pomyselnej priamky. Ak hľadáme najbližšieho suseda pre inštanciu, ktorá sa nachádza naľavo od pomyselnej priamky, tak najbližšia inštancia z druhej sekcie pre danú dimenziu sa bude nachádzať práve na pomyselnej priamke. Každá inštancia, ktorá sa nachádza v pravej sekcii je viac vzdialená od inštancie v ľavej sekcii ako inštancia nachádzajúca sa na pomyselnej priamke.

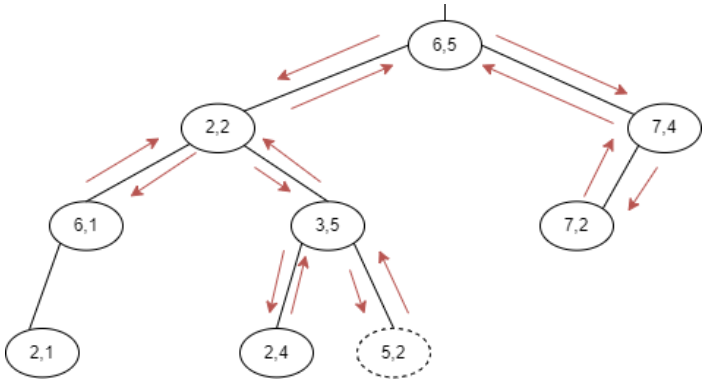
Pre lepšie pochopenie danej problematiky obrázok 7 graficky znázorňuje prípad, kedy zisťujeme vzdialenosť od najbližšieho možného bodu v danej sekcii. Priamka *r*‘ predstavuje danú vzdialenosť.



Obr. 7 Novo pridaná inštancia v 2d priestore

Algoritmus zistenia troch najbližších susedov novo pridanej inštancie s hodnotami {5, 2} na danom datasete je grafický znázornený prostredníctvom obrázku 8. Ako je možné

vidieť, nepreskúmal sa iba jeden vrchol, v tomto prípade je to inštancia {2,1}, no v skutočnosti sa nepreskúmal celý podstrom bez ohľadu na počet prvkov. Znázornené šípky predstavujú traverzovanie algoritmu po štruktúre Kd-Tree.



Obr. 8 Novo pridaná inštancia v Kd strome

Vykonaním algoritmu získame najbližšie inštancie, čiže K-najbližších susedov. Susedia nesú informáciu o cieľovej triede a ich vzdialenosť od novo pridanej inštancie. Tieto informácie sú prehľadne zobrazené v tabuľke 1, pričom dáta predstavujú dáta z pôvodného datasetu, ktorý bol predstavený ako príklad.

Tabuľka 1: *k*(3) najbližšie inštancie danej inštancii {5, 2} algoritmu KNN

Označenie inštancie	Inštancia	Euklidovská vzdialenosť $d(x, x')$	Výsledná trieda
x_8	7,2	2,0	1
x_3	6,1	1,414	0
x_2	7,4	2,828	0

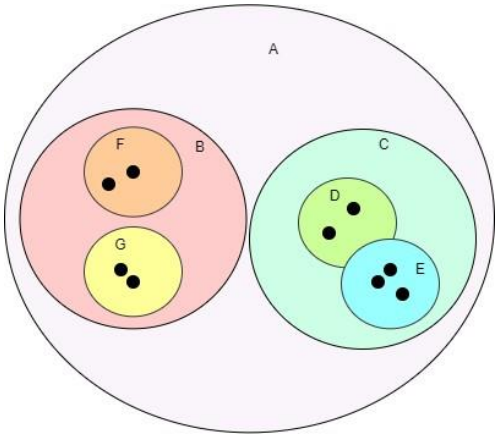
Ako bolo popísane v kapitole KNN vieme, že po získaní K-najbližších susedov nastane takzvaná fáza hlasovania. V základnom variante algoritmu KNN prejdeme zoznam inštancií a vyberieme tú cieľovú triedu, ktorá by mala najväčšie zastúpenie. Pre náš príklad, kde berieme do úvahy dve cieľové triedy je zastúpenie zobrazené v tabuľke 1. Výsledná trieda pre našu inštanciu je 0.

V prípade, že atribúty inštancií nachádzajúcich sa na prvých pozíciách v datasete sú približne rovnaké hodnoty a posledné atribúty sú veľmi rozdielne, môže to viesť k neefektívnej implementácií Kd-Tree. Kd-Tree nezohľadňuje dáta pri vytváraní stromovej

štruktúry. Jedným zo spôsobom ako túto možnosť eliminovať je zostupné usporiadanie indexov atribútov na základe rozptylu. Táto možnosť v Kd-Tree implementovaná.

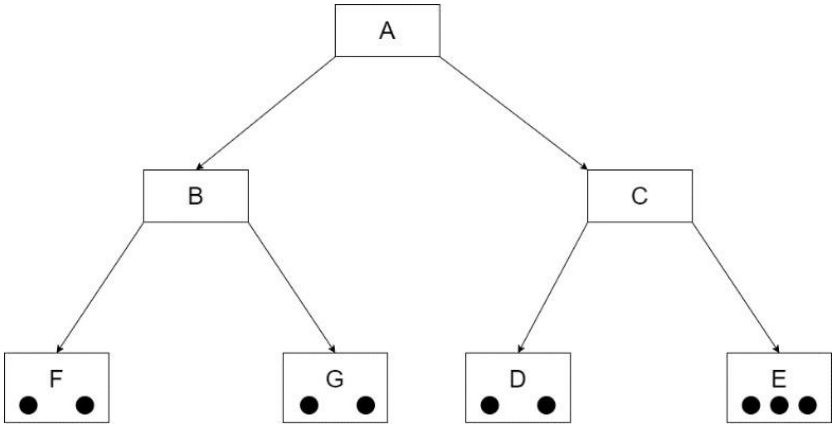
2.3 Ball-Tree

Je pokročilá údajová štruktúra, ktorá ako všetky ostatné údajové štruktúry slúži na efektívnejšie uchovanie dát. Jej rozdiel oproti Kd-Tree spočíva vo využití takzvaných kruhov v prípade 2d priestoru alebo takzvaných gulí v prípade 3d priestoru. Počet atribútov predstavuje počet dimenzií. Keďže sa v praxi málokedy využívajú len dve, respektíve tri hodnoty je potrebné pracovať s takzvanou n-guľou alebo hypersférou. Hypersféra funguje v n-dimenzionálnom priestore, tým pádom je možné využiť počet atribútov o veľkosti *j*. Pre lepšiu predstavu štruktúry Ball-Tree bol pridaný obrázok 9 [22].



Obr. 9 Štruktúra Ball-Tree v priestore

Je potrebné si uvedomiť, že Ball-Tree je v pamäti stromová štruktúra. Dáta, ktoré boli znázornené prostredníctvom obrázku 9, musia byť zobrazené aj prostredníctvom stromovej štruktúry. Výsledok tohto usporiadania bol znázornený na obrázku 10.



Obr. 10 Stromová reprezentácia Ball-Tree

Konštrukcia Ball-Tree

Konštrukciu Ball-Tree je možné popísať prostredníctvom nasledovného pseudokódu. Vstupom pre tento pseudokód je premenná *k*, ktorá ako vždy udáva počet najbližších susedov. Počas konštrukcie je využívaná aj základná údajová štruktúra front. Front sa využíva počas typu prehliadky Level Order. Inštancia je udávaná premennou *x*. *S* je množina všetkých inštancií. Pred začatím algoritmu je potrebné vytvoriť koreň, nastaviť mu parametre medzi ktoré radíme zoznam inštancií a rádius. Keď nastavíme potrebné atribúty pridáme vrchol do frontu. Vzhľadom na to, že prerozdeľujeme inštancie pre daný vrchol po prerozdelení vymažeme inštancie z daného vrcholu. Inštancie totiž uchovávame len v listoch a nie vo vnútorných vrcholoch, ako to bolo pri Kd-Tree [23].

Pseudokód [23]:

- Pokiaľ front nie je prázdny vykonaj nasledovné:
  - Vyber vrchol z fronty.
  - Náhodne vyber inštanciu x_0 z *S*.
  - Vyber najvzdialenejšiu inštanciu x_1 od x_0.

$$x_1 = \arg max_{x \in S}\, d(x_0, x)$$

  - Vyber najvzdialenejšiu inštanciu x_2 od x_1.

$$x_2 = \arg max_{x \in S}\, d(x_1, x)$$

  - Vykonaj projekciu inštancií x_i na vektor $\rightarrow (x_1 - x_2)$ alebo na vektor $\rightarrow (x_2 - x_1)$ na základe väčšieho rozptylu. Výsledkom tohto kroku bude vypočítaná hodnota z_i pre každú inštanciu.

$$(x_1 - x_2)^T x_i = [(x_1 - x_2)_1\ (x_1 - x_2)_2\ \cdots (x_1 - x_2)_n] \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \vdots \\ x_{i_n} \end{bmatrix}$$

$$= [(x_1 - x_2)_1 x_{i_1} + (x_1 - x_2)_2 x_{i_1} + \cdots + (x_1 - x_2)_n x_{i_1}]$$

$$z_i = (x_1 - x_2)^T x_i\ ;\ \forall i = 1 \cdots |S|$$

  - Vyber medián m

$$m = median\ (z_i \cdots z_{|S|})$$

- Vykonaj rozdelenie inštancií

$$S_L = \{x \epsilon S\} : (z_i < m)$$
$$S_R = \{x \epsilon S\} : (z_i \geq m)$$

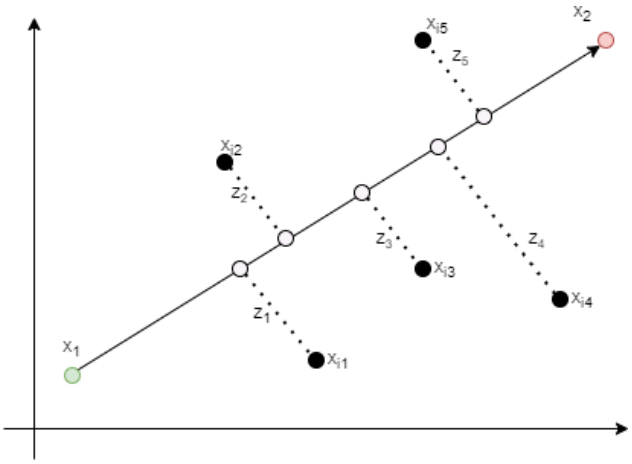
- Nastav atribúty vrchola – centroid (c), rádius (r), inštancie synov (S_L, S_R)

$$c = \frac{\sum_{i=1}^{|S|} X_i}{|S|}$$

$$r = \arg max_{xgS}\, d(x, c)$$

- Pridaj pravého / ľavého syna do fronty ak $|S_R| > k$; $|S_L| > k$

V pseudokóde je popísaná projekcia inštancií na vektor. Medzi inštanciou x_1 a inštanciou x_2 vznikne vektor, na ktorý je potrebné pre každú inštanciu vykonať projekciu, tak ako je znázornené na obrázku 11.



Obr. 11 Projekcia inštancií na vektor

K-najbližší susedia

Aby sme mohli využiť štruktúru Ball-Tree ako súčasť algoritmu KNN je nutné, aby táto štruktúra podporovala hľadanie K-najbližších susedov. Vstupom pre tento algoritmus je v prvom prípade novo pridaná inštancia (x‘), pre ktorú hľadáme K-najbližších susedov a samotné k, ktoré vyjadruje počet najbližších susedov. Výstupom je zoznam inštancií, čiže najbližší susedia. Pre uchovanie vrcholov sa používa štruktúra zásobník. Pre uchovanie inštancií sa využila základná údajová štruktúra max-first-queue-priority (Q). Premenná B udáva práve spracovávaný vrchol.

Pseudokód [24]:

- Ak nie je vrchol *B* null.
  - Vlož vrchol *B* do zásobníka.
  - Ak je vrchol *B* list (nemá pravého ani ľavého syna), spracuj vrchol *B* podľa daného pravidla:
    - Pre každú inštanciu *x* vo vrchole *B*, vykonaj nasledovné:
      - Vypočítaj vzdialenosť z.

$$z = d(x', x) - d(x', Q.prvý)$$

      - V prípade že z < 0 potom:
        - Pridaj inštanciu p do prioritného frontu *Q*.
        - Ak veľkosť $Q > k$ potom:
          - Odstráň prvý, najvzdialenejší prvok z *Q*.
  - Ak je vrchol *B* vnútorný vrchol, vykonaj nasledovné:
    - Vypočítaj hodnoty pre premenné *l* a *r*.
      - Ak B.ĽavýSyn je null potom *l* = Double.MaxValue.
      - Inak $l = d(x', B.ĽavýSyn.Centroid)$.
      - Ak B.PravýSyn je null potom *r* = Double.MaxValue.
      - Inak $r = d(x', B.PravýSyn.Centroid)$.
    - Na základe menšej hodnoty *l*, *r* vyber bližšieho syna a vyhlás ho za vrchol *B*.
- Ak vrchol *B* nie je null.
  - Vyber vrchol zo zásobníka.
    - Ak je vrchol list, nastav vrchol *B* na null.
    - Ak má aj pravého a ľavého syna a obaja synovia už boli spracovaný, nastav vrchol *B* na null.
    - Ak má len ľavého syna a tento syn bol spracovaný, nastav vrchol *B* na null. Dané pravidlo platí aj pre pravého syna.
    - Ak je nasledujúca rovnica pravdivá, nastav vrchol *B* na null.

$$d(x', B.centroid) - B.radius \geq d(x', Q.prvý)$$

Na základe vlastností Ball-Tree je v listoch pri operácii konštrukcia Ball-Tree uložených *k* inštancií. V prípade, že hodnota *k* nie je zadaná nastaví sa premenná *k* na hodnotu 40 tak ako je to aj v prípade knižnice WEKA. Ball-Tree pracuje s viac dimenzionálnymi dátami efektívnejšie ako Kd-Tree vzhľadom na to, že nerozdeľujeme dáta v strome na základe jednej dimenzie v jednom momente.

2.4 Varianty KNN

Existujú mnohé varianty algoritmu KNN. Hlavným cieľom týchto variant je potlačenie neefektívnych prístupov algoritmu. Medzi implementované algoritmy patria varianty fuzzy KNN alebo FKNN, weighted KNN alebo WKNN a variant založený na princípe harmonického stredu alebo HMDKNN.

2.4.1 FKNN

Problémom, s ktorým sa počas fungovania algoritmu stretávame je jeho nedostatok vo fáze hlasovania. Predstavme si nasledovnú situáciu. Z troch najbližších susedov, ktorých sme našli je vzdialenosť dvoch susedov 9,521 a cieľová trieda týchto dvoch susedov je 1. Tretí najbližší sused má ale vzdialenosť od inštancie 1 a jeho cieľová trieda je 0. Na základe klasického princípu hlasovania bude novej inštancii priradená hodnota 1, hoci by sme mali rozumne predpovedať triedu 0. Problémom, ktorý totiž nastáva pri klasickom variante KNN je jeho „lenivosť" zobrať do úvahy vzdialenosti výsledných K-najbližších susedov. Jedným z riešením tohto problému je aj implementovaný variant KNN takzvané FKNN [25].

FKNN alebo fuzzy KNN je algoritmus, ktorý ku každej hodnote cieľovej triedy priradí príslušnosť pre danú triedu. Táto variácia z časti rieši aj problém takzvanej patovej situácií v hlasovaní. Hoci počet výsledných hlasov pre cieľovú triedu môže byť rovnaký ich príslušnosť na základe pravidiel fuzzy KNN už rovnaká byť nemusí [25].

Algoritmus nájdenia K-najbližších susedov je rovnaký. Čo sa ale mení je spôsob priradenia výslednej triedy novo pridanej inštancie x'. Pri tejto operácií postupujeme podľa nasledovného algoritmu.

Pseudokód FKNN [25]**:**

- Nájdi K-najbližších susedov.
- Inicializuj premennú $i = 1$ a premennú m, ktorá predstavuje počet možných hodnôt cieľovej triedy.

- Vypočítaj príslušnosť $\mu_i(x')$ pre každú triedu prostredníctvom vzťahu. Premenná *z* je definovaná používateľom variantu fuzzy KNN.

$$\mu_i(x') = \frac{\sum_{j=1}^{k} \mu_{ij} \left(\frac{1}{d(x_j x')^{\frac{2}{z-1}}}\right)}{\sum_{j=1}^{k} \left(\frac{1}{d(x_j, x')^{\frac{2}{z-1}}}\right)} \qquad i = 1,2,\cdots,m;\ 1 \le j \le k$$

Premenná μ_{ij} definuje nasledovné pravidlo. Ak je cieľová trieda inštancie x_j zhodná s cieľovou triedou y^i hodnota premennej je 1 v opačnom prípade 0.

- Vyber cieľovú triedu y^* na základe nasledujúceho vzťahu.

$$y^* = \arg\max \mu_i(x') \qquad i = 1,2,\cdots,m$$

Príklad použitia

Na základe tabuľky 1 poznáme vzdialenosti inštancie od konkrétneho suseda, čo je zobrazené vo vzťahu ako $d(x, x')$. Definujme hodnotu *m* ako 2. Pre výslednú triedu 0 boli priradené dve inštancie x_3 a x_2. Pre triedu 1 bola priradená jedna inštancia x_8.

$$\mu_0(x') = \frac{\frac{0}{(2)^2} + \frac{1}{(1{,}414)^2} + \frac{1}{(2{,}828)^2}}{\frac{1}{(2)^2} + \frac{1}{(1{,}414)^2} + \frac{1}{(2{,}828)^2}} = 0{,}71429$$

$$\mu_1(x') = \frac{\frac{1}{(2)^2} + \frac{0}{(1{,}414)^2} + \frac{0}{(2{,}828)^2}}{\frac{1}{(2)^2} + \frac{1}{(1{,}414)^2} + \frac{1}{(2{,}828)^2}} = 0{,}28571$$

Prostredníctvom FKNN sme na základe výberu najväčšej pravdepodobnosti dospeli k rovnakému variantu ako pri pôvodnom KNN. Ak by sme hlasovanie pri základnom KNN previedli na pravdepodobnosť, tak by výsledok bol nasledovný.

$$\mu_0(x') = 0{,}333$$

$$\mu_1(x') = 0.6666$$

Na základe variantu FKNN sme sa o 5% viac presvedčili, že výsledná trieda pre novo pridanú inštanciu je práve 1, čiže pacient je zdravý.

Premenná *z* určuje s akou váhou berieme do úvahy vypočítané vzdialenosti pre K-najbližších susedov. V prípade väčšej hodnoty *z* ako je 2 sú susedia rovnomernejšie vážený a ich vzdialenosti majú na celkový výsledok menší účinok. Ak sa *z* približuje k hodnote 1, vzdialenosti najbližších susedov sú vážene oveľa viac, čo vedie ku zníženiu počtu susedov, ktorý ovplyvňujú cieľovú triedu novo pridanej inštancie [25].

2.4.2 HMDKNN

Z princípu fungovania algoritmu KNN vyplýva, že hodnota premennej *k* silno vplýva na celkovú presnosť klasifikácie. Nesprávne vybraná hodnota môže výrazným spôsobom degradovať celkovú presnosť. Jedným zo spôsobom ako znížiť citlivosť algoritmu KNN na nesprávne vybranú hodnotu k je práve implementovaný variant HMDKNN [26].

HMDKNN predstavuje harmonický priemer vzdialeností K-najbližších susedov. Algoritmus variácie HMDKNN rovnako ako FKNN nemení algoritmus nájdenia K-najbližších susedov. Jeho rozdiel spočíva v zmene spracovania získaných vzdialeností, následného procesu hlasovania a výberu cieľovej triedy [26].

Pseudokód HMDKNN [26]:

- Nájdi K-najbližších susedov.
- Inicializuj premennú *m*, ktorá predstavuje počet možných hodnôt pre cieľovú triedu. Premenná *x'* predstavuje novo pridanú inštanciu.
- Vypočítaj priemernú inštanciu u_j prostredníctvom vzťahu pre každú možnú cieľovú triedu inštancie. Premenná x_{lj} predstavuje konkrétnu inštanciu pre konkrétnu triedu.

$$u_j = \frac{1}{i}\sum_{l=1}^{i} x_{lj} \quad ; 1 \le i \le k; \qquad j = 1, 2, \cdots, m$$

- Vypočítaj harmonický stred $h(x', U_j)$ vzdialeností prostredníctvom nasledovného vzťahu. Premenná *r* je definovaná používateľom.

$$h(x', U_j) = \frac{r}{\sum_{j=1}^{m} \frac{1}{d(x', u_j)}} \qquad j = 1,2, \cdots, m; 1 \le r \le k$$

- Definuj nový harmonický stred $H(y, h^j)$ pre každú cieľovú triedu y^i na základe vzťahu.

$$H(x', h^j) = \frac{k}{\sum_{j=1}^{m} \frac{1}{h(x', U_j)}} \qquad j = 1,2,\cdots,m$$

- Vyber novú cieľovú triedu y^* na základe definovaného pravidla (najmenšiu hodnotu).

$$y^* = \arg\min H(x', h^j) \qquad j = 1,2,\cdots,m$$

Príklad použitia

Aj pri variante HMDKNN vychádzame z tabuľky 1. Na základe pseudokódu budeme postupovať nasledovne.

1. V prvom kroku vytvoríme priemernú inštanciu pre každú cieľovú triedu.

$$u_0 = 6.5, 2.5 \qquad u_1 = 7,2$$

2. Následne vypočítame harmonický stred $h(x', U_j)$. Vypočítame vzdialenosť medzi x' a priemernou inštanciou definovanou v prvom kroku pre každú cieľovú triedu. Pričom premenná *r* nadobúda hodnotu 3.

$$h(x', U_0) = 1.6$$

$$h(x', U_1) = 1.5$$

3. Definujeme nový harmonický stred $H(x', h^j)$.

$$H(x', h^0) = 4.81$$

$$H(x', h^1) = 4.50$$

4. Vyberieme minimum $(x', h^1) = 4.50$
5. Novo pridanej inštancie x' priradíme triedu 1.

2.4.3 WKNN

WKNN alebo weighted KNN je variant algoritmu KNN, ktorý rieši spomínané problémy medzi, ktoré radíme výber hodnoty *k* a fázu hlasovania. Pri tejto variante KNN je

ku každej inštancii spomedzi K-najbližších susedov pridaná váha prostredníctvom funkcie $w(x_i)$. Hlavná myšlienka variantu spočíva v tom, že inštancie, ktoré sa nachádzajú vo väčšej blízkosti budú mať pri hlasovaní väčší vplyv na rozhodnutie ako tie, ktoré sú vzdialenejšie. Funkcia $w(x_i)$ môže byť implementovaná viacerými spôsobmi. Pri našej implementácii bola využitá inverzná vzdialenostná funkcia [27].

Pseudokód WKNN

- Nájdi K-najbližších susedov.
- Každej inštancii pridaj váhu $w(x_i)$ podľa nasledovného vzťahu.

$$w(x_i) = \frac{1}{d(x', x_i)}$$

- Urči cieľovú (výslednú) triedu y^* pre novo pridanú inštanciu x' pre údaj prostredníctvom hlasovania nasledovným vzťahom. Premenná v predstavuje cieľovú triedu inštancie x, čiže cieľovú triedu najbližšieho suseda. Premenná y_i predstavuje možné hodnoty cieľovej triedy.

$$y^* = \arg max \sum_{i=1}^{k} w_i \times I(v = y^i) \qquad I(áno) = 1;\ I(nie) = 0$$

Príklad použitia

Na príkladné zobrazenie, kde hodnota premennej k = 5, variantu WKNN sme využili nasledovnú tabuľku 2.

Tabuľka 2: Príklad použitia WKNN

Inštancia	Výsledná trieda inštancie	Euklidovská vzdialenosť	Váha
x_1	0	0.1	10
x_2	0	0.4	2.5
x_3	1	0.5	2
x_4	1	1	1
x_5	1	1	0.5

Pri klasickom algoritme KNN by sme pre novo pridanú inštanciu na základe hlasovania zvolili výslednú triedu 1. S použitím variantu KNN sme dospeli k nasledovnému rozdeleniu.

0: 10 + 2.5 = 12.5

1: 2 + 1 + 0.5 = 3.5

Na základe algoritmu je potrebné vybrať najväčšiu hodnotu, čo ma za následok zmenu výslednej triedy z 1 → 0.

3 Validácia a hodnotenie klasifikačných algoritmov

V predchádzajúcej kapitole sme popísali implementované variácie algoritmu KNN. Dôvodom vzniku týchto variácií bolo zvýšenie celkovej presnosti klasifikačného algoritmu. Ako zistíme, ktorý implementovaný variant ponúka najpresnejšiu klasifikáciu? Ako získame hodnotu celkovej presnosti? Nielen na tieto otázky zodpovieme v tejto kapitole.

3.1 Matica zámen

Predstavuje tabuľku so 4 rôznymi kombináciami s predikovanými a aktuálnymi hodnotami [28].

Tabuľka 3: Matica zámen

	Pozitívny výsledok (Skutočná hodnota)	Negatívny výsledok (Skutočná hodnota)
Pozitívny výsledok (Predikovaná hodnota)	TP	FP
Negatívny výsledok (Predikovaná hodnota)	FN	TN

Vysvetlenie kombinácií v matici zámen je nasledovný [28]:

- TP (true positive) - Predikovali sme pozitivitu a predikcia bola správna.
- FP (false positive) - Predikovali sme pozitivitu napriek negatívnemu výsledku.
- FN (false negative) - Predikovali sme negativitu napriek pozitivite.
- FN (true negative) - Predikovali sme negativitu a predikcia bola správna.

Na základe matici zámen sa vyhodnocujú tzv. ukazovatele, ktoré nám poskytujú informácie o citlivosti (senzitivite), pozitívnej predikovanej hodnote a F_1 miere [28].

Senzitivita

Cieľom je najvyššia hodnota. Predstavuje pozitívne prípady, ktoré boli správne predikované. Senzitivita je definovaná podľa nasledovného vzťahu [28].

$$TPR = \frac{TP}{TP + FN}$$

Pozitívna predikovaná hodnota

Cieľom je najvyššia hodnota. Pozitívna predikovaná hodnota popisuje informáciu zo všetkých prípadov, ktoré sme definovali ako pozitívne, koľko ich pozitívnych skutočne bolo. Matematicky je túto hodnotu možné definovať nasledovne [28].

$$PPV = \frac{TP}{TP + FP}$$

F_1 miera

Je náročné porovnávať dva modely s nízkou pozitívnou predikovanou hodnotou a vysokou senzitivitou. Aby sme ich mohli porovnať používame F_1 mieru, ktorá namiesto aritmetického priemeru využíva harmonický priemer. F_1 miera je definovaná nasledovne [28].

$$F_1\, miera = \frac{2 * TPR * PPV}{PPV + TPR}$$

3.2 Krížová validácia

Vieme, že dataset sa rozdeľuje na trénovaciu a testovaciu množinu. Krížová validácia zabezpečí rozdelenie dát do takzvaných blokov. Postup krížovej validácie je definovaný nasledovne [29].

- Rozdeľ dataset do x blokov. Každý blok obsahuje rovnaký počet inštancií.
- Pre každý blok vykonaj nasledovné:
    - Definuj jeden blok ako testovaciu množinu. Je dôležité, aby sa ako testovacia množina vždy definovala iná množina. Zvyšné množiny definuj ako trénovacie množiny.
    - Vykonaj vyhodnotenie.
- Výsledky pre každý blok spriemeruj a vyhodnoť.

V diplomovej práci sa používa 10-násobná krížová validácia. Veľká výhoda spočíva hlavne pri menšom množstve dát, vzhľadom na jej pretestovanie celej množiny dát.

4 Implementačný návrh a implementácia algoritmov

V kapitole 2 sme sa venovali popísaniu klasifikačnému algoritmu KNN. Preskúmali sme možnosti, ktoré by mohli eliminovať jeho nevýhody. Charakterizovali sme rôzne varianty, ktoré môžu viesť k zvýšeniu celkovej presnosti algoritmu. Po získaní týchto znalostí sme mohli pristúpiť k samotnej implementácii algoritmu KNN a jeho variant. Prácu sme implementovali prostredníctvom programovacieho jazyku Java, pričom sme využili aj knižnicu WEKA. Knižnica WEKA nebola len využitá, ale aj rozšírená o vlastnú implementáciu údajových štruktúr Kd-Tree a Ball-Tree. Knižnica bola rozšírená o varianty fuzzy KNN, weighted KNN a algoritmu na princípe harmonického stredu HMDKNN.

4.1 WEKA

Vzhľadom na nekonečné množstvo prípadov využitia hĺbkovej analýzy dát existuje množstvo knižníc, ktoré poskytujú možnosti na prácu s hĺbkovou analýzou dát. Medzi tieto knižnice sa radí aj knižnica WEKA, ktorá je voľno dostupná. Je to kolekcia klasifikačných algoritmov a nástrojov na prípravu dát, klasifikáciu, zhlukovania, tvorbu asociačných pravidiel, validáciu výsledkov a aj výslednú grafickú vizualizáciu. Napísaná je v programovacom jazyku Java [30].

Počas práce s knižnicou WEKA sme sa najčastejšie stretávali s nasledovnými pojmami [31]:

- ARFF – preferovaný formát dát, ktorý WEKA využíva na ukladanie súborov.
- Instances – objekt reprezentujúci naše inštancie, čiže dáta, ktoré spracovávame a na základe ktorých chceme robiť predikcie a experimenty.
- Classifier – klasifikátor vytvorený na základe inštancií.
- Evaluation – objekt, ktorý využíva validačné nástroje ako maticu zámen a krížovú validáciu pre ohodnotenie implementovaného algoritmu.
- Filter – objekt zabezpečujúci transformáciu numerických hodnôt na nominálne hodnoty.

WEKA využíva na klasifikovanie inštancií prostredníctvom algoritmu KNN objekt IBk. Tento klasifikátor umožňuje na základe parametrov(options) definovať hodnotu premennej *k*, prípadne dokáže definovať rôzne typy variant na základe určitých pravidiel

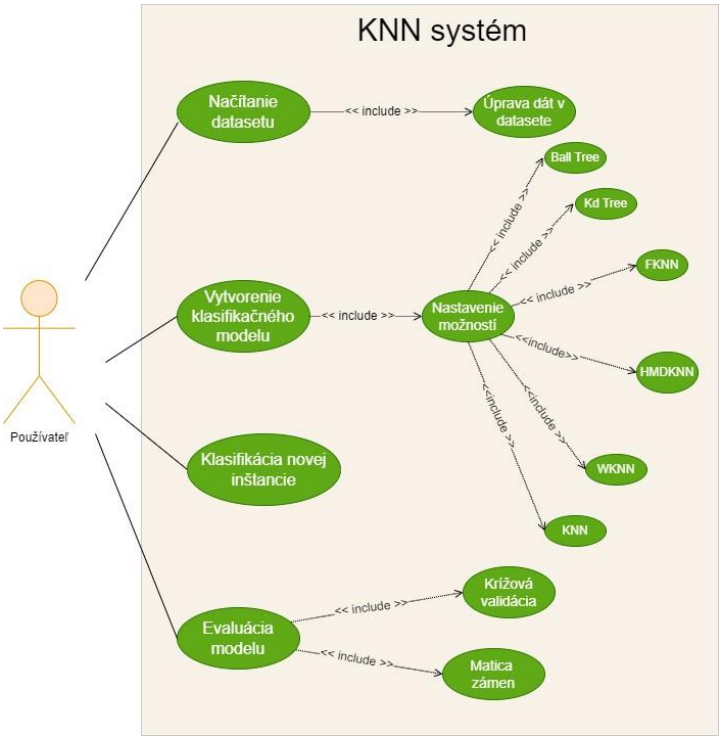
a rôzne iné. Nanešťastie knižnica neposkytuje spôsob, akým je možné definovať popísane varianty FKNN a HMDKNN. Tieto nedostatky, vzhľadom na voľnú dostupnosť knižnice, môžu byť doplnené a knižnica môže byť o tieto rozšírenia obohatená. Spôsob, akým sme implementovali algoritmus KNN musel preto spĺňať všetky podmienky na to, aby mohla byť knižnica WEKA rozšírená práve o túto časť.

Aby sme mohli túto knižnicu využiť, implementovali sme, prostredníctvom nástroja na automatizáciu Gradle, nasledovnú závislosť.

```
dependencies {
	implementation group: 'nz.ac.waikato.cms.weka',
			name: 'weka-stable',
			version: '3.8.0'
}
```

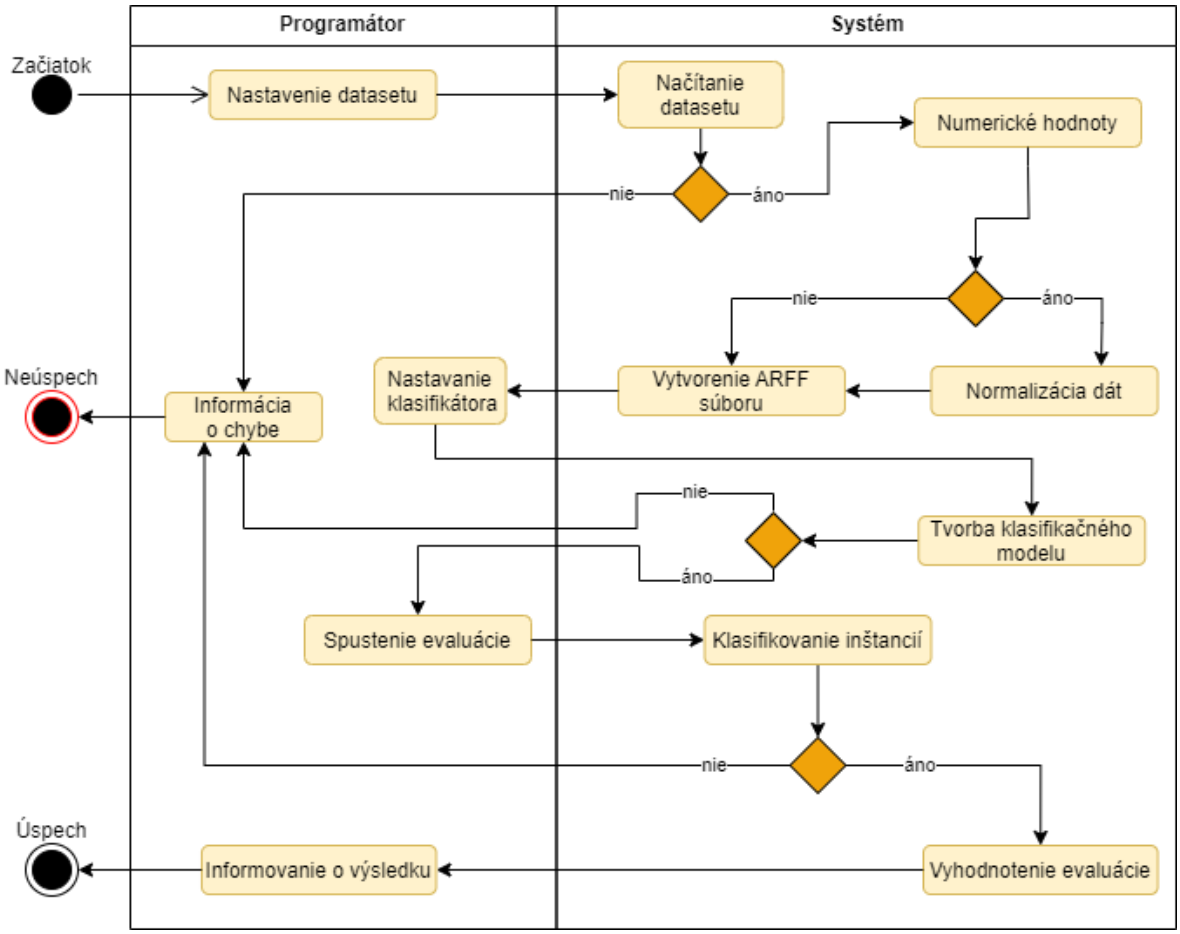
4.2 Návrh

Výsledkom práce je implementovanie rozšírenia knižnice WEKA o možnosti využitia Kd-Tree a Ball-Tree spoločne s možnosťou zadefinovania variant FKNN, HMDKNN a WKNN. Pre podrobnejší popis funkčných požiadaviek sme pridali nasledovný diagram prípadov použitia zobrazený na obrázku 12.



Obr. 12 Diagram prípadov použitia

Zobrazený diagram popisuje rôzne prípady použitia rozšírenia knižnice WEKA spoločne s aplikáciou. Prípad použitia aktivuje účastník, ktorý je prezentovaný programátorom, ktorý rozšírenie využije. Diagram aktivít je znázornený na nasledovnom obrázku 13.



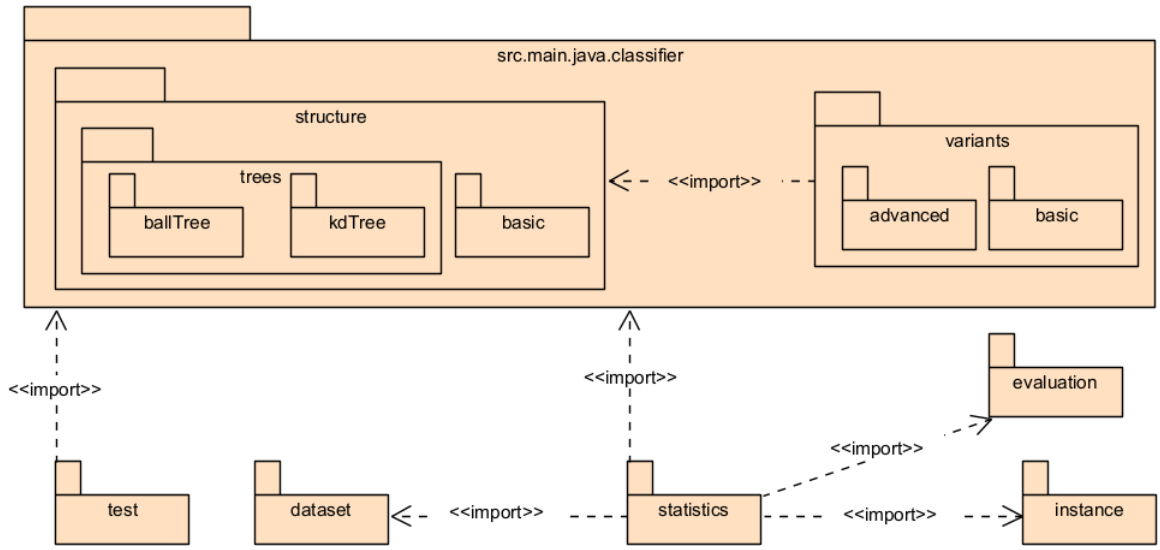
Obr. 13 Diagram aktivít procesu klasifikácie

4.3 Implementácia

Implementačný proces je možné rozdeliť do nasledovných etáp:

- Spracovanie datasetu prostredníctvom knižnice WEKA.
- Spracovanie inštancií vytvorených z datasetu v spolupráci s knižnicou WEKA.
- Implementácia štruktúr.
- Implementácia algoritmu KNN a variant.
- Implementácia štatistík za účelom porovnania.
- Implementácia testov na overenie funkčnosti implementovaných štruktúr.

Etapy implementačného procesu sa odzrkadľujú aj v návrhu samotnej implementácii, vzhľadom na rozloženie balíčkov, ktoré je možné vidieť na nasledovnom obrázku 14.



Obr. 14 Model balíčkov v aplikácii

Balíček dataset

Logika balíčku je implementovaná prostredníctvom triedy DatasetManager, pričom spočíva v nasledujúcom.

- Načítanie datasetu prostredníctvom CSVLoader.
- Transformácia numerických hodnôt na nominálne hodnoty pomocou filtra.
- Vytvorenie ARFF súboru.

Balíček instance

Trieda InstanceManager zabezpečuje vykonanie nasledovných úloh.

- Načítanie údajov z ARFF súboru.
- Vytvorenie objektu Instances spoločne s nastavením cieľovej triedy spomedzi všetkých atribútov.
- Rozdelenie inštancií na trénovaciu a testovaciu množinu.

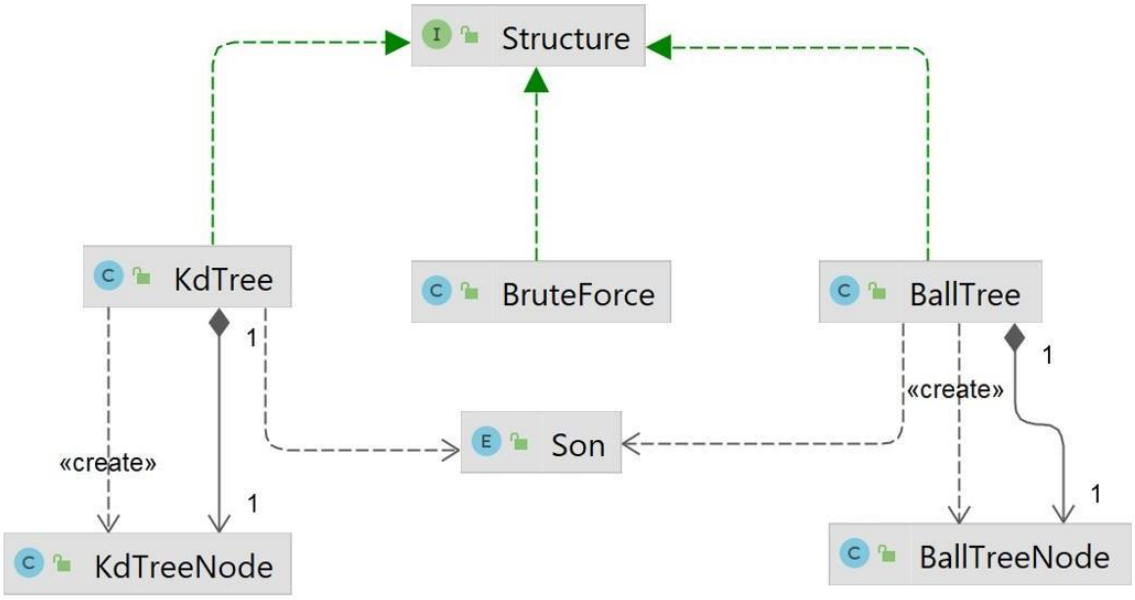
Balíček classifier

Balíček predstavuje „nosnú časť“ celej aplikácie. Obsahuje totiž triedu, ktorej abstraktným predkom je trieda AbstractClassifier. Abstraktná trieda je súčasťou knižnice WEKA. V prípade, že programátor chce vytvoriť rozšírenie knižnice so zameraním na klasifikáciu je

nutné dediť od tejto triedy. Spôsob nastavenia algoritmu ako typ štruktúry alebo variantu je preto implementovaný na základe pravidiel a konvencií knižnice WEKA. Súčasťou balíčka je aj naša implementácia euklidovskej vzdialenosti. WEKA poskytuje rôzne metriky na rátanie vzdialeností. Pričom využíva rôzne postupy, metodiky a normalizácie, ktoré rozširujú základný výpočet vzdialeností. Pri experimentovaní sme preto skúmali aj rozdiely v celkovej presnosti a rýchlosti pri použití týchto vzdialeností. Nami implementovanú vzdialenosť je možné použiť aj v knižnici WEKA, vzhľadom na to, že implementuje rozhranie DistanceFunction. Balíček classifier obsahuje balíček structure a balíček variant.

Balíček structure

Hlavným cieľom balíčku je implementácia popísaných štruktúr Ball-Tree, Kd-Tree a hrubej sily prostredníctvom popísaných slovných popisov, pseudokódov a príslušných algoritmov. Diagram tried spoločne s ich závislosťami, ktoré sú súčasťou balíčka je zobrazený na obrázku 15.



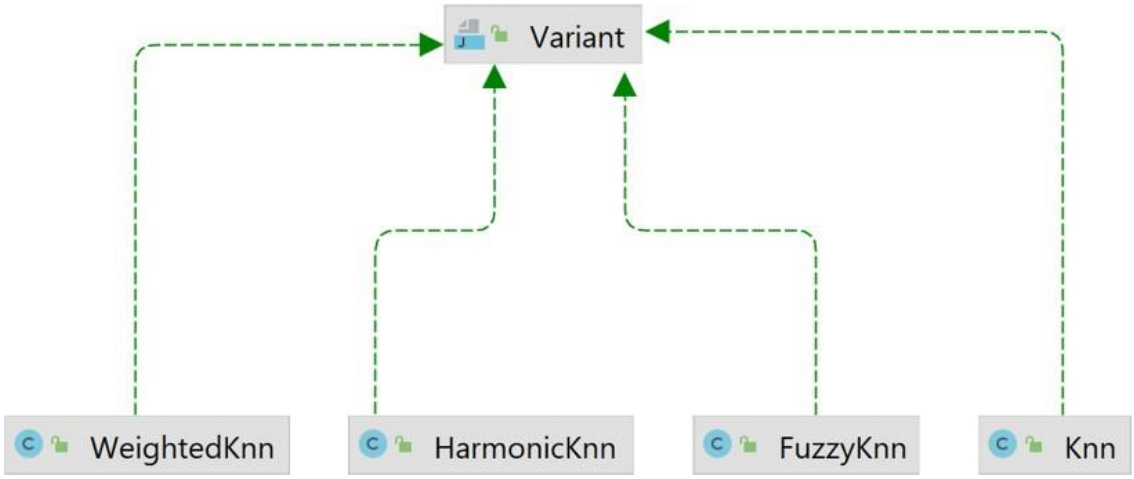
Obr. 15 Diagram tried v balíčku structure

Rozhranie Structure predstavuje „vstupnú bránu“ balíčka. Každá štruktúra, ktorá je implementovaná, prípadne štruktúra, ktorá bude implementovaná v budúcnosti, musí implementovať toto rozhranie. Rozhranie zabezpečí, že štruktúra podporuje algoritmus vytvorenia štruktúry, algoritmus nájdenia K-najbližších susedov, získanie všetkých atribútov inštancie a získanie informácií o nastavených možnostiach. Súčasťou rozhrania je aj vnútorná trieda, ktorá predstavuje pár atribútov <inštancia, vzdialenosť> za účelom efektívnejšieho spracovania základných údajových štruktúr ako je prioritný front. Trieda

Son je špeciálny typ triedy enum. Predstavuje možnosti spracovania potomkov v stromovej štruktúre.

Balíček variant

Balíček obsahuje varianty, ktoré sú v rámci práce implementované. Každý variant je implementovaný ako samostatný objekt, ktorý musí implementovať rozhranie variant. V prípade, že sa v budúcnosti objaví iný variant na základe novej štúdie, ktorý bude prínosom pre knižnicu WEKA, je potrebné, aby implementoval toto rozhranie. Rozhranie Variant zahŕňa klasifikovanie novej inštancie. To znamená, že výstupnou hodnotu bude výsledná cieľová trieda novo pridanej inštancie. Taktiež zahŕňa pravdepodobnostné rozdelenie možných cieľových tried pre novo pridanú inštanciu. Toto rozdelenie je potrebné implementovať vzhľadom na použitie 10-násobnej krížovej validácie. Diagram tried balíčka variant je znázornený na nasledovnom obrázku 16.



Obr. 16 Diagram tried balíčka classifier

Balíček tests

Jeho cieľom je otestovať funkcionalitu jednotlivých štruktúr a tried. Princíp testov je možné rozdeliť do dvoch kategórií. Princíp prvej kategórie spočíva v schopnosti nájdenia všetkých K-najbližších susedov, ktorý sú určený na základe pseudogenerátora. Hodnoty týchto susedov sú totiž z iného intervalu a preto, keď algoritmus nájde suseda, ktorý do tohto intervalu nepatrí, test odhalil nesprávnosť danej štruktúry. Princíp druhej kategórie testov spočíva v porovnávaní nájdených susedov so susedmi, ktorý boli nájdený prostredníctvom hrubej sily. Ak štruktúra Ball-Tree alebo Kd-Tree nemá 100% zhodu vo vzdialeností, test odhalil nesprávnosť implementácie danej štruktúry. Je potrebné si uvedomiť, že predmetom porovnávania musia byť vzdialenosti K-najbližších inštancií od novo pridanej inštancie a nie

samotné inštancie. Dôvodom je, že viaceré inštancie môžu mať rovnaké vzdialenosti a preto by test vyhodnotil štruktúru ako neúspešnú, hoci by bola implementovaná správne.

Balíček statistics

Logika tried obsiahnutá v balíčku ma za účel vykonať rôzne štatistické porovnávania a experimenty. Výsledky, postupy a účel porovnávaní implementovaných štruktúr a variant bude popísaný v nasledujúcej kapitole 5.

Balíček evaluation

Vytvára evalvačný model na základe klasifikátora a testovacej množiny. Implementuje maticu zámen a 10-násobnú krížovú validáciu. Informuje používateľa o výsledkoch evaluácie.

Na základe implementačného návrhu sme mohli implementovať samotné algoritmy. Implementované štruktúry ako Ball-Tree a Kd-Tree je možné následne porovnať a vykonať experimenty predovšetkým z hľadiska časovej efektivity a presnosti. Implementácia variant fuzzy KNN, weighted KNN a HMDKNN umožňuje vykonať porovnania z hľadiska celkovej presnosti daného algoritmu.

5 Porovnanie implementovaných algoritmov

Vzhľadom na to, že popísané algoritmy boli implementované, mohli sa vykonať rôzne experimenty a štatistiky. V úvodnej kapitole sme popísali, že veľmi dôležitým indikátorom algoritmu v hĺbkovej analýze dát je jeho presnosť s akou dokáže predikovať rôzne ochorenia. Následne sme sa snažili nájsť efektívnejšie riešenie z časového hľadiska pre operáciu nájdenia K-najbližších susedov, oproti klasickému použitiu hrubej sily. V rámci tejto kapitoly boli vykonané nasledujúce porovnania:

- Porovnanie implementovaných štruktúr z časového hľadiska.
- Porovnanie implementovaných variant z hľadiska výslednej presnosti identifikovania výslednej triedy pre novo pridanú inštanciu.
- Dodatočné porovnania (experimenty).

5.1 Porovnanie implementovaných štruktúr

V predchádzajúcich kapitolách sme popísali, že existujú tri spôsoby akým je možné získať K-najbližších susedov v algoritme KNN. Medzi tieto spôsoby zaraďujeme hrubú silu, Ball-Tree a v neposlednom rade Kd-Tree. Aby sme dokázali overiť, ktorá implementovaná štruktúra je pre prácu s algoritmom KNN najvýhodnejšia, vytvorili sme nasledovný scenár. Princíp scenára spočíval v porovnaní štruktúr v dvoch kategóriách. Prvá kategória zaznamenávala čas, za ktorý boli nájdení K-najbližší susedia pre novo pridanú inštanciu. Druhá kategória merala čas, počas ktorého dokáže štruktúra vytvoriť príslušný strom. Vzhľadom na to, že pri spôsobe hrubou silou sa nevytvára žiadna stromová štruktúra, tento spôsob nebol zaznamenávaný. Súčasťou porovnávania sú aj štruktúry Kd-Tree a Ball-Tree implementované knižnicou WEKA.

Najvýhodnejšie je rýchlosť štruktúry otestovať na veľkom a náhodnom množstve dát. Vzhľadom na absenciu týchto dát, boli dáta pre tento účel generované prostredníctvom generátora pseudonáhodných čísel z Java knižnice. Počas testovania sa vykoná 10krát opakovanie cyklu, kde sa vždy nastaví iná násada generátora. Hodnoty atribútov sú náhodne generované, aby sa zabezpečila, čo najväčšia diverzita.

Scenár č. 1

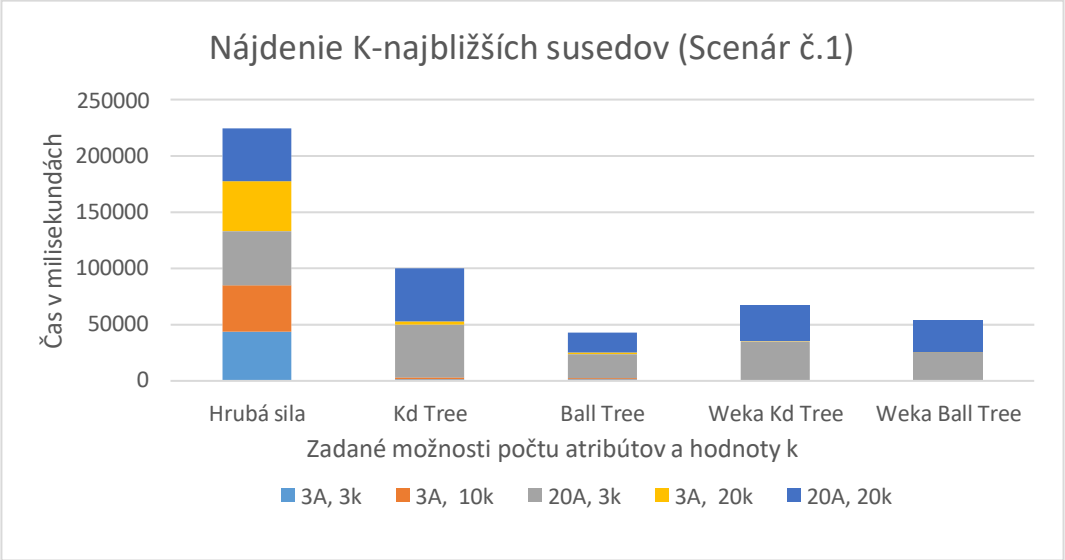
Cieľom tohto scenára je zistiť poradie jednotlivých štruktúr z hľadiska rýchlosti nájdenia K-najbližších susedov. Štruktúra sa naplní prostredníctvom vygenerovaných

10 000 inštancií. Pričom sa vyhľadávajú najbližší susedia pre 2 000 inštancií. Počet atribútov označené prostredníctvom písmena A, hodnota premennej *k* a výsledok scenára je zobrazený prostredníctvom nasledovnej tabuľky 4. Pod každým názvom štruktúry sa nachádza aj časová hodnota vyjadrená v milisekundách. Hodnota udáva celkový čas, za ktorý štruktúra vykonala scenár č. 1.

Tabuľka 4: Výsledky štatistického porovnania scenára č. 1

A	k	Výsledné poradie				
		5.	**4.**	**3.**	**2.**	**1.**
3	3	Hrubá sila	Kd-Tree	Ball-Tree	Weka Kd	Weka Ball
		43 772	1 390	1 289	221	96
3	10	Hrubá sila	Kd-Tree	Ball-Tree	Weka Kd	Weka Ball
		41 257	1 824	1 130	234	134
20	3	Hrubá sila	Kd-Tree	Weka Kd	Weka Ball	Ball-Tree
		47 926	46 717	34 724	25 340	21 354
3	20	Hrubá sila	Kd-Tree	Ball-Tree	Weka Kd	Weka Ball
		44 419	2 892	1 449	346	248
20	20	Kd-Tree	Hrubá sila	Weka Kd	Weka Ball	Ball-Tree
		47 409	47 076	31 847	28 324	17 602

Dáta zobrazené v tabuľke ukazujú, že nami implementovaný Ball-Tree a Ball-Tree, implementovaný ako súčasť knižnice WEKA, sa jednoznačne preukazujú ako najefektívnejšie štruktúry zo všetkých štruktúr. Na základe tohto porovnania algoritmov vieme dokázať, že implementované štruktúry majú zmysel oproti použitiu hrubej sily. Na základe tabuľky 4 môžeme usúdiť, že počet atribútov a počet K-najbližších susedov nemá výrazný vplyv na čas pri použití hrubej sily. V prípade porovnania výsledkov experimentu medzi Kd-Tree a Ball-Tree sa rozhodujúcim faktorom javí počet atribútov. Čím je počet atribútov väčší, tým je štruktúra Ball-Tree výhodnejšia. Príliš veľký počet atribútov bude robiť problémy pre všetky algoritmy. Informácie z tabuľky sú aj zobrazené prostredníctvom skladaného stĺpcového grafu zobrazeného na obrázku 17.



Obr. 17 Grafická reprezentácia scenáru č. 1

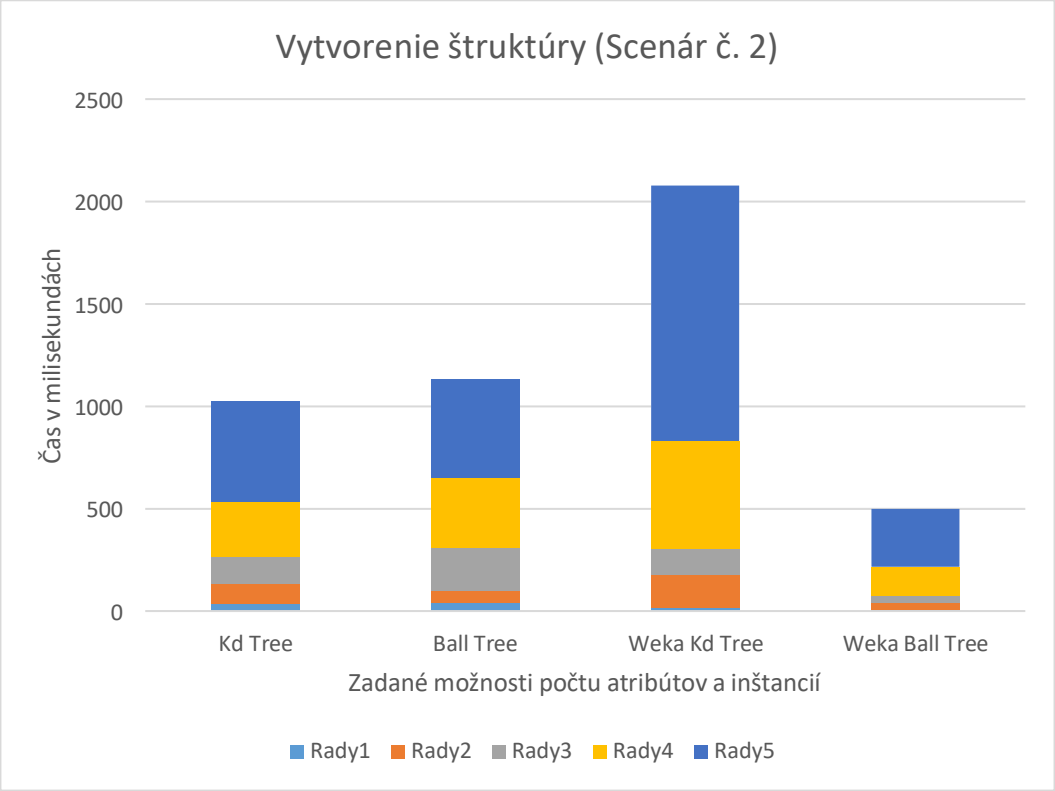
Scenár č. 2

Cieľom tohto scenára je zistiť aká štruktúra dokáže najrýchlejšie spracovať danú množinu dát na následne hľadanie K-najbližších susedov. Výsledok scenára č.2 je rovnako ako pri scenári č.1 zobrazený prostredníctvom tabuľky 5.

Tabuľka 5: Výsledky štatistického porovnania scenára č. 2

Počet atribútov	Počet inštancií	Výsledné poradie			
		4.	**3.**	**2.**	**1.**
3	10 000	Kd-Tree	Ball-Tree	Weka Kd-Tree	Weka Ball-Tree
		40	38	16	5
50	10 000	Weka Kd-Tree	Kd-Tree	Ball-Tree	Weka Ball-Tree
		165	96	61	37
3	50 000	Ball-Tree	Kd-Tree	Weka Kd-Tree	Weka Ball-Tree
		210	133	126	33
20	50 000	Weka Kd-Tree	Ball-Tree	Kd-Tree	Weka Ball-Tree
		524	340	269	142
50	50 000	Weka Kd-Tree	Kd-Tree	Ball-Tree	Weka Ball-Tree
		1247	478	478	283

Informácie zobrazené v tabuľke sú zobrazené aj prostredníctvom skladaného stĺpcového grafu na obrázku 18.



Obr. 18 Grafická reprezentácia scenáru č. 2

Na základe tabuľky 5 a obrázku 18 sa štruktúra Ball-Tree implementovaná prostredníctvom knižnica WEKA javí ako jednoznačne najrýchlejšia možnosť. Keďže operácia nájdi K-najbližších susedov sa využíva častejšie ako operácia vytvorenia štruktúry je pre nás viac smerodajný scenár č.1.

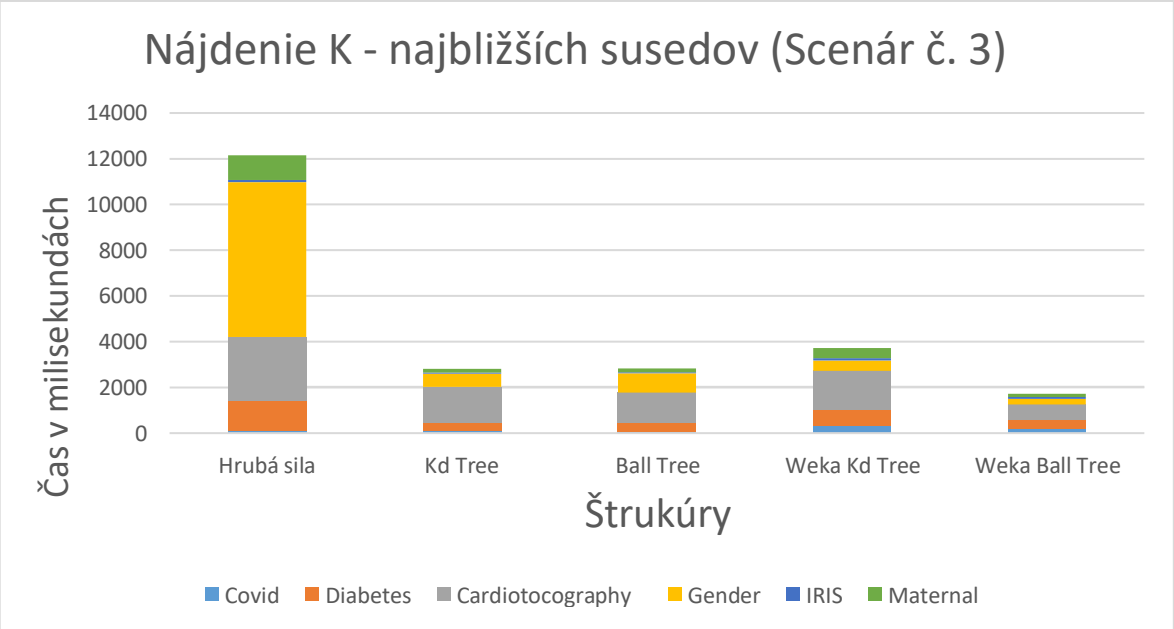
Scenár č. 3

Scenár č. 1 a scenár č. 2 boli vykonávané na vygenerovaných číslach prostredníctvom pseudogenerátora. Vzhľadom na skutočnosť, že náhodné čísla sú pseudonáhodne vytvárajú sa určité vzory a závislosti. Práve kvôli tejto nevýhode sme sa rozhodli porovnať algoritmy pre operáciu nájdenia K-najbližších susedov na datasetoch.

- Dataset Covid-19 – Vzhľadom na to, že ochorenie Covid-19 zapríčinilo celosvetovú pandémiu, dataset s informáciami o tomto ochorení bol predmetom porovnávania algoritmov [32].

- Dataset Diabetes – Súbor údajov obsahujúci informácie extrahované zo súboru Messiderových obrázkov na predikciu, či daný obrázok obsahuje príznaky diabetickej retinopatie alebo nie. Atribúty predstavujú informácie, či je na napríklad na obrázku zistená lézia [33].
- Dataset Cardiotocography – Súbor údajov pozostáva z meraní srdcovej frekvencie plodu a kontrakcie maternice na kardiotokogramoch klasifikovanými odbornými pôrodníkmi. Ako cieľ experimentu pri porovnávaní bola stanovená predikcia stavu plodu [34].
- Dataset Gender – Dataset s medicínskymi dátami určený na testovanie klasifikačných algoritmov. Cieľom je predikcia pohlavia na základe vymyslených dát [35].
- Dataset IRIS – Najznámejší dataset, ktorý predstavuje vstupnú bránu do sveta strojového učenia. Cieľom je predikcia výslednej triedy pre kosatca [36].
- Dataset Maternal – Množina údajov obsahujúca informácie o zdravotných rizikách matky pri tehotenstve, medzi ktoré radíme napríklad vek alebo systolický krvný tlak [37].

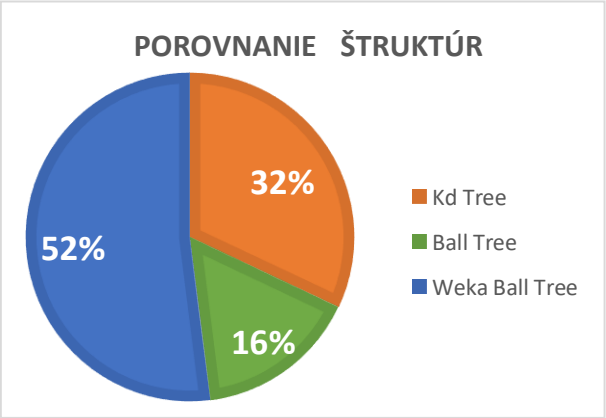
Aby sme zabezpečili čo najlepšie podmienky pre porovnávanie algoritmov, rozhodli sme sa pre každý jeden dataset nastaviť možnosti nájdenia 3, 10, 20, 50 K-najbližších susedov. Výsledky boli následne spracované a sú zobrazené na obrázku 19.



Obr. 19 Grafická reprezentácia scenáru č. 3

Výsledkom je, že Ball-Tree implementovaný knižnicou WEKA spracováva sprístupnené datasety s najvyššou rýchlosťou. Táto skutočnosť môže byť iná pre rôzne zoznamy datasetov.

Scenár bol vykonaný na 6 datasetov, pričom premenná *k* nadobúdala 4 hodnoty. Výsledný počet prípadov porovnania bol teda 24. Princíp ako sme zisťovali, kedy bola štruktúra najrýchlejšia je nasledovný. Pre každý prípad, ktorý bol predmetom porovnania, štruktúra, ktorá bola najrýchlejšia získala jeden bod. Hrubá sila a WEKA Kd-Tree neboli ani v jednom prípade najrýchlejšie, preto ich percentuálny podiel je 0 percent. Percentuálne rozdelenie prípadov zvyšných štruktúr je zobrazené prostredníctvom obrázku 20.



Obr. 20 Percentuálne rozdelenie jednotlivých štruktúr

5.2 Porovnanie implementovaných variant

Štatistické porovnanie a experimenty za účelom porovnania implementovaných variant boli vykonané na spomínaných datasetoch. Datasety boli popísané veľmi stručne. Pre bližšiu predstavu o datasetoch, na ktorých sa vykonávali experimenty sme sa rozhodli jeden dataset z hľadiska medicíny viac priblížiť. Ako príklad je možné uviesť dataset, ktorý obsahoval informácie o pacientoch so zameraním na ochorenie Covid-19. Informácie je možné rozdeliť na dve kategórie. Prvá kategória predstavuje nominálne hodnoty. Hodnoty, ktoré môže daná informácia nadobúdať sú {TRUE, FALSE, UNKNOWN}, prípadne iné nominálne hodnoty. Medzi tieto dáta respektíve informácie radíme nasledovné:

- **Tromboembólia** je ochorenie, pri ktorom sa krvná zrazenina, ktorá sa tvorí v cieve, zvaná trombus uvoľní a následne prenesie do krvného obehu, čo spôsobí zablokovanie inej cievy [38].
- **Hypertenzia (HTN)** alebo vysoký krvný tlak.

- **Ischemická choroba srdca (IHD)** predstavuje súhrnné označenie pre srdcové problémy zapríčinené zúženými srdcovými tepnami, čo vedie k nedostatku kyslíka a v konečnom dôsledku až k infarktu [39].
- **Chronická obštrukčná choroba pľúc (COPD)** je chronické zápalové ochorenie pľúc, ktoré spôsobuje obštrukciu prúdenia vzduchu z pľúc [40].
- **Chronické ochorenie obličiek (CKD)** predstavuje stav, pri ktorom dochádza k poškodeniu obličiek, pričom sa znižuje ich schopnosť udržať si zdravie prostredníctvom filtrovania odpadu z krvi [41].
- **Vazopressor** je liek, ktorý sa používa na zvýšenie kontrakcie srdca u pacientov so šokom [42].
- **Renálna substitučná liečba (RRT)** je terapia, ktorá nahrádza primárnu funkciu obličiek predstavujúcu filtráciu krvi [42].
- **Extra telesné membránové okysličenie (ECMO)** je forma kardiopulmunálnej podpory života, pri ktorej sa krv odvádza z cievneho systému, cirkuluje mimo tela mechanickou pumpou a potom sa vracia späť do obehu [42].
- **Metylprednisolón** je liek určený na liečbu alergických stavov, artritídy, dlhodobého udržiavania astmy a iné [42].
- Ochorenie astma, cukrovka, závislosť pacienta na fajčení, pohlavie, ...

Druhá kategória predstavuje numerické hodnoty, pričom medzi tieto hodnoty radíme nasledovné:

- Vek pacienta, index telesnej hodnoty, teplota.
- **Pozitívny tlak na konci výdychu (PEEP)** je hodnota, ktorú je možné nastaviť u pacientov, ktorí dostávajú invazívnu alebo neinvazívnu mechanickú ventiláciu [42].
- **Podiel vdychovaného kyslíka (FiO_2)** je koncentrácia kyslíka v zmesi plynov [42].
- **(PaO_2)** je hodnota udávajúca parciálny tlak kyslíka v alveolách [42].
- **Reaktívny proteín C (CRP)** udáva hodnotu proteínu syntetizovaného pečeňou, ktorého hladina sa zvyšuje v reakcii na zápal pľúc [42].
- **Feritín** predstavuje v klinickej medicíne hodnotu popisujúcu celkové zásoby železa v tele [42].

- **D – Dimér** je vedľajším produktom procesu zrážania krvi a rozkladu, ktorý je možno merať analýzou vzorky krvi. Za normálnu hodnotu sa považuje menšia hodnota ako 0.5 [42].

Scenár č. 4

Cieľom scenára je prostredníctvom 10-násobnej krížovej validácie určiť presnosť jednotlivých variant na daných datasetoch. Výsledok scenára je rozdelený do dvoch častí. V prvej časti budú odprezentované výsledky na datasete Covid-19 a datasete Diabetes pre rôzne hodnoty *k*. V druhej časti budú výsledky spracované pre všetky datasety pre jednu hodnotu *k*.

Scenár č. 4 – 1

Výsledky vykonané na datasete Covid-19 sú prezentované prostredníctvom nasledovnej tabuľky 6, pričom boli porovnávané nasledovné implementované varianty {KNN (K), fuzzy KNN (F), weighted KNN (W), HMDKNN(H) } a varianty knižnice WEKA {KNN (wK), weighted KNN (wW), Weighted-I KNN (wI) }. Weighted-I KNN sa líši od klasického weighted KNN tým, že váha sa ráta ako 1 – vzdialenosť. Pre lepšiu prehľadnosť je najlepšia presnosť pre daný dataset zobrazená farebne.

Tabuľka 6: Výsledky štatistického porovnania scenára č. 4 - 1

	Dataset Covid-19			Dataset Diabetes		
Hodnota *k* / Variant	3	7	11	3	7	11
K	67.03%	67.54%	67.83%	62.18%	62.42%	62.38%
F	67.17%	67.39%	67.83%	62.14%	62.38%	62.42%
W	66.96%	67.10%	67.61%	62.10%	62.30%	62.29%
H	67.03%	67.03%	66.67%	62.18%	62.24%	62.26%
wK	71.67%	71.45%	71.45%	60.77%	60.85%	61.01%
wW	71.67%	71.45%	71.45%	60.83%	60.86%	61.06%
wI	71.67%	71.45%	71.45%	60.77%	60.85%	61.02%

Knižnica WEKA mala pre dataset Covid-19 najvyššiu presnosť. Pre dataset Diabetes mali najvyššie presnosti náš KNN algoritmus a jeho varianty WKNN a HMDKNN.

Scenár č. 4 – 2

Príprava scenára spočívala v nájdení najväčších presností bez ohľadu na vybranú hodnotu *k*. Scenár bol vykonaný na každom jednom datasete, pre každý jeden variant. Výsledok je zobrazený prostredníctvom nasledovnej tabuľky 7.

Tabuľka 7: Výsledky štatistického porovnania scenára č. 4 - 2

	K	W	F	H	wK	wW	wI
Covid	67.83%	67.83%	67.61%	67.03%	71.67%	71.67%	71.67%
Diabetes	62.42%	62.42%	62.30%	62.26%	61.01%	61.06%	61.02%
Cardiotocography	89.51%	89.38%	89.37%	89.58%	98.85%	98.84%	98.85%
Gender	96.29%	96.28%	96.25%	96.23%	95.93%	95.93%	95.93%
IRIS	91.89%	91.83%	91.78%	91.78%	91.44%	91.56%	91.56%
Maternal	83.04%	82.65%	82.66%	84.25%	83.06%	84.50%	83.30%

Na základe tabuľky vidíme, že nami implementované KNN je pri 3 datasetoch lepšie ako knižnica WEKA. V prípade, že pri porovnávaní neberieme do úvahy knižnicu WEKA, tak HMDKNN je najlepšie v dvoch datasetoch, WKNN taktiež v dvoch a klasický variant KNN je najlepší v 4 datasetoch.

Scenár č. 5 – 1

V predchádzajúcej kapitole sme popísali, že sme implementovali počítanie vzdialeností na základe štandardného euklidovského vzťahu. Samozrejme naše varianty a štruktúry dokážu implementovať aj vzdialenosti implementované knižnicou WEKA, no rýchlosť štruktúr sa dramaticky zníži, čo je zobrazené prostredníctvom scenára č. 5 – 2. Na druhej strane nastane zmena aj v presnosti na daných datasetoch. Princíp scenára zostáva rovnaký ako je v scenári č. 4 – 2. Nové presnosti sú zobrazené v nasledovnej tabuľky 8.

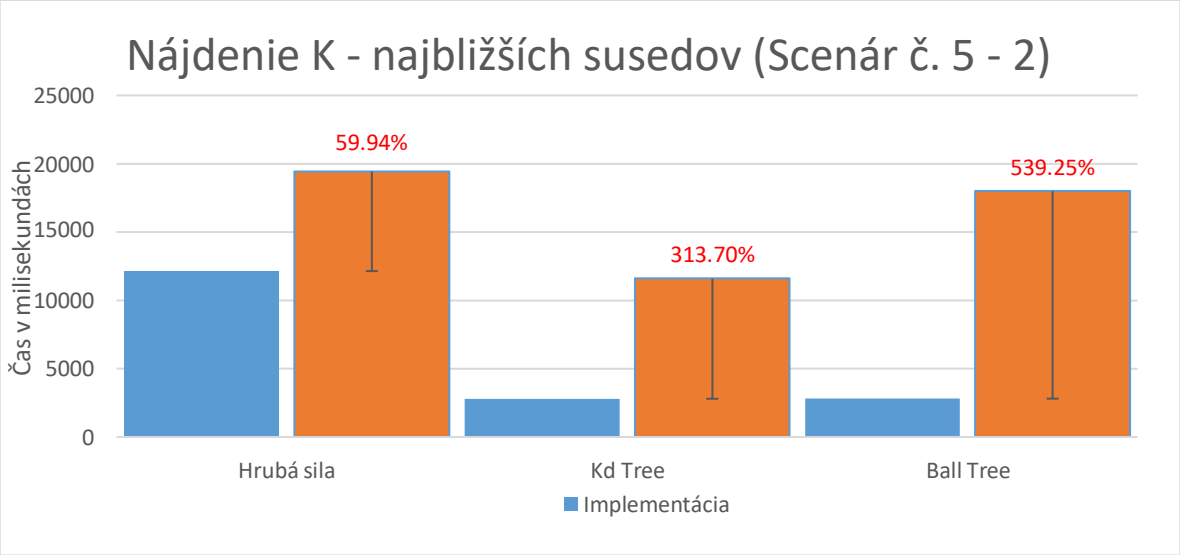
Tabuľka 8: Výsledky štatistického porovnania scenára č. 5 - 1

	K	W	F	H	wK	wW	wI
Covid	72.10%	72.10%	72.17%	70.43%	71.67%	71.67%	71.67%
Diabetes	61.04%	60.94%	60.89%	60.90%	61.01%	61.06%	61.02%
Cardiotocography	98.94%	98.94%	98.94%	98.89%	98.85%	98.84%	98.85%
Gender	95.95%	95.90%	95.89%	95.83%	95.93%	95.93%	95.93%
IRIS	91.56%	91.39%	91.39%	91.44%	91.44%	91.56%	91.56%
Maternal	83.30%	82.91%	82.98%	84.49%	83.06%	84.50%	83.30%

Rozdiely v presnosti sú nasledovné: Covid (+4.34%), Diabetes (-1.38%), Cardiotocography (+9.36%), Gender (-0.34%), IRIS (-0.33%), Maternal (+0.24%). Zvýšená presnosť je viditeľná hlavne pri datasetoch Covid a Cardiotocography. Ako je možné vidieť presnosť pre dva datasety sa dramaticky zvýšila. V nasledujúcom scenári sa pozrieme na ovplyvnenie rýchlosti.

Scenár č. 5 – 2

Zvýšená presnosť, vďaka využitiu spôsobu na rátanie vzdialenosti od knižnice WEKA, nie je zadarmo. Problém je znázornený prostredníctvom operácie hľadania K-najbližších susedov na nasledujúcom obrázku 21.



Obr. 21 Grafická reprezentácia scenáru č. 5 – 2

Všetky štruktúry spomalili proces hľadania K-najbližších susedov. Najvýraznejší rozdiel je pri štruktúre Ball-Tree. Domnievame sa, že je to práve spôsobené tým, že táto štruktúra využíva počítanie vzdialeností najviac krát ako napríklad hľadanie najvzdialenejších inštancií.

5.3 Experimenty

V predchádzajúcich scenároch sme porovnali implementované varianty a údajové štruktúry. Pri porovnávaní presnosti sme zistili, že v prípade využitia implementácie metriky na počítanie vzdialenosti od knižnice WEKA sa zvýši celková presnosť. Táto skutočnosť nás zaujala a preto sme sa rozhodli dodatočne vykonať ďalší scenár.

Scenár č. 6

Cieľom toho scenára je porovnať celkovú presnosť predikcie pri datasete Covid. Predmetom experimentu je porovnanie rôznych metrík na počítanie vzdialeností, ktoré sú implementované knižnicou WEKA. Pre účely experimentu bol použitý základný klasifikačný algoritmus WEKA bez akýchkoľvek variant. Výsledky experimentu sú znázornené prostredníctvom tabuľky 9.

Tabuľka 9: Výsledky štatistického porovnania scenára č. 6

	3	7	11
Euklidovská	71.67%	71.45%	71.45%
Euklidovská (WEKA)	71.09%	71.09%	71.09%
Manhattanovská (WEKA)	71.09%	71.09%	71.09%
Minkowski (WEKA)	71.09%	71.09%	71.09%
Chebyshev (WEKA)	71.09%	71.09%	71.09%

Prostredníctvom štatistického porovnania sme zistili, že použitie nami implementovanej vzdialenosti zvýši celkovú presnosť o 0.56% v najlepšom možnom prípade. Experiment poukázal na skutočnosť, že pri datasete Covid nezáleží akú metriku na počítanie vzdialeností použijeme. Aj takáto skutočnosť vie byť výsledkom experimentu a preto je veľmi dobré a vhodné vziať ju do úvahy. Táto skutočnosť bola vykonaná len pre dataset Covid. Ostatné datasety neboli súčasťou daného scenára.

Scenár č. 7

Pri analyzovaní variant sme skúmali rôzne štúdie, kde sa analyzoval význam týchto variant. Jeden z týchto variant je aj spomínaný variant HMDKNN. V rámci prvých krokov sme vykonali aj definovanie takzvaných priemerných inštancií. Následne sme od priemerných inštancií zistili vzdialenosť novo pridanej inštancie pre každú triedu. Na rozdiel od HMDKNN takzvaný variant LMKNN ukončuje svoj algoritmus práve v tomto bode. Pri variante HMDKNN sme ešte definovali nový harmonický stred. Variant založený na lokálnych priemerov (LMKNN) má za úlohu redukovať negatívny vplyv na algoritmus KNN takzvaných „odchýliek“. Pre porovnanie týchto veľmi podobných variant sme vykonali experiment za cieľom zistiť rozdiel v celkovej presnosti práve pre tieto dva varianty. Výsledky experimentu sú znázornené prostredníctvom tabuľky 10 [43].

Tabuľka 10: Výsledky štatistického porovnania scenára č. 7

k / Variant / Dataset	3		7		11	
	HMD	LM	HMD	LM	HMD	LM
Covid	67.03%	67.03%	67.03%	67.03%	66.67%	66.67%
Diabetes	62.18%	62.15%	62.24%	62.23%	62.26%	62.26%
Cardiotocography	89.58%	89.58%	89.54%	89.54%	89.56%	89.56%
Gender	96.20%	95.38%	96.23%	96.17%	96.21%	96.18%
IRIS	91.44%	91.44%	91.44%	91.44%	91.67%	91.67%
Maternal	84.25%	80.34%	83.58%	81.36%	83.00%	81.96%

Experiment, ktorý sme vykonali dokázal, respektíve opodstatnil použitie posledného kroku v algoritme pri variante HMDKNN. Ani v jednom prípade nebol variant LMKNN totiž lepšou voľbou. Pri datasete Maternal rozdiel v celkovej presnosti bol výrazný a pri hodnote 3 premennej *k* dosahoval +3.91%.

5.4 Diskusia

Výsledok práce sme vyhodnotili prostredníctvom experimentov, ktorých cieľom bolo porovnať implementované varianty a údajové štruktúry fungujúce s klasifikačným algoritmom KNN.

Výsledok porovnania rýchlosti spracovania inštancií a následného hľadania K-najbližších susedov nám ukázal, že existujú datasety a prípady, kedy je vhodné použiť z hľadiska rýchlosti nami implementovanú štruktúru Kd-Tree a Ball-Tree. Táto skutočnosť ale platí v prípade použitia nami implementovanej metriky na rátanie vzdialeností. V opačnom prípade sa rýchlosť štruktúr zníži no zvýši sa presnosť. Tento rozdiel je pri nami implementovaných štruktúrach výrazný narozdiel od štruktúr implementovaných knižnicou WEKA. Dôvod prečo knižnica dosahuje v tomto prípade väčšiu rýchlosť môže byť, že implementácia je určitým spôsobom prispôsobená prostredníctvom určitých algoritmov.

Ak by sme použili nami implementovanú euklidovskú vzdialenosť rozdiel celkovej presnosti pri určitých datasetoch bol výrazný. Nanešťastie rozdiel bol v neprospech pre nami implementované algoritmy. Jeden z možných dôvodov prečo metriky na počítanie vzdialeností dosahujú vyššie výsledky je možnosť určitého sofistikovanejšieho spracovania vzdialeností, normalizácií alebo iných algoritmov. Skúmanie týchto rozdielov alebo príčin môže byť súčasťou ďalšej štúdie.

Záver

V úvode sme definovali niekoľko cieľov diplomovej práce. Prvým cieľom bolo sa oboznámiť zo základmi hĺbkovej analýzy dát. Bolo potrebné analyzovať techniky a algoritmy, ktoré sa pri práci s hĺbkovou analýzou dát používajú. Pričom sme sa primárne venovali klasifikačnému algoritmu KNN. Následne sme spracovali rôzne štúdie, ktoré prostredníctvom hĺbkovej analýzy dát predikovali rôzne ochorenia, detegovali škodlivé lieky alebo odhaľovali podvody v medicíne.

Hlavným pilierom diplomovej práce sa stal klasifikačný algoritmus KNN fungujúci na princípe hľadania K-najbližších susedov. Na začiatok sme popísali tento algoritmus teoreticky. Jeho popis zahŕňal nielen analýzu jednotlivých štruktúr a variant, ale aj vytvorenie jednotlivých krokov, ktoré slúžili ako návod pri implementácií. Praktická časť diplomovej práce zahŕňala implementačný návrh a samotnú implementáciu. Počas implementácie sme získali prehľad o princípe fungovania veľmi populárnej knižnice WEKA. Znalosti, ktoré boli získane pri práci s knižnicou, môžu byť následne využité v budúcej profesijnej kariére.

Vytvorili sme určité varianty a štruktúry, vďaka ktorým môže byť knižnica obohatená. Medzi implementované štruktúry patrí hrubá sila a pokročilé údajové štruktúry Ball-Tree a Kd-Tree. Varianty ktoré boli implementované sú weighted KNN, fuzzy KNN, KNN založené na princípe harmonického stredu (HMDKNN) a variant založený na princípe lokálnych priemerov (LMKNN).

Implementované varianty boli z hľadiska presnosti následne porovnávané a vykonávali sa na nich experimenty. Variant weighted KNN bol najpresnejší v troch prípadoch. Variant fuzzy KNN bol najpresnejší v dvoch prípadoch a variant harmonického stredu HMDKNN bol najpresnejší v troch prípadoch. Pričom do úvahy boli zobrané scenár č. 4 – 2 a scenár č. 5 – 1. V prípade datasetu Covid, obohatenie knižnice o variant fuzzy KNN WEKA bude mať za výsledok o 0.4% presnejšiu predikciu ochorenia s využitím knižnice na počítanie vzdialenosti. V prípade datasetu Diabetes využitie variantu HMD KNN bude knižnica WEKA produkovať presnejšie výsledky o 1.24% za predpokladu využitia nami implementovanej metriky pre počítanie vzdialenosti.

Záverečným cieľom diplomovej práce bolo vytvorenie scenárov za účelom porovnania implementovaných štruktúr a variant. Dáta, ktoré boli použité ako vstup pre

jednotlivé scenáre boli buď generované prostredníctvom pseudogenerátora náhodných čísel alebo boli využité reálne datasety so zameraním na medicínu. Výsledky boli spracované prostredníctvom tabuliek alebo v grafickej forme.

Na základe vytvoreného procesu implementácie štruktúr a variant je možné vytvoriť rozšírenie, ktoré by zahŕňalo buď novú štruktúru alebo nový variant algoritmu. Ďalším rozšírením práce môže byť analýza vplyvu parametrov na implementované varianty fuzzy KNN a HMDKNN. Pri variante fuzzy KNN je možné analyzovať citlivosť parametra *z* a pri HMDKNN sa predmetom skúmania citlivosti stáva parameter *r*.

Zoznam použitej literatúry

[1] "14 areas where data mining is widely used." https://www.researchgate.net/post/14-areas-where-data-mining-is-widely-used (accessed Jan. 23, 2022).

[2] "How does data mining help healthcare? | Data Management, Medical Software, HIPAA, Compliance Certification, Healthcare, Data Analytics Software | Cprime Studios." https://cprimestudios.com/blog/how-does-data-mining-help-healthcare (accessed Jan. 23, 2022).

[3] "What Is Data Mining? How It Works, Techniques & Examples | NetSuite." https://www.netsuite.com/portal/resource/articles/data-warehouse/data-mining.shtml (accessed Jan. 23, 2022).

[4] "Strojové učenie - Definícia." https://smnd.sk/mcibula/zakl_info/definicia.html (accessed Mar. 01, 2022).

[5] "Data Standards in Healthcare: Codes, Documents, and Exchange Formats | AltexSoft." https://www.altexsoft.com/blog/data-standards-healthcare/ (accessed Mar. 01, 2022).

[6] J. Santos-Pereira, L. Gruenwald, and J. Bernardino, "Top data mining tools for the healthcare industry," *Journal of King Saud University - Computer and Information Sciences*, Jun. 2021, doi: 10.1016/J.JKSUCI.2021.06.002.

[7] P. Ahmad, S. Qamar, and S. Qasim Afser Rizvi, "Techniques of Data Mining In Healthcare: A Review," *International Journal of Computer Applications*, vol. 120, no. 15, pp. 38–50, Jun. 2015, doi: 10.5120/21307-4126.

[8] "Basic Concept of Classification (Data Mining) - GeeksforGeeks." https://www.geeksforgeeks.org/basic-concept-classification-data-mining/ (accessed Jan. 26, 2022).

[9] "K-Nearest Neighbors: A Simple Machine Learning Algorithm | RapidMiner." https://rapidminer.com/blog/k-nearest-neighbors-laziest-machine-learning-technique/ (accessed Feb. 16, 2022).

[10] S. Zhang *et al.*, "Using the K-Nearest Neighbor Algorithm for the Classification of Lymph Node Metastasis in Gastric Cancer," *Computational and Mathematical Methods in Medicine*, vol. 2012, p. 11, 2012, doi: 10.1155/2012/876545.

[11] "Data Mining in the Healthcare Industry: Key Benefits & Examples." https://demigos.com/blog-post/data-mining-in-the-healthcare-industry-benefits-examples/ (accessed Jan. 25, 2022).

[12] F. A. Khan, K. Zeb, M. Al-Rakhami, A. Derhab, and S. A. C. Bukhari, "Detection and Prediction of Diabetes Using Data Mining: A Comprehensive Review," *IEEE Access*, vol. 9, pp. 43711–43735, 2021, doi: 10.1109/ACCESS.2021.3059343.

[13] S. Kharya and S. Assistant, "USING DATA MINING TECHNIQUES FOR DIAGNOSIS AND PROGNOSIS OF CANCER DISEASE," *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, vol. 2, no. 2, 2012, doi: 10.5121/ijcseit.2012.2206.

[14] P. Singh, S. Singh, and G. S. Pandi-Jain, "Effective heart disease prediction system using data mining techniques," *Int J Nanomedicine*, vol. 13, no. T-NANO 2014 Abstracts, pp. 121–124, 2018, doi: 10.2147/IJN.S124998.

[15] "7 Advantages and Disadvantages of Data Mining | Limitations & Benefits of Data Mining." https://www.hitechwhizz.com/2021/04/7-advantages-and-disadvantages-limitations-benefits-of-data-mining.html (accessed Mar. 01, 2022).

[16] R. J. Howard and D. L. Cornell, "Ethical Issues in Organ Procurement and Transplantation," *Bioethics - Medical, Ethical and Legal Perspectives*, Dec. 2016, doi: 10.5772/64922.

[17] J. Santos-Pereira, L. Gruenwald, and J. Bernardino, "Top data mining tools for the healthcare industry," *Journal of King Saud University - Computer and Information Sciences*, Jun. 2021, doi: 10.1016/J.JKSUCI.2021.06.002.

[18] "Data Science - MATLAB & Simulink." https://www.mathworks.com/solutions/data-science.html (accessed Apr. 02, 2022).

[19] S. Huang, Q. Qi, J. Liu, and W. Liu, "Tunnel surrounding rock stability prediction using improved knn algorithm," *Journal of Vibroengineering*, vol. 22, no. 7, pp. 1674–1691, Nov. 2020, doi: 10.21595/JVE.2020.21427.

[20] "KNN(K-Nearest Neighbour) algorithm, maths behind it and how to find the best value for K | by i-king-of-ml | Medium." https://medium.com/@rdhawan201455/knn-k-nearest-neighbour-algorithm-maths-behind-it-and-how-to-find-the-best-value-for-k-6ff5b0955e3d (accessed Mar. 02, 2022).

[21] "K-D Trees and KNN Searches Or, 'How Do We Figure Out What Nodes Are In Our Stencil?'".

[22] "Tree algorithms explained: Ball Tree Algorithm vs. KD Tree vs. Brute Force | by Hucker Marius | Towards Data Science." https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940 (accessed Feb. 28, 2022).

[23] "16: KD Trees." https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote16.html (accessed Mar. 02, 2022).

[24] T. Liu, A. W. Moore, and A. Gray, "New Algorithms for Efficient High-Dimensional Nonparametric Classification," *Journal of Machine Learning Research*, vol. 7, pp. 1135–1158, 2006.

[25] M. R. G. JAMES M. KELLER and JR. JAMES A. GIVENS, "A Fuzzy K-Nearest Neighbor Algorithm." https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6313426&casa_token=LJChoF-CD2QAAAAA:EFDYmf9Mn2t13Dt1Cb3AL58EZUGDRY9HfkIXznnhqAHmgVybEde8Sy1usuiuvF_Id_jXlsYqlxOpSg&tag=1 (accessed Feb. 17, 2022).

[26] Hongxing Ma, Jianping Gou, Weihua Ou, Shaoning Zeng, Yunbo Rao, and Hebiao Yang, "A new nearest neighbor classifier based on multi-harmonic mean distances." https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8304246&casa_token=HJeUh6k29yIAAAAA:mwDsqn_PbT8NqhJJXyXQQfjYcGfD91HCwkGzwhDz9pzx0An9EYBA-hDBcsaJaXr0BT1Xy2TaAog-qw&tag=1 (accessed Feb. 19, 2022).

[27] "Weighted K-NN - GeeksforGeeks." https://www.geeksforgeeks.org/weighted-k-nn/ (accessed Mar. 01, 2022).

[28] "Understanding Confusion Matrix | by Sarang Narkhede | Towards Data Science." https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (accessed Feb. 19, 2022).

[29] "Cross-Validation in Machine Learning: How to Do It Right - neptune.ai." https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right (accessed Feb. 19, 2022).

[30] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java." https://www.cs.waikato.ac.nz/ml/weka/ (accessed Feb. 19, 2022).

[31] "Use weka in your java code - Weka Wiki." https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/ (accessed Feb. 19, 2022).

[32] A. Takhar *et al.*, "Timing of Tracheostomy for Prolonged Respiratory Wean in Critically Ill Coronavirus Disease 2019 Patients: A Machine Learning Approach," *Critical Care Explorations*, vol. 2, no. 11, p. e0279, Nov. 2020, doi: 10.1097/CCE.0000000000000279.

[33] "UCI Machine Learning Repository: Diabetic Retinopathy Debrecen Data Set Data Set." https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set# (accessed Mar. 21, 2022).

[34] "UCI Machine Learning Repository: Cardiotocography Data Set." https://archive.ics.uci.edu/ml/datasets/Cardiotocography (accessed Mar. 21, 2022).

[35] "Gender Classification Dataset | Kaggle." https://www.kaggle.com/datasets/elakiricoder/gender-classification-dataset (accessed Mar. 21, 2022).

[36] "UCI Machine Learning Repository: Iris Data Set." https://archive.ics.uci.edu/ml/datasets/Iris (accessed Mar. 21, 2022).

[37] "UCI Machine Learning Repository: Maternal Health Risk Data Set Data Set." https://archive.ics.uci.edu/ml/datasets/Maternal+Health+Risk+Data+Set# (accessed Mar. 21, 2022).

[38] "Thromboembolism: Types, Symptoms, Diagnosis, Treatment." https://www.verywellhealth.com/thromboembolism-1745803 (accessed Mar. 09, 2022).

[39] "Silent Ischemia and Ischemic Heart Disease | American Heart Association." https://www.heart.org/en/health-topics/heart-attack/about-heart-attacks/silent-ischemia-and-ischemic-heart-disease (accessed Mar. 12, 2022).

[40] "COPD - Symptoms and causes - Mayo Clinic." https://www.mayoclinic.org/diseases-conditions/copd/symptoms-causes/syc-20353679 (accessed Mar. 12, 2022).

[41] “Chronic kidney disease (CKD) - Symptoms, causes, treatment | National Kidney Foundation.” https://www.kidney.org/atoz/content/about-chronic-kidney-disease (accessed Mar. 12, 2022).

[42] “National Center for Biotechnology Information.” https://www.ncbi.nlm.nih.gov/ (accessed Mar. 12, 2022).

[43] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, and H. Yang, “A generalized mean distance-based k-nearest neighbor classifier,” *Expert Systems with Applications*, vol. 115, pp. 356–372, Jan. 2019, doi: 10.1016/J.ESWA.2018.08.021.

Zoznam príloh

Prílohy

Príloha A: Programátorská príručka k rozšíreniu knižnice WEKA

Rozšírenie knižnice WEKA je voľno dostupné na nasledujúcom odkaze. Odkiaľ je možné zdrojový kód stiahnuť alebo naklonovať prostredníctvom GIT.

https://github.com/MwAajj/dp

Prácu s rozšírením je možné popísať prostredníctvom nasledujúcich krokoch.

1. Načítanie datasetu. (prevod datasetu z .csv do .arff)

```
DatasetManager dM = new DatasetManager(fileName, classIndex:0);
```

2. Vytvorenie inštancií (načítanie arff súboru)

```
InstanceManager iM = new InstanceManager(fileName, classIndex:0);
```

3. Vytvorenie klasifikátora. (Vlastnosti klasifikátora {štruktúra, variant, hodnota k} je možné nastaviť prostredníctvom možností)

```
Classifier alg = new MyAlgorithm(k:9);
```

4. Spracovanie inštancií. (testovacia množina)

```
alg.buildClassifier(iM.getTrain());
```

5. Definovanie evaluácie. (testovacej množiny, všetkých inštancií pre 10 – násobnú krížovú validáciu, nastavanie možných výstupných tried, hodnota seedu)

```
EvaluationManager eM = new EvaluationManager
                            (
                               iM.getTest(),
                               iM.getAll(),
                               new String[]{"healthy", "ill"},
                               10
                             );
```

6. Vykonanie evaluácie.

```
eM.evaluateModel(alg);
```

7. Informovanie o výsledkoch evaluácie.

```
eM.infoPrintSoft();
```

Nastavanie možností klasifikátora

Prostredníctvom nasledovného príkazu,, kde parameter je pole typu String, je možné definovať variantu, štruktúru alebo hodnotu k pre klasifikátor.

```
alg.setOptions(options);
```

Zoznam možností je možné definovať nasledujúco.

{"-K", "-3"}, - Definovanie hodnoty premennej *k*, pre tento prípad 3.

{"-B"}, - Definovanie využitia štruktúry Ball-Tree.

{"-D"}, - Definovanie využitia štruktúry Kd-Tree.

{"-V"}, - Definovanie využitia rozptylu pri Kd-Tree.

{"-W"}, - Definovanie využitia variantu weighted KNN.

{"-F", "-2"}, - Definovanie využitia variantu fuzzy KNN spoločne s hodnotou premennej *z*.

{"-H", "3"}, - Definovanie využitia variantu HMDKNN spoločne s hodnotou premennej *r*.

{"-L", "3"}, - Definovanie využitia variantu LMKKNN spoločne s hodnotou premennej *r*.

Štruktúry a varianty medzi sebou nie je možné kombinovať.

Príloha B: Obsah DVD

Priložené DVD obsahuje:

- Práca v elektronickej podobe (formát PDF)
- Zdrojový kód k aplikácií