VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA KLASIFIKAČNÍCH METOD

ANALYSIS OF CLASSIFICATION METHODS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAKUB JURÍČEK**
AUTHOR

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

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Juríček Jakub, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Analýza klasifikačních metod**
Analysis of Classification Methods
Kategorie: Data mining

Zadání:

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

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Burgetová Ivana, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 30. října 2018

Abstrakt

Táto práca pojednáva o klasifikačných metódach využívaných pri získavaní znalostí z dát a rozoberá možnosti ich validácie a porovnania. Prostredníctvom experimentov sa zameriava na analýzu štyroch vybraných metód: jednoduchý Bayesovský klasifikátor, rozhodovací strom, neurónová sieť a SVM. Skúmané sú faktory ovplyvňujúce základné vlastnosti ako rýchlosť trénovania, rýchlosť klasifikácie, presnosť. Súčasťou práce je desktopová aplikácia, ktorá tvorí prostriedok k trénovaniu, testovaniu a validácii jednotlivých metód. Pre potreby experimentov je vybraných jedenásť referenčných dátových sád. V závere práce sú zhrnuté experimentálne získané výsledky porovnania a pozorované vlastnosti klasifikačných metód.

Abstract

This work deals with the classification methods used in the knowledge discovery from data process and discusses the possibilities of their validation and comparison. Through experiments, the work focuses on the analysis of four selected methods: Naive Bayes classificator, decision tree, neural network and SVM. Factors influencing basic characteristics such as training speed, classification speed, accuracy are examined. A part of the thesis is a desktop application, which is a tool for training, testing and validation of individual methods. Eleven reference data sets are selected for experimental purposes. At the end of this work experimental results of comparison and observed characteristics of classification methods are summarized.

Kľúčové slová

dolovanie z dát, znalosti, klasifikácia, analýza, dátová sada, neurónová sieť, SVM, jednoduchý Bayes, rozhodovací strom, strojové učenie, desktopová aplikácia, experimenty, presnosť, časová náročnosť

Keywords

data mining, knowledge, classification, analysis, data set, neural network, SVM, Naive Bayes, decision tree, machine learning, desktop application, experiments, accuracy, duration

Citácia

JURÍČEK, Jakub. *Analýza klasifikačních metod.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivana Burgetová, Ph.D.

Analýza klasifikačních metod

Prehlásenie

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

.....
Jakub Juríček
18. mája 2019

Poďakovanie

Chcel by som poďakovať svojej vedúcej, pani Ing. Ivane Burgetovej Ph.D., za jej odbornú pomoc a rady, ktoré mi boli poskytnuté pri vypracovávaní diplomovej práce.

Obsah

Kapitola 1

Úvod

Žijeme v dobe plnej informačných technológií, kde je denne vyprodukované obrovské množstvo nových dát. Súkromné správy a dokumenty, popisy produktov, návody, videá, obrázky, zvuky, profily užívateľov, obchodné transakcie, lekárske záznamy, recenzie, vyhľadávané frázy či mnohé ďalšie prvky tvoria enormnú masu dát. Ich vhodnou analýzou je možné dospieť k novým informáciám, predpovedať vývoj udalostí či dáta automaticky kategorizovať. Procesu získavania znalostí z obrovských dátových kolekcií sa hovorí *data mining* [6]. Vzhľadom na množstvo dát spracovanie prebieha s využitím výpočtovej sily často i niekoľkých počítačov. K dispozícii máme dnes mnoho účinných algoritmov, ktoré je možné uplatniť ako pre samotné spracovanie dát, tak i pre dolovanie potrebných znalostí.

Klasifikácia je forma analýzy dát, ktorá odvodzuje modely, ktoré sú schopné popisovať určité triedy (kategórie) dát. Na dopredu daných dátach (napr. lekárske záznamy), u ktorých poznáme kategórie jednotlivých vzoriek, je možné klasifikačné metódy naučiť zaraďovať informácie do týchto kategórií – napr. pacient „má rakovinu“ alebo „nemá rakovinu“. Následne algoritmus dostane nové informácie o pacientovi a na základe toho, čo sa naučil, dokáže s istou presnosťou určiť pacientovu diagnózu. V dnešnej dobe ide o veľmi populárnu formu spracovania veľkého objemu dát, ktorá poskytuje veľmi dobré výsledky v rade oborov.

Cieľom diplomovej práce je implementácia desktopovej aplikácie, ktorá umožňuje prácu s vybranými klasifikačnými metódami a rôznymi dátovými sadami, a následná analýza vlastností týchto metód na základe výsledkov experimentov. Aplikácia umožňuje načítať rôzne dátové sady a vykonať základné úpravy dôležité pre ich ďalšie použitie. Hlavnými funkciami je možnosť výberu metódy alebo metód klasifikácie a následné trénovanie a testovanie. Validáciu je možné vykonať niekoľkými metódami a vo výsledku je možné pozorovať vybrané vlastnosti – rýchlosť učenia, rýchlosť klasifikácie a výslednú presnosť. Záver práce je venovaný spracovaniu výsledkov a vykonaným experimentom, na základe ktorých bolo možné odhaliť špecifické vlastnosti klasifikačných metód a validačných techník.

Kapitola 2 sa zaoberá úvodom do data miningu, popisuje dôležité pojmy a pojednáva o metódach prípravy dát tak, aby boli vhodné pre ďalšie spracovanie a analýzu. Kapitola 3 je zameraná na samotnú klasifikáciu. Postupne predstavuje jednotlivé vybrané metódy, bližšie ich vysvetľuje a zameriava sa na princíp ich fungovania a prípadné výhody a nedostatky. Okrem metód popisuje i existujúce validačné techniky vhodné k ich analýze. V kapitole 4 sú vymenované a stručne popísané zvolené dátové sady a dôvod ich výberu. Rôznorodé dátové sady dokážu odhaliť viaceré nedostatky či silné stránky jednotlivých metód a prispieť tak k dôveryhodnejším výsledkom analýzy. Kapitola 5 je praktického charakteru a popisuje kompletný proces vývoja aplikácie, od špecifikácie požiadaviek, návrhu, cez grafické spra-

covanie až po implementačné detaily a inštaláciu. Predposledná kapitola 6 tvorí významnú časť práce – popisuje experimenty s jednotlivými metódami a dátovými sadami s využitím aplikácie, obsahuje rôzne porovnania, pohľady a závery. Cieľom kapitoly je poskytnúť čitateľovi prehľad o tom, ako veľmi sa jednotlivé metódy líšia pri využití rovnakých dát, aké metódy sú vhodnejšie v ktorom prípade a ktoré spotrebujú najviac času učením/klasifikovaním. Kapitola 7 je zhrnutím výsledkov celej práce a zdôrazňuje dosiahnuté poznatky.

Kapitola 2

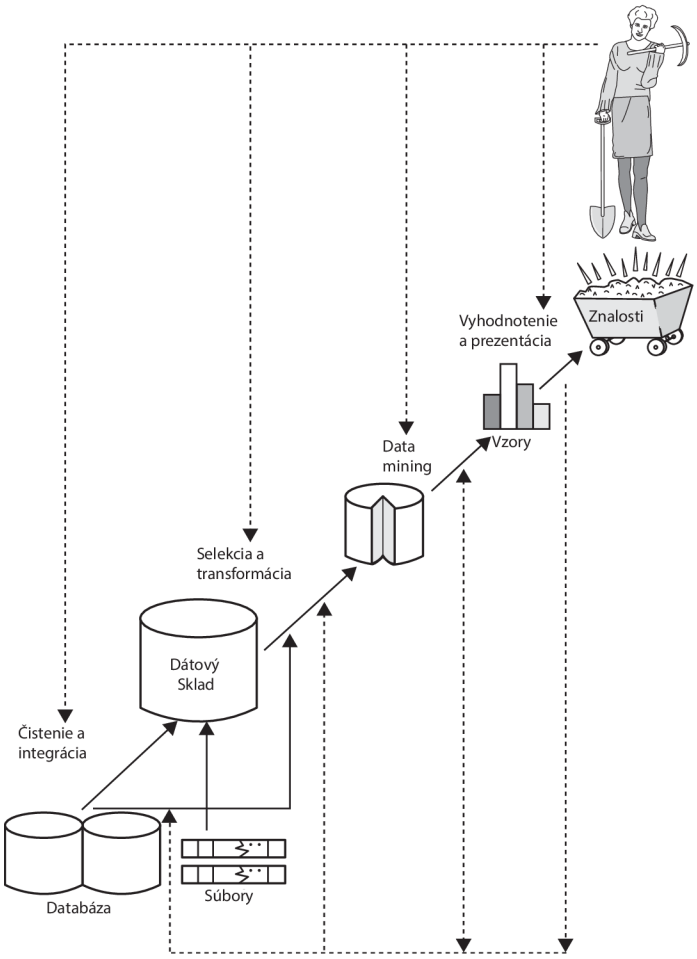
Data mining

Kapitola prináša ľahký úvod do problematiky data miningu, vysvetľuje základné pojmy a súvislosti, ktorých znalosť je kľúčová pre pochopenie ďalších kapitol. V úvode kapitoly sú popísané rôzne druhy dát, ktoré sú súčasťou rôznych dátových sád. Vysvetlené sú odlišné typy dát, s ktorými je možné sa stretnúť a problémy, ktoré treba riešiť. Predstavené sú techniky opravy chýb alebo nedostatkov vo vstupných dátach a spôsoby výberu dôležitých dát. Aby mohli byť sady spracované pokročilejšími metódami, je často nutné previesť hodnoty na iný tvar, vykonať zmenu typov alebo redukovať ich množstvo, k čomu existuje opäť sada zaužívaných techník prebraných v druhej polovici kapitoly. V závere sú popísané druhy získateľných vzorov, ktoré dokážu poskytnúť zaujímavé znalosti o skúmaných dátach. Väčšina informácií v tejto kapitole je z knihy [6].

2.1 Druhy dát

Data mining (dolovanie dát) je pojem, ktorý môže byť definovaný rôznymi spôsobmi. Pokiaľ hovoríme o dolovaní dát, myslíme tým ich získavanie. Keďže dáta väčšinou máme (v podobe dátovej sady), je data mining považovaný skôr za synonymum k termínu získavanie znalostí z dát (hľadanie zaujímavých vzorov) alebo za krok v procese získavania znalostí. Proces získavania znalostí je ukázaný na obrázku 2.1 a skladá sa z iteratívnej sekvencie nasledujúcich krokov [6]:

1. **Čistenie dát** – odstránenie šumu a nekonzistentných dát.
2. **Integrácia dát** – kombinácia niekoľkých zdrojov dát a vytvorenie jednej dátovej sady.
3. **Selekcia dát** – výber dát, ktoré sú vhodné pre konkrétnu plánovanú analýzu.
4. **Transformácia dát** – premena dát do foriem vhodných na dolovanie vykonávaním súhrnných alebo agregačných operácií.
5. **Data mining** – základný proces, pri ktorom sa používajú inteligentné metódy na extrakciu dátových vzorov.
6. **Vyhodnotenie vzorov** – identifikácia len tých vzorov, ktoré reprezentujú znalosti, ktoré nás zaujímajú.
7. **Prezentácia znalostí** – vizualizácia a reprezentácia znalostí vhodnou formou užívateľovi.



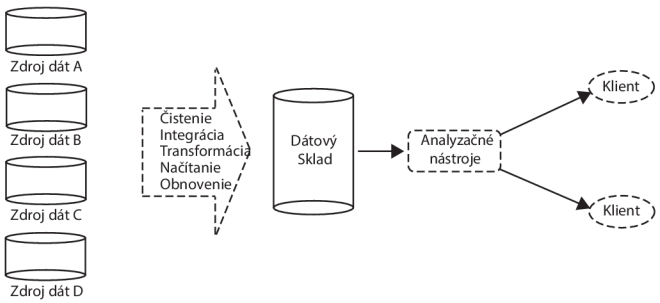
Obr. 2.1: Data mining ako krok v procese získavania znalostí [6]

Niektorým krokom bude podrobnejšie venovaná kapitola 2.3, v ktorej budú spomenuté zaužívané algoritmy a postupy.

Data mining môže byť aplikovaný na zdroje dát rôznych druhov. Medzi najzákladnejšie formy patria databázové dáta, transakčné dáta a dáta z dátových skladov; medzi vedľajšie zdroje môžu patriť i dátové prúdy, sekvenčné, textové, multimediálne dáta či WWW [6].

Databázové dáta – Databázový systém pozostáva z kolekcie vzájomne súvisiacich údajov známych ako databáza a skupiny softvéru, ktorý má za úlohu správu a prístup k údajom. Významným zdrojom dát je napríklad relačná databáza. Je to zbierka tabuliek, z ktorých každej je priradený unikátny názov. Každá tabuľka sa skladá zo súboru atribútov (stĺpcov alebo polí) a zvyčajne ukladá veľkú sadu n-tíc, inak povedané záznamov alebo riadkov. Každá n-tica v relačnej tabuľke predstavuje objekt identifikovaný jedinečným kľúčom a opísaný súborom hodnôt atribútov. Pri dolovaní z relačných databáz môžeme vyhľadávať trendy alebo dátové vzory. Data mining systémy môžu pomôcť odhaliť odchýlky, teda napríklad položky s predajmi, ktoré sa výrazne líšia od očakávaných predajov v porovnaní s predchádzajúcim rokom. Takéto odchýlky je možné ďalej skúmať a zistiť, že napríklad nastala zmena v balení produktu alebo výrazná zmena ceny. Relačné databázy sú jedným z najčastejšie dostupných a najbohatších úložísk informácií, a preto sú hlavným zdrojom dát pre data mining [6].

Dátový sklad – Dátový sklad je úložisko informácií zozbieraných z viacerých zdrojov, ktoré sú uložené v jednotnej schéme a zvyčajne bývajú na jednom mieste. Dátové sklady sú konštruované prostredníctvom procesu čistenia dát, integrácie dát, transformácie dát, načítania dát a periodického obnovovania údajov. Údaje sú zvyčajne uložené tak, aby poskytovali informácie z historického hľadiska (napríklad v posledných 6 – 12 mesiacoch). Skôr ako ukladanie podrobností o každej predajnej transakcii, môže dátový sklad ukladať napríklad súhrn transakcií pre každý typ položky, pre každý obchod alebo pre každý predajný región. Na obrázku 2.2 je znázornený typický príklad konštrukcie a použitia dátového skladu.



Obr. 2.2: Schéma konštrukcie a použitia dátového skladu [6]

Dátový sklad je aktualizovaný len v určitých časových intervaloch (mesačne, ročne). Všetky dáta predstavujú časový snímok dát z produkčných databáz vytvorený v určitom okamihu. Analýza tak prebieha oddelene, takže prípadné porušenie dátového skladu neovplyvní operatívne riadenie, ktoré využíva dáta z produkčnej databázy. Vďaka stálosti dátového skladu dotazy od užívateľov (analytikov) nespôsobujú zmenu uložených dát. Dáta uložené v dátovom sklade predstavujú neutrálny dátový priestor, ktorý nie je vytváraný s myšlienkou konkrétnych analýz. Preto je vhodné vytvárať v náväznosti na dátový sklad špecializované dátové trhy, kam sa presunú iba dáta relevantné pre určitý typ analýz [21].

Transakčné dáta – Každý záznam v transakčnej databáze zachytáva jednu transakciu – nákup zákazníka, rezervácia letu či jedno kliknutie na webovej stránke. Typicky transakcia zahŕňa unikátny identifikátor a zoznam položiek (napr. nakúpené produkty). Data mining nad transakčnými dátami dokáže objaviť napríklad frekventované skupiny položiek, ktoré môžu predstavovať napríklad produkty, ktoré sú často predávané spolu [6].

Iné typy dát – K spomenutým formám dát existuje niekoľko iných druhov, ktorých forma a štruktúra je vhodná pre data mining – historické záznamy, dáta zo senzorov, videozáznam z priemyselných kamier, mapy, dizajnérske návrhy, schémy integrovaných obvodov, text, obrázky, sociálne siete, web... V počítačových sieťach je možné napríklad dolovať anomálie, ktoré indikujú narušenia. V mapách je možné hľadať vzory, ktoré opisujú zmeny v mierach chudoby vo veľkomestách vzhľadom na vzdialenosti od hlavných ciest. Dolovaním z textových dát, napríklad článkov, je možné identifikovať najdiskutovanejšie témy posledných rokov [6].

2.2 Typy atribútov a prevody medzi nimi

Dátové sady sú tvorené dátovými objektami, ktoré reprezentujú jednotlivé entity (vzorky alebo inštancie) vstupných dát (napr. zákazníci, pacienti, obchodné položky). Dátové objekty sú popísané atribútmi. V databázach riadky odpovedajú dátovým objektom a stĺpce atribútom. Atribút je dátové pole reprezentujúce vlastnosť dátového objektu. Súbor atribútov používaných na opis daného objektu sa nazýva vektor atribútov. Typ atribútu závisí na množine možných hodnôt, ktoré môže nadobudnúť – nominálne, binárne, ordinálne alebo numerické hodnoty [6].

Nominálne atribúty – Hodnoty nominálnych (kategorických) atribútov sú symboly, stavy, kategórie alebo názvy vecí. Príkladom môže byť atribút *farba*, ktorý nadobúda hodnoty *červená, zelená, modrá*. Jednotlivým farbám dokážeme priradiť i číselnú hodnotu – *0* pre červenú, *1* pre zelenú a *3* pre modrú, čísla však nie sú použité kvantitatívne. Kľúčovou vlastnosťou nominálneho atribútu je, že matematické operácie nad jednotlivými hodnotami nemajú zmysel. Nominálne atribúty nemajú definované klasické usporiadanie hodnôt a preto je nezmyselné hľadať medián alebo priemer. Zaujímavé však môže byť hľadanie najčastejšie sa vyskytujúcej hodnoty – modus.

Binárne atribúty – Ide o nominálne atribúty s dvoma kategóriami, typicky s hodnotami 0 a 1. Často predstavujú absenciu/prítomnosť daného atribútu alebo v prípade Booleovských hodnôt pravdu/klamstvo. Binárny atribút je symetrický, ak oba stavy majú rovnakú váhu – neexistuje preferencia. O asymetrickom binárnom atribúte hovoríme pokiaľ je jedna hodnota atribútu považovaná za dôležitejšiu alebo významnejšiu.

Ordinálne atribúty – Jedná sa o hodnoty atribútov, ktoré majú zmysluplné poradie, ale rozsah medzi postupnými hodnotami nie je známy. Oproti nominálnym atribútom je možné určiť medián, priemer je však stále nezmyselný. Príkladom môžu byť hodnoty atribútu *pohár – malý, stredný, veľký*.

Numerické atribúty – Oproti predchádzajúcim atribútom sú numerické atribúty kvantitatívne. Je možné merať množstvo a reprezentovať ho celým alebo desatinným číslom. Rozdeľujú sa na [23]:

- **Intervalové** – Mierka hodnôt je rozdelená približne lineárne, je možné určiť rozdiel. Neexistuje nulový bod a hodnoty majú poradie.
- **Pomerové** – Hodnoty majú implicitný nulový bod a sme schopný určiť podiel dvoch hodnôt. Hodnoty sú zároveň len kladné a majú určené poradie.

Mnoho dát, ktoré majú byť skúmané, sú často heterogénne a obsahujú atribúty niekoľkých typov. Z pohľadu analytika tak vzniká pomerne komplikovaná situácia, kedy je teoreticky nútený vytvoriť alebo pozmeniť algoritmus tak, aby využíval konkrétnu kombináciu typov. Tento prístup je však časovo náročný a často nepraktický. Preto je možné previesť niektoré typy atribútov na iné, čím však vždy dochádza k istej strate presnosti a výpovednej hodnoty. Tabuľka 2.1 ukazuje rôzne možnosti prevodu dátových typov. V tejto práci si vystačíme s jednoduchými numerickými a kategorickými typmi, ktoré sú najrozšírenejšie v rade dolovacích algoritmov, a s prevodmi medzi nimi (diskretizácia a binarizácia), popísané v nasledujúcich odstavcoch [2].

Diskretizácia – Jedná sa o prevod numerických dát na kategorické a patrí medzi najčastejšie druhy konverzie dát. Proces diskretizácie rozdeľuje rozsah numerických hodnôt

Zdrojový dátový typ	Cieľový dátový typ	Metódy prevodu
Numerický	*Kategorický*	*Diskretizácia*
Kategorický	*Numerický*	*Binarizácia*
Text	*Numerický*	*Latentná sémantická analýza (LSA)*
Časová rada	*Diskrétna sekvencia*	*Symbolická agregačná aproximácia (SAX)*
Časová rada	*Numerický multidimenzionálny*	*Diskrétna vlnková alebo Furierova transformácia (DWT, DFT)*
Diskrétna sekvencia	*Numerický multidimenzionálny*	*DWT, DFT*
Priestorový	*Numerický multidimenzionálny*	*2-d DWT*
Graf	*Numerický multidimenzionálny*	*Multirozmerné škálovanie (MDS), spektrálna transformácia*
Ľubovoľný typ	*Graf*	*Similarity graph*

Tabuľka 2.1: Prevody rozličných dátových typov [2]

na ϕ menších intervalov. Novým rozsahom je možné následne priradiť hodnoty (napr. 1 až ϕ), ktoré ale tentokrát predstavujú kategorické označenie. Vo výsledku dochádza k určitej strate informácie, ale pokiaľ je parameter ϕ rozumne volený, je často táto strata zanedbateľná pre potreby konkrétnej analýzy.

Proces diskretizácie môže byť vykonaný rôznymi spôsobmi v závislosti od charakteru dát a cieľov analýzy [2]:

1. **Rozdelenie rovnakej šírky** – Zistí sa $[min, max]$ celého rozsahu hodnôt a následne sa rozdelí na ϕ intervalov rovnakej šírky $w = (max - min)/\phi$ [23]. Rozdelenie spĺňa podmienku, že pre každý nový rozsah $[a, b]$ je rozdiel $b - a$ rovnaký. Problémom sú nerovnomerne rozložené dáta, kde jeden interval môže zahŕňať príliš mnoho dát, narozdiel od ostatných, a naopak.
2. **Rozdelenie rovnakej hĺbky** – Rozsah je rozdelený na ϕ intervalov s rovnakým počtom záznamov. Rozdelenie je väčšinou vykonané nad zoradenou postupnosťou hodnôt atribútu.
3. **Logaritmické rozdelenie** – Všetky nové intervaly $[a, b]$ spĺňajú podmienku, že rozdiel $\log b - \log a$ majú rovnaký. Tento druh rozsahu môže byť užitočný, pokiaľ atribút vykazuje exponenciálne rozloženie hodnôt.

Binarizácia – Ide o prevod kategorických dát na numerické. Využíva toho, že binárne dáta sú špeciálnou formou ako numerických, tak i kategorických dát. Ak nadobúda kategorický atribút ϕ rôznych hodnôt, potom je vytvorených ϕ binárnych atribútov zodpovedajúcich jednotlivým hodnotám. Potom práve jeden z týchto atribútov nadobúda hodnotu 1 a ostatné hodnotu 0. Takto prevedené dátové sady je možné následne využiť v algoritmoch vyžadujúcich čisto numerické vstupné dáta. Binarizácia sa často označuje pojmom *One Hot Encoder*.

2.3 Predspracovanie dát

Obrovské množstvo dát nachádzajúcich sa v rôznych databázach alebo produkované rôznymi zdrojmi zvyšuje riziko výskytu chýbajúcich hodnôt, nekonzistencií alebo šumu. Dáta nízkej kvality vedú i k nízkej kvalite získaných znalostí. Medzi hlavné faktory kvality patria *presnosť*, *kompletnosť*, *konzistencia*, *včasnosť*, *dôveryhodnosť* a *interpretovateľnosť*.

Existuje mnoho dôvodov, prečo môžu byť dáta *nepresné* – hodnoty atribútov majú nesprávne hodnoty. Chybné môžu byť nástroje na zber dát alebo nepresnosť nastane pri zadávaní hodnôt na vstupe (napr. zvolenie predvolenej hodnoty pre dátum narodenia) alebo je chyba spôsobená limitáciou konkrétnej technológie. *Nekompletné* dáta môže spôsobiť napríklad nefunkčnosť vybavenia, ktoré malo dané hodnoty zaznamenať, nedorozumenie alebo jednoducho dáta neboli zahrnuté, pretože v dobe ich vzniku neboli považované za dôležité. Ak sú dáta z viacerých zdrojov *nekonzistentné*, môže to viesť k potrebe odstrániť konfliktné časti, čo prispieva opäť k ich nekompletnosti. *Včasnosť* hrá významnú rolu hlavne pokiaľ dochádza k analýze dát za nejaké časové obdobie. Ak napríklad neboli včas doplnené dáta za prvý mesiac, môže analýza takýchto nekompletných dát na začiatku druhého mesiaca negatívne ovplyvniť predpoveď. *Dôveryhodnosť* hovorí, nakoľko užívatelia daným dátam veria (napríklad ak dáta užívateľom spôsobili nepresné odhady, nemusia im veriť). Posledným faktorom, ktorý vplýva na celkové vnímanie kvality dát je *interpretovateľnosť*. Dáta môžu obsahovať mnoho rôznych kódov, ktoré nie je analytik schopný interpretovať a dáta tak napriek splneniu všetkých predošlých vlastností môže považovať za nekvalitné.

Proces predspracovania dát sa snaží odstrániť všetky spomenuté problémy a zvýšiť tak kvalitu dát. Predspracovanie je dôležitý krok v procese získavania znalostí, pretože kvalitné rozhodnutia musia byť založené na kvalitných dátach. V nasledujúcich kapitolách sú vysvetlené jednotlivé kroky procesu – čistenie, integrácia, redukcia a transformácia dát [6].

2.3.1 Čistenie dát

Krok čistenia dát má za ciel doplniť chýbajúce hodnoty, vyhladiť šum, identifikovať odľahlé hodnoty a opraviť nezrovnalosti. Vyriešením týchto problémov je možné výrazne zvýšiť kvalitu danej dátovej sady [6].

Chýbajúce hodnoty – Chýbajúce hodnoty je možné doplniť niekoľkými spôsobmi:

1. *Ignorovanie záznamu* – Ak v konkrétnom zázname chýba hodnota, ignoruje sa celý. Tento prístup nie je efektívny, pokiaľ percento chýbajúcich hodnôt u jednotlivých atribútov značne kolísa a vo výsledku by nám zostalo málo dát. Zvyčajne sa vykonáva pri klasifikácii, keď chýba označenie triedy.
2. *Ručné doplnenie hodnôt* – Všeobecne ide o časovo náročný a často nemožný proces pri obrovskom množstve dát.
3. *Doplnenie globálnou konštantou* – Jedná sa o doplnenie všetkých chýbajúcich hodnôt rovnakou konštantou, napr. „*neznáma*“ alebo $-\infty$. Problémom je, že dolovací algoritmus si môže myslieť, že doplnené hodnoty tvoria zaujímavý koncept, pretože zdieľajú spoločnú hodnotu.
4. *Doplnenie hodnotou charakterizujúcou stred* – Spočítame medián alebo modus hodnôt daného atribútu a výslednú hodnotu priradíme všetkým neznámym hodnotám toho istého atribútu. Hodnoty opäť môžu tvoriť mylný koncept.

5. *Doplnenie hodnotou charakterizujúcou stred hodnôt z rovnakej triedy* – Metóda je podobná predchádzajúcej s tým rozdielom, že medián alebo modus nepočítame zo všetkých hodnôt, ale iba z tých, ktoré patria do záznamu rovnakej triedy ako ten s chýbajúcou hodnotou.
6. *Doplnenie najpravdepodobnejšou hodnotou* – S využitím ostatných známych atribútov je možné zostrojiť rozhodovací strom (alebo použiť inú metódu) a predpovedať chýbajúcu hodnotu.

Chýbajúce hodnoty doplnené metódami 3 až 6 nemusia byť správne. Metóda 6 je však pomerne populárna kvôli predikcii chýbajúcej hodnoty zvážením ostatných atribútov. Chýbajúca hodnota nie vždy musí znamenať chybu v dátach. Často sa užívateľ rozhodol danú hodnotu nezadať, pretože tým chcel vyjadriť, že o niečo nemá záujem, niečo nemá, niečo nedostal alebo nespravil. Rozhodnúť však o význame prázdneho atribútu až vo fáze predspracovania, môže byť často nemožné.

Vyhladenie šumu – Šum je náhodná chyba alebo odchýlka v nameraných dátach. K jeho odstráneniu (vyhladeniu) je možné použiť niektorú z následujúcich techník:

1. *Binning* – Vyhladzovanie prebieha nad jednotlivými zoradenými hodnotami dát a berie do úvahy ich susedné hodnoty. Postupnosť je rozdelená do niekoľkých „košov“. Počet košov a počet hodnôt v košoch závisí na použitej metóde rozdelenia, ako bolo popísané v procese diskretizácie v sekcii 2.2. Následne je možné každú hodnotu v koši nahradiť mediánom alebo modusom hodnôt v danom koši. Inou možnosťou je takzvané vyhladenie pomocou hraníc koša, kde minimálna a maximálna hodnota koša tvoria hranice, pričom každá hodnota v koši je následne nahradená hodnotou najbližšej hranice.
2. *Regresia* – Technika ktorá prispôsobuje dátové hodnoty funkcii. Lineárna regresia zahŕňa nájdenie najlepších dvoch atribútov, kde hodnota jedného atribútu môže byť použitá na predpoveď druhého.

Detekcia odľahlých hodnôt – Odľahlé hodnoty sú hodnoty, ktoré sa značne odlišujú od ostatných hodnôt. Často ich označujeme ako abnormality, odchýlky alebo anomálie. Vzniknúť môžu chybou v generujúcom procese. Pokiaľ zameriavame analýzu práve na hľadanie nezvyčajného správania, odľahlé hodnoty poskytujú užitočné informácie. K detekcii sa používa hlavne zhlukovanie. Zhluky obsahujú hlavné dáta a malé množstvo dát, ktoré sú mimo zhlukov, predstavujú odľahlé hodnoty [2].

2.3.2 Integrácia dát

Aby získané znalosti boli čo najdôveryhodnejšie, je často potrebné spojiť dáta z rôznych dátových úložísk. Správnou integráciou je možné redukovať nadbytočné a nekonzistentné dáta vo výslednej dátovej sade. Integrácia prináša mnoho problémov, z ktorých hlavným je takzvaný problém identifikácie entity. Ani človek, ani počítač si nemôže byť bez akýchkoľvek ďalších informácií istý, či atribút `produkt_id` v jednej databáze znamená to isté ako `produkt_name` v druhej. Z toho dôvodu by mali existovať metadata, ktoré jasne definujú meno, význam, dátový typ, rozsah hodnôt a pravidlá pre prácu s prázdnymi (null) hodnotami pre každý atribút.

Dôležitý môže byť i problém redundancie. Nejaký atribút môže byť redundantný, pokiaľ sa dá odvodiť od iného atribútu alebo skupiny atribútov. Na detekciu redundancie sa vy-

užíva *korelačná analýza*, ktorá poskytuje rozdielne prístupy k nominálnym a numerickým atribútom [6].

Integrácia zahŕňa i detekciu a riešenie konfliktov dátových hodnôt. Dva zdroje totiž môžu obsahovať významovo tie isté dáta, ale hodnoty sú v iných jednotkách, napríklad cena v rôznej mene.

2.3.3 Redukcia dát

Aplikácia dolovacích algoritmov na dátové sady obrovských rozmerov môže trvať priveľmi dlhú dobu. Techniky redukcie dát majú za cieľ získať redukovanú reprezentáciu dátovej sady, ktorá má menší objem, ale jej informačná hodnota je značne podobná originálnej sade. Medzi hlavné stratégie patria: redukcia dimenzionality, redukcia početnosti a dátová kompresia.

Redukcia dimenzionality je proces, ktorý zvažuje a redukuje množstvo náhodných premenných alebo atribútov. Používajú sa metódy DWT (angl. discrete wavelet transform) alebo PCA (angl. principal components analysis), ktoré transformujú originálne dáta do menšieho priestoru. Metóda výberu podmnožiny atribútov detekuje a odstraňuje irelevantné, málo relevantné alebo nadbytočné atribúty.

V tejto práci budú využité sady malých až stredne veľkých rozmerov s časovo nie veľmi náročnou analýzou. Sady boli vybrané tak, aby nebola potrebná redukcia niektorou zo spomenutých metód, preto tejto problematike nie je venovaná väčšia pozornosť [6].

2.3.4 Transformácia dát

Dáta sú transformované do podoby, ktorá prispieva k účinnosti dolovacích metód a k lepšej pochopiteľnosti nájdených vzorov. Pozostávajú z niekoľkých druhov transformácie, pričom niektoré boli popísané ako súčasť iných častí procesu predspracovania. Stratégie transformácie zahŕňajú vyhladzovanie dát, konštrukciu atribútov, agregáciu, normalizáciu, diskretizáciu a generovanie konceptuálnej hierarchie pre nominálne dáta.

Významným krokom, ktorý doteraz nebol spomenutý, je normalizácia (alebo tiež štandardizácia). Dáta jednotlivých atribútov sú transformované tak, aby spadali do menších intervalov hodnôt – najčastejšie -1.0 až 1.0 alebo 0.0 až 1.0. Cieľom je priradiť atribútom rovnakú váhu, čo je užitočné pre klasifikačné algoritmy ako neurónové siete, klasifikačné metódy založené na meraní vzdialenosti alebo zhlukovanie. Medzi najznámejšie metódy normalizácie patrí min-max normalizácia, normalizácia pomocou z-skóre a normalizácia desatinnou stupnicou, ktoré sú vysvetlené v nasledujúcich odrážkach [6] [23].

- **Min-max normalizácia** – Lineárna transformácia pôvodných dát. Nech min_A a max_A sú minimálna a maximálna hodnota atribútu A. Min-max normalizácia mapuje hodnoty v_i atribútu A na nové hodnoty v'_i v rozsahu $[new\_min_A, new\_max_A]$ nasledujúcim výpočtom (rovnica 2.1):

$$v'_i = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A. \tag{2.1}$$

  Nové hodnoty zachovávajú vzťahy medzi pôvodnými hodnotami. Keďže rozsahy sú dané minimálnou a maximálnou hodnotou v pôvodných dátach, akákoľvek nová hodnota pridaná do normalizovaných dát mimo týchto hraníc spôsobí chybu normalizácie.

- **Normalizácia pomocou z-skóre** – Hodnoty atribútu A sú normalizované na základe priemeru a smeru odchýlky atribútu A. Hodnota v_i atribútu A je normalizovaná

na novú hodnotu v_i' rovnicou (rovnica 2.2):

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}, \tag{2.2}$$

kde σ_A je smerodatná odchylka atribútu A a $\bar{A}$ je priemer.

- **Normalizácia desatinnou stupnicou** – Normalizácia spočíva v posune desatinnej čiarky hodnôt atribútov. Počet desatinných miest závisí od najväčšej absolútnej hodnoty atribútu A. Výpočet novej normalizovanej hodnoty v_i' z pôvodnej v_i atribútu A výpočtom (rovnica 2.3):

$$v_i' = \frac{v_i}{10^j}, \tag{2.3}$$

kde j je najmenšie celé číslo, pre ktoré platí $max(|v_i'|) < 1$.

Kapitola 3

Klasifikácia

Kapitola nadväzuje na problematiku dolovania znalostí, upúšťa od všeobecného pojednávania o základných pojmoch z oblasti spracovania dát a predstavuje konkrétnu formu ich analýzy – klasifikáciu. Prvá časť má za cieľ oboznámiť čitateľa o základných pojmoch, princípe klasifikácie a predstavené sú aj aplikačné oblasti. Pozornosť je venovaná spôsobom validácie klasifikačných metód. Následne sú jednotlivé metódy a prístupy systematicky prezentované. Každá sekcia je venovaná samostatnej metóde, podrobne popisuje myšlienku, princíp a prípadne výhody či nevýhody. Kapitola poskytuje teoretický podklad pre pochopenie značnej časti aplikácie, ktorá je základom pre získané výsledky tejto diplomovej práce.

3.1 Klasifikácia obecne

Klasifikácia je forma analýzy dát, ktorá odvodzuje modely popisujúce dôležité dátové triedy. Tieto modely sú nazývané klasifikátory. Používajú sa k predpovedi (odhadu) identifikátorov skupiny – triednych označení (angl. *class labels*, ďalej len tried) pre nové dátové vzorky s neznámou triedou. Dáta, ktorých triedu poznáme, sú správne označované ako trénovacia dátová sada, naopak dáta, ktorých triedu nepoznáme, sú testovacia dátová sada. Príkladom využitia môže byť výskum v medicíne, ktorý chce na základe analýzy dát rakoviny prsníka predpovedať jednu z troch špecifických liečebných procesov, ktoré by mal pacient podstúpiť [2].

Samotná klasifikácia je pomerne jednoduchý proces. Skladá sa z dvoch hlavných fáz, ktoré sú prítomné u všetkých klasifikačných algoritmov, ktoré budú v tejto práci rozoberané [6]:

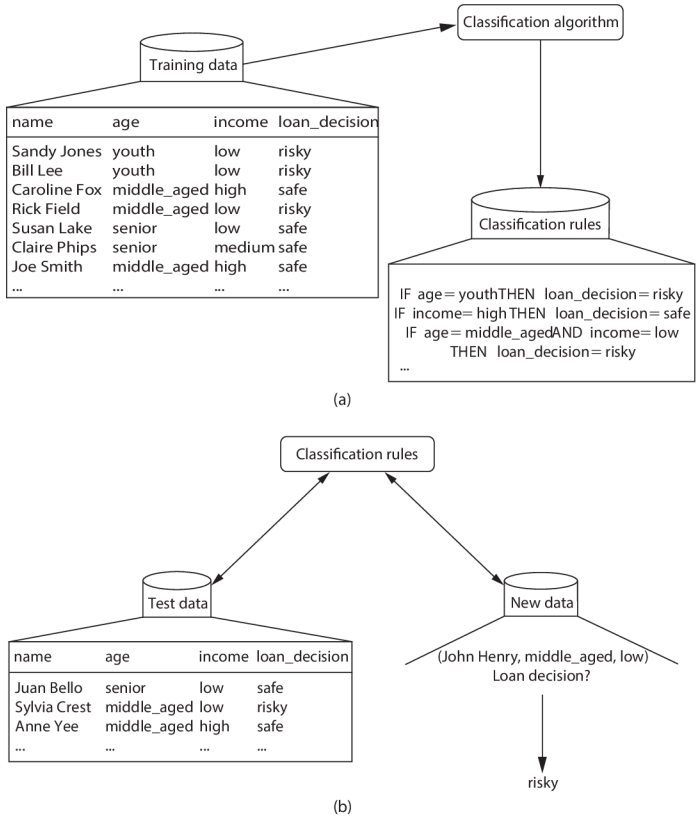
1. **Fáza učenia** – Aby bolo samotné klasifikovanie možné, je nutné vytvoriť klasifikačný model. V tejto fáze teda dochádza k vytvoreniu klasifikátorov analýzou trénovacích dát. Trénovacie dáta obsahujú vzorky, u ktorých poznáme triedu. Formálne je možné trénovacie dáta definovať nasledovne. Nech n-tica X je reprezentovaná n-rozmerným vektorom atribútov $X = (x_1, x_2, ..., x_n)$, zobrazujúcim n meraní vykonaných na n-tici z n databázových atribútov $A_1, A_2, ...A_n$. Každá n-tica X patrí to preddefinovanej triedy určenej ďalším, triednym atribútom (triedne označenie).

   Tento proces tiež môžeme vidieť ako proces učenia mapovacej funkcie $y = f(X)$, ktorá predpovedá priradenie triedy y k danej n-tici X. N-tice sa tiež často nazývajú vzorky, objekty či dátové body.

2. **Fáza klasifikovania** – V tejto fáze je klasifikačný model použitý na určenie triedy neznámych vzoriek dát, teda na klasifikáciu. Výstup klasifikačného algoritmu môže mať jednu z nasledujúcich foriem:

   - *Diskrétna značka* – Každá vzorka má pridelenú jednu triedu.
   - *Numerické skóre* – Pre každú triedu a testovaciu kombináciu je vypočítané numerické skóre. Skóre môže byť pre danú testovaciu kombináciu konvertované na diskrétnu značku výberom triedy s najvyšším skóre pre túto kombináciu. Výhodou je možnosť porovnať náchylnosť príslušnosti testovacej vzorky do istej triedy dôležitosti. Numerické skóre sa často využíva na detekciu vzácnych tried, kde pôvodná distribúcia tried je značne nevyvážená a nájdenie konkrétnych tried je dôležitejšie než ostatných.

Na obrázku 3.1 (*a, b*) je možné vidieť znázornené obe fázy klasifikácie. Trénovacie dáta sú použité na vytvorenie klasifikačných pravidiel klasifikátora, čím je možné ho považovať za naučený (a). Pravidlá sú následne použité k odhadu triedy klasifikovaného dátového objektu (b). Každý klasifikátor používa rozdielne postupy trénovania i klasifikovania. Pred nasadením klasifikátoru na reálne dáta je dôležité ho otestovať a vyhodnotiť – validovať. Jednotlivé spôsoby validácie sú popísané v kapitole 3.2.



Obr. 3.1: Názorná ukážka trénovania (a) a testovania s klasifikáciou (b) [6]

Pokiaľ v prvej fáze algoritmus využíva trénovacie dáta so známou triedou jednotlivých vzoriek, hovoríme o *učení s učiteľom*. Naopak, v *učení bez učiteľa* nie sú dopredu známe jednotlivé triedy dátových objektov a často nemusí byť známy ani ich počet. Reprezentantom učenia bez učiteľa sú zhlukovacie metódy, ktoré nebudú v tejto práci zahrnuté.

3.1.1 Aplikačné oblasti

Existuje mnoho oblastí a mnoho rôznorodých dát, na ktoré je možné uplatniť klasifikačné metódy a pozorovať zaujímavé výsledky. Niektoré z nich bude možné vidieť v kapitole 4 priamo na vybraných dátových sadách. Medzi iné aplikačné oblasti môže patriť [1]:

- **Marketing cielený na zákazníka** – Klasifikácia je v tejto oblasti veľmi populárna. Využíva zozbierané informácie o zákazníkovi a na základe nich predpovedá, o ktoré z produktov by mohol mať záujem.
- **Diagnóza ochorení v medicíne** – Jednotlivé vzorky sú získané z lekárskych záznamov a využité na predpoveď, či pacient trpí nejakou chorobou, alebo či môže ochorieť.
- **Analýza biologických dát** – Biologické dáta sú často reprezentované ako diskrétna postupnosť nejakých hodnôt (napríklad DNA sekvencia), v ktorých je žiadaná predikcia vlastností konkrétnej sekvencie.
- **Kontrolovaná detekcia udalostí** – Jednotlivé triedy môžu byť priradené k časovým razítkam nezvyčajných udalostí, ako napríklad neoprávnené vniknutie. Klasifikácia časových údajov môže byť následne veľmi užitočná pri odhaľovaní rizikových momentov.
- **Analýza multimediálnych dát** – Klasifikácia multimediálnych dát je často výzvou, kvôli množstvu asociácií, ktoré k nim môžu existovať. Klasifikáciou obrázkov, zvukov, videí a iných dát je možné automaticky rozoznať osoby, informácie o nich, miesta, rozlúštiť poškodené texty či identifikovať predmety.
- **Kategorizácia dokumentov a filtrovanie** – Služby poskytujúce aktuálne informácie v elektronickej podobe využívajú klasifikáciu obrovského množstva dokumentov v reálnom čase. Dokumenty sa triedia a filtrujú, čo je dôležité napríklad pre výskumné oblasti.
- **Analýza sociálnych sietí** – Sociálne siete sú obrovským zdrojom dát. Klasifikácia sa dá využiť v mnohých smeroch – predpoveď charakteristík konkrétneho užívateľa, bezpečnostné analýzy, hľadanie nákupných trendov či klasifikovanie aktuálnych problémov.

3.2 Hodnotenie klasifikácie

Dôležitým problémom v oblasti klasifikácie je vyhodnocovanie klasifikačných algoritmov. Zaujíma nás hlavne presnosť klasifikátora, prípadne časová náročnosť učenia či klasifikácie. Na základe výsledkov je možné porovnať dva rozdielne klasifikačné modely a vybrať najvhodnejší pre dané dáta. Problémom pri validácii môže byť voľba vhodnej metodológie a spôsobu reprezentácie výsledkov. Tieto problémy sú definované nasledovne [2]:

1. **Metodologický problém** – Metodologické problémy sú spojené s rozdelením dát so známou triedou na trénovaciu dátovú sadu a na testovacie vzorky určené na validáciu klasifikátora. Výber vhodnej metodológie má priamy vplyv na vyhodnocovací proces a môže znamenať nadcenenie či podcenenie presnosti klasifikátora. Medzi známe prístupy patrí *holdout*, *krížová validácia* alebo *bootstrap*, vysvetlené v sekcii 3.2.1. Zmyslom validácie klasifikačných metód na dostupnej sade dát je zistenie výkonnosti klasifikačného modelu pred jeho použitím na neznámych dátach v reálnom nasadení.

2. **Kvantifikačný problém** – Problém je spojený s poskytovaním výsledkov numerických meraní kvality metód po aplikovaní špecifickej metodológie. Merania môžu zahrňovať rôzne varianty presnosti alebo krivku charakterizujúcu kompromis medzi skutočne pozitívnymi (angl. true positive) a nesprávne pozitívnymi (angl. false positive) výsledkami klasifikácie.

Vyhodnocovací proces vyžaduje dáta so známymi triedami, ktoré reprezentujú *skutočnosť* a neboli použité v procese trénovania. Odlišnosť testovacích a trénovacích dát je veľmi dôležitá, inak dochádza k preceneniu výslednej presnosti metód, pretože klasifikátor bol na dané dáta naučený, a teda vykazuje 100 % úspešnosť pri testovaní. Nedozvieme sa tak, že pri učení mohlo dôjsť k naučeniu presných charakteristík trénovacích dát a metóda nemusí byť aplikovateľná na všeobecné dáta danej oblasti. Tento stav nazývame tiež *preučenie* (angl. overfitting) [2].

Okrem presnosti je dobré zvážiť pri vyhodnocovaní i iné veličiny. Časová zložitosť jednotlivých algoritmov môže byť vždy vypočítaná teoreticky, ale je dobré mať naviac výsledky praktického merania. Dôležité je poznať dve hodnoty. Prvou je čas potrebný k zostaveniu modelu na trénovacích dátach a druhou je čas potrebný na klasifikáciu jednotlivých vzoriek pomocou tohto modelu.

Ďalšou dôležitou veličinou môže byť pamäťová náročnosť modelu. Po naučení modelu na vstupných dátach môže byť model uložený a počiatočné dáta nie sú potrebné. U niektorých klasifikátorov postačuje uloženie výsledného modelu po jeho naučení (napr. rozhodovací strom), u iných (napr. K-najbližších susedov) je ale nutné uložiť celú dátovú sadu [1].

3.2.1 Metódy rozdelenia dát

Pre zaručenie rozdielnosti trénovacej a testovacej sady je potrebné z trénovacej sady dát odstrániť niektoré vzory a tie použiť na testovanie. Problémom je, že odstránením dát, ktoré mohli byť použité na trénovanie, znižujeme silu klasifikátora, ktorý je na týchto hodnotách založený [1]. Existuje preto niekoľko metód rozdelenia dát tak, aby došlo k čo najmenšiemu ovplyvneniu nameranej presnosti klasifikátora [2].

Holdout – Metóda rozdelenia dát založená na náhodnom rozdelení do dvoch disjunktných podmnožín – trénovacej a testovacej. Pomer oboch podmnožín je väčšinou volený tak, aby trénovacie dáta predstavovali aspoň dve tretiny alebo tri štvrtiny pôvodného množstva dát a zvyšok boli dáta testovacie. Problémom tohto prístupu je, že triedy, ktoré sú v trénovacích dátach zastúpené väčšinovo, môžu byť zároveň menej zastúpené v testovacích dátach. Náhodný výber má preto výrazný vplyv na výsledky vyhodnotenia.

Veľký problém nastane, pokiaľ máme sadu 1000 vzoriek, z toho 990 patrí do jednej triedy a 10 do druhej. V tomto prípade je vysoko pravdepodobné, že testovacia sada s 200 prvkami nebude obsahovať ani jednu vzorku s priradením k vzácnej triede. V tomto prípade nebude vyhodnotenie presnosti dávať zmysel. Riešením je implementovať metódu holdout tak, že sa náhodne vyberajú vzorky s ohľadom na zastúpenie jednotlivých tried v pôvodných dátach.

Krížová verifikácia – Metodológia rozdelenia dát na m disjunktných podmnožín rovnakej veľkosti n/m. Najčastejšia hodnota parametru m je približne 10. Jeden z m segmentov použitý na testovanie a zvyšných $m - 1$ na trénovanie. Následne je tento postup opakovaný, dokým nie je všetkých m segmentov použitých pre testovanie. Veľkosť

trénovacej sady je potom $(m-1)n/m$. Pokiaľ sa veľkosť m blíži počtu vzoriek n, chybovosť metódy je veľmi blízka tej, ktorú by sme dosiahli s použitím všetkých pôvodných dát na trénovanie. Celková presnosť krížovej verifikácie je často veľmi reprezentatívna. Špeciálnym prípadom je takzvaná *leave-one-out* krížová verifikácia, kedy $m = n$ a teda na trénovanie je použitých $n-1$ vzoriek. Tento prípad je pomerne náročný pri rozsiahlych dátových sadách, avšak nemusí tomu tak byť pri všetkých klasifikačných metódach. Existuje i *stratifikovaná krížová validácia*, ktorá využíva proporčné zastúpenie jednotlivých tried vo výsledných sadách.

Bootstrap – Bootstrap predstavuje najkomplikovanejšiu z troch predstavených metód. Vzorky pre trénovaciu sadu sú vyberané náhodne, ale s navrátením. To znamená, že je možné opakovane vybrať rovnakú vzorku. Počet opakovaní výberu sa rovná počtu vzoriek v pôvodnej sade. Trénovacia sada má veľkosť pôvodnej sady, ale takmer určite obsahuje duplikáty, a niektoré vzorky jej naopak chýbajú.

Pravdepodobnosť, že vzorka nie je obsiahnutá v jednom výbere, je $1-1/n$ a $(1-1/n)^n$ pre n výberov. Pre vysoké hodnoty n sa výraz aproximuje $1/e$, kde e je základ prirodzeného logaritmu. Z toho vyplýva, že pri n výberoch je vybraných približne 63.2 % unikátnych vzoriek do trénovacej sady. Celková presnosť je vypočítaná použitím pôvodnej sady dát ako testovacej sady, avšak jedná sa o veľmi pozitívne vyhodnotenie, kedže niekoľko prvkov z testovacej množiny sa mohlo podieľať i na trénovaní [2].

Lepšou alternatívou je váhovaná kombinácia presnosti A_1 získanej použitím testovacej sady obsahujúcej prvky, ktoré neboli súčasťou trénovacej sady a presnosti A_2 testovaním na celej sade pôvodných dát. Výsledná presnosť je (rovnica 3.1) [1]:

$$A = (0.368) \cdot A_1 + (0.632) \cdot A_2 \tag{3.1}$$

Procedúra je opakovaná niekoľkokrát nad rôznymi bootstrap vzorkami a konečná presnosť je udaná ako priemer všetkých postupných presností.

3.2.2 Reprezentácia výsledkov

Z výsledkov numerických meraní kvality metód nás najčastejšie zaujíma výsledná presnosť modelu, teda porovnanie predikovaných tried so skutočnými triedami daných vzoriek. Existuje niekoľko spôsobov prezentácie meraní [2]:

1. **Presnosť** – Všeobecná presnosť je daná ako pomer správne klasifikovaných vzoriek k počtu všetkých vzoriek. Najčastejšie sa udáva v percentách.

2. **Váhovaná presnosť** – Nie vo všetkých dátach sú triedy vzájomne rovnako dôležité. Ako príklad môžu poslúžiť medicínske dáta, ktoré klasifikujú pacientov do dvoch tried – „zdravý“ a „chorý“. V tomto prípade môže byť nesprávna klasifikácia chorého pacienta menej tolerovaná ako nesprávna klasifikácia zdravého pacienta. Tento fakt je možné zahrnúť do výslednej presnosti zavedením váh (cien) $c_1, ..., c_k$ pri nesprávnej klasifikácií jednotlivých k tried. Nech $n_1, ..., n_k$ sú počty testovacích vzoriek patriacich do každej triedy a $a_1, ..., a_k$ sú všeobecné presnosti vyjadrené zlomkom pre jednotlivé triedy. Potom celková váhovaná presnosť je vypočítaná ako (rovnica 3.2):

$$A = \frac{\sum_{i=1}^{k} c_i n_i a_i}{\sum_{i=1}^{k} c_i n_i} \tag{3.2}$$

Ak sú ceny rovnaké, dostávame rovnakú presnosť ako v prípade neváhovanej presnosti.

3. **Matica zámen** – V niektorých prípadoch je pri testovaní dobré poznať, v koľkých prípadoch klasifikátor určil triedu správne a v koľkých naopak nesprávne, a pritom zobraziť informáciu o tom, akej chyby sa dopustil. K tomu môže slúžiť takzvaná matica zámen. Jeden rozmer tvoria skutočné triedy vzoriek a druhý predikované triedy. Bunky matice obsahujú počet výskytov danej kombinácie. Názorná matica pre binárnu klasifikáciu je znázornená tabuľkou 3.1.

		Predikované triedy	
		Trieda A	**Trieda B**
Skutočné triedy	**Trieda A**	Správne klasifikované vzorky triedy A	Vzorky triedy A nesprávne klasifikované do triedy B
	Trieda B	Vzorky triedy B nesprávne klasifikované do triedy A	Správne klasifikované vzorky triedy B

Tabuľka 3.1: Príklad matice zámen

3.3 Jednoduchá Bayesovská klasifikácia

Jednoduchá Bayesovská klasifikácia, nazývaná tiež naivná Bayesovská klasfikácia (angl. Naive Bayes), je založená na Bayesovom teoréme a vychádza z predpokladu vzájomnej nezávislosti atribútov. Ide o jednoduchú metódu, ktorá v rade prípadov poskytuje dobré výsledky s vysokou presnosťou. Výhodou je aj rýchlosť pri aplikácii na objemné dátové sady.

Bayesov teorém je definovaný vzťahom (rovnica 3.3):

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}, \tag{3.3}$$

kde $p(Y)$ a $p(X)$ sú pravdepodobnosti javu Y, X a $p(Y|X)$, $p(X|Y)$ sú podmienené pravdepodobnosti. $p(X)$ pritom môžeme väčšinou považovať za konštantu [1].

Podstatu fungovania jednoduchej Bayesovskej klasifikácie je možné popísať nasledovne [6]:

1. Nech D je množina trénovacích vzorov. Nech $X = (x_1, x_2, ..., x_n)$ je vzorka dát s n atribútmi $A_1, A_2, ..., A_n$.

2. Uvažujme m tried $C_1, C_2, ..., C_m$. Bayesovský klasifikátor zaradí vzorku X do jednej z týchto tried len vtedy, ak je jej $p(C_i|X)$ maximálna (rovnica 3.4):

$$p(C_i|X) = \frac{p(X|C_i)P(C_i)}{p(X)}. \tag{3.4}$$

3. Pretože $p(X)$ je konštanta pre všetky triedy, hľadáme len maximálne $p(X|C_i)P(C_i)$. Pravdepodobnosti jednotlivých tried je možné vypočítať ako (rovnica 3.5):

$$p(C_i) = \frac{|C_{i,D}|}{|D|}, \tag{3.5}$$

kde $|C_{i,D}|$ je počet vzoriek v trénovacej sade D priradených do triedy C_i.

4. Za predpokladu, že hodnoty jednotlivých atribútov sú vzájomne nezávislé, $p(X|C_i)$ môžeme vypočítať nasledovne (rovnica 3.6):

$$p(X|C_i) = \prod_{k=1}^{n} p(x_k|C_i). \tag{3.6}$$

Nech x_k je hodnota atribútu A_k vzorku X. Výpočet $p(x_k|C_i)$ sa bude líšiť pre kategorické a numerické hodnoty.

(a) Ak A_k je kategorický atribút, potom (rovnica 3.7):

$$p(X_k|C_i) = \frac{|C_{i,D,k}|}{|C_{i,D}|}, \tag{3.7}$$

kde $|C_{i,D,k}|$ je počet vzoriek triedy C_i v D, ktorých hodnota atribútu A_k je x_k.

(b) Ak A_k je numerický atribút, je výpočet o niečo komplikovanejší. Predpokladá sa, že hodnoty majú Gaussovo rozloženie s priemerom μ a smerodatnou odchýlkou σ, definované rovnicou (rovnica 3.8):

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\Pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.8}$$

z čoho plynie (rovnica 3.9):

$$p(X_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \tag{3.9}$$

Problém nastáva, pokiaľ pre niektoré k platí, že $p(x_k|C_i) = 0$, a teda i $p(X|C_i) = 0$. Riešením je takzvaná *Laplaceova korekcia*, ktorá pridá ku každej množine jeden prvok.

5. K úspešnému určeniu triedy vzorku X, je $p(X|C_i)P(C_i)$ vyhodnotené pre každú triedu. Výsledná trieda je potom trieda C_i, pre ktorú bolo $p(X|C_i)P(C_i)$ maximálne.

Výhodou naivného Bayesovského klasifikátora je jednoduchosť, zrozumiteľnosť a rýchlosť. Podľa niektorých štúdií je jeho účinnosť porovnateľná s rozhodovacími stromami alebo niektorými neurónovými sieťami. Využitie nachádza často v oblastiach klasifikácie textu, filtrovania spamu alebo štatistike. Výhodou je tiež to, že ku každej triede je určený jej pravdepodobnostný popis. Hlavnou nevýhodou je naivné predpokladanie nezávislosti atribútov [4].

3.4 Rozhodovacie stromy

Rozhodovacie stromy patria medzi najintuitívnejšie nástroje pre klasifikáciu. Rozdeľujú priestor vstupných hodnôt, kým nenájdu taký podpriestor, ktorý je asociovaný s konkrétnou triedou. Ich aplikácia je rozšírená v medicíne, biznise i vede, pretože poskytujú intuitívny pohľad na riešenie zložitých rozhodovacích úloh.

Rozhodovací strom je vo svojej podstate graf stromovej štruktúry. Každý vnútorný uzol odpovedá čiastočnému rozhodnutiu (test hodnoty atribútu) a každý listový uzol predstavuje triedu. Klasifikáciu si je možné predstaviť tak, že každá vzorka od koreňa prechádza stromom. Na každom vnútornom uzle je rozdeľovacím pravidlom rozhodnuté, k akému ďalšiemu uzlu bude vzorka poslaná. Po dosiahnutí listového uzlu je výsledná trieda vzorky určená uzlom.

Každá hrana stromu môže byť preložená ako Booleovský výraz (napr. $X > 5$). Strom je tak možné previesť na množinu pravidiel, kde každá cesta od koreňa k listu generuje pravidlá konjunkciou (logický AND) všetkých čiastočných rozhodnutí (testov) vnútorných uzlov.

Pravidlá môžu byť jednorozmerné (každé rozdelenie zvažuje len jeden atribút) alebo viacrozmerné (uvažuje sa viacero atribútov). Väčšina stromov je binárnych, kde každé rozdelenie rozdeľuje priestor dát práve na dva podpriestory. Rozdelenia na viacero podpriestorov sa vyskytujú tiež, no klesá tým stabilita stromu a zvyšuje sa jeho zložitosť [1].

Proces učenia sa štruktúry rozhodovacieho stromu, a teda konštrukcia klasifikátora sa nazýva indukcia rozhodovacieho stromu. Algoritmus je možné skoro celý napísať ako jednu rekurzívnu funkciu. Vstupom sú dáta z trénovacej množiny, ktorých atribúty sú prevedené na diskrétne hodnoty, označme ich S, a zoznam atribútov L. Výstupom je rozhodovací strom. Algoritmus pre vytvorenie rozhodovacieho stromu je nasledovný (algoritmus 1):

Algoritmus 1: Rekurzívny algoritmus tvorby rozhodovacieho stromu [23]

```
1  Function CreateTree(S, L):
2      Vytvor nový uzol N;
3      if vzorky S sú v rovnakej triede C then
4          return N ako list danej triedy C
5      if zoznam atribútov L je prázdny then
6          return N ako list najbežnejšej triedy v množine vzoriek S
7      Vyber atribút A zo zoznamu L s „najvyššou rozhodovacou schopnosťou“ a
        odstráň ho z tohto zoznamu;
8      Pomenuj uzol N menom vybraného atribútu A;
9      for každá možná hodnota a_i atribútu A do
10         Vytvor vetvu z uzlu N pre podmienku „A == a_i“;
11         Nech s_i je podmnožina vzoriek z S, u ktorých „A == a_i“;
12         if s_i je prázdna then
13             pripoj k vetve list s najbežnejšou triedou v množine S
14         else
15             pripoj k vetve podstrom vzniknutý rekurzívnym volaním
                CreateTree(s_i,L)
16         end
17     end
```

Kľúčovou časťou algoritmu je výber atribútu s najvyššou rozhodovacou schopnosťou, čo je atribút, ktorý sa bude testovať v rámci uzlu, a prispeje k najlepšej výslednej presnosti. Hovoríme tiež o rozdeľovacom pravidle, pretože určuje, na základe čoho budú vzorky rozdelené na danom uzle. Pokiaľ sú hodnoty vybraného atribútu spojité alebo sme obmedzení len na použitie binárneho stromu, je potrebné určiť rozdeľovaciu podmnožinu (angl. *splitting subset*) alebo rozdeľovací bod (angl. *split point*). Rôzne varianty rozhodovacích stromov (ID3, CART, C4.5) používajú rôzne metriky pre výber atribútu (informačný zisk, Gini index) [6].

Kvantitatívne meranie vhodnosti atribútu udáva štatistická vlastnosť informačný zisk, ktorú využíva napríklad metóda ID3. Aby mohol byť stanovený informačný zisk, je potrebné zadefinovať entropiu. Ak cieľový atribút má c rôznych hodnôt, potom entropia S, ktorá

zodpovedá klasifikácii s c triedami, je definovaná ako (rovnica 3.10):

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i, \tag{3.10}$$

kde p_i je podiel objektov z trénovacej množiny S spadajúci do triedy i. Informačný zisk je vlastne očakávané zmenšenie entropie zapríčinené rozdelením vzoriek týkajúcich sa daného atribútu. Informačný zisk $Gain(S, A)$ v rámci rozdelenia pomocou atribútu A je definovaný ako (rovnica 3.11):

$$Gain(S, A) = Entropy(S) - \sum_{h=hodnoty(A)} \frac{|S_v|}{|S|} Entropy(S_v), \tag{3.11}$$

kde $hodnoty(A)$ je množina všetkých hodnôt atribútu A a S_v je podmnožina S, pre ktorú atribút A má hodnotu v. Pre rozdelenie je následne možné vybrať atribút A s najväčšou hodnotou $Gain(S, A)$ [9].

Pri vytváraní rozhodovacieho stromu môžu vznikať vplyvom šumu vetvy, ktoré vedú k zníženiu presnosti predpovedi a väčšej zložitosti. Tieto vetvy je potrebné odstrániť, k čomu slúžia dve základné metódy orezania stromu [23]:

- **Prepruning** – Vetvy sa negenerujú už pri samotnom vytváraní stromu. Namiesto testu atribútu na ďalšiu podmienku sa táto časť priamo ukončí listom. List je ohodnotený triedou, ktorá je priradená najväčšiemu počtu vzoriek danej časti stromu. Táto metóda sa vyznačuje jej nízkou výpočtovou náročnosťou.
- **Postpruning** – „Nepotrebné“ vetvy sú odstránené až po vytvorení celého stromu. Metóda je spoľahlivejšia, ale výpočtovo náročnejšia.

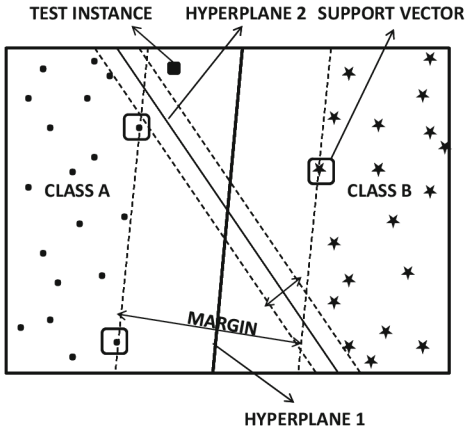
Orezanie je vhodné vykonávať i ako prevenciu proti preučeniu klasifikátoru.

Výhodou rozhodovacích stromov je, že dokážu generovať zrozumiteľné pravidlá, klasifikácie dosahujú bez zložitejších výpočtov a dokážu pracovať s kontinuálnymi i kategorickými hodnotami. Nevýhodou je chybovosť pri klasifikácii do mnohých tried s malým počtom trénovacích vzorov a v niektorých prípadoch vyššia časová náročnosť trénovania [9].

3.5 SVM

SVM je skratka pre klasifikačnú metódu s názvom Support Vector Machines. SVM používa separačnú hyperrovinu ako rozhodovaciu hranicu medzi dvoma triedami. Optimalizácia SVM spočíva v určení maximálneho okraja (angl. *margin*) hyperroviny. Okrajom je myslená oblasť v okolí hyperplochy, ktorá jasne oddeľuje dve rozdielne triedy a neobsahuje žiadne trénovacie prvky – *hard margin*. Pre lepšiu predstavu je rozdelenie zobrazené na obrázku 3.2. Viditeľné sú dve hyperroviny, jedna s úzkym okrajom a druhý so širokým. Predpokladáme, že testovacia vzorka má byť zaradená do triedy A. Hyperrovina 1 tak zaradila vzorku správne do triedy A, no hyperrovina 2 nie. Rozdelenie dát na dve triedy je jednoduché, pokiaľ sú dáta lineárne separovateľné (v reálnych dátach tomu tak väčšinou nie je).

Majme dátovú sadu D s d vzorkami, označené ako $(\bar{X}_1, y_1)...(\bar{X}_n, y_n)$, kde $\bar{X}_i$ je n-dimenzionálny vektor predstavujúci jeden dátový objekt a $y_i \in \{-1, +1\}$ je binárna trieda,



Obr. 3.2: Separácia vzoriek do dvoch tried pomocou hyperrovín [2]

do ktorej je vzorka zaradená. Oddeľujúca hyperrovina je potom definovaná ako (rovnica 3.12):

$$\bar{W} \cdot \bar{X} + b = 0. \tag{3.12}$$

$\bar{W} = (w_1, ..., w_n)$ je n-dimenzionálny vektor, ktorý reguluje orientáciu hyperroviny a b (bias) je skalár, ktorý reguluje vzdialenosť hyperroviny od jej východzej polohy. Za predpokladu, že dáta sú lineárne, všetky vzorky $\bar{X}_i$ s $y_i = +1$ budú ležať na jednej strane hyperroviny a všetky vzorky s triedou $y_i = -1$ na opačnej strane hyperroviny. Platí teda (rovnice 3.13, 3.14):

$$\bar{W} \cdot \bar{X}_i + b \geq 0 \quad \forall i : y_i = +1 \tag{3.13}$$

$$\bar{W} \cdot \bar{X}_i + b \leq 0 \quad \forall i : y_i = -1. \tag{3.14}$$

Veľmi podobne potom môžeme určiť i dve roviny vymedzujúce okraj, ktoré sa budú dotýkať najbližších vzoriek k hyperrovine (takzvaných *support vectors*) (rovnice 3.15, 3.15):

$$\bar{W} \cdot \bar{X}_i + b \geq +1 \quad \forall i : y_i = +1 \tag{3.15}$$

$$\bar{W} \cdot \bar{X}_i + b \leq -1 \quad \forall i : y_i = -1. \tag{3.16}$$

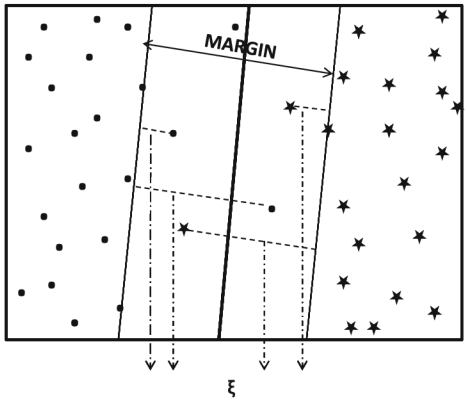
Cieľom metódy SVM je teda maximalizovať vzdialenosť medzi týmito dvoma rovinami (šírku okraja), čo je pomerne komplexný problém. Výstupom SVM klasifikátora sú potom nájdené váhy w_i, ktoré sú schopné klasifikovať danú vzorku dát do niektorej z tried.

Doteraz sme uvažovali len priestor lineárne separovateľných dát. SVM je ale možné použiť i na nelineárne dáta. V tomto prípade sa spomínaný okraj stáva mäkkým okrajom (*soft margin*), pretože povoľujeme niektorým vzorkám z trénovacích dát porušiť obmedzenia okraja za cenu penalizácie. Obe hranice okraja môžu byť potom vyjadrené ako (rovnice 3.17, 3.18):

$$\bar{W} \cdot \bar{X}_i + b \geq +1 - \xi_i \quad \forall i : y_i = +1 \tag{3.17}$$

$$\bar{W} \cdot \bar{X}_i + b \leq -1 + \xi_i \quad \forall i : y_i = -1, \tag{3.18}$$

kde premenné $\xi_i \geq 0 \quad \forall_i$ predstavujú vzdialenosti trénovacích dát od hyperrovín, ako je možné vidieť na obrázku 3.3. Hodnoty vzdialenosti sú 0, pokiaľ dáta ležia na správnej strane oddeľujúcich hyperrovín. Samozrejme je žiaduce, aby hodnoty $\xi_i > 0$ malo čo najmenej trénovacích vzoriek. Preto sú tieto porušenia penalizované ako $C\xi_i^r$, kde C a r sú užívateľom definované parametre regulujúce striktnosť modelu. Malé hodnoty C predstavujú väčšie



Obr. 3.3: Separácia vzoriek do dvoch tried s mäkkým okrajom (upravené z [2])

okraje, zatiaľ čo veľké hodnoty C minimalizujú chybu trénovacích dát, čím dochádza k zúženiu okraja. Nastavením vhodne vysokého C zamedzíme výskytu chyby na trénovacích dátach, avšak môže dôjsť k zhoršeniu generalizácie problému. Parameter r sa často volí 1 [2].

Na mnohé nelineárne dáta by nebolo možné aplikovať ani „zmäkčovanie“ okrajov a tak je často nelineárny priestor rôznymi spôsobmi prevádzaný na lineárny, napríklad za cenu zvýšenia počtu atribútov. Väčšinou však ide o komplikovaný problém [23].

3.6 Neurónové siete

Neurónová sieť bola pred pár rokmi znovuobjavená ako dôležitá alternatíva k rozličným štandardným klasifikačným metódam. Neurónové siete sú univerzálnym funkčným aproximátorom schopným aproximovať akékoľvek mapovanie vektorového priestoru. Keďže klasifikácia je v podstate mapovanie vektorového priestoru na nominálny, neurónová sieť je teoreticky schopná vykonať akúkoľvek klasifikačnú úlohu.

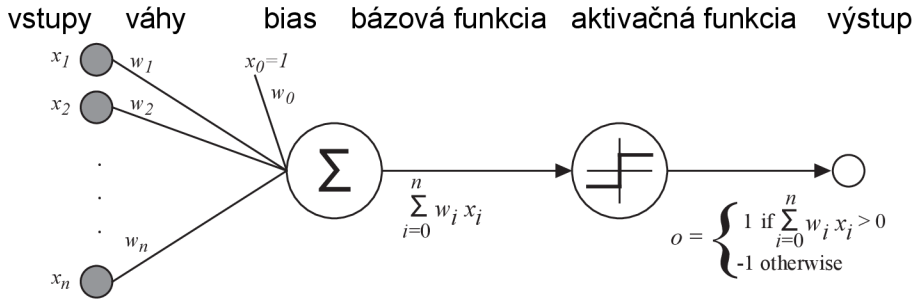
Objaveniu neurónových sietí predchádzal objav (Cajal 1892) funkčnosti nervového systému človeka zloženého z neurónov. Neurón prijíma rôzne signály od ostatných neurónov, spracováva ich a rozhoduje, či potlačiť alebo odoslať signál založený na nejakej internej logike.

Umelá neurónová sieť je graf, ktorý spája jednotky reprezentujúce matematický model biologického neurónu. Prepojenia medzi neurónmi môžu byť jednosmerné i obojsmerné a sú im priradené váhy, ktoré reprezentujú silu spojenia. Z pohľadu klasifikácie je neurónová sieť charakterizovaná nasledovne [1]:

- *Matematický model neurónu* – Popisuje, akým spôsobom neurón produkuje výstup na základe vstupov.
- *Architektúra* – Topológia, ktorá zobrazuje spojenie medzi neurónmi a zahrňuje dobre definovanú množinu vstupov a výstupov neurónu.
- *Kódovanie dát* – Popisuje, ako sú vstupné dáta alebo triedy reprezentované v neurónovej sieti.
- *Trénovací algoritmus* – Algoritmus použitý na určenie optimálnej množiny váh spojených s každým neurónom.

Matematický model jedného neurónu sa nazýva perceptrón a je zobrazený na obrázku 3.4. Vnútorná logika transformuje množinu vstupov $X = (x_1, x_2, ..., x_n)$ na jeden výstup o zložením nasledujúcich funkcií:

1. **Bázová funkcia** – Kombinuje vstupy a príslušné váhy.
2. **Aktivačná funkcia** – Počíta výstup neurónu na základe hodnoty bázovej funkcie.



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Obr. 3.4: Model jedného neurónu (upravené z [8])

Bázová aj aktivačná funkcia môžu byť rôznych typov, čím je daný i výsledný typ neurónovej siete. Najjednoduchším prípadom je použitie váhovanej sumy ako bázovej funkcie a skokovej aktivačnej funkcie. Potom výstup bázovej funkcie v sa vypočíta ako (rovnica 3.19) [23]:

$$v = \sum_{i=1}^{n} x_i w_i + w0. \tag{3.19}$$

w_0 sa nazýva Bias a je to vnútorná konštanta neurónu. Následne skoková aktivačná funkcia len vráti $0, 1$ podľa v, ako je možné vidieť na obrázku 3.4.

Učenie spočíva v určení množiny váh, s ktorými je možné dosiahnuť vyriešenie klasifikačnej úlohy. Učenie perceptronu je možné popísať nasledovne (vzťah 3.20) [8]:

$$w_i \leftarrow w_i + \eta(t - o)x_i, \tag{3.20}$$

kde $t = c(\vec{x})$ je cieľová hodnota, o je výstup perceptronu a η je malá konštanta ovplyvňujúca rýchlosť učenia. Perceptron je možné naučiť klasifikovať len lineárne separovateľné hodnoty.

Spojením niekoľkých neurónov vzniká neurónová sieť. Ak sieť obsahuje len vstupnú a výstupnú vrstvu neurónov, hovoríme o jednovrstvovej neurónovej sieti. Touto sieťou je možné reprezentovať ľubovoľnú booleovskú funkciu. Do siete je možné pridať akýkoľvek počet ďalších vrstiev a vzniká viacvrstvová neurónová sieť. Každá funkcia môže byť aproximovaná s ľubovoľnou presnosťou neurónovou sieťou s dvoma skrytými vrstvami [8]. Vo všeobecnosti platí, že sa volí skôr menší počet skrytých vrstiev.

Jednou z neurónových sietí, ktoré je možné použiť ku klasifikácii je sieť *Backpropagation*, ktorej princíp je popísaný algoritmom (algoritmus 2).

Algoritmus 2: Pseudokód algoritmu Backpropagation [23]

input : **S** ← trénovacia množina
output: Klasifikátor naučený metódou Backpropagation pre vzorky **S**

Inicializácia váh a biasov náhodnými hodnotami z intervalu $\langle -1, 1 \rangle$
while *neurónová sieť nenaučená* **do**
  for *každý prvok* $\boldsymbol{X}$ *z* $\boldsymbol{S}$ **do**
    Postupne od prvej vrstvy pre každý neurón **j** sa vyhodnotí

$$I_j = \sum_i w_{ij} O_i + \theta_j,$$

    kde O_i sú výstupy neurónov z minulej vrstvy, ktoré sú vstupom pre neuróny s indexom **j**, w_{ij} je váha medzi neurónom i a aktuálnym neurónom **j**, θ_j je bias aktuálneho neurónu, I_j je výstup bázovej funkcie aktuálneho neurónu.
    Výstup neurónu O_j sa spočíta

$$O_j = \frac{1}{1 + e^{-I_j}},$$

    Pre každý neurón **j** výstupnej vrstvy sa spočíta chyba

$$Err_j = O_j(1 - O_j)(T_j - O_j),$$

    kde T_j je očakávaná hodnota na výstupe, O_j je skutočná hodnota na výstupe.
    Pre každý neurón **j** v skrytých vrstvách smerom od poslednej sa spočíta

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk},$$

    kde O_j je skutočná hodnota na výstupe, w_{jk} je váha medzi neurónom **j** a nasledujúcim neurónom **k**.
    Všetky váhy sú zmenené

$$\Delta w_{ij} = \mu Err_j O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}.$$

    Všetky biasy sú zmenené

$$\Delta \theta_j = \mu Err_j$$
$$\theta_j = \theta_j + \Delta \theta_j,$$

    kde μ je koeficient učenia z intervalu $\langle 0, 1 \rangle$.
  end
end

Je zložená zo vstupnej vrstvy (tá len predá hodnoty na vstupe), jednej alebo viacerých skrytých vrstiev a výstupnej vrstvy. Na začiatku sa nastavia váhy a biasy na náhodné

malé čísla. Na vstup privedieme vstupný vektor, hodnoty sú predané skrytej vrstve a každý neurón skrytej vrstvy spočíta svoj výstup aktivačnej funkcie ako sumu súčinov jednotlivých váh so vstupnými hodnotami. Aktivačná funkcia má spojitý charakter (rovnica 3.21)

$$y = \frac{1}{1 + e^{-x}}. \tag{3.21}$$

Výstupná hodnota neurónu je potom šírená do ďalších vrstiev. Výstupná vrstva obsahuje jeden neurón pre každú triedu. Neurón, ktorý predstavuje výslednú triedu má ideálne hodnotu 1 a ostatné hodnotu 0. V štádiu učenia musí prebehnúť fáza spätného šírenia chyby (odtiaľ Backpropagation). Výstupný vektor sa porovná s vektorom očakávaných hodnôt a podľa rozdielu dochádza spätne k zmenám váh a biasov. Táto fáza sa opakuje pre všetky trénovacie vzorky, dokým sa sieť nenaučí produkovať správne výsledky pre všetky z nich (alebo nedosiahne maximálneho stanoveného počtu iterácii). [23].

Kapitola 4

Dátové sady

Neodmysliteľnou súčasťou klasifikácie je trénovanie na dátach s dopredu známou triedou jednotlivých vzoriek. Tieto dáta sú zoskupené do dátových sád jednoduchého formátu. Každá sada môže obsahovať informácie z rôznych odvetví, popisujúce rôzne problémy alebo zaznamenávajúce rôzne udalosti. Kapitole postupne predstavuje dátové sady vybrané po dohode s vedúcou práce. Hlavná pozornosť je venovaná jednotlivým parametrom dátovej sady, ako počet vzoriek, atribútov, tried, typ údajov či výskyt chýbajúcich hodnôt. V tejto práci sú sady využité na odhalenie a analýzu jednotlivých vlastností klasifikačných metód, k čomu má dopomôcť najmä ich diverzita.

4.1 Popis dátových sád

Všetky dátové sady použité v tejto práci boli získané z verejne dostupného repozitára pre strojové učenie *UCI Machine Learning Repository* [3]. Názvy sád, ktoré sú použité v texte môžu byť odlišné od pôvodných názvov. Každá sada prešla ručnou kontrolou a formát bol zjednotený. Zmeny sú uvedené pri popise jednotlivý sád v nasledujúcich odstavcoch.

Táto práca sa zameriava na sady reprezentované formou atribútov a položiek. Všetky sady sú vo formáte CSV (z angl. *Comma-separated values*), pričom stĺpce sú oddelené čiarkou a záznamy novým riadkom. Niektoré sady sú často výsledkom už vykonaného procesu predspracovania, iné tvoria reálne hodnoty meraní. Každá vzorka má priradenú triedu a niektoré hodnoty môžu chýbať. Jednotný znak pre označenie chýbajúcej hodnoty je *'?'*. Súčasťou prvého riadku niektorých dátových súborov je i hlavička, ktorá udáva názvy jednotlivých stĺpcov, nie je však povinná.

Pre potreby tejto práce nie je nutné poznať presnú sémantiku všetkých dát. Číselné metriky popisujúce použité dátové sady sú uvedené v tabuľke 4.1. Stĺpec „počet vzoriek“ uvádza súčet vzoriek trénovacej i testovacej sady, pokiaľ boli dostupné ako samostatné a odlišné sady. Symbol „∗“ v stĺpci s počtom atribútov upozorňuje na zmenu atribútov vzhľadom na pôvodnú sadu. Originálny popis je dostupný na stránkach UCI Machine Learning Repository alebo na CD/DVD nosiči výslednej práce v zložke s dátovými sadami (súbory s názvom *.names). V nasledujúcich bodoch sú stručne popísané hlavné charakteristiky vybraných sád [3]:

- **Adult Salary** – (Orig. názov: *Adult*). Dátová sada obsahuje vzorky so socio-ekonomickými informáciami o rôznych osobách. Jedná sa o údaje ako vek, pohlavie, rasa, typ zamestnania, dosiahnuté vzdelanie, zisk, dĺžka pracovnej doby a iné. Hodnoty atribútov sú nominálne i numerické. Každá vzorka je priradená do jednej z dvoch tried

Názov	Počet vzoriek	Počet atribútov	Stĺpec s triedou	Triedy	Chýbajúce hodnoty
Adult Salary	32561	14 + 1	15.	>50K, <=50K	2399
Arrhythmia	452	279 + 1	280.	1 ... 16	408
Breast Cancer Coimbra	116	9 + 1	10.	1 , 2	0
Breast Cancer Wisconsin	699	9 + 1 *	10.	2, 4	16
Habermans Survival	306	3 + 1	4.	1, 2	0
MAGIC Gamma Telescope	19020	10 + 1	11.	g, h	0
Mammographic Mass	961	5 + 1	6.	0, 1	162
MONKs Problems	432	6 + 1 *	1.	0, 1	0
Occupancy Detection	20561	5 + 1 *	6.	0, 1	0
Poker Hand	1025010	10 + 1	11.	0 ... 9	0
Zoo	101	16 + 1 *	17.	1 ... 7	0

Tabuľka 4.1: Číselné informácie o jednotlivých sadách

– „>50K“ alebo „<=50K“, ktoré vyjadrujú fakt, či daná osoba zarába viac/menej ako 50000 $ ročne. Vzoriek z triedy „>50K“ je v sade prítomných len 24 %.

- **Arrhythmia** – Sadu tvoria základné údaje o pacientoch a výsledky merania ich srdcovej činnosti. Jednotliví pacienti sú zaradení do 16 rôznych tried podľa závažnosti a typu srdcovej arytmie. Z toho však tri triedy nie sú priradené žiadnej vzorke v sade. Celkovo sada obsahuje až 280 atribútov, numerického i nominálneho typu.

- **Breast Cancer Coimbra** – Sadu tvoria lekárske záznamy vyšetrení pacientov – konkrétne rozbory odberu krvi. Všetky údaje sú numerické a pacienti buď majú diagnostikovanú rakovinu prsníka (trieda „2“) alebo nie (trieda „1“). Medzi dátami nie je prítomná žiadna chýbajúca hodnota. Distribúcia tried je pomerne vyrovnaná, pričom 55 % vzoriek tvoria pacienti z triedy „2“.

- **Breast Cancer Wisconsin** – Dátová sada je opäť zameraná na prítomnosť rakoviny prsníka u pacientov, obsahuje však iný druh meraní. Všetky atribúty sú celočíselné a nadobúdajú hodnôt v rozmedzí od 1 – 10. Trieda „2“ (66 %) označuje zdravých pacientov a trieda „4“ (34 %) chorých. Pôvodná sada obsahovala atribút obsahujúci čísla vzoriek, ktorý bol zo sady odstránený. Distribúcia tried je pomerne vyrovnaná.

- **Haberman's Survival** – Databáza pacientov, ktorí podstúpili operáciu pre odstránenie nádoru prsníka. O pacientoch sú známe len 3 údaje – vek, rok operácie a počet pozitívnych podpazušných uzlín. Pacienti sú klasifikovaní do dvoch tried – na tých, ktorí prežili viac ako 5 rokov (trieda „1“) a tých, ktorí neprežili (trieda „2“). Všetky atribúty sú numerické a sada neobsahuje chýbajúce hodnoty. Prevažujú vzorky pacientov triedy „1“, ktorej zastúpenie v sade je 74 %.

- **MAGIC Gamma Telescope** – Celý názov znie *Major Atmospheric Gamma Imaging Cherenkov Telescope project* a dáta sú náhodne generované veličiny, simulujúce príjem pozemného Chrenkovho gamma teleskopu. Na základe týchto dát sa určuje, či prijatý signál predstavuje gamma lúče (trieda „g“) alebo šum pozadia (trieda „h“). Atribúty okrem triedy sú numerické, bez chýbajúcich hodnôt. Pokiaľ by bola do úvahy

braná i sémantika danej sady, obecná presnosť klasifikácie by bola pre tieto dáta zavádzajúca. Je horšie klasifikovať šum ako gamma signál, než klasifikovať signál ako šum. Distribúcia hodnôt jednotlivých tried je v sade značne odlišná od skutočnosti – vzoriek z triedy „g“ je 65 % a vzoriek z triedy „h“ je 35 %, pričom v skutočných dátach trieda „h“ predstavuje značnú väčšinu udalostí.

- **Mammographic Masses** – Dátová sada, ktorá je výstupom vyšetrení na prítomnosť rakoviny prsníka pomocou mamografu. Táto sada môže byť použitá pre predpoveď závažnosti nálezu pri mamografickom vyšetrení. Jeden atribút (vek) je numerický, zvyšné sú ordinálne a nominálne. Klasifikačnou úlohou je opäť zaradiť pacientov do tried – nemá zhubný nádor „0“ alebo má „1“. Rozloženie tried je pomerne vyrovnané, 54 % vzoriek z triedy „0“ a 45 % z triedy „1“.

- **MONKs Problems** – Dátová sada ktorá vznikla v roku 1991 práve pre porovnanie rôznych klasifikačných metód. Každá vzorka predstavuje robota reprezentovaného 6 kategorickými atribútmi reprezentovanými číslom. Roboti sú klasifikovaní do dvoch tried – „1“, „2“. Všetkých možných kombinácií tvoriacich unikátnych robotov je 432. Dostupné sú tri skupiny trénovacích a testovacích dát:

  1. Testovacia sada – 432 vzoriek, trénovacia – 124.
  2. Testovacia sada – 432 vzoriek, trénovacia – 169.
  3. Testovacia sada – 432 vzoriek, trénovacia – 124. 5 % trénovacích dát tvorí šum, resp. nesprávne klasifikované vzorky.

  Trénovacia sada je podmnožinou testovacej a neobsahuje teda žiadne unikátne vzorky, ktoré by neboli obsiahnuté v testovacej sade. V pôvodnom výskume bol ku každej sade priradený ciel učenia a na základe jeho naplnenia boli jednotlivé metódy porovnávané [22]. V tejto práci je použitá len prvá skupina dát.

- **Occupancy Detection** – Názov je možné preložiť ako „detekcia obsadenosti“ a obsahuje dáta získané sledovaním charakteristík ovzdušia pravidelne využívanej kancelárie. Dáta je možné použiť pre rozhodovanie, či sa v danej miestnosti osoba nachádza (trieda „1“) alebo nie (trieda „0“). Rozhodovanie je pritom založené len na vlastnostiach ovzdušia. Vzorky sú popísané 5 numerickými atribútmi. Originálna sada obsahovala atribút „date“ predstavujúci časové razítko ponechané z procesu vzorkovania pôvodného časového záznamu a unikátne identifikačné číslo vzorky. Oba tieto atribúty nemá zmysel uvažovať a preto boli zo sady odstránené. Dátová sada pozostáva z trénovacej sady s 2665 vzorkami a dvoch testovacích sád s 8143 a 9753 vzorkami, pričom väčšina experimentov využíva k testovaniu len väčšiu z dvoch sád.

- **Poker Hand** – Najrozsiahlejšia dátová sada, kde každý záznam predstavuje ruku hráča pozostávajúcu z piatich pokerových kariet zo štandardného balíčka 52 kariet. Každá karta je popísaná pomocou dvoch atribútov – znak a číslo. Celkovo sú teda karty na ruke popísané 10 ordinálnymi atribútmi. Triedny atribút určuje skupinu kariet – nič na ruke, jedna dvojica, trojica, full house, kráľovská postupka, atď. Chýbajúce hodnoty nie sú prítomné. Sada je rozdelená na trénovaciu (25010 vzoriek) a testovaciu (1000000 vzoriek) sadu, pričom percentuálne zastúpenie jednotlivých tried je veľmi podobné zastúpeniu v doméne všetkých možných kombinácií kariet na ruke.

- **Zoo** – Jednoduchá sada, ktorej vzorky predstavujú popis živočícha na základe jeho charakteristických čŕt (povrch tela, rozmnožovanie, počet nôh, ...). Živočích je následne priradený k jednej zo 7 skupín zvierat zdieľajúcich spoločné vlastnosti. Všetky atribúty okrem atribútu „počet nôh“ (numerický) sú binárne a vyjadrujú, či je daná vlastnosť prítomná alebo nie. V originálnej sade je prítomný i atribút predstavujúci unikátny názov daného živočícha, ktorý bol zo sady odstránený. Pokrytie jednotlivých tried je veľmi nerovnomerné, najviac (41) vzoriek reprezentuje triedu „1“ (cicavce) a najmenej vzoriek (4) triedu „5“ (žaby).

4.2 Uváženie vhodnosti

Pri výbere sád bol braný ohľad nielen na sémantickú rôznorodosť, ale i na diverzitu v počte hodnôt, počte atribútov a použitých typoch. Rôznorodosť dátových sád má za cieľ odhaliť slabé a silné stránky jednotlivých klasifikačných metód a tým umožniť ich porovnanie. Hlavnými sledovanými parametrami je presnosť a čas strávený učením alebo testovaním metód na jednotlivých sadách. Analýza vhodnosti je dôležitým krokom pri výbere dátových sád a na jej výsledkoch je možné neskôr založiť celú radu experimentov. V nasledujúcich bodoch sú menované hlavné parametre vybraných dátových sád. Niektoré z nich nachádzajú využitie v kapitole experimenty 6:

- **Počet záznamov** – Sady s veľkým počtom záznamov je možné využiť hlavne pre porovnanie časovej náročnosti trénovacej alebo testovacej fáze klasifikačných metód. Sada *Poker Hand* obsahuje najrozsiahlejšiu testovaciu sadu a sada *Adult Salary* naopak najväčšiu trénovaciu sadu. Rozdiely v rýchlostiach by preto mali byť najviditeľnejšie práve pri týchto dátach. Množstvo záznamov tiež môže ovplyvňovať i presnosť jednotlivých metód. Problémy s veľmi rozsiahlou množinou možných záznamov potrebujú často čo najväčšiu množinu trénovacích vzoriek, aby bola klasifikácia presná. Naopak, existujú i problémy, ktoré sa je možné naučiť i s menším počtom vzoriek.

- **Počet atribútov** – Atribútov je spravidla menej než záznamov, avšak ich dopad na výslednú presnosť alebo časovú náročnosť môže byť oveľa väčší. Zaujímavé tak môže byť pozorovanie sady *Arrhythmia*, pričom atribúty je možné systematicky odstraňovať a sledovať výkyvy presnosti alebo nižšie časy výpočtov.

- **Počet tried** – Implementácia vybraných metód bola volená tak, aby všetky metódy boli schopné viac-triednej klasifikácie (viac v kapitole 5.2.1). Výkonnosť jednotlivých metód pri klasifikácii do viacerých tried môže byť rôzna, avšak výsledky nie sú veľmi jednoznačné. Problémom je, že pri všeobecnom porovnávaní metód je ťažké určiť, či sú slabšie výsledky spôsobené počtom hodnôt alebo počtom atribútov.

- **Typ hodnôt** – Typ hodnôt môže ovplyvniť najmä metódy, ktoré striktne vyžadujú určitý typ dát alebo s nimi dokážu lepšie pracovať. Mnohé implementácie požadujú vstupné dáta v numerickom tvare, pričom kategorické dáta sa často prevádzajú binarizáciou na viacero atribútov. Táto transformácia môže spôsobiť v niektorých prípadoch navýšenie atribútov na extrémne množstvo, čo je potrebné pri porovnávaní zohľadniť.

- **Chýbajúce hodnoty** – Niektoré z dátových sád obsahujú neúplné hodnoty, ktoré môžu byť v procese učenia zavádzajúce a skresľovať výslednú presnosť. Hodnoty je možné dopĺňať spôsobmi popísanými v kapitole 2.3.1.

- **Šum** – Šum je najlepšie pozorovateľný hlavne v sadách, u ktorých sú známe všetky možné záznamy a ich triedy. Ovplyvnenie presnosti v dôsledku šumu je závislé na dôležitosti zmenenej hodnoty (z pohľadu vnútornej reprezentácie dát daným klasifikátorom) a môže byť skresľujúce.
- **Nerovnomerné zastúpenie tried** – V niektorých trénovacích sadách môžu byť počty vzoriek patriacich do jednotlivých tried natoľko rozdielne, že niektorá trieda má výrazne nižšie zastúpenie. Potom môže dochádzať k lepšiemu naučeniu prevažujúcich tried, a tým k nesprávnej klasifikácii napríklad dôležitejších minoritných vzoriek.

Kapitola 5

Návrh a implementácia aplikácie

Jedným z cieľov tejto práce je vytvoriť desktopovú aplikáciu, ktorá umožňuje prácu s vybranými klasifikačnými metódami a poskytuje informácie užitočné k ich porovnaniu. Táto kapitola sa bude venovať postupne jednotlivým fázam vývoja aplikácie, od špecifikácie požiadaviek, výberu konkrétnych metód, cez návrh grafického užívateľského rozhrania, realizačné prostriedky, implementáciu až po testovanie a zhodnotenie funkčnosti. Pozornosť bude venovaná hlavne nárokom na funkcionalitu a spôsobu, ktorým bolo dosiahnuté ich pokrytie.

5.1 Špecifikácia požiadaviek

Aplikácia musí byť spustiteľná aspoň na operačnom systéme Windows a poskytovať intuitívne užívateľské rozhranie. Vo všeobecnosti musí umožňovať prácu s rôznymi dátovými sadami a vybranými klasifikačnými metódami. Užívateľ si môže vybrať dátovú sadu, s ktorou chce pracovať, a klasifikačné metódy, ktoré chce porovnávať. Prípadné chyby v dátových sadách, konflikty typov či akékoľvek iné dôvody nepoužiteľnosti dátovej sady sú hlásené užívateľovi alebo automaticky opravené. Výsledky testovania sú prehľadne zobrazené užívateľovi takým spôsobom, aby neboli potrebné žiadne ďalšie výpočty založené na implementačných detailoch jednotlivých použitých metód.

Dátová sada musí byť uložená vo formáte CSV, pričom jednotlivé hodnoty musia byť jednoduchých typov – `int`, `float`, `string`. Aplikácia neumožňuje pokročilé techniky predspracovania dát. Vstupom preto musí byť dátová sada vo vhodnej podobe, ktorá bude použitá priamo na trénovanie jednotlivých klasifikátorov.

Podrobnejšiu špecifikáciu funkčnosti možno popísať rozdelením do niekoľkých pracovných fáz. Každá fáza zahŕňa niekoľko rozdielnych procesov a úloh, ktoré sú vykonávané automaticky, alebo ktoré môže užívateľ priamo ovplyvniť. Nasledujúce fázy a ich popis sa dá chápať i ako výčet všetkých funkcií a vymožeností aplikácie, a zároveň výčet nárokov a obmedzení niektorých vstupov či výstupov.

1. **Spustenie** – Po spustení aplikácie je užívateľ povinný vybrať dátovú sadu. Dátová sada musí byť uložená vo formáte CSV v súbore s príponou „.csv“ alebo „.txt“. Jednotlivé záznamy musia byť oddelené novým riadkom a hodnoty čiarkou. Hodnota reprezentovaná symbolom „?“ alebo „NaN“ je považovaná za chýbajúcu hodnotu. Užívateľ môže nastaviť, či chce odstrániť všetky záznamy s chýbajúcimi hodnotami alebo numerické hodnoty doplniť najčastejšie sa vyskytujúcou hodnotou daného atribútu. Samozrejmosťou je možnosť zobrazenia nápovede.

2. **Príprava dát** – Táto fáza sa zaoberá načítaním, kontrolou a správnym nastavením dátovej sady. Výsledkom sú plne pripravené dáta pre ďalšie spracovanie.

   (a) *Kontrola sady* – Pri načítavaní sú dáta skontrolované na správnosť formátu a o prípadnej chybe je užívateľ informovaný a sada je odmietnutá. Typy jednotlivých atribútov sú odvodené na základe hodnôt. Prvý riadok dátového súboru môže obsahovať hlavičku s pomenovaním jednotlivých atribútov. Hlavička je však nepovinná a jej prítomnosť môže byť zistená automaticky analýzou dát.

   (b) *Vytvorenie vnútornej reprezentácie* – Každá sada je postupne prevedená do internej reprezentácie aplikácie. Užívateľ si môže vybrať, či chce názvy atribútov generovať alebo použiť prvý riadok súboru alebo automaticky rozhodnúť na základe obsahu na prvom riadku. Generované názvy stĺpcov dát sú pomenované automaticky ako $(attr\_1, attr\_2, ..., attr\_n)$. Pokiaľ mali byť záznamy s chýbajúcimi hodnotami odstránené, dôjde k ich odstráneniu.

   Pre pokračovanie do ďalšej fázy užívateľ musí zo zoznamu vybrať atribút, ktorý predstavuje klasifikačnú triedu. Potvrdením výberu dochádza k prevodu všetkých kategorických atribútov (mimo triedneho) pomocou binarizácie (one-hot-encoder) na ich numerickú podobu.

   (c) *Zobrazenie informácií o sade* – Pre každú sadu sú zozbierané sumárne informácie, ktoré sú dostupné používateľovi: celkový počet záznamov v sade, počet atribútov, zoznam atribútov. Tiež je možné zobraziť celú sadu so všetkými hodnotami v podobe tabuľky a overiť tak, či je správne načítaná.

3. **Rozdelenie dát** – Aby mohli byť všetky metódy testované za rovnakých podmienok, užívateľ musí rozhodnúť, akým spôsobom budú dáta rozdelené na trénovaciu a testovaciu sadu. Volí sa tak spôsob validácie jednotlivých metód. Na výber sú nasledujúce spôsoby rozdelenia (popísané v kapitole 3.2.1):

   (a) *Holdout* – Načítanú dátovú sadu je možné rozdeliť v ľubovoľnom pomere na trénovaciu a testovaciu sadu. Jednotlivé záznamy sú rozdeľované náhodne. Avšak kvôli replikácii výsledkov je možné ovplyvňovať náhodnosť výberu. Voľba veľkosti testovacej sady s hodnotou 0.0 vytvorí testovaciu i trénovaciu sadu obsahujúcu všetky prvky načítanej sady.

   (b) *Leave-one-out* – Postupné rozdeľovanie nasledované testovaním. Priebežné výsledky sú zaznamenávané a na konci spriemerované.

   (c) *Krížová validácia* – Užívateľ môže zvoliť počet rozdelení. Záznamy pred rozdelením môžu byť náhodne premiešané alebo sa môžu vyberať postupne.

   (d) *Vlastná testovacia sada* – Vlastná testovacia sada podlieha spomenutým obmedzeniam pre dátové sady. Sada zvolená pri spustení aplikácie slúži celá na trénovanie. Obe sady musia mať zhodný formát dát. O prípadných chybách je užívateľ informovaný.

   K rozdeleniu dochádza až po spustení validácie metód a aktívny môže byť len jeden spôsob rozdelenia.

4. **Nastavenie klasifikačných metód** – Najdôležitejšou časťou z pohľadu experimentov je nastavenie klasifikačných metód. V ponuke sú dostupné štyri klasifikátory – Naive Bayes (jednoduchý Bayes), Decision Tree (rozhodovací strom), Neural Network

(neurónová sieť), SVM. Užívateľ môže modifikovať chovanie metód dostupnými konfiguračnými parametrami alebo ponechať východzie nastavenie. Viac informácií k jednotlivým klasifikátorom vrátane ich implementácie je uvedených v kapitole 5.2.1.

5. **Validácia** – V momente, kedy sú známe metódy rozdelenia dát a nastavené metódy, užívateľ môže spustiť validáciu. Je možné validovať každý klasifikátor samostatne alebo spustiť sekvenčnú validáciu všetkých metód. Postupne sa vykoná rozdelenie sady definovaným spôsobom, prebehne učenie klasifikátora na trénovacích dátach a automaticky sa spustí testovanie na testovacej sade. V prípade krížovej validácie alebo leave-one-out sa učenie a testovanie opakuje n-krát podľa nastavenia rozdeľovacej metódy. Výsledkom je niekoľko hodnotiacich veličín – *presnosť, čas strávený trénovaním, čas strávený testovaním.* Výsledné hodnotenia sú priemerom čiastkových výsledkov a sú kalkulované pre každý klasifikátor samostatne.

   Proces trénovania a testovania je monitorovaný a všetky prípadné chyby sú hlásené užívateľovi spolu s popisom možnej príčiny a návodom k jej odstráneniu. K upozorneniu dochádza i pokiaľ je zaznamenaná konfigurácia klasifikátoru, ktorá nevedie k jeho konvergencii.

6. **Zobrazenie výsledkov** – Po úspešnej validácii sú výsledky automaticky zobrazené v samostatnom okne. V prípade analýzy všetkých metód zároveň sú zobrazené všetky výsledky naraz po jej skončení. Reprezentácia je nasledovná:

   (a) *Presnosť* – Obecná presnosť *acc* klasifikačnej metódy je vyjadrená vzťahom (rovnica 5.1) a užívateľovi je zobrazená v tvare desatinného čísla. Druhou formou zobrazenia presnosti je chybová matica alebo tiež matica zámen. Použitím leave-one-out alebo krížovej validácie dochádza k sčítaniu čiastkových matíc pre poskytnutie detailnejšieho pohľadu na klasifikáciu modelu.

   $$acc = \frac{počet\ správne\ klasifikovaných\ vzoriek}{počet\ všetkých\ vzoriek} \tag{5.1}$$

   (b) *Čas trénovania* – Čas od zahájenia trénovacej fázy do jej ukončenia z dôvodu dosiahnutia povolenej chyby alebo ustálenia učenia. Čas je uvedený v milisekundách.

   (c) *Čas testovania* – Čas strávený testovaním klasifikátora na testovacej sade uvedený v milisekundách.

   Aplikácia uchováva posledné výsledky validácií počas jej behu a je možné ich kedykoľvek zobraziť. Maticu zámen aj s percentuálnym údajom o presnosti je možné uložiť do súboru.

7. **Ukončenie** – Program je možné kedykoľvek ukončiť. Ukončenie predstavuje uvoľnenie všetkých alokovaných zdrojov a uzatvorenie aplikácie. Možné je aj uzatvorenie aktuálne používanej dátovej sady a uvedenie užívateľského rozhrania do východzej podoby. Posledné nastavenia metód tak zostávajú zachované a môžu byť použité pre nasledujúcu dátovú sadu. Výsledky predchádzajúcich meraní sú vymazané.

5.2 Realizácia

Táto kapitola sa venuje programovému riešeniu aplikácie. Predstavené sú použité technológie a ich prínos pre výsledný produkt a pozornosť je venovaná jednotlivým implementovaným modulom a možnostiam konfigurácie vybraných metód. Aplikácia spĺňa požiadavky uvedené v kapitole 5.1.

Za implementačný jazyk bol zvolený *Python* vo verzii *3.7*. Jedná sa o interpretovaný, dynamicky typovaný jazyk s automatickou správou pamäte. Hlavnou výhodou je množstvo dostupných modulov umožňujúcich efektívnu prácu s veľkými dátami, strojové učenie, vizualizáciu dát a vytvorenie grafického užívateľského rozhrania (ďalej len GUI). Okrem toho je možné Python spustiť pod systémami Windows, Linux, Mac OS, neoficiálne napríklad i Android či iOS, a teda je možné vytvoriť plne prenosnú aplikáciu [12]. GUI je vytvorené pomocou knižnice *PyQt5* a viac informácií je uvedených v kapitole 5.3. Pre vývoj boli použité vývojové prostredia PyCharm a Qt Designer.

Aplikácia využíva vo svojej implementácii i niekoľko voľne dostupných Python modulov určených pre potreby spracovania a analýzy rozsiahlych dát, strojového učenia a vizualizácie – menovite hlavne *Pandas*, *Scikit-learn* a *Matplotlib*.

Pandas – je voľne dostupná knižnica poskytujúca kvalitné a jednoducho použiteľné dátové štruktúry a analyzačné nástroje [10].

Scikit-learn – je Python modul, ktorý zaisťuje širokú škálu algoritmov strojového učenia (s učiteľom i bez). V tejto práci je zdrojom implementácie všetkých klasifikačných metód. Ponúka tiež rôzne spôsoby rozdelenia dát a validáciu metód. Modul je ľahko integrovateľný do iných knižníc, využíva len moduly – *numpy* (základné dátové štruktúry), *scipy* (štatistika, matice, lineárna algebra), *cython* (vrstva nad C implementáciou SVM metódy) [11] [19].

Matplotlib – je 2D vykresľovacia knižnica, ktorá produkuje obrázky grafov podľa zadaných dát a parametrov [7].

Kompletná implementácia aplikácie je rozdelená do 10 modulov (rovnomenných súborov). V tabuľke 5.1 sú vymenované jednotlivé moduly spolu so stručným popisom problému, ktorý implementujú. V nasledujúcich odstavcoch sú popísané kľúčové časti jednotlivých modulov, predstavené sú použité technológie a moduly tretích strán.

Modul (súbor)	**Popis**
app	*Hlavná programová slučka.*
DataLoader	*Načítanie súboru a vytvorenie dátovej sady.*
DataSet	*Objekt dátovej sady a spôsoby jej rozdelenia.*
Classificators	*Trénovanie a testovanie klasifikátorov.*
Visualization	*Vytvorenie tabuľky a matice zámen.*
UIController	*Obsluha hlavného okna aplikácie.*
Errors	*Definície výnimiek a spracovanie chýb.*
ClassificationApp	*GUI hlavného okna aplikácie.*
DatasetTable	*GUI zobrazenia dátovej sady.*
Results	*GUI pre zobrazenie výsledkov.*

Tabuľka 5.1: Názvy jednotlivých modulov a ich stručný popis

Východzím bodom aplikácie je súbor `app.py`, ktorý obsahuje hlavnú programovú slučku, ktorá zabezpečuje beh užívateľského rozhrania. O interakciu užívateľa s aplikáciou sa stará modul `UIController`, ktorý bude spolu s grafickou reprezentáciou (moduly `ClassificationApp`, `DatasetTable`, `Results`) bližšie popísaný v kapitole 5.3.

Fázu prípravy a rozdelenia dát (viď 5.1) implementujú moduly `DataLoader`, `DataSet`. `DataLoader` zabezpečuje načítanie CSV súboru s dátami a ich uloženie do špeciálneho dátového rámca `pandas.DataFrame`. Skúmaním prvých riadkov dát dochádza k zisteniu prítomnosti hlavičky – pomenovania atribútov. `DataSet` obsahuje triedu, ktorá poskytuje všetky potrebné metódy na prácu s načítaným dátovým rámcom. Jedná sa o získavanie súhrnných informácií o dátach, určenie triedneho atribútu (anglicky *label*), konvertovanie kategorických atribútov a delenie dát na viacero sád. Zaisťuje i rozdelenie dát využitím modulu Scikit-learn.

Modul `Classificators` implementuje triedu pre jednoduché inicializovanie a validáciu jednotlivých klasifikačných metód. Podrobnejšie informácie o spôsobe použitia vrátane konfigurovateľných parametrov sú uvedené v kapitole 5.2.1. K meraniu časovej náročnosti učenia a klasifikácie bol využitý štandardný modul `time`.

Štruktúry potrebné pre zobrazovanie dátových sád a výsledkov validácie sú obsiahnuté v module `Visualization`. Pre generovanie matice zámen je využitý modul Matplotlib, dátová sada je zobrazená formou tabuľky priamo s využitím knižnice PyQt5. Aplikácia informuje užívateľa o vzniknutých chybách vizuálnou formou. Jednotlivé výnimky sa nachádzajú v module `Errors` spolu s metódami potrebnými k ich zobrazeniu.

Najzákladnejší princíp fungovania aplikácie je možné s vynechaním užívateľského rozhrania a s využitím implementovaných metód popísať zjednodušeným pseudo-algoritmom (algoritmus 3).

Algoritmus 3: Zjednodušená ukážka behu programu

```
1  Function main():
2      dl = DataLoader();
3
4      # načítanie dátovej sady
5      data_set = dl.load_csv("dataset.csv");
6
7      # nastavenie triedy
8      data_set.set_label("Class");
9
10     # vytvorenie klasifikátora typu Neurónová sieť
11     NN = Classificator(TYPE_NN, max_iter=10000, hidden_layers=(20, 20, 20));
12
13     # validácia s rozdelením leave-one-out
14     NN.validate(data_set, VALIDATION_LOO);
15
16     # zobrazenie výsledkov
17     print(NN.result_acc);
18     print(NN.result_training_time);
19     print(NN.result_testing_time);
```

5.2.1 Konfigurácia vybraných klasifikačných metód

Cieľom tejto diplomovej práce je porovnanie vybraných klasifikačných metód a vytvorenie aplikácie, ktorá toto porovnanie umožní. Konkrétne sa jedná o metódy, ktorých základný princíp bol popísaný už v kapitole 3, menovite teda: jednoduchý Bayesovký klasifikátor, rozhodovací strom, neurónová sieť, SVM. Aby bolo možné s metódami pohodlne pracovať a experimentovať s ich nastaveniami, sú integrované priamo do aplikácie.

Knižnica Scikit-learn je jednou z najpoužívanejších Python knižníc v oblasti strojového učenia pre jednoduché až stredne náročné problémy. Implementácie porovnávaných klasifikačných metód sú preto použité práve z tejto knižnice. Výsledky tak môžu byť zaujímavé pre veľkú skupinu programátorov. Napriek tomu, že Scikit-learn sa zameriava na jednoduchosť použitia a je zväčša napísaný vo vysokoúrovňovom jazyku, ohľad bol braný aj na maximalizáciu výpočtového výkonu [11].

Klasifikačné metódy boli vybrané tak, aby ich princíp výpočtu a učenia bol odlišný. Bayesovská klasifikácia používa štatistiku a rozhoduje na základe pravdepodobnosti, rozhodovacie stromy reprezentujú pravidlá opísateľné prirodzeným jazykom. Neurónové siete predstavujú čiernu skrinku, ktorá rôzne váhuje a kombinuje vstupy a kalkuluje výstupy a SVM konvertuje vstupný priestor do lineárne separovateľného a hľadá jeho ideálne rozdelenie. Scikit-learn obsahuje rôzne varianty jednotlivých metód, ktorých efektivita sa môže líšiť. Niektoré varianty môže užívateľ meniť priamo v aplikácii. Podrobný popis implementácie týchto metód je nad rámec tejto práce. K metódam je preto pristupované ako k čiernym skrinkám, u ktorých je možné manipulovať s niektorými vstupnými parametrami a nezaujíma nás ich možná optimalizácia. Vo výsledku je tak možné porovnať všeobecné rozdiely vo vlastnostiach metód implementovaných knižnicou Scikit-learn. V nasledujúcich odstavcoch je uvedený stručný popis vybraných klasifikačných metód knižnice Scikit-learn z pohľadu možností ich konfigurácie v rámci užívateľského rozhrania aplikácie.

Jednoduchá Bayesovská klasifikácia – Scikit-learn implementuje rôzne typy tejto metódy. Odlišujú sa najmä predpokladmi v súvislosti s distribúciou $P(X_i|y)$, y je trieda a $(x_1...x_i)$ je vzorka dát s i atribútmi. Aplikácia umožňuje voliť medzi troma typmi označovanými ako `GaussianNB`, `MultinomialNB`, `ComplementNB`. Pokiaľ je to nutné, automaticky sa uplatňuje Laplaceova korekcia [17].

`GaussianNB` predpokladá, že pravdepodobnosť vzorov má Gaussovo rozloženie. Produkuje dobré výsledky pre veľké množstvo klasifikačných úloh.

`MultinomialNB` implementuje jednoduchú Bayesovskú klasifikáciu pre multinomiálne rozložené dáta a je jednou z klasických variant používaných pre klasifikáciu textu.

`ComplementNB` označovaný ako CNB, je adaptácia štandardného multinomiálneho Bayesovského klasifikátora, ktorý je určený obzvlášť pre dátové sady s nerovnomernou distribúciou tried. CNB používa na výpočet váh modelu štatistiky z doplnku každej triedy. Varianta často prekonáva vo výsledkoch multinomiálny NB.

Rozhodovací strom – Použitá trieda sa nazýva `DecisionTreeClassifier` a je schopná viactriednej klasifikácie. Parametre ovplyvňujúce veľkosť stromu (hĺbka, počet vzoriek v listoch) sú pevne nastavené tak, že vedú k plne vygenerovanému stromu bez orezania. Nastaviteľnými parametrami sú spôsob merania kvality rozdelenia a inicializácia generátora náhodných čísel [16].

`Gini/Enthropy` sú funkcie na meranie kvality rozdelenia, buď „Gini“ alebo „Enthropy“ pre informačný zisk.

`Random state` umožňuje nastaviť číslo, ktoré bude použité pre inicializáciu generátora náhodných čísel. Vlastnosti sú vždy náhodne permutované pri každom rozdelení. Môže sa preto stať, že nájdenie najlepšieho rozdelenia sa môže líšiť i pri rovnakých trénovacích dátach.

Neurónová sieť – Trieda `MLPClassifier` implementuje algoritmus viacvrstvového perceptronu (jedná sa vlastne o viacvrstvovú doprednú sieť), ktorý pri trénovaní využíva spätné šírenie chyby (angl. backpropagation) a je schopný naučiť sa nelineárne problémy. Predpokladajme τ trénovacích vzoriek, μ atribútov, σ skrytých vrstiev, každá s η neurónmi, υ výstupných neurónov a i iterácií. Časová náročnosť backpropagation je $O(\tau \cdot \mu \cdot \eta^{\sigma} \cdot \upsilon \cdot i)$. Preto je dobré začať trénovanie s malým počtom skrytých vrstiev a neurónov. Scikit-learn neposkytuje podporu GPU pre výpočty. Aplikácia poskytuje pre neurónovú sieť niekoľko parametrov k nastaveniu [18].

`Activation function` je aktivačná funkcia a môže byť typu `relu` $[f(x) = max(0, x)]$, `tanh` $[f(x) = tanh(x)]$ a `logistic` $[f(x) = \frac{1}{(1+exp(-x)}]$.

`Solver` predstavuje použitú modifikáciu/optimalizáciu algoritmu backpropagation, ktorá môže byť:

- `sgd` – (Stochastic Gradient Descent) aktualizuje váhy pomocou gradientu stratovej funkcie (chyby), z ohľadom na ich aktuálne hodnoty.
- `adam` – je optimalizér podobný SGD, ktorý dokáže automaticky prispôsobiť zmeny váh na základe adaptívnych odhadov momentov nižšieho rádu (viac v [5]).
- `lbfgs` – používa implementáciu L-BFGS z knižnice Scipy. L-BFGS konverguje rýchlejšie a s lepšími výsledkami na malých dátových sadách. U veľkých dátových sád je často lepšie použiť algoritmus Adam alebo SGD.

`Hidden layers size` požaduje na vstupe reťazec tvaru „$int_1 int_2 ... int_n$“ kde n predstavuje počet skrytých vrstiev a int_i je celé číslo udávajúce počet neurónov vo vrstve i. Napríklad „20 20 20“ predstavuje 3 skryté vrstvy, každú s 20 neurónmi.

`Max iterations` udáva maximálny počet iterácií. Algoritmus beží kým nekonverguje alebo nedosiahne daný počet iterácií.

`Tolerance error` predstavuje presnosť riešenia. Pokiaľ nedošlo k zlepšeniu aspoň o danú presnosť, konvergencia je považovaná za dosiahnutú.

`No change tol iterations` špecifikuje počet iterácií počas ktorých dochádza k overovaniu presnosti. Pokiaľ sa presnosť riešenia nemení aspoň o hodnotu danú parametrom `Tolarence error` po dobu `No change tol iterations` iterácií, učenie je ukončené a je prehlásená konvergencia.

`Random state` umožňuje nastaviť číslo, ktoré bude použité pre inicializáciu generátora náhodných čísel a umožňuje tak replikáciu výsledkov.

SVM – Scikit-learn poskytuje 3 rôzne implementácie metódy SVM, pričom vybraná bola najefektívnejšia z nich – `LinearSVC`. Zvyšné dve, označené ako SVC a NuSVC, využívajúce implementáciu z knižnice libsvm, vykazovali pri testovaní vhodnosti mnohonásobne dlhšie časy učenia. `LinearSVC` interne používa implementáciu z knižnice `liblinear` napísanú v jazyku C. Jadrová (angl. Kernel) funkcia je lineárna. Pre redukciu problému viactriednej klasifikácie na viaceré binárne klasifikačné problémy

používa stratégiu *one-vs-the-rest* (trénovanie jedného klasifikátora pre každú triedu) [20].

`Penalty parameter C` je regularizačný parameter. Čím je hodnota menšia, tým väčší minimálny okraj vieme nájsť, avšak i za cenu nezahrnutia niektorých okrajových hodnôt. Naopak čím je C väčšie, tým je možné klasifikovať trénovacie dáta presnejšie, ale môže sa stratiť generalizačná schopnosť klasifikátora [2].

`Max iterations` udáva maximálny počet iterácií behu.

`Tolerance error` predstavuje zastavovacie kritérium pre beh algoritmu – presnosť.

`Random state` opäť slúži k inicializácii generátora náhodných čísel pre replikáciu výsledkov.

5.3 Užívateľské rozhranie

Súčasťou aplikácie je grafické užívateľské rozhranie umožňujúce jej pohodlné a jednoduché používanie. Spomedzi niekoľkých dostupných nástrojov je pre jeho realizáciu zvolené *Qt*. Qt je voľne dostupná multiplatformová sada nástrojov pre tvorbu grafických užívateľských rozhraní, originálne dostupná pre jazyk C++. Pre použitie v jazyku Python slúži knižnica *PyQt5*, ktorá poskytuje prepojenie s Qt verzie 5.

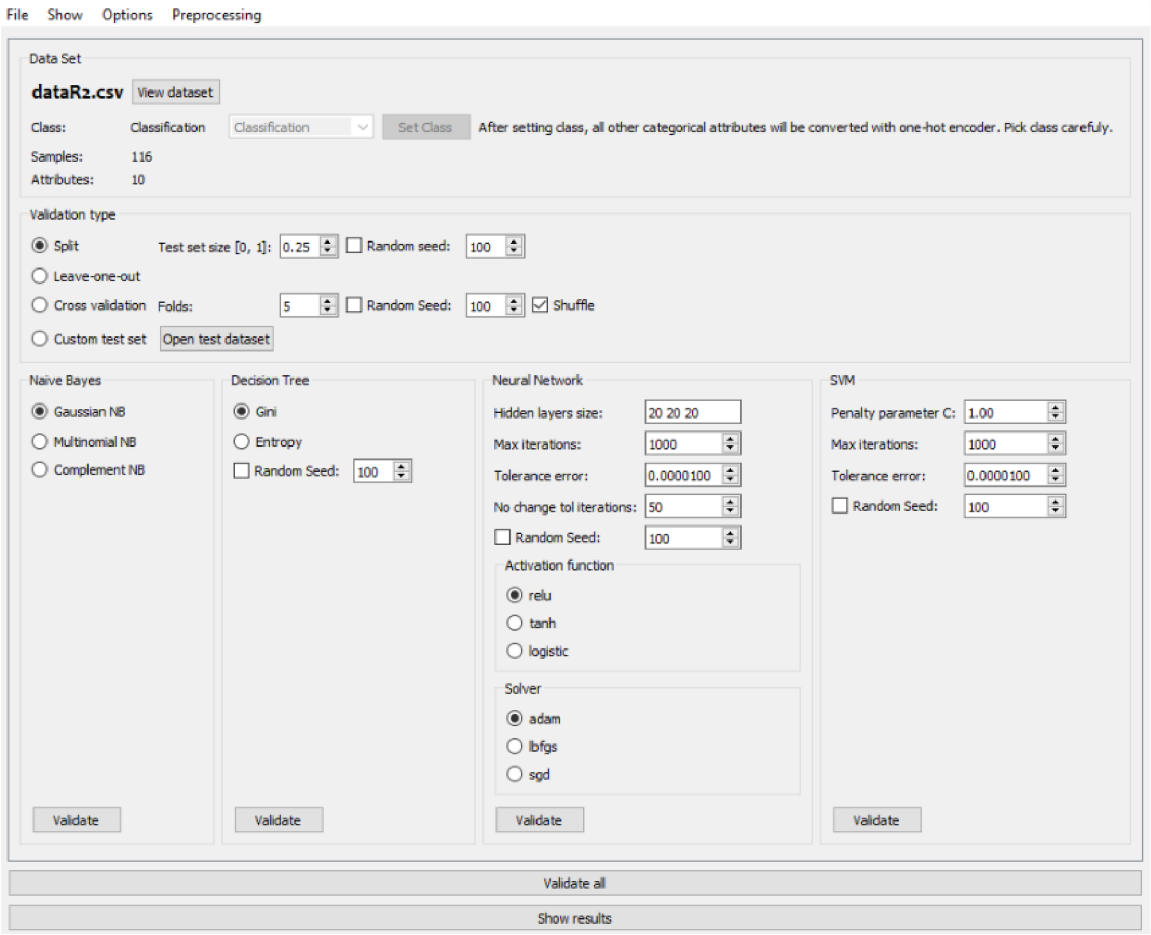
Vzhľad rozhrania je možné vytvárať a editovať vizuálne pomocou aplikácie *QtDesigner*, ktorá je súčasťou PyQt5. Výstupom dizajnéra je XML súbor s príponou „.ui“ obsahujúci vytvorený grafický návrh. PyQt5 poskytuje nástroj `pyuic` s konzolovým rozhraním, ktorý umožňuje prevod „.ui“ súboru na Python modul. Modul následne vytvára navrhnuté grafické užívateľské rozhranie [13].

O prepojenie užívateľského rozhrania s funkciami na pozadí sa stará modul `UIController`. Obsahuje triedy potrebné k inicializácii všetkých zobrazovaných okien. Spravuje aktiváciu a deaktiváciu vizuálnych častí rozhrania a kontroluje správnosť zadaných hodnôt parametrov. Na základe jednotlivých nastavení má za úlohu spustenie príslušných validácií a zozbieranie výsledkov.

Grafické rozhranie jednotlivých okien aplikácie bolo vytvorené nástrojom QtDesigner a následne prevedené na Python modul, výsledkom čoho sú moduly `ClassificationApp`, `DatasetTable`, `Results`. V nasledujúcich odstavcoch sú jednotlivé moduly popísané z pohľadu ich významu v rámci aplikácie.

ClassificationApp – Modul obsahuje kompletný vzhľad hlavného okna aplikácie. To je vizuálne rozdelené do troch častí, podľa jednotlivých fáz aplikácie. Prvú časť tvorí oblasť zahrňujúca nastavenie a informácie o dátovej sade. Druhá časť pokrýva výber a nastavenie spôsobov rozdelenia dát použitých na validáciu. Tretia časť je tvorená jednotlivými klasifikačnými metódami. Skutočnú podobu v systéme Windows 10 je možné vidieť na obrázku 5.1.

Pri spustení sú všetky prvky v okne deaktivované a nie je možné s nimi interagovať. Užívateľ je tak nasmerovaný do menu, kde pre zahájenie práce musí pridať dátovú sadu voľbou „File -> Open Dataset“. Menu obsahuje rýchly prístup k akciám ako zobrazenie dátovej sady, uzatvorenie aktuálnej sady, zobrazenie výsledkov a iné, ale i špeciálne akcie, ktoré je nutné nastaviť pred načítaním sady (predspracovanie) – doplnenie chýbajúcich hodnôt či spôsob pomenovania atribútov. Po pridaní sady sa aktivuje prvá oblasť. V nej zohráva kľúčovú úlohu voľba atribútu predstavujúceho klasifikačnú triedu. Tú je možné nastaviť výberom atribútov z ponuky a potvrdením



Obr. 5.1: GUI hlavného okna aplikácie

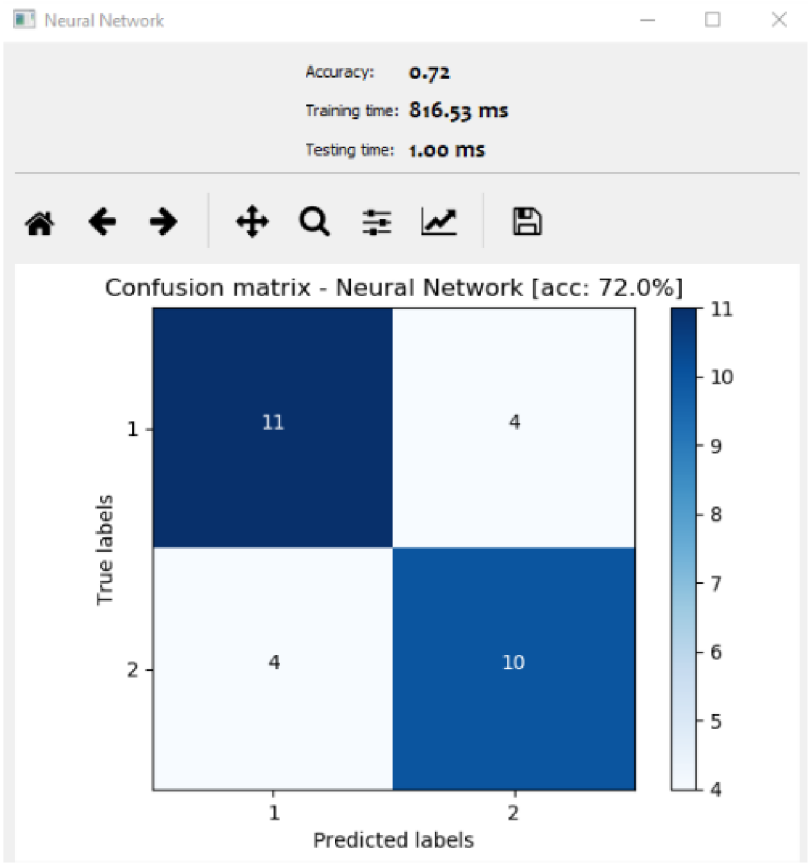
tlačidlom „Set Class“. Následne sa sprístupní i validačná časť aplikácie a jednotlivé klasifikačné metódy.

DatasetTable – Načítanú dátovú sadu je možné zobraziť vo forme tabuľky (tlačidlo „View dataset“). Je tak možné napríklad skontrolovať, či došlo k správnej detekcii hlavičky a zistiť vygenerované názvy atribútov. Po nastavení triedy je možné zobrazením dátovej sady overiť výsledok binarizácie kategorických atribútov (pokiaľ k nej došlo). Tabuľka sa otvára v novom okne, ktorého vzhľad je definovaný v module DatasetTable a je zobrazené na obrázku 5.2.

Results – Modul obsahuje vzhľad posledného samostatného okna, ktoré zabezpečuje zobrazenie výsledkov. K jeho otvoreniu dochádza po úspešnej validácií metód (tlačidlo „Validate“) alebo je možné zobraziť posledné získané výsledky tlačidlom „Show results“. Predchádzajúce výsledky sú po otvorení novej dátovej sady zmazané. Okno je možné vidieť na obrázku 5.3. Samotná matica zámen je vykreslená knižnicou matplotlib a je zasadená do prostredia aplikácie spolu s ovládacími prvkami. Ovládacie prvky umožňujú (na obrázku ikony zľava doprava) nastaviť východzie zobrazenie, vracať a opakovať posledné zmeny, pohybovať so zobrazovacou plochou, približovať, meniť rozmery grafu, meniť celkový vzhľad grafu (napr. farby) alebo uložiť maticu do súboru.

Adiponectin	Age	BMI	Glucose	HOMA	Insulin	Leptin	MCP.1	Resistin
9.7024	48	23.5	70	0.46740866700000006	2.707	8.8071	417.11400000000003	7.99585
5.429285	83	20.69049454	92	0.706897333	3.115	8.8438	468.786	4.06405
22.43204	82	23.12467037	91	1.009651067	4.498	17.9393	554.697	9.27715
7.169560000000001	68	21.36752137	77	0.612724933	3.2260000000000004	9.8827	928.22	12.765999999999998
4.81924	86	21.11111111	92	0.8053864000000001	3.549	6.6994	773.92	10.57635
13.67975	49	22.85445769	92	0.7320869329999999	3.2260000000000004	6.8317	530.41	10.3176
5.5898650000000005	89	22.7	77	0.890787333	4.69	6.9639999999999995	1256.083	12.9361
13.251320000000002	76	23.8	118	1.8832013330000001	6.47	4.311	280.694	5.1042
10.358725	73	22.0	97	0.801543333	3.35	4.47	136.855	6.28445
11.57899	75	23.0	83	1.013839467	4.952	17.127	318.302	7.0913
13.11	34	21.47	78	0.6674356	3.469	14.57	354.6	6.92
26.72	29	23.01	82	1.145436133	5.662999999999999	35.59	174.8	4.58
23.67	25	22.86	82	0.8272706670000001	4.09	20.45	313.73	5.14
36.06	24	18.67	88	1.33	6.107	8.88	632.22	6.85
17.95	38	23.34	75	1.0696700000000001	5.782	15.26	165.02	9.35
20.32	44	20.76	86	1.6	7.553	14.09	63.61	7.64
38.04	47	22.03	84	0.59	2.8689999999999998	26.65	191.72	3.32
7.780255	61	32.03895937	85	3.790144333	18.077	30.7729	444.395	13.68392
5.462619999999999	64	34.529722799999995	95	1.0373936670000001	4.427	21.2117	252.449	6.70188

Obr. 5.2: GUI okna zobrazujúceho dátovú sadu



Obr. 5.3: GUI okna s výsledkami validácie

5.4 Inštalácia

Aplikácia vyžaduje na cieľovom systéme nainštalovaný Python 3.7, nástroj `pip3` a zdrojové súbory. Potom je možné nainštalovať všetky závislosti potrebné k spusteniu aplikácie príkazom `$ pip3 -r requirements.txt` v zložke so zdrojovými súbormi aplikácie. Aplikáciu je možné následne spustiť príkazom `$ python app.py`.

Jednoduchší spôsob distribúcie aplikácie poskytuje nástroj `Pyinstaller`. S jeho pomocou je vytvorený adresár so spustiteľným súborom `app`, resp. `app.exe`, obsahujúci všetky potrebné knižnice a závislosti. Aplikáciu je tak možné spustiť na ktoromkoľvek systéme bez nutnosti inštalovať Python a použité moduly. Tento súbor je dostupný pre systém Windows (app.exe) i pre systém Linux (app) v zložke `app/dist` na priloženom CD. Proces inštalácie bol testovaný na systémoch Windows 10 a Ubuntu 16.04.03.

Kapitola 6

Experimenty a analýza vybraných klasifikačných metód

Posledná kapitola sa venuje experimentovaniu s implementovanou aplikáciou, zvolenými sadami a klasifikačnými metódami. Na začiatku je predstavená hlavná myšlienka analýzy a dôležitosť správneho zamerania experimentov. Postupne sú popisované jednotlivé experimenty, ich cieľ, prípadné obmedzenia a výsledky. Časť experimentov sa zameriava na analýzu presnosti klasifikačných metód a časť na čas strávený učením a klasifikovaním. V závere kapitoly je uvedené zhrnutie významných vlastností klasifikátorov pozorovaných v experimentoch.

6.1 Myšlienka experimentov

Súčasťou tejto práce je analýza klasifikačných metód. Jedná sa hlavne o ich porovnanie z pohľadu presnosti klasifikácie a časovej náročnosti trénovania a testovania na rôznych dátových sadách. Experimenty majú za cieľ nájsť kľúčové aspekty, ktoré ovplyvňujú dané veličiny a detailne ich opísať. Výsledkom je zhodnotenie vhodnosti použitia jednotlivých metód pre rôzne druhy dátových sád.

Experimenty sa nezameriavajú na optimalizáciu konkrétnych metód pre získanie najlepších výsledkov pre každú zo sád. Je nutné ujasniť, že pokiaľ sa niektorá z metód prehlási za najrýchlejšiu, je tým myslený kontext aplikácie. Vzhľadom na množstvo existujúcich implementácií je samozrejmé, že môže existovať rýchlejšia implementácia (paralelná, využívajúca GPU) inej metódy. Istým spôsobom vlastne experimenty analyzujú metódy knižnice Scikit-learn.

Aby bol zrejmý všeobecný rozdiel medzi metódami, je nutné manipulovať hlavne s dátovými sadami, avšak vhodným spôsobom. Ak existuje sada, ktorej dáta nie sú normalizované a experiment má porovnať presnosť metód len všeobecne, upravenie dát pre niektorú špecifickú metódu by znehodnotilo výsledky. Nakoniec, i formát dát je dôležitým faktorom pri ich porovnávaní. Vo väčšine prípadov je možné dáta upraviť na mieru ktorémukoľvek klasifikátoru, čo by vyvrcholilo v ohromné množstvo kombinácií nutných k analyzovaniu a nakoniec by experimenty nemuseli dospieť k žiadnemu záveru. Tiež je nutné spomenúť, že i jednotlivé klasifikačné metódy je možné konfigurovať na mieru niektorej dátovej sade. V tejto práci sa však pokúsime vyhnúť i tomuto prístupu, nakoľko je nutné vyskúšať veľké množstvo sád. Cieľom nie je nájsť optimálne riešenie klasifikačného problému udaného dátovou sadou, ale nájsť a potvrdiť rozdiel vo vlastnostiach jednotlivých metód vo všeobecnom merítku.

6.2 Dôležitosť výberu spôsobu validácie

Výber spôsobu validácie má na výsledky klasifikácie obrovský dopad. Zároveň rozličnými spôsobmi rozdelenia dát je možné experimentovať s rôznymi vlastnosťami bez nutnosti zmeny dátovej sady.

Spôsobom holdout je možné sadu rozdeliť v ľubovoľnom pomere na trénovacie a testovacie dáta. Je tak možné sledovať závislosť presnosti klasifikácie na počte vzoriek v trénovacej sade. Opakovaním rozdelenia je možné v dôsledku náhodnosti výberu zistiť, že môže nastať prípad, kedy sa klasifikátor na základe vybraných dát nenaučí generalizovať daný problém. Výnimočne môže dôjsť napríklad k vytvoreniu trénovacej sady, ktorej všetky vzorky patria len do jednej triedy. Ide o najjednoduchšiu metódu validácie, pričom pre dosiahnutie vierohodných výsledkov je nutné vykonať niekoľko opakovaní.

Validácia leave-one-out môže byť časovo veľmi náročná pre rozsiahle sady. Uvažujme sadu s 1000 vzorkami. V jej prípade dochádza 1000-krát k učeniu na 999 vzorkách. Využité sú však všetky dáta a výsledky automaticky spriemerované. Vzhľadom na využitie skoro celej sady na trénovanie sa výsledok validácie blíži výkonnosti klasifikátora v reálnom nasadení. Spôsob sa hodí najmä pre menšie dátové sady.

Pre zníženie časovej náročnosti je možné použiť krížovú validáciu, ktorá predstavuje kompromis medzi spôsobmi holdout a leave-one-out. Narozdiel od metódy holdout nie sú celkové výsledky ovplyvnené vyčlenením časti dát. Avšak celkový odhad validácie je pesimistickejší než u metódy leave-one-out. Krížová validácia je ideálna pre rozsiahlejšie dátové sady.

Môžu nastať prípady, kedy chceme použiť celú trénovaciu sadu, a zároveň máme dostupnú i úplne inú sadu určenú na testovanie. Obe sady môžu mať napríklad rovnakú distribúciu tried alebo môžu obsahovať len konkrétne vzorky, ktoré majú byť otestované. V tom prípade je možné použiť vlastnú testovaciu sadu a na učenie bude použitých 100 % trénovacích dát.

6.3 Obecná presnosť metód

Prvý vykonaný experiment sa zameriava na overenie schopnosti metód poradiť si s ľubovoľnou dátovou sadou. Neberie sa ohľad na to, či sú dáta optimalizované pre konkrétnu metódu, teda či sú normalizované alebo štandardizované. Niektoré sady obsahujú čisto desatinné numerické dáta – Breast Cancer Coimbra, MAGIC Gamma Telescope, Occupancy Detection. Iné sady zase obsahujú len celočíselné dáta s malým rozsahom hodnôt – Breast Cancer Wisconsin, Haberman's Survival, Mammographic Masses, MONKs Problems, Poker Hand, Zoo; pričom niektoré sady obsahujú kategorické dáta reprezentované číslom. Sady Adult Salary a Arrhythmia obsahujú zmiešaný typ dát a v Adult Salary dochádza k binarizácii kategorických atribútov. Metriky jednotlivých dátových sád boli uvedené v tabuľke 4.1 v kapitole 4.

Validačné metódy sú volené podľa charakteru dát. Dátové sady, ktorých počet vzoriek nie je väčší ako 400, sú validované metódou leave-one-out. Pre väčšie sady je použitá krížová validácia s počtom rozdelení 5 (`Folds`). Pokiaľ je pre daný problém dostupná samostatná testovacia sada, je použitá tá, a trénovanie prebieha na celej trénovacej sade. Spôsob rozdelenia holdout je použitý len pre rýchle orientačné hľadanie vhodných parametrov klasifikačných metód. Ako u validačných metód, tak i u klasifikátorov je povolená náhodnosť, a preto sa výsledky presnosti medzi jednotlivými behmi alebo iteráciami môžu líšiť. Rozdiely sú však minimálne.

Problémom experimentu je, že nie je možné vyskúšať všetky možné kombinácie nastavení jednotlivých metód, a preto je nutné mať na pamäti, že výsledky sú len orientačné. Pre každú dátovú sadu boli najskôr vykonané rýchle pokusy (s využitím holdout rozdelenia bez náhodnosti), pri ktorých sa hľadalo nastavenie, ktoré speje k najlepšiemu výsledku. Najprv bola hľadaná najlepšia varianta daného klasifikátora (algoritmus, funkcie), až potom boli ladené jeho numerické parametre. V momente, kedy sa výsledná presnosť zmenou numerických parametrov výrazne nemení, sú posledné nastavenia vedúce k najlepšej presnosti prehlásené za „najvhodnejšie“. Snahou je i udržanie približne rovnakých hodnôt parametrov medzi sadami, aby sa znížil dopad na rozdielnosť výsledkov u experimentov s časovou náročnosťou. Nastavenia parametrov sú uvedené v prílohe A.

Postup pri vykonávaní experimentu so známymi parametrami metód je možné popísať nasledujúcimi krokmi:

1. Je vybraná dátová sada.
2. Pokiaľ má trénovacia sada viac ako 400 vzoriek, je zvolená krížová validácia, v opačnom prípade je použitá leave-one-out. Ak je dostupná, použije sa na testovanie testovacia sada.
3. Nastavené sú „najvhodnejšie“ parametre jednotlivých klasifikátorov a spustená je validácia.
4. Výsledky validácie sú zaznamenané a validácia je opakovaná 2 až 5-krát v závislosti od veľkosti dátovej sady.
5. Následne je najlepší z výsledkov zaznačený do tabuľky 6.3.
6. Postup je opakovaný pre zvyšné dátové sady.

Dataset/Metóda	**Jednoduchý Bayes**	**Rozhodovací Strom**	**Neurónová sieť**	**SVM**
Adult Salary	79.52	81.77	80.83	80.00
Arrhythmia	16.6	66.58	62.18	59.95
Breast Cancer Coimbra	63.88	75.83	72.36	74.96
Breast Cancer Wisconsin	96.28	94.71	95.56	96.85
HabermansSurvival	75.82	66.63	76.13	74.52
MAGIC Gamma Telescope	72.72	82.2	86.11	77.45
Mammographic Mass	80.33	78.15	82.62	82.31
MONKs Problems	67.56	93.52	100	66.67
Occupancy Detection	97.74	99.26	98.87	98.04
Poker Hand	50.12	51.00	99.44	50.15
Zoo	96.04	96.00	97.09	96.00
Priemerná presnosť	**72.42**	**80.51**	**86.47**	**77.9**

Tabuľka 6.1: Presnosť klasifikačných metód naprieč dátovými sadami (%)

Výsledky experimentu popisujú všeobecnú presnosť jednotlivých metód. Z pohľadu dôležitosti sú jednotlivé triedy brané rovnocenne. Z tabuľky je možné vidieť, že vo viacerých prípadoch je dosahovaná presnosť jednotlivých metód podobná. Výkyvy je možné pozorovať hlavne u sád Arrhythmia a Poker Hand. Je možné predpokladať, že nízka presnosť

jednoduchého Bayesovho klasifikátora u sady Arrhythmia je spôsobená množstvom a redundaciou atribútov. Táto skutočnosť je skúmaná v experimente 6.7. Sada Poker Hand predstavuje zložitý problém, kde je nutné uvažovať vzájomné vzťahy medzi jednotlivými kartami a nezávisí na ich poradí v ruke. S touto sadou si výborne dokázala poradiť iba neurónová sieť. Ostatné metódy by potrebovali pokročilé predspracovanie dát, doplnenie atribútov označujúce zhodné karty v ruke a iné.

Priemerná presnosť predstavuje schopnosť jednotlivých klasifikačných metód reprezentovať rôzne druhy dát v rôznych formátoch. Najlepšie sa spomedzi vybraných metód umiestnila neurónová sieť. Jednoduchý Bayes ju prekonal u sady Breast Cancer Wisconsin, čo znamená, že nie vždy je možné jednoducho určiť najlepší klasifikátor. Vo všeobecnosti sa však zdá, že z pohľadu presnosti je neurónová sieť najlepšou prvou voľbou.

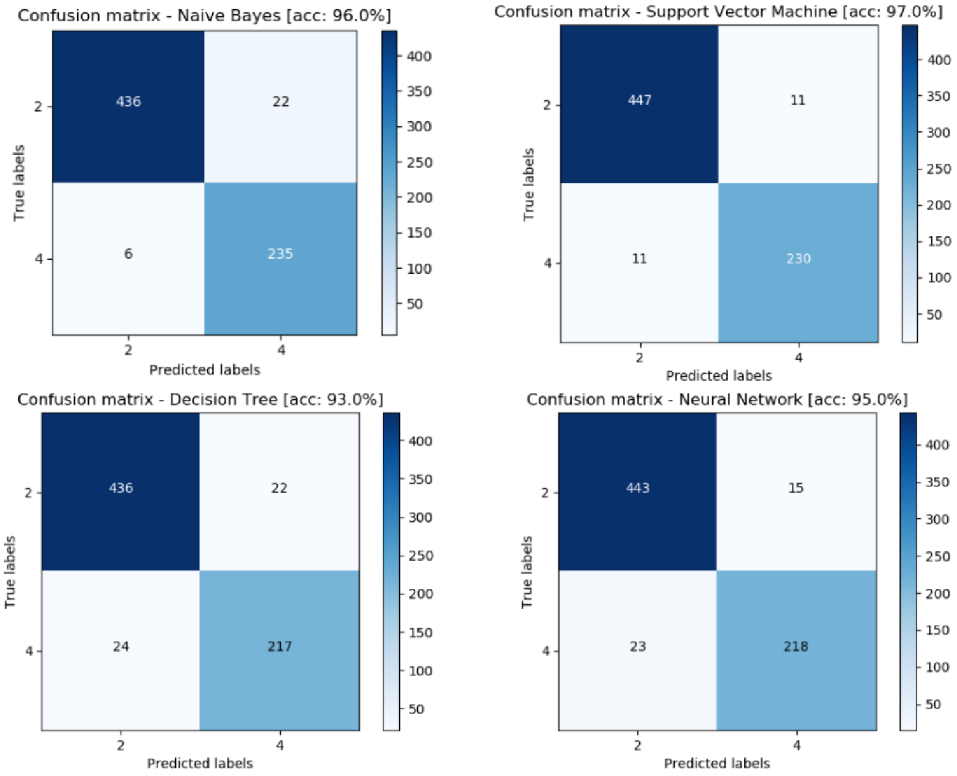
Z pohľadu konfigurácie je Bayes najjednoduchšou voľbou. Typ GaussianNB produkuje najlepšie výsledky na všetkých skúmaných dátových sadách, ComplementNB je tesne za ním. U rozhodovacieho stromu majú uplatnenie obe dostupné varianty výberu atribútov, pričom výsledky sú veľmi podobné. Neurónová sieť poskytuje najväčšiu škálu nastavení. Na základe použitých nastavení je možné povedať, že vo všetkých dátových sadách bolo dosiahnutých najlepších výsledkov s použitím algoritmu Adam. Logistická aktivačná funkcia sa vo vybraných sadách tiež neuplatnila. Čo sa týka ostatných parametrov, hodnoty boli rôzne pokusne volené. U metódy SVM je najdôležitejším parametrom počet iterácii. Metóda potrebuje pomerne veľký počet iterácií, aby konvergovala. Regulačná konštanta bola u väčšiny sád ponechaná na počiatočnej hodnote 1.0, nakoľko jej zmena nemala výrazný vplyv na výsledky. Náhodnosť u jednotlivých metód nehrá významnú rolu a výslednú presnosť ovplyvňuje približne o ±2-3 %.

6.4 Nevyvážené sady

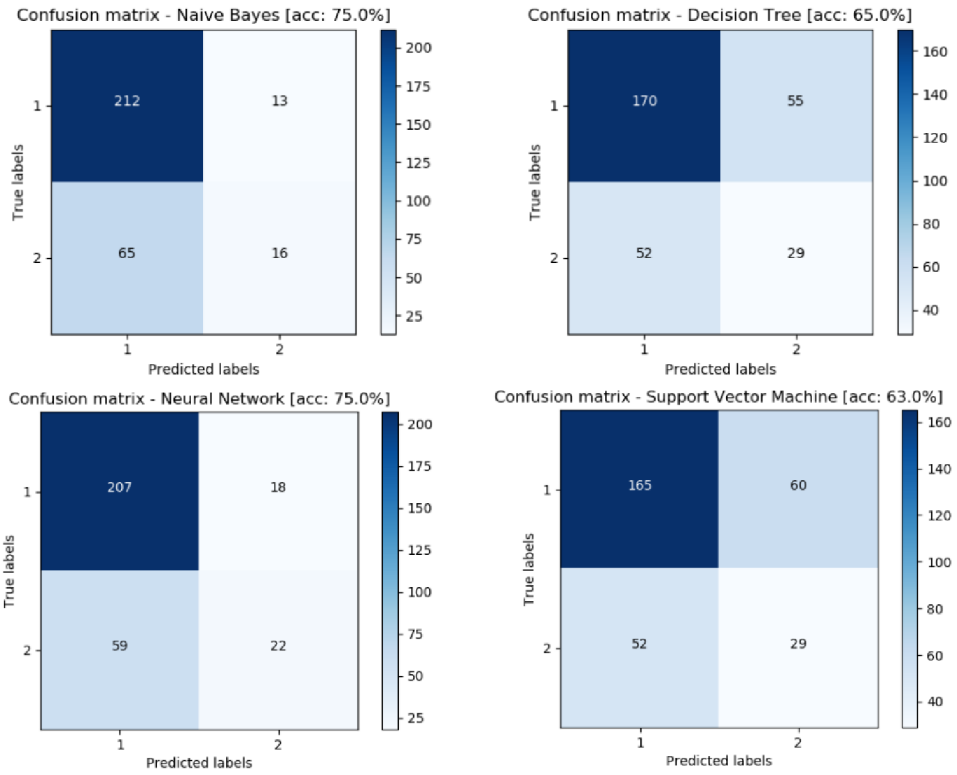
Cieľom tohto experimentu je zistiť správanie klasifikátorov na dátach s nevyváženým rozložením tried. Vhodné dáta pre experiment poskytujú sady Haberman's Survival a Breast Cancer Wisconsin. Pre validáciu bola použitá krížová validácia s piatimi rozdeleniami a nastavenie metód bolo rovnaké ako v experimente 6.3. Na obrázku 6.1 je možné vidieť chybové matice jednotlivých klasifikátorov pre sadu Breast Cancer Wisconsin. Rozdiely medzi jednotlivými metódami nie sú veľmi odlišné a úspešne bola klasifikovaná i minoritná trieda „4". Naopak je tomu u sady Haberman's Survival. Všetky metódy vykazovali značne nedostatočnú schopnosť naučiť sa správne klasifikovať vzorky patriace do minoritnej triedy „2", ako je možné vidieť na obrázku 6.2.

Výsledky sú medzi metódami opäť podobné a nie je možné pozorovať výrazne lepšiu metódu. Z výsledkov je možné vyvodiť záver, že nevyváženosť dát môže aj nemusí mať vplyv na výslednú presnosť, pričom voľba metódy nemá výrazný vplyv na elimináciu chyby. V prípade nevyváženosti dát a nízkej presnosti je preto nutné hľadať riešenie v úprave dát. Na časovú náročnosť nemá nevyvýženosť výrazný vplyv.

V závislosti od problému, ktorý dáta predstavujú, je možné buď navýšiť počet vzoriek menšej triedy (anglicky *oversampling*) alebo znížiť počet vzoriek väčšej triedy (anglicky *undersampling*) [6]. Úpravy je nutné robiť len na trénovacej sade (po rozdelení), inak je možné, že výsledky validácie budú podobné ako pred úpravou.



Obr. 6.1: Výsledky validácie sady Breast Cancer Wisconsin

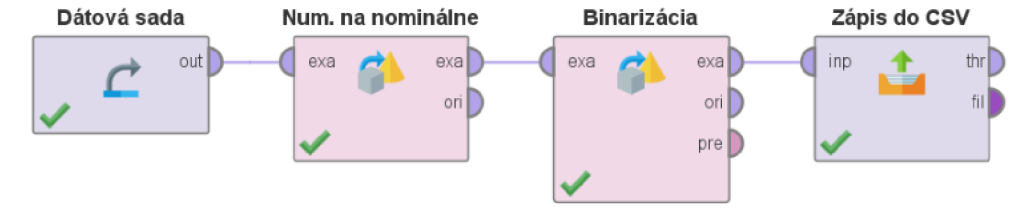


Obr. 6.2: Výsledky validácie sady Haberman's Survival

6.5 Binarizácia atribútov

Aplikácia automaticky prevádza kategorické atribúty reprezentované reťazcom na numerickú podobu prevodom nazvaným binarizácia. Ak však dátová sada obsahuje kategorické atribúty reprezentované číslom, k prevodu nedochádza. Klasifikačné algoritmy sa tak k týmto atribútom stavajú rovnako, ako keby boli numerické. Ako príklad môže poslúžiť atribút *Farba*, ktorý nadobúda troch hodnôt – *červená, zelená, modrá*. Pokiaľ tieto hodnoty zamením za čísla *1, 2 ,3*, vytvoríme medzi hodnotami usporiadanie. Klasifikačné algoritmy sa tak môžu k hodnotám správať odlišným spôsobom, môžu ich dodatočne spájať do intervalov, násobiť alebo vzájomne kombinovať. Klasifikátor je schopný sa dané dáta naučiť, no je možné, že z dôvodu nesprávnej interpretácie môže v niektorých prípadoch dochádzať k zníženiu výslednej presnosti.

Cieľom experimentu je zistiť, či postačuje len konverzia reťazcov na číslo (angl. label encoder) alebo je vhodnejšie použiť binarizáciu. Na pokusy sú vhodné sady MONK's Problems a Mammographic Masses. Obe sady obsahujú kategorické atribúty reprezentované číslom, pričom u MONK's Problems sú to všetky atribúty a u Mammographic Masses dva. Binarizácia konkrétnych atribútov v dátových sadách je vykonaná prostredníctvom programu *Rapid Miner*, ktorý poskytuje obrovskú sadu nástrojov z oblasti predspracovania dát, strojového učenia a dolovania znalostí. Prevod a vytvorenie novej sady je možné „naprogramovať" pomocou grafických blokov programu. Na obrázku 6.3 je zobrazený príslušný proces.



Obr. 6.3: Proces binarizácie dát v programe Rapid Miner

Nové modifikované sady boli použité pre trénovanie a testovanie klasifikačných metód s rovnakými nastaveniami ako v prvom experimente 6.3. V prípade sady Mammographic Masses bola výsledná presnosť takmer identická s presnosťou dosiahnutou bez konverzie atribútov. Presnosť MONK's Problems však klesla o 30–40 % u všetkých klasifikátorov. Problém je spôsobený tým, že binarizácia vytvorí nové atribúty pre každú jednu hodnotu pôvodného atribútu a nové atribúty sa stanú lineárne závislými. Niektoré metódy preto vyžadujú, aby bol jeden nový atribút pri binarizácii vypustený. Každý nový atribút, ktorý odpovedá aktuálnej nominálnej hodnote, nadobúda hodnotu 1 a všetky ostatné hodnotu 0. Ak však nominálna hodnota odpovedá vypustenému atribútu, všetky ostatné nové atribúty nadobúdajú hodnotu -1 [14]. Rapid Miner ponúka tento spôsob prevodu pod názvom *Effect Coding*.

Experiment bol zopakovaný pre obe sady, tentoraz s použitím prevodu typu Effect Coding. V tabuľke 6.2 sú uvedené výsledné presnosti pre obe dátové sady pred a po vykonaní prevodu kategorických atribútov na numerické metódou Effect Coding. U sady MONK's Problems je vidieť výrazného zlepšenia, pritom jediné čo sa zmenilo je forma reprezentácie

dát. Okrem zlepšenia presnosti došlo i k viac ako dvojnásobnému zrýchleniu fázy trénovania u neurónovej siete a SVM.

Dataset/Metóda	**Jednoduchý Bayes**		**Rozhodovací strom**		**Neurónová sieť**		**SVM**	
	Pred	**Po**	**Pred**	**Po**	**Pred**	**Po**	**Pred**	**Po**
Mammographic Masses	80.33	77.00	78.15	78.20	82.62	82.73	82.31	82.83
MONK's Problems	67.56	74.04	93.52	96.30	100	100	66.67	71.53

Tabuľka 6.2: Presnosť pred a po Effect Coding prevode kategorických atribútov (%)

6.6 Obecná časová náročnosť metód

Experiment na overenie všeobecnej časovej zložitosti bol súčasťou experimentu na overenie obecnej presnosti 6.3. Krížová validácia alebo leave-one-out sú výhodné i pre meranie času, nakoľko je vykonaných niekoľko nezávislých behov a aplikáciou namerané časy v každom behu sú spriemerované do výslednej časovej náročnosti. V tabuľke 6.3 sú uvedené časy trvania učenia a testovania u jednotlivých dátových sád. Nastavenie parametrov je uvedené v prílohe A a je zhodné s nastaveniami pre experiment 6.3.

Dataset/ Metóda	**Jednoduchý Bayes**		**Rozhodovací strom**		**Neurónová sieť**		**SVM**	
	Učenie	Test	Učenie	Test	Učenie	Test	Učenie	Test
Breast Cancer Wisconsin	2.0	0.8	1.8	1.0	3103.4	2.0	106.7	1.0
MONKs Problems	2.0	1.0	2.0	1.0	6260.2	3.0	18.0	1.0
Adult Salary	197.1	54.0	543.5	13.2	31382.8	24.0	48827.0	13.4
Arrhythmia	5.8	5.0	56.2	1.0	6349.8	2.2	14215.6	2.0
Breast Cancer Coimbra	1.6	1.6	2.0	1.0	4208.8	1.3	57766.6	1.0
Habermans Survival	1.6	0.8	1.6	1.0	2535.4	1.0	5351.9	1.2
MAGIC Gamma Telescope	27.0	2.0	1102.2	3.2	102335.3	10.4	220305.4	1.6
Mammographic Mass	1.8	0.8	2.8	0.6	3855.0	1.2	5672.2	1.2
Occupancy Detection	3.4	1.2	16.4	0.8	43073.0	8.0	35849.5	1.0
Poker Hand	20.0	3392.0	185.9	455.7	120540.2	1599.1	44094.8	328.8
Zoo	2.0	1.0	1.0	1.0	2583.7	0.6	7.2	0.6
Priemerný čas	**24.0**	**314.6**	**174.1**	**43.6**	**29657.1**	**150.2**	**39292.3**	**32.1**

Tabuľka 6.3: Časová náročnosť klasifikačných metód naprieč dátovými sadami (ms)

Z jednotlivých časov je pomerne zložité určiť súvislosti medzi dátami a jednotlivými metódami. Na základe merania môžeme prehlásiť, že pokiaľ chceme dosiahnuť relatívne presné výsledky a zároveň je pre nás kľúčová časová náročnosť trénovania, tak neurónové siete a SVM sú vo väčšine prípadov zlá voľba. Jednoduchý Bayes a SVM tvoria zaujímavý kontrast; Bayes vyniká v učení a SVM v samotnej klasifikácii. Najlepšou priemernou voľbou je v tomto prípade rozhodovací strom.

Priame porovnanie časov je však len orientačné. Porovnávanie SVM a neurónovej siete medzi sebou je komplikovanejšie. V oboch prípadoch vystupujú užívateľom definované parametre (počet neurónov, počet iterácií, presnosť), ktoré dokážu mnohonásobne predĺžiť jednotlivé fázy. Vzhľadom na teoreticky nekonečné množstvo kombinácií hodnôt parametrov je ich optimalizácia náročný proces, ktorý je závislý na konkrétnom klasifikačnom probléme.

Vzhľadom na spomenuté problémy je preto zaujímavejšie overiť závislosti medzi jednotlivými metódami a dátami. V nasledujúcich kapitolách sú vykonané experimenty, ktorých cieľom je zistiť, do akej miery je časová náročnosť klasifikátorov ovplyvnená počtom vzoriek a do akej napríklad nevhodným formátom dát.

6.7 Vplyv počtu atribútov

Experiment využíva sadu Arrhythmia, ktorá pozostáva z 279 atribútov (bez triedy). Všetky hodnoty sú reprezentované číslom. Postupným odstraňovaním atribútov je skúmaná zmena rýchlosti trénovania a presnosti. Na odstránenie atribútov je použitý program *Rapid Miner*. Ten ponúka modul na odstránenie nepotrebných atribútov.

Pri prvom pokuse sú odstránené atribúty tvorené aspoň z 95 % rovnakými hodnotami a aspoň z 95 % unikátnymi hodnotami. Tým je počet atribútov zredukovaný z 279 na 259. Trénovanie a testovanie je vykonané opäť s rovnakými nastaveniami ako v prvom experimente 6.3. Výsledná presnosť a časová náročnosť sa vôbec nezmenili.

V druhom pokuse sú postupne ručne odstraňované atribúty, až do finálneho počtu 149 atribútov. Jediný zrejmý pokles časovej náročnosti je u neurónovej siete, ako je možné vidieť v tabuľke 6.4. Ostatné metódy vykazovali žiadne alebo len nepatrné zníženie doby trénovania. Metóda SVM sa na základe výsledkov experimentov javí ako nezávislá na počte atribútov. Presnosť metód sa výrazne nezmenila.

Počet atribútov	**280**	**259**	**184**	**149**
Jednoduchý Bayes	6.9	6.4	6.2	3.9
Rozhodovací strom	63.5	73.6	57.3	52.9
Neurónová sieť	8030	7236	6700	5200
SVM	13982	14350	15150	14052

Tabuľka 6.4: Časová náročnosť trénovania v závislosti na počte atribútov (ms)

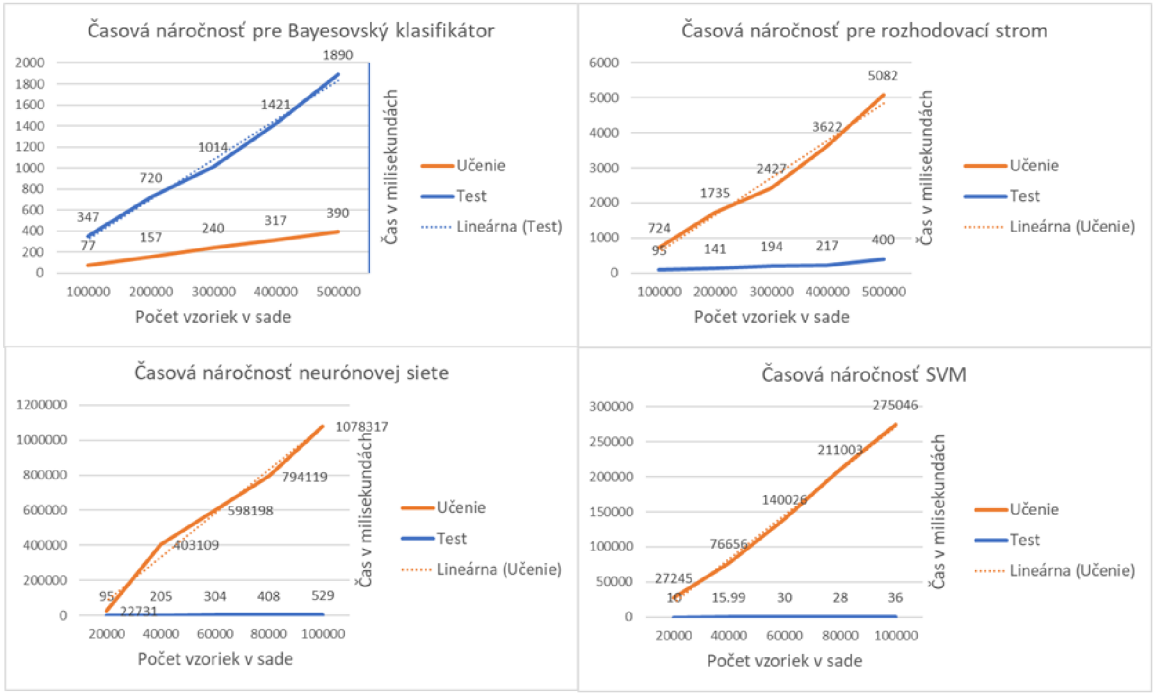
Z experimentu je možné vyvodiť, že počet atribútov má vplyv hlavne na časovú náročnosť trénovania neurónovej siete. Čo sa týka presnosti, pre danú sadu nepomohlo ani odobratie niekoľkých atribútov. Očakávané zlepšenie presnosti (z kapitoly 6.3) pre Bayesovský klasifikátor teda nenastalo, a bola by pravdepodobne potrebná znalosť problematiky a množstvo experimentov s úpravou dát do vhodnej podoby, aby k nemu došlo.

6.8 Vplyv počtu vzoriek

Najväčšou dostupnou sadou je sada Poker Hand s počtom testovacích vzoriek 1000000. Táto sada je pre potreby merania použitá ako trénovacia i testovacia. V tomto experimente je ignorovaná výsledná presnosť a zameranie je len na trvanie jednotlivých fáz. Uplatnenie tu nachádza validačná metóda holdout. Postupne je ňou možné „ukrajovať“ zo vstupnej sady a tvoriť tak menšiu/väčšiu trénovaciu alebo testovaciu sadu. Pritom je možné sledovať nárast či pokles časovej náročnosti v závislosti od zmeny počtu dát.

Prvotné testy odhalili obrovskú časovú náročnosť pri použití celého miliónu vzoriek. Preto je nakoniec použitých maximálne 500000 vzoriek, u neurónovej siete a SVM len 100000. Z nich je postupne použitých 100 %, 80 %, ..., 20 % na trénovanie a zvyšok na testovanie. Každé meranie bolo vykonané 5-krát a výsledky spriemerované. Neurónová sieť bola nastavená na nižší počet neurónov (30) kvôli zníženiu časovej náročnosti experimentu a max počet iterácií rovnako ako u SVM bol obmedzení na 1000.

Jednotlivé výsledky sú zobrazené formou grafu. Všetky grafy je možné vidieť na obrázku 6.4. Z grafov je možné pozorovať, že doba trénovania i doba testovania stúpa približne lineárne s počtom vzoriek. Taktiež je možné vidieť výrazný rozdiel v časoch Bayesovkého klasifikátora a rozhodovacieho stromu oproti neurónovej sieti a SVM, a to i na výrazne menšom počte vzoriek. Jednoduchý Bayes exceluje v rýchlosti trénovania za cenu pomal-



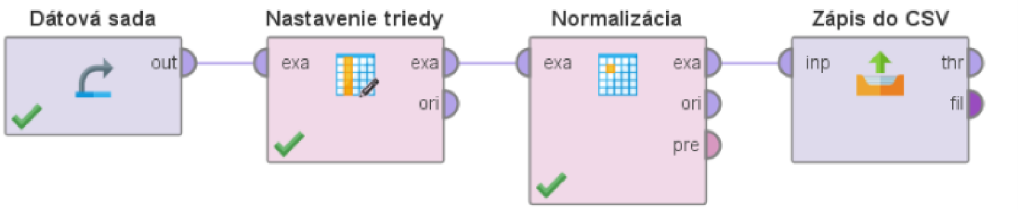
Obr. 6.4: Výsledky testu časovej náročnosti jednotlivých metód (ms)

šieho klasifikovania. V rýchlosti klasifikácie je rýchlejší rozhodovací strom a SVM. Nakoľko bola neurónová sieť z pohľadu presnosti najlepšou voľbou, z pohľadu časovej náročnosti sa jedná o jednu z najpomalších metód. Doba trénovania SVM je pri menšom počte vzoriek vyššia, ale s pribúdajúcimi vzorkami rastie pomalšie ako u neurónovej siete. Samozrejme čím je zvolená zložitejšia topológia siete, tým bude jej časová náročnosť opäť vyššia.

6.9 Normalizácia hodnôt

Posledný z radu experimentov, má za cieľ ukázať dopad normalizácie hodnôt na výsledky jednotlivých klasifikačných metód. Spomedzi sád sú vybrané sady s numerickými atribútmi, ktorých intervaly hodnôt boli výrazne odlišné – Breast Cancer Coimbra, MAGIC Gamma Telescope a Occupancy Detection.

K normalizácii dát je znovu použitý program *Rapid Miner*. Na obrázku 6.5 je znázornená ukážka procesu v programe Rapid Miner, ktorá má na starosť normalizáciu dát do intervalu $\langle -1.0, 1.0 \rangle$ a zápis výslednej sady do súboru CSV, ktorý je možné ďalej použiť pre potreby experimentu [15].



Obr. 6.5: Proces normalizácie dát v programe Rapid Miner

Normalizované dáta sú následne využité pre naučenie všetkých klasifikátorov. Sú použité rovnaké nastavenia ako v prípade prvého experimentu 6.3 (uvedené v prílohe A). V tabuľke 6.5 je uvedená časová náročnosť trénovacej fázy jednotlivých metód pred a po aplikovaní normalizácie na danú dátovú sadu. Jednoznačne je vidieť, že doba trénovania SVM klasifikátora sa drasticky znížila po normalizovaní dát. U neurónovej siete je tiež v dvoch prípadoch značný pokles, a preto je možné prehlásiť, že použitie normalizácie dát u oboch klasifikátorov je rozhodne žiaduce. Naopak, na Bayesovský klasifikátor ani na rozhodovací strom nemá žiadny citeľný pozitívny vplyv.

Dataset/Metóda	**Jednoduchý Bayes**		**Rozhodovací srom**		**Neurónová sieť**		**SVM**	
	Pred	**Po**	**Pred**	**Po**	**Pred**	**Po**	**Pred**	**Po**
Breast Cancer Coimbra	1.6	1.8	2.0	2.0	4208	992	57766	2.0
MAGIC Gamma Telescope	27	29	1102	646	102335	223415	220305	371
Occupancy Detection	3.7	7.0	16	19	43073	20441	35849	29

Tabuľka 6.5: Trvanie trénovania pred a po normalizácii dát (ms)

Vo všetkých troch prípadoch bola monitorovaná i výsledná presnosť, ktorej zmena sa pohybovala v rozpätí $\pm 2\,\%$. Rozdiel bol medzi sadami nepravidelný, a preto nie je možné jednoznačne určiť pozitívny alebo negatívny vplyv normalizácie na presnosť niektorej z metód z daného experimentu.

6.10 Zhrnutie výsledkov

Cieľom vykonaných experimentov bolo objavenie či overenie určitých vlastností jednotlivých klasifikačných metód a ich súvislosť s predloženými dátami. Skúmané boli faktory ovplyvňujúce presnosť a časovú náročnosť. Výsledky experimentov sú zhrnuté v nasledujúcich odstavcoch v podobe opisu pozorovaných vlastností metód.

Jednoduchý Bayes – Metóda s najjednoduchším princípom klasifikácie a v základnej verzii bez nutnosti nastavovania parametrov. Nevhodná pre dátové sady, ktorých atribúty sú vzájomne závislé. V praxi poskytuje prekvapivo dobré výsledky porovnateľné s ostatnými metódami na väčšine sád. Dokáže pracovať s numerickými i kategorickými dátami a dobre zvláda i klasifikáciu do viacerých tried. Vo vykonaných experimentoch sa najlepšie umiestnila varianta GaussianNB, ktorá je vhodná pre numerické dátové sady. Spomedzi vybraných metód je Bayes bezkonkurenčne najrýchlejší v procese učenia. Naopak, testovacia fáza je najpomalšou, no i tak sa pohybuje len v rádoch stoviek milisekúnd na stotisíc vzoriek. Metóda nevyžaduje normalizáciu dát a nevyužíva generátor náhodných čísel, čiže výsledky učenia na rovnakých dátach sú plne reprodukovateľné.

Rozhodovací strom – Z pohľadu konfigurácie je to jednoduchý klasifikátor, kde obe varianty výberu atribútov - Gini i Entropy produkujú dobré výsledky. Proces učenia i testovania je rýchly i pri veľkom počte vzoriek. Pracuje s numerickými a kategorickými dátami. Nevyžaduje žiadne špeciálne predspracovanie dát, no menšie množstvo atribútov alebo tried je výhodou. Tiež nie je veľmi citlivý na chýbajúce alebo vychýlené hodnoty.

Neurónová sieť – Zo všetkých metód poskytuje najbohatšie možnosti konfigurácie. Zdĺhavé môže byť nájdenie vhodnej kombinácie parametrov. Teoreticky postačujú len dve skryté vrstvy s vhodným počtom neurónov, pre ktorúkoľvek sadu. Rovnako je náročný i proces trénovania, ktorý časovo mnohonásobne prekračuje učenie Bayesovského klasifikátora alebo rozhodovacieho stromu. Odmenou je však veľmi vysoká presnosť (spomedzi metód najvyššia) a spoľahlivosť naprieč rôznymi druhmi sád. Urýchlenie je možné v niektorých prípadoch dosiahnuť normalizáciou dát. Kategorické atribúty je nutné konvertovať na čísla alebo ideálne previesť binarizáciou. Veľký počet atribútov ani ich vzájomná závislosť nepredstavuje problém.

SVM – Metóda je v mnohých ohľadoch podobná neurónovej sieti. Na jednoduchších sadách dosahuje podobnej (dobrej) presnosti. Časová náročnosť učenia je výrazne závislá na formáte dát. Normalizácia numerických dát do intervalu $\langle -1, 1 \rangle$ dokáže trénovaciu fázu urýchliť v niektorých prípadoch viac než tisícnásobne a metóda dokáže predbehnúť i rozhodovací strom. V opačnom prípade ide často o najpomalšiu z metód. Rýchlosť trénovania nie je závislá na počte atribútov. Proces testovania je bez ohľadu na formát dát najrýchlejší spomedzi všetkých metód. Konfigurácia je jednoduchá, dôležitejšia môže byť voľba maximálneho počtu iterácii tak, aby metóda konvergovala.

Všetky metódy zvládajú klasifikáciu do viacerých tried. Použitá implementácia (a často i iné) vyžaduje, aby dáta boli numerické. Prostá konverzia kategorických hodnôt na numerickú reprezentáciu je vo väčšine prípadov nedostatočná a lepšej presnosti je možné dosiahnuť binarizáciou všetkých kategorických atribútov, a to najlepšie pomocou metódy Effect Coding, ktorá naruší závislosť nových atribútov.

Vykonané experimenty sa pokúsili nájsť odpovede na najčastejšie otázky, ktoré sa môžu pri riešení klasifikačného problému vyskytnúť. Výber vhodnej klasifikačnej metódy je najmä závislý na charaktere dát a probléme, ktorý reprezentujú; ideálne je však dobré skúsiť vždy všetky dostupné metódy. Nie je možné jednoznačne povedať, ktorá metóda je najlepšia.

Kapitola 7

Záver

Cieľom tejto diplomovej práce je analýza a porovnanie vybraných klasifikačných metód a implementácia aplikácie, ktorá ponúka základnú konfiguráciu, trénovanie, testovanie a validáciu metód, čím umožňuje experimentálne získanie výsledkov použitých k porovnaniu a vyhodnoteniu vlastností metód.

Prácu je možné rozdeliť do niekoľkých vzájomne prepojených častí, ktoré tvoria teoretický, praktický i experimentálny rozbor témy. Prvá časť bola venovaná všeobecným teoretickým poznatkom z oblasti spracovania rozsiahlych dát, ich typovej rôznorodosti a metódam transformácie do podoby vhodnej pre ďalšie spracovanie. Teoretická časť následne pokračovala predstavením princípu klasifikácie a detailnejšie boli opísané spôsoby validácie klasifikačných metód, ako i samotné vybrané metódy – jednoduchý Bayesovský klasifikátor, rozhodovací strom, neurónová sieť, SVM. Druhá časť práce zahrňovala výber a rozbor dátových sád a návrh a implementáciu aplikácie. Popisovala postupne všetky požiadavky na funkcionalitu a detailnejšie sa venovala prostriedkom a mechanizmom, ktoré tvorili výslednú aplikáciu. Posledná časť práce bola venovaná experimentom s klasifikačnými metódami a dátovými sadmi.

Programový výstup práce tvorí desktopová aplikácia spustiteľná v systémoch Windows i Linux. Poskytuje možnosť správneho načítania dátovej sady, voľbu triedy pre potreby klasifikácie, výber validačnej metódy a samozrejme samotnú validáciu všetkých štyroch klasifikačných metód. Jednotlivé metódy je možné konfigurovať. Výsledná presnosť je prezentovaná percentuálne a vo forme matice zámen. Čas trénovania a čas testovania je meraný zvlášť a to v milisekundách.

Samotná analýza klasifikačných metód prebiehala na základe výsledkov niekoľkých experimentov. Tie reprezentovali rôzne scenáre, na základe ktorých bolo možné sledovať správanie jednotlivých metód. Testovaná bola všeobecná presnosť a časová náročnosť metód naprieč všetkými dátovými sadami. Okrem toho však boli vykonané i experimenty zamerané na konkrétnejšie faktory, ako vplyv normalizácie hodnôt, nutnosť binarizácie dát, prístup metód k nevyváženým dátam alebo závislosť času či presnosti na počte vzoriek či atribútov v sadách.

Podrobenie klasifikačných metód rôznym experimentom odhalilo niektoré ich silné i slabé stránky. Jednoduchý Bayes je v priemere jednoznačne najrýchlejšia metóda, ktorá nevyžaduje normalizáciu dát. Rozhodovací strom dosahuje o niečo vyššej presnosti na úkor vyššej časovej náročnosti, no javí sa ako zlatá stredná cesta medzi vybranými metódami. Neurónová sieť je naopak veľmi pomalá, no správnou konfiguráciou je možné dosiahnuť najvyššej presnosti spomedzi metód. Zrýchlenie je možné dosiahnuť normalizáciou. SVM je vlastnos-

ťami veľmi podobná neurónovej sieti. Normalizáciou dát sa z nej stáva metóda rýchlejšia než rozhodovací strom.

Budúce rozšírenia práce by mohli spočívať v doplnení nových dátových sád iných typov, rozšírení aplikácie o ďalšie klasifikačné metódy alebo navrhnutí nových experimentov skúmajúcich ďalšie vlastnosti metód. Aplikáciu by bolo tiež možné smerovať viac do oblasti spracovania dát a doplniť komplexnejšie metódy manipulácie s dátami – predspracovanie, transformácia, zbieranie štatistík.

Za vlastný prínos je možné označiť vytvorenie desktopovej aplikácie, výber dátových sád, navrhnutie vhodných experimentov a získanie výsledkov, a to všetko na základe nadobudnutých znalostí z komplexnej oblasti spracovania dát, klasifikácie a vývoja aplikácií.

Literatúra

[1] Aggarwal, C. C.: *Data Classification: Algorithms and Applications*. Taylor & Francis Group, 2015, ISBN 978-1-4665-8674-1, 671 s.

[2] Aggarwal, C. C.: *Data Mining: The Textbook*. Springer International Publishing Switzerland, 2015, ISBN 978-3-319-14141-1, 734 s.

[3] Asuncion, A.; Newman, D.: *UCI Machine Learning Repository*. [Online; navštívené 15.01.2019].
URL https://archive.ics.uci.edu/ml/datasets.php

[4] Brédová, L.; Kováčová, A.: *Naivný Bayesovský klasifikátor*. 2011–2012, [Online; navštívené 15.02.2019].
URL http://oz.koncz.sk/attachments/article/173/oz_bredova_kovacova.pdf

[5] Diederik P. Kingma and Jimmy Ba : *Adam: A Method for Stochastic Optimization*. 2015, [Online; navštívené 17.05.2019].
URL https://arxiv.org/abs/1412.6980

[6] Han, J.; Kamber, M.; Pei, J.: *Data mining: Concepts and techniques*. Morgan Kaufmann, tretie vydanie, 2012, ISBN 978-0-12-381479-1, 703 s.

[7] Hunter, J.; Dale, D.; Firing, E.; aj.: *Matplotlib*. [Online; navštívené 16.04.2019].
URL https://matplotlib.org/

[8] Janousek, V.: *Prednáška: Machine Learning and Neural Networks*. 2007, [Online; navštívené 10.01.2019].
URL
https://www.fit.vutbr.cz/study/courses/SIN/private/lectures/SIN-NN.pdf

[9] Matuška, M.: *Rozhodovacie stromy*. 2004, [Online; navštívené 1.05.2019].
URL http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0405/matuska/zs.htm

[10] Pandas: *Python Data Analysis Library*. [Online; navštívené 15.04.2019].
URL https://pandas.pydata.org/index.html

[11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: *Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research*, , č. 12, 2011: s. 2825–2830.

[12] Python: *BeginnersGuide/Overview*. [Online; navštívené 20.03.2019].
URL https://wiki.python.org/moin/BeginnersGuide/Overview

[13] Python Software Foundation: *PyQt5 5.12.1*. [Online; navštívené 29.04.2019].
URL https://pypi.org/project/PyQt5/

[14] RapidMiner GmbH: *Nominal to Numerical*. [Online; navštívené 10.05.2019].
URL https://docs.rapidminer.com/latest/studio/operators/blending/attributes/types/nominal_to_numerical.html

[15] RapidMiner GmbH: *Normalize*. [Online; navštívené 04.05.2019].
URL https://docs.rapidminer.com/latest/studio/operators/cleansing/normalization/normalize.html

[16] Scikit-learn: *Decision Trees*. [Online; navštívené 24.04.2019].
URL https://scikit-learn.org/stable/modules/tree.html

[17] Scikit-learn: *Naive Bayes*. [Online; navštívené 24.04.2019].
URL https://scikit-learn.org/stable/modules/naive_bayes.html

[18] Scikit-learn: *Neural network models (supervised)*. [Online; navštívené 28.04.2019].
URL
https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[19] Scikit-learn: *Scikit-learn: Machine Learning in Python*. [Online; navštívené 16.04.2019].
URL https://scikit-learn.org/stable/index.html

[20] Scikit-learn: *Support Vector Machines*. [Online; navštívené 29.04.2019].
URL https://scikit-learn.org/stable/modules/svm.html

[21] Svoboda, P.: *Metody klasifikace www stránek*. Diplomová práca, FIT VUT v Brně, Brno, 2009.

[22] Thrun, S.; aj.: *The MONK's Problems: A Performance Comparison of Different Learning Algorithms*. 1991, [Online; navštívené 15.03.2019].
URL
https://www.researchgate.net/publication/2293492_The_MONK's_Problems-_A_Performance_Comparison_of_Different_Learning_Algorithms

[23] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: *Získávání znalostí z databází, Studijní opora*. 2010.

Príloha A

Parametre klasifikačných metód pre obecné experimenty

Tabuľka A.1, A.2 obsahuje parametre jednotlivých metód, ktoré boli použité pri niektorých experimentoch, a ktoré je možné meniť prostredníctvom užívateľského rozhrania aplikácie.

Dataset/ Metóda	**Jednoduchý Bayes**	**Rozhodovací strom**	**Neurónová sieť**	**SVM**
Breast Cancer Wisconsin	GaussianNB	Gini	Layers: 30 30 30 Max iter: 1000 Tol: 0.00001 No change: 60 Activaton: relu Solver: adam	C: 1.0 Max iter: 100000 Tol: 0.000001
MONKs Problems	GaussianNB	Gini	Layers: 20 20 Max iter: 5000 Tol: 0.0000001 No change: 40 Activation: tanh Solver: adam	C: 1.0 Tol: 0.0000001 Max iter: 10000
Adult Salary	GaussianNB	Gini/Entropy	Layers: 10 10 10 10 Max iter: 10000 Tol: 0.000001 No change: 40 Activation: relu Solver: adam	C: 300 Max iter: 10000 Tol: 0.00001
Arrhythmia	GaussianNB	Gini	Layers: 50 50 Max iter: 10000 Tol: 0.00001 No change: 40 Activation: relu Solver: adam	C: 1.0 Max iter: 100 000 Tol: 0.00001

Tabuľka A.1: Nastavenia metód použité pre obecné experimenty (časť 1.)

Dataset/ Metóda	**Jednoduchý Bayes**	**Rozhodovací strom**	**Neurónová sieť**	**SVM**
Breast Cancer Coimbra	GaussianNB	Entropy	Layers: 20 20 10 Max iter: 10000 Tol: 0.00001 No change: 60 Activation: relu Solver: adam	C: 1.0 Max iter: 10000000 Tol: 0.00001
Habermans Survival	GaussianNB	Gini	Layers: 20 20 20 10 Max iter: 10000 Tol: 0.0000001 No change: 60 Activation: relu Solver: adam	C:1.0 Max iter: 1000000 Tol: 0.00001
MAGIC Gamma Telescope	GaussianNB	Entropy	Layers: 20 20 20 Max iter: 10000 No change: 50 Tol: 0.00001 Activation: relu Solver: adam	C 1.0 Max iter: 100 000 Tol: 0.00001
Mammographic Mass	GaussianNB	Entropy	Layers: 20 20 20 Max iter: 10000 Tol: 0.00001 No change: 50 Activation: relu Solver: adam	C: 1.0 Max iter: 100000 tol 0.00001
Occupancy Detection	GaussianNB	Gini/Entropy	Layers 20 20 20 20 max iter 10000 tol 0.000001 no change 50 relu adam	C: 1.0 Max iter: 500000 Tol: 0.00001
Poker Hand	GaussianNB	Entropy	Layers: 64 64 Max iters: 2000 Tol: 0.00001 No change: 20 Activation: tanh Solver: adam	C: 1.0 Max iter: 100000 Tol: 0.00001
Zoo	GaussianNB	Gini	Layers: 15 15 Max iter: 10000 Tol: 0.0000001 No change: 20 Activation: relu Solver: adam	C: 1.0 Max iter: 2000 Tol: 0.00001

Tabuľka A.2: Nastavenia metód použité pre obecné experimenty (časť 2.)

Príloha B

Obsah pamäťového média

Nasledujúci adresárový strom popisuje štruktúru dát uložených na pamäťovom médiu – CD, priloženom k tejto diplomovej práci.

```
/ ..................................................Koreňový adresár CD
├── app/ ..........................................Adresár s aplikáciou
│   ├── dist/ ..............Spustiteľné verzie aplikácie (Windows i Linux)
│   │   ├── Linux/ ..................................Pre systém Linux
│   │   └── Windows/ ................................Pre systém Windows
│   └── src/ ..................................Zdrojové súbory aplikácie
├── datasety/ ..................................Použité dátové sady
├── technicka_sprava/ ..........................Text práce a šablóna
│   ├── latex/ ..............................Zdrojové súbory práce v LaTeX
│   └── technicka_sprava.pdf ....................Text diplomovej práce
└── README.txt ..................................................Popis obsahu CD
```