VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

PŘEHLED SOUČASNÝCH PŘÍSTUPŮ KE KLASIFIKACÍM

OVERVIEW OF ACTUAL APPROACHES TO CLASSIFICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ BREZÁNSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. FRANTIŠEK ZBOŘIL, CSc.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Brezánský Tomáš**
Program: Informační technologie
Název: **Přehled současných přístupů ke klasifikacím**
Overview of Actual Approaches to Classifications
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte zadanou literaturu.
2. Vyberte několik různých algoritmů (alespoň 3: rozhodovací stromy, neuronové sítě, Bayesovské klasifikátory) a navrhněte program pro demonstraci jejich činnosti.
3. Navržený program implementujte.
4. Proveďte experimenty s vybranými klasifikačními úlohami.
5. Porovnejte a zhodnoťte výsledky dosažené jednotlivými algoritmy.

Literatura:

- Zhang, Ch., Liu, Ch., Zhang, X., Almpanidis, G.: An up-to-date comparison of state-of-the-art classification algorithms, Expert Systems With Applications 82, 2017
- Kotsiantis, S.B.: Supervised Machine Learning: A Review of ClassificationTechniques, Informatica 31, 2007
- Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, Wiley & Sons, 2000

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

Vedoucí práce: **Zbořil František V., doc. Ing., CSc.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Táto bakalárska práca sa zaoberá prehľadom súčasných prístupov ku klasifikáciám. Popisuje rôzne prístupy ku klasifikáciám a ich algoritmy, zameriava sa na neuronové siete, bayesové klasifikátory a rozhodovacie stromy. Hlavnou úlohou tejto prace je vykonať experimenty s tromi klasifikačnými algoritmami, konkrétne sú to, algoritmus ID3, RCE neurónová sieť a naivný bayesov klasifikátor. Práca obsahuje experimenty s danými algoritmami a vyhodnocuje získané výsledky.

Abstract

This bachelor thesis deals with an overview of current approaches to classifications. It describes various approaches to classifications and their algorithms, focuses on neural networks, Bayesian classifiers and decision trees. The main task of this work is to perform experiments with three classification algorithms, namely, the ID3 algorithm, the RCE neural network and the naive Bayesian classifier. The work contains experiments with given algorithms and evaluates the obtained results.

Kľúčové slová

klasifikácia, klasifikačné algoritmy, neuronové siete, rozhodovacie stromy ,naivný bayesov klasifikátor, algoritmus ID3 ,Restricted Coulomb Energy (RCE) neurónová sieť

Keywords

classification, classification algorithms, neural networks, decision trees, naive bayes classifier, ID3 algorithm, Restricted Coulomb Energy (RCE) neural network

Citácia

BREZÁNSKÝ, Tomáš. *Přehled současných přístupů ke klasifikacím*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, CSc.

Přehled současných přístupů ke klasifikacím

Prehlásenie

Prehlasujem , že som túto bakalársku prácu vypracoval samostatne pod vedením pána doc. Ing. František Vítězslav Zbořil, CSc. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.

. . . . . .
Tomáš Brezánský
6. mája 2021

Poďakovanie

Týmto by som sa chcel poďakovať pánovi doc. Ing. Františkovi V. Zbořilovi CSc. za jeho čas a cenné odborné rady pri písaní tejto bakalárskej práce. Taktiež by som sa chcel poďakovať svojim rodičom a priateľom za ich podporu a trpezlivosť.

Obsah

Kapitola 1

Úvod

Umelá inteligencia sa stáva čoraz viac súčasťou nášho života, je zastúpená v rôznych oblastiach, ktoré zasahujú do každodenného života. Napriek jej širokému uplatneniu, bežne laická verejnosť nepozná princípy mechanizmov, na základe ktorých funguje umelá inteligencia.

Témou záverečnej bakalárskej práce je téma prístupov ku súčasným klasifikáciám v umelej inteligencii. Po preskúmaní danej problematiky, úlohou bolo vybrať niekoľko klasifikačných algoritmov, navrhnúť a implementovať program, ktorý demonštruje činnosti daných klasifikačných algoritmov. S použitím implementovaného programu následne urobiť experimenty o rôznych praktických klasifikačných problémoch, riešených vybranými algoritmami a zhodnotiť výsledky daných experimentov. Záverečná bakalárska práca je bližšie venovaná popisu a aplikácii algoritmu ID3, neurónovej sieti RCE a Naivného Bayesového klasifikátora.

Hlavným cieľom záverečnej bakalárskej práce je implementácia klasifikačných algoritmov a následne experimenty so špecifickými dátovými množinami.

V teoretickej časti záverečnej práce sú primárne opísané nasledovné teoretické témy: Rozhodovacie stromy, Neuronové siete a Bayesovské klasifikátory.

Druhá kapitola sa zameria na podrobný, teoretický popis klasifikácie, fungovaniu jednotlivých algoritmov a ich následné príkladné využitie v praxi. Okrem vyššie spomenutého, zahrňuje terminológiu a determinovaný príklad s následným využitím. Poznatky, ktoré sú uvedené v danej kapitole slúžia ako podklad pre ďalšiu kapitolu, ktorá je venovaná teoretickej aj praktickej časti, kde sú popísané konkrétne vybrané algoritmy a ich následné jednotlivé implementácie vybraných algoritmov.

V tretej kapitole je opísaný výber jednotlivých klasifikačných algoritmov, ktoré sú použité v aplikačnej časti práce. Algoritmy sú opísané pomocou procesu ich učenia. Taktiež je uvedené ako boli dané algoritmy implementované.

Štvrtá kapitola je venovaná vytvorenému programu pre danú prácu s klasifikačnými algoritmami. V kapitole je opísaná štruktúra programu, aké technológie boli použité pri jeho tvorbe. Ďalej je v kapitole popísaný návrh programu a užívateľské rozhranie.

Piata kapitola je venovaná testovaniu a experimentovaniu problémov pri použití jednotlivých algoritmov. Venovali sme sa nasledujúcim problémom, problém Cancer, ktorý bol riešený všetkými implementovanými algoritmami. Ďalšie problémy boli problém Card, Diabetes a Mushrooms, ktoré boli riešené pomocou RCE neurónovej siete a pomocou Naivného Bayesovho klasifikátora. V tejto kapitole sú zhrnuté aj výsledky experimentovania.

Záver práce obsahuje zhrnutie nadobudnutých poznatkoch o klasifikačných algoritmoch, ktoré sme získali ako výsledky z testovania a experimentovania.

Kapitola 2

Prehľad prístupov ku klasifikáciám

V tejto kapitole si predstavíme klasifikačné algoritmy. Kapitola definuje rôzne vetvy smerovania klasifikácie od Rozhodovacích stromov, cez Neuronové siete až po Bayessovské klasifikátory. Na začiatku kapitoly je uvedená definícia významu klasifikácie, opísaný proces ako si užívateľ vyberá algoritmus klasifikácie a vymenované jej rôzne druhy. Ďalej kapitola pokračuje v predstavovaní vybraných klasifikačných algoritmov. Pri každom algoritme sú uvedené nasledovné atribúty: princíp, štruktúra, jednotlivé pod-algoritmy, miesta využitia daných jednotlivých algoritmov. V poslednej časti kapitoly sú rozobrané výhody a nevýhody vybraných algoritmov.

2.1 Klasifikácia

2.1.1 Informácie

V bakalárskej práci sa často vyskytuje pojem klasifikácia, preto si hneď na začiatku priblížime jej význam v našom kontexte.

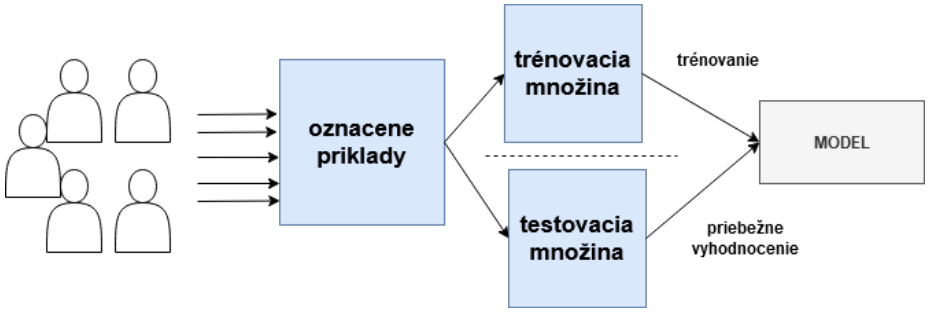
Klasifikácia je v odbore umelej inteligencie druh problému, ktorý určuje ako rozdeliť dáta do skupín na základe logického rozhodovania, tak aby čo najlepšie zodpovedala skutočnému roztriedeniu. Cieľom klasifikácie je teda vytvoriť funkciu, ktorá zaradí príklad do jednej z preddefinovaných tried.

V terminológii strojového učenia sa klasifikácia najčastejšie používa ako metóda učenia s učiteľom, kde máme k dispozícii trénovaciu množinu správne klasifikovaných príkladov. Vyskytuje sa aj pri metóde učenia bez učiteľa (zhlukovanie a klasifikácia podľa vzdialenosti). Algoritmus, ktorý implementuje klasifikáciu sa nazýva **klasifikátor**[1]. Pojem "klasifikátor" sa niekedy vzťahuje aj na matematickú funkciu implementovanú algoritmom klasifikácie, ktorá mapuje vstupné dáta do tried.

Hlavná úloha klasifikácie je identifikovať, ku ktorej kategórii patrí nový prípad na pozorovanie. Táto klasifikácia sa robí na základe trénovacej sady dát, obsahujúcej pozorovania, pre ktoré už je známa ich kategória. Táto klasifikácia dát prebieha pri trénovacej časti. Druhá testovacia časť je, keď sa klasifikátor snaží klasifikovať nové testovacie dáta[12].

Strojové učenie s učiteľom (Supervised Machine Learning) poskytuje nástroj na klasifikáciu a spracovanie údajov pomocou strojového jazyka. Pri učení pod dohľadom používame na odvodenie algoritmu učenia označené údaje, čo predstavuje súbor údajov, ktorý bol kla-

[1] Klasifikátor je algoritmus, ktorý implementuje klasifikáciu dát. Mapuje vstupné dáta na kategórie.



Obr. 2.1: Proces trénovania modelu pri učení s učiteľom

sifikovaný. Súbor údajov sa používa ako základ predikcie klasifikácie ďalších neoznačených údajov pomocou algoritmov strojového učenia.

Proces trénovania strojového učenia s učiteľom je zobrazený na obrázku 2.1.

Proces aplikovania strojového učenia s učiteľom na problém v reálnom svete je zobrazený na obrázku 2.2.

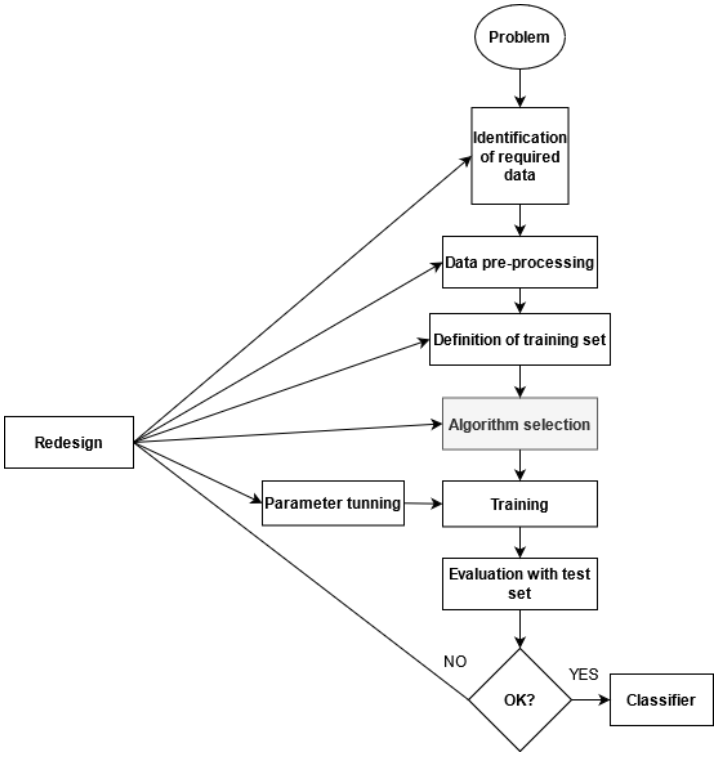
Prvý dôležitý krok v metóde učenia s učiteľom je zhromaždenie vstupnej množiny údajov. Pri riešení problému z reálneho sveta je najlepšie konzultovať s expertom na konkrétny problém. Takýto expert môže vybrať, ktoré vlastnosti, atribúty sú najinformatívnejšie. Ak nemáme experta k dispozícii na konkrétny problém, existuje aj metóda hrubá sila (brute force). Táto metóda pracuje na jednoduchom princípe, merá všetko dostupne v nádeji, že možno izolovať informatívne a relevantné znaky. Druhý krok v metóde je príprava a predspracovanie údajov. V prípade, že medzi údajmi chýbajú nejaké dáta, tak sú k dispozícií rôzne metódy na riešenie chýbajúcich údajov. Tieto metódy sú bližšie popísané v článku [5].

Ďalšie metódy, ktoré sa používajú pri príprave údajov sú napríklad výber podmnožiny vlastnosti a výber inštancie. Výber podmnožiny vlastnosti (Feature subset selection) je proces identifikácie a odstránenia toľkých irelevantných a nadbytočných vlastností ako je to možné. Táto metóda umožňuje, aby algoritmy pracovali rýchlejšie a efektívnejšie. Výber inštancie (Instance selection) sa používa na optimalizovanie problému a snaží sa udržať kvalitu pri minimálnej veľkosti vzorky.

Výber algoritmu, ktorý použijeme na implementáciu klasifikácie je dôležitý kritický krok v procese, pretože akonáhle je predbežne testovanie hotové a my sme spokojní s výsledkami priebežného testovania, tak klasifikátor je pripravený k dispozícii na použitie. Ohodnotenie klasifikátora je najčastejšie založené na presnosti predikcie (počet správnych predikcii / celkový počet predikcii). Na výpočet presnosti predikcie klasifikátora poznáme aspoň dve techniky, ktoré môžeme použiť na počítanie presnosti predikcie [5].

Prvá technika delí trénovacie dáta pomerom dvoch tretín na trénovanie a poslednú tretinu na odhad výkonu.

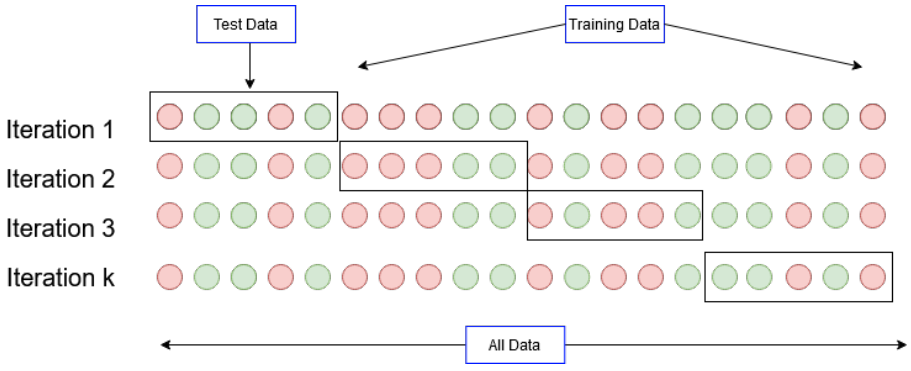
Druhá technika je krížová validácia (cross validation). Pri tejto technike prichádza k rozdeleniu vstupnej množine dát na dve podmnožiny. Jedna podmnožina slúži ako testovacia množina a druhá podmnožina slúži na trénovanie konkrétneho algoritmu. Klasifikátor natrénuje model na množine na trénovanie a pomocou testovacej množiny testuje presnosť a výkonnosť tohto modelu. Tento proces sa niekoľkokrát opakuje, vždy s inou podmnožinou pre trénovanie a testovanie. Pre každú podmnožinu sa spočíta chybovosť. Z týchto sa potom vypočíta priemerná chybovosť pre celú dátovú množinu. Priemerná chybovosť celej dátovej množiny je odhadovaná ako priemerná chybovosť klasifikátora. Špeciálny typ krížovej



Obr. 2.2: Zobrazenie procesu strojového učenia s učiteľom

validácie je k-násobná krížová validácia, ktorá obsahuje jeden parameter k, ktorý odkazuje na počet podmnožín, do ktorých sa vstupne dátová množina rozdelí. Krížová validácia typu LOOCV (Leave-one out) je konfigurácia krížovej validácie k-násobku, kde parameter k je nastavený na počet príkladov v vstupnej množine. Tento typ validácie sa používa v prípade, keď potrebujeme dosiahnuť najpresnejší potrebný odhad chybovosti klasifikátora. Veľkou nevýhodou krížovej validácie typu LOOCV je výpočtová náročnosť.

Princíp k-násobnej krížovej validácie je zobrazený na obrázku 2.3.



Obr. 2.3: Zobrazenie princípu cross validation

Ak ohodnotenie klasifikátora nie je uspokojivé, tak sa vraciame do predošlého kroku modelu procesu strojového učenia s učiteľom (podla obrázka 2.2) a preskúmame ďalšie faktory. Najprv skúsime upraviť parametre algoritmu. V prípade, že klasifikátor nedosa-

huje stále uspokojivé výsledky, tak preskúmame krok prípravy a predspracovania údajov, napríklad či neboli relevantné vlastnosti problému neskúmané. Ďalší možný faktor môže byť komplexnosť skúmaného problému, ktorý môže byť príliš náročný, prípadne vybraný konkrétny algoritmus je pre skúmaný problém nevyhovujúci, v tomto prípade zmeníme predtým vybraný algoritmus za iný algoritmus, ktorý bude lepši pre daný prípad. V poslednom prípade, že ani zmena algoritmu nepriniesla lepši výsledok, potom preskúmame či sme správne identifikovali najinformatívnejšie vlastnosti.

Bežnou metódou na porovnávanie algoritmov je štatistické porovnanie presnosti trénovaných klasifikátorov na špecifickej dátovej skupine. Ak máme dostatok dát, môžeme vytvoriť niekoľko trénovacích množín o veľkosti N. Na tieto množiny spustíme 2 algoritmy, ktoré chceme porovnať a pre každý z algoritmov odhadujeme rozdiel v presnosti pre každý pár klasifikátorov na testovacej množine. Priemer týchto rozdielov je odhadovanou očakávanou rozdielovou chybou pre všetky možné trénované množiny o veľkosti N. Ich variancia predstavuje odhad rozptylu klasifikátorov v celej trénovacej množine [5].

2.1.2 Druhy algoritmov

V strojovom učení je klasifikácia kontrolovaným trénovacím prístupom, pri ktorom sa počítačový program učí z vložených údajov a potom pomocou tohto učenia sa klasifikuje nové pozorovanie. Táto množina údajov môže byť dvoj-triedna (napríklad identifikácia, či ide o muža alebo ženu, alebo či sa jedná o nevyžiadanú poštu, alebo vyžiadanú poštu), alebo môže ísť o viac-triednu klasifikáciu. Niektoré známe príklady klasifikačných problémov sú: rozpoznávanie reči, rozpoznávanie rukopisu, biometrická identifikácia, klasifikácia dokumentov a iné. Spomenuté klasifikačné problémy môžu byť vyriešené pomocou klasifikačných algoritmov. Výber príkladov klasifikačných algoritmov je uvedený nižšie s jednoduchou definíciou. Výber týchto konkrétnych algoritmov s metódu učenia s učiteľom tzv. 'Supervised Learning' je zobrazený nižšie.

- Logistic Regression
- Naive Bayes Classifier
- Support Vector Machines
- Decision Trees
- Random Forest
- Neural Networks

Logistic Regression, (logistická regresia) predstavuje štatistickú metódu pre analýzu dátovej množiny, v ktorej je jedna, alebo viac nezávislých premenných, ktoré určujú výsledok. Výsledok regresie sa meria pomocou dichotomickej premennej. V dichotomickej premennej (alebo tiež v tzv. binárnej premennej) existujú len dve možné výsledky. Cieľom logistickej regresie je získať najvhodnejší model na opis vzťahu medzi dichotomickou charakteristikou a skupinou nezávislých premenných. Tento algoritmus patrí k lepším algoritmom binárnej klasifikácie ako napríklad nearest neighbor (najbližší sused) z dôvodu, že vysvetľuje tiež kvantitatívne faktory, ktoré smerujú ku klasifikácii. Metóda má široké využitie v rôznych oboroch, používa sa napríklad v bankovníctve, medicíne alebo v ekonómii.

Naive Bayes Classifier (ďalej ako NBC) je klasifikačná metóda založená na Bayesovom teoréme s predpokladom nezávislosti medzi predikátormi. Metóda predpokladá, že

prítomnosť konkrétneho prvku v triede nesúvisí s prítomnosťou akéhokoľvek iného prvku v triede. Aj keď tieto vlastnosti závisia na sebe, alebo existencii ďalších prvkov, všetky tieto vlastnosti nezávisle prispievajú k pravdepodobnosti. Hlavnou výhodou NBC je, že svojou jednoduchosťou prekonáva aj iné vysoko sofistikované klasifikačné metódy. Primárne použitie NBC je v lingvistike, v oblasti spracovania prirodzeného jazyka.

Support Vector Machines (ďalej ako SVM) známa tiež ako metóda podporných vektorov. SVM je algoritmus slúžiaci na klasifikáciu a regresiu. Cieľom metódy je nájsť optimálnu nadrovinu, ktorá priestor príznakov rozdeľuje tak, aby dáta z trénovanej metódy pátracie k odlišným triedam, ležali v opačných pol-priestoroch. Optimálna nadrovina je taká, v ktorej hodnota minima vzdialenosti bodu od roviny je čo najväčšia. Uplatnenie SVM je hlavné pre klasifikáciu, reálna aplikácia je implementovaná napríklad v oblasti spracovania textu, alebo pri rozpoznávaní tvári.

Decision Trees (ďalej ako DT) rozhodovacie stromy. DT zostavujú klasifikačné alebo regresívne modely vo forme stromovej štruktúry. Výsledkom DT je strom s rozhodovacími a listovými uzlami. Rozhodovací uzol ma dve, alebo viac vetiev a listový uzol predstavuje klasifikáciu, alebo rozhodnutie. Hlavné uplatnenie DT je využitie pri datách miningu a pri strojom učení.

Random Forest (ďalej ako RF) Náhodný les. RF predstavuje súhrnnú metódu učenia sa na klasifikáciu, regresiu a iné úlohy. Metóda RF vytvorí viacero rozhodovacích stromov pri učení a následne vyselektuje modus tried vrátených jednotlivými stromami. Táto metóda RF našla uplatnenie v bankovom sektore, napríklad pri odhaľovaní podvodníkov medzi ostatnými zákazníkmi.

Neural Networks (ďalej ako NS) Neurónové siete. NS je metóda určená na štruktúru pre distribuované a paralelné spracovanie dát. Skladá sa z jednotiek (neurónov), usporiadaných do vrstiev, ktoré prevádzajú vstupné vektory na výstup. Každá jednotka zoberie vstup, aplikuje sa na ňu napríklad nelineárna funkcia a potom tá predá výstup na ďalšiu vrstvu. Siete sú definované ako posun vpred: jednotka dodáva svoj výstup všetkým jednotkám v nasledujúcej vrstve, ale k predchádzajúcej vrstve neexistuje spätná väzba.

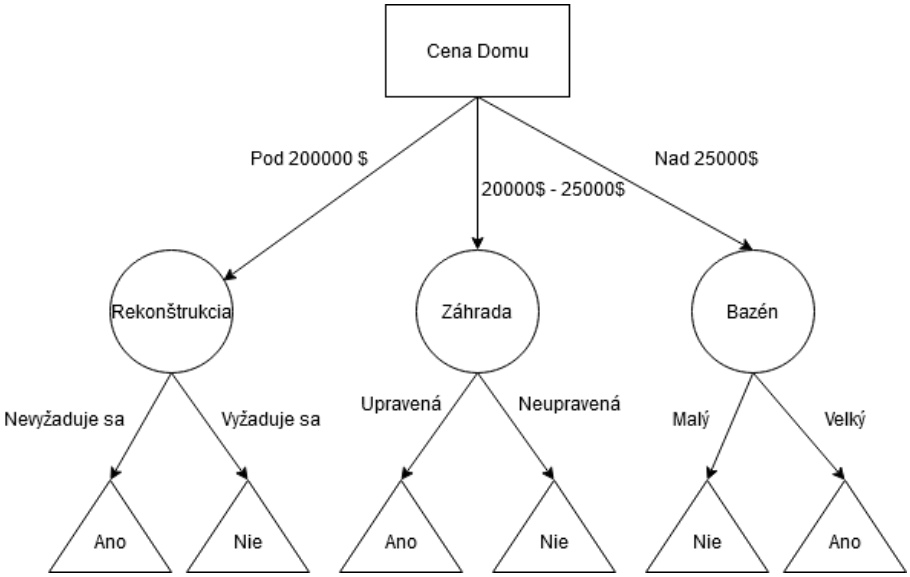
2.2 Rozhodovacie stromy

Cieľom tejto podkapitoly je priblíženie problematiky rozhodovacích stromov, definovanie ich vlastnosti, komponentov, typov, heuristických funkcií, ich vybraných algoritmov ID3 a C4.50, predstavenie frameworku CART.

2.2.1 Informácie

Rozhodovací strom je základom stromovej reprezentácie procedúry, pomocou ktorej je objektu na základe jeho vlastností priradená jedna z vopred definovaných tried. Rozhodovacie stromy známe taktiež ako Decission Tree (ďalej ako DT), slúžia pre binárnu klasifikáciu objektov na základe ich vlastnosti. DT sa zvyčajne kreslia zľava doprava začínajúc koreňom a pokračujúc smerom nadol. Z uzla môžu vychádzať dve až N vetiev. Každý uzol je atribút, ktorý sa pomocou vetiev rozdeľuje na podskupiny možných hodnôt samotného atribútu. Listy stromu reprezentujú možné hodnoty, ku ktorým sa potrebujeme dopracovať.

Ak už máme existujúci strom a potrebujeme pri ňom predpovedať požadovaný objekt na základe známych parametrov (atribútov), postupujeme od koreňa smerom nadol. S pomocou všetkých atribútov sa dopracujeme k správnej ceste a správnemu výsledku. Proces je cyklicky a vyžaduje jednoduché rozhodovacie podmienky. Pre DT platí pravidlo, že strom



Obr. 2.4: Obr. znázornujucí rozhodovací strom pre príklad 1.

Tabuľka 2.1: Tabuľka: vstupné dáta pre učiaci algoritmus

Dom	Cena	Rekonštrukcia	Záhrada	Bazén	Kúpiť?
Dom č.1	150000$	Nevyžaduje	Neupravená	Malý	Ano
Dom č.2	220000$	Vyžaduje sa	Upravená	Malý	Ano
Dom č.3	300000$	Nevyžaduje	Upravená	Velky	Ano
Dom č.4	20000$	Nevyžaduje	Neupravená	Malý	Nie
Dom č.5	340000$	Nevyžaduje	Neupravená	Malý	Nie
Dom č.6	140000$	Vyžaduje	Neupravená	Malý	Nie
Dom č.7	180000$	Nevyžaduje	Neupravená	Velky	Áno

rozdeľuje dáta do vetiev bez straty. Z toho vychádzame, že počet záznamov spĺňajúcich podmienku rodičovského uzla sa rovná súčtu záznamov spĺňajúcich podmienku dcérskych uzlov. Ak postupujeme smerom nadol počet záznamov, ktoré vyhovovali podmienkam cesty v strome, sa zmenšujú. DT v podstate rozdeľujú N-dimenzionálny priestor na oblasti s možnými riešeniami. Počet dimenzií je určený počtom atribútov použitých v strome.

Na bližšie ozrejmenie rozhodovacieho stromu uvádzame ilustračný príklad rozhodovacieho problému. **Príklad 1**: manželia XY sa rozhodujú o kúpe domu. Atribúty príkladu sú: Cena domu, rekonštrukcia, záhrada a bazén. Možné varianty rozhodovania pre kúpu domu sú: áno alebo nie. Zobrazenie DT pre tento príklad je znázornené na obrázku 2.4.

DT je graficko-analytická metóda, ktorej hlavná výhoda spočíva v prehľadnosti a interpretovaní, ktorá umožňuje užívateľom rýchlo a ľahko vyhodnocovať získané výsledky. DT sa často používajú v rozhodovacích analýzach, ktoré pomáhajú určiť najpravdepodobnejšiu stratégiu na dosiahnutie cieľa. Ďalšie primárne využitie nájdeme v nástrojoch pri strojovom učení a v dolovaní dát (data mining).

Pre DT je prirodzené a intuitívne klasifikovať objekt prostredníctvom sekvencie otázok, v ktorom každá ďalšia otázka zavisí od odpovede na aktuálnu otázku. Tento pristúp 'otázok' je obzvlášť užitočný pre nemetrické údaje, pretože všetky otázky môžeme klásť typu

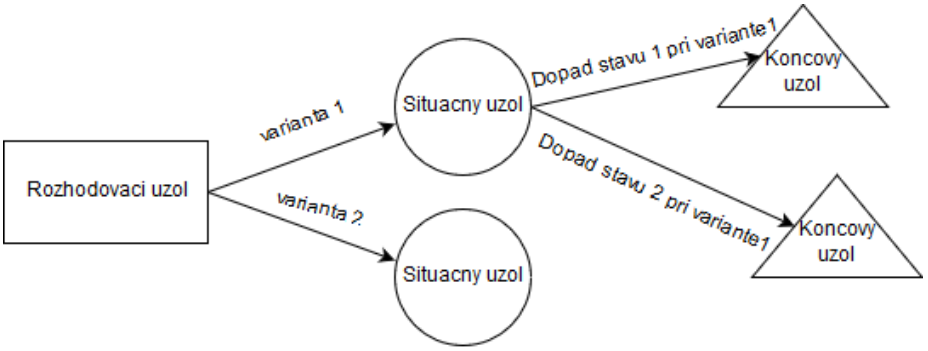
áno/nie, alebo pravda/nepravda, alebo hodnota (vlastnosť), ktorá patrí sade hodnôt, ktoré nevyžadujú žiadne metriky. Takáto sekvencia otázok sa zobrazuje v riadenom DT, alebo jednoducho v strome [11].

Zobrazenie DT predstavuje zvláštny prípad grafu skladajúci sa z uzlov a hrán. Uzol v strome predstavuje funkciu v prípadoch, ktorá sa má klasifikovať. Hrana v strome predstavuje hodnotu, ktorú uzol, môže očakávať. DT sú zoradené od vrcholu, kde je koreňový uzol, ktorí je spojený s vetvami. Tie sú buď spojené s ďalšími vetvami alebo koncovým uzlom.

Klasifikácia určitého objektu pomocou metódy DT prebieha od koreňa k listu. Každý vnútorný uzol predstavuje test určitej vlastnosti objektu (určenie jednej z hodnôt atribútu). Vstup začína v koreňovom uzle, ktorý sa pýta na konkrétnu vlastnosť objektu. V každom kroku je objekt otestovaný podla otázky v aktuálnom uzle DT a pokračuje po uzle, ktorý je zhodný s konkrétnym výsledkom otázky. Na základe odpovede nasledujeme príslušný odkaz na nasledujúci, alebo nadradený uzol. Listy musia byť navzájom odlišné, to znamená, že iba jeden list bude nasledovaný. Ak objekt dôjde až ku koncovému uzlu je klasifikovaný triedou pre daný koncový uzol DT.

Existujú 3 typy uzlov [13], zobrazené na obrázku 2.5

- **Rozhodovací uzol**, fáza rozhodovania, pri ktorej sa subjekt rozhodovania sám rozhodne pre niektorú z možných variantov riešenia, typicky je reprezentovaný štvorcom
- **Situačný uzol**, stav kedy situácia riešenia nezávisí od subjektu rozhodovania, ale od pôsobenia náhodných faktorov, typický je reprezentovaný kruhom
- **Koncový uzol**, konečný stav riešenia, typický je reprezentovaný trojuholníkom



Obr. 2.5: Obrázok znázornujucí štruktúru DT

Pri DT rozoznávame 2 hlavné typy DT, **Klasifikačné stromy** a **Regresívne stromy**. Obidve stromy majú určité podobnosti, ale aj určité rozdiely, ako napríklad postup na určenie miesta, kde strom rozdelíme.

Klasifikačné stromy rozdeľujú premenné hlavne do 2 skupín, ktoré môžeme číselne kategorizovať ako 0 alebo 1, alebo ako Áno a Nie. Teda výsledok je kategorický a diskrétny. Tento typ teda používame, keď je množina tried konečná.

V prípade, že množina tried predstavuje množinu reálnych čísel hovoríme o **regresívnych stromov**. Príkladom použitia je odhad ceny domu, ide o spojitú premennú, ktorej výsledok zaleží od ďalších spojitých premenných, napríklad rozloha domu a záhrady. Tento

typ používame, keď máme problém odhadu (predikcie). Ďalej budeme uvádzať pod pojmom rozhodovací strom klasifikačný strom, ak nie je uvedené inak.

Pre konštrukciu optimálneho DT hľadáme efektívnu heuristiku. Čo znamená heuristika? Heuristika je metóda a spôsob riešenia problémov, ktorá skracuje a zjednodušuje tradičné spôsoby riešenia a objavovania problémov. Funkcia, ktorá najlepšie rozdeľuje trénovaciu množinu, je znázornená vo forme koreňového uzla stromu. Poznáme niekoľko metód na nájdenie tejto funkcie pre rozdelenie DT napríklad : **Entropy**, **Information gain**, **Gini index**.

2.2.2 Algoritmus ID3 [9]

Iterative Dichotomiser 3 (ďalej ako ID3) je algoritmus, objavený austrálskym inžinierom Roosom Quinlan, ktorý sa používa na generovanie DT z dátovej množiny. ID3 je najznámejším klasifikačným algoritmom pre generovanie rozhodovacích stromov metódou zhora nadol, pričom generuje minimálny strom neinkrementálne. Názov ID3 algoritmu by sme mohli jednoducho preložiť ako iteratívny dvojtriedny klasifikátor. Jeho použitie je časté v strojovom učení, ale používa sa napríklad aj v doménach pri spracovaní prirodzeného jazyka.

ID3 generuje rozhodovací strom na základe vstupnej množiny príkladov, ktorú označujeme ako trénovacia množina. Vygenerovaný klasifikačný model v podobe rozhodovacieho stromu sa potom použije na klasifikáciu budúcich príkladov. Každý príklad má niekoľko atribútov a je zaradený do jednej z tried (napr. áno alebo nie). Listové uzly rozhodovacieho stromu tvoria klasifikačné triedy. Nelistové uzly sú rozhodovacie uzly, ktoré reprezentujú testovacie atribúty, pričom každá vetva obsahuje jednu možnú hodnotu tohto atribútu.

ID3 hľadá medzi atribútmi trénovacích príkladov atribút a tento potom vyberá, ktorý najlepšie roztriedi dané príklady. Ak atribút úplne klasifikuje trénovaciu množinu, potom ID3 končí, inak rekurzívne pokračuje na n rozdelených podmnožín, kde n je počet možných hodnôt daného atribútu. Algoritmus používa lačné hľadanie (z angl. greedy search) v priestore možných rozhodovacích stromov. To znamená, že vyberie najlepší atribút a nikdy sa nevracia späť, aby prehodnotil skoršie rozhodnutia.

Popis algoritmu ID3 .Táto verzia algoritmu pracuje len s diskrétnymi hodnotami, a preto musia byť spojité hodnoty diskretizované (prevedené na diskrétne hodnoty).

- 1. Na začiatku sa vytvorí uzol, ktorý reprezentuje všetky trénovacie príklady
- 2. Ak budú všetky príklady rovnakej triedy, potom sa tento uzol stane listovým uzlom pomenovaným touto triedou
- 3. V opačnom prípade sa použije metóda informačný zisk na určenie atribútu, ktorý bude najlepšie rozdeľovať vstupné príklady do jednotlivých tried. Tento atribút sa stane buď „testom“ alebo „rozhodnutím – triedou“.
- 4. Pre každú možnú hodnotu atribútu je vytvorená samostatná vetva a príklady sú priradené jednotlivým vetvám.
- 5. Algoritmus rovnakým spôsobom vytvára rozhodovací strom v každej časti. V prípade, že sa atribút použije ako testovacie kritérium, nesmie byť znovu použitý pre testovanie v tejto časti stromu.

- 6. Opakované delenie môže skončiť, pokiaľ je splnená niektorá z nasledujúcich podmienok
  - všetky príklady z daného uzla patria do rovnakej triedy
  - nie sú žiadne zostávajúce atribúty, podľa ktorých by mohlo dôjsť k ďalšiemu deleniu. V tomto prípade prebehne tzv. „väčšinová voľba“, ktorá zahŕňa prevod daného uzla na list a označí sa triedou, ktorá sa vyskytuje vo väčšine príkladov tohto uzla
  - nie sú žiadne ďalšie vzorky patriace k danej vetve. V tomto prípade je list vytvorený z väčšinovej triedy v príkladoch

Výber atribútu

Dôležitým kritériom v algoritme ID3 je výber atribútu pre testovanie v každom rozhodovacom uzle rozhodovacieho stromu. Cieľom je vybrať taký atribút, ktorý najlepšie klasifikuje príklady. Dobrým kvantitatívnym meradlom vhodnosti atribútu je štatistická vlastnosť, ktorá sa nazýva informačný zisk. Ten meria, ako dobre daný atribút rozdelí trénovacie príklady do ich cieľovej klasifikácie. Informačný zisk sa používa pri výbere z kandidátskych atribútov v každom kroku pri budovaní rozhodovacieho stromu. Aby sme mohli stanoviť informačný zisk, potrebujeme zadefinovať entrópiu. Informácie ohľadom ID3 boli prebraté z [9].

Algoritmus ID3 používa pre výber testovacieho atribútu Shannonovu teóriu informácie, ktorá na meranie množstva informácie používa entrópiu. Ak správy x_1, x_2, ... , x_N sú možné s pravdepodobnosťami p(x_1), p(x_2), ... , p(x_N) a tieto vytvárajú úplný súbor pravdepodobností, teda:

$$\sum_{j=1}^{N} p(x_j) = 1 \tag{2.1}$$

potom entrópiu, resp. neurčitosť daného súboru správ **S** môžeme vyjadriť nasledovne:

$$\mathbf{H(S)} = \sum_{\mathbf{j=1}}^{\mathbf{N}} -\mathbf{p(x_j)} \log_{\mathbf{2}} \mathbf{p(x_j)} \tag{2.2}$$

kde p(x_j) je pravdepodobnosť, že nejaký príklad patriaci uzlu **S** bude klasifikovaný do triedy x_j, kde x_j je súbor tried v **S**. Tento vzťah je známy ako Shannonová entrópia. Čím je entrópia súboru správ vyššia, tým menej určitý je obsah budúcej správy a tým väčšie množstvo informácie získame, keď správu príjmeme. Pre vetvenie stromu sa vyberie atribút s najmenšou entrópiou. Entrópia je nulová, ak uzol obsahuje len príklady rovnakej triedy. Potom každý listový uzol stromu má nulovú entropiu. A naopak každý nelistový uzol má nenulovú entropiu. Entrópia s hodnotou 1 nastane v prípade, že počet príkladov v uzle stromu jednej triedy sa rovná počtu príkladov druhej triedy.

Nech uzol **S** obsahuje n_1 príkladov klasifikovaných do triedy T_1 a n_2 príkladov zaradených do triedy T_2. Potom pravdepodobnosť, že nejaký príklad prislúchajúci uzlu **S** rozhodovacieho stromu bude klasifikovaný do triedy T_1 alebo T_2 je:

$$P(n_1|T_1) = \frac{n_1}{n_1 + n_2} resp. P(n_2|T_2) = \frac{n_2}{n_1 + n_2} \tag{2.3}$$

Entrópiu v danom uzle S vypočítame nasledovne:

$$\mathbf{H(S)} = \sum_{\mathbf{j=1}}^{\mathbf{2}} -\frac{\mathbf{n_j}}{\mathbf{n_1 + n_2}} \log_{\mathbf{2}} \frac{\mathbf{n_j}}{\mathbf{n_1 + n_2}} \tag{2.4}$$

Predpokladajme, že je možné v uzle **S** použiť atribút **A** s hodnotami a_1 a a_2 pre rozdelenie príkladov z uzla **S** do dvoch disjunktných podmnožín S_1 a S_2. Ak v uzle **S** má m_1 príkladov hodnotu atribútu **A** rovnú a_1 a m_2 príkladov má hodnotu a_2, potom celková entrópia v uzle **S** s použitím atribútu **A** sa určí ako:

$$\mathbf{H(S|A)} = \sum_{\mathbf{j=1}}^{\mathbf{2}} \frac{\mathbf{m_j}}{\mathbf{m_1 + m_2}} \mathbf{H(S_j)} \tag{2.5}$$

kde H(S_j) označuje entrópiu súborov pod uzlov uzla S.

Informácia získaná použitím atribútu **A** v uzle **S** sa označuje ako **informačný zisk** (information gain) získava sa nasledovne:

$$\mathbf{IG(A)} = \mathbf{H(S)} - \mathbf{H(S|A)} \tag{2.6}$$

kde H(S|A) je podmienená entrópia vzhľadom na hodnotu premennej A. H(S) entrópia množiny S. Informačný zisk sa v algoritme ID3 použije namiesto entrópie na vypočítanie **IG** pre každý atribút. Atribút s najväčším informačným ziskom sa používa na rozdelenie množiny **S** na tejto iterácii.

Algoritmus ID3 môžeme po úprave použiť i pre príklady, ktoré budú klasifikované do viacerých tried, kde atribúty môžu nadobúdať viac ako dve hodnoty. Ak jednotlivé triedy budú T_1, T_2, ..., T_N, potom vzťah pre výpočet entrópie uzla S bude mať nasledovný tvar:

$$\mathbf{H(S)} = \sum_{\mathbf{j=1}}^{\mathbf{N}} -\mathbf{p(T_j)} \log_{\mathbf{2}} \mathbf{p(T_j)} \tag{2.7}$$

kde p(T_j) je pravdepodobnosť, že nejaký príklad patriaci uzlu S bude klasifikovaný do triedy T_j. T_j je súbor tried v **S**.

Ak atribút **A** dosahuje hodnoty a_1,a_2,...,a_m, tak vzorec pre výpočet neurčitosti uzla S s testovacím atribútom A bude:

$$\mathbf{H(S|A)} = \sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{p(a_j)H(S_j)} \tag{2.8}$$

p(a_j) je pravdepodobnosť toho, že nejaký príklad patriaci uzlu **S** má hodnotu atribútu A=a_j. H(S_j) označuje entrópiu súborov pod uzlom uzla **S**.

Zhrnutie procesu fungovania algoritmu ID3 [14].

- Vypočítanie entrópie každého atribútu **a** z dátovej množiny **S**.
- Rozdelenie množiny **S** na podmnožiny pomocou atribútu, pre ktorý je výsledná entrópia po rozdelení najmenšia alebo je zisk informácii najväčší.
- Vytvorenie uzla rozhodovacieho stromu obsahujúci tento atribút.
- Aplikovanie rekurzie na podmnožiny pomocou zvyšných atribútov.

2.2.3 Algoritmus C4.5 [9]

Ďalším zaujímavým algoritmom je algoritmus C4.5, ktorý taktiež navrhol Ross Quinlan. Ide o algoritmus ID3, ktorý je vylepšený a doplnený. Algoritmus C4.5 rozlišuje dva typy atribútov: diskrétne (nominálne) a spojité (reálne). Ak je potrebné používať iné typy atribútov, je potrebné ich vhodne pretransformovať na jeden z týchto atribútov. Je to neinkrementálny algoritmus[2], budujúci strom zhora nadol. Tento algoritmus je založený na metóde "Rozdeľuj a panuj" panuje nad a rozdeľuje množinu **M** trénovacích príkladov. Predpokladajme, že trénovacie príklady majú byť klasifikované do tried T_1, T_2, ...T_K.

- Množina M obsahuje jeden alebo viac príkladov patriacich do tej istej triedy T_i. Rozhodovací strom pre množinu M je listový uzol s triedou T_i.
- Množina M neobsahuje žiadne príklady. Rozhodovací strom je opäť listový uzol, ale triedu nie je možné určiť z množiny M. Môže byť určená napr. ako majoritná trieda zo všetkých trénovacích príkladov, alebo ako majoritná trieda v rodičovskom uzle.
- Množina M obsahuje príklady patriace do viacerých tried. Vyberie sa test vedúci k rozdeleniu množiny M do podmnožín, obsahujúcich príklady s rovnakou hodnotou testovacieho atribútu. Testovacia podmienka predstavuje atribút, podľa hodnôt ktorého a_1, ...a_n sa M rozdelí do podmnožín M_1, M_2, ...M_n, kde Mi obsahuje všetky príklady z M, vyhovujúce podmienke s hodnotou atribútu a_1. Rozhodovací strom sa v tomto prípade skladá z rozhodovacieho uzla s testovacím atribútom a jednou podmienkou pre každú hodnotu atribútu. Tento postup je opakovaný rekurzívne pre každú vetvu rozhodovacieho uzla.

Neskôr Ross Quinlan ďalej vylepšil aj tento algoritmus C4.5, z ktorého vznikol nový algoritmus C5.0.

2.2.4 CART

CART je anglická skratka pre Classification and Regresion Trees (ďalej ako CART). CART poskytuje užívateľom všeobecný framework [3], ktorý môže byť použitý rôznymi spôsobmi na vytvorenie rôznych rozhodovacích stromov. Inými slovami termín CART označuje algoritmus rozhodovacieho stromu, ktorý sa môže použiť na klasifikáciu, ale i na regresívne problémy predikatívneho modelovania. Pri použití CART metódy vzniká niekoľko všeobecných druhov otázok.

- A. Koľko listov bude vychádzať z jedného uzla?
- B. Ktorá vlastnosť by mala byť testovaná v uzle?
- C. Kedy sa uzol zmení na list?
- D. Ak sa stane strom "príliš veľkým", ako sa dá zmenšiť a zjednodušiť?
- E. Ak je listový uzol nečistý, ako by sa mu malo priradiť označenie kategórie?
- F. Ako by sa malo zaobchádzať v prípade chýbajúcich údajov?

[2] Neikrementalný algoritmus je taky, ktorý spracováva celu množinu trénovacich príkladov odrazu.
[3] Framework je štruktúra, ktorá slúži na podporu pri zostavovaní

A. Koľko listov bude vychádzať z jedného uzla?

Každý výsledok rozhodnutia v uzle sa nazýva rozdelenie, pretože zodpovedá rozdeleniu podmnožine trénovacích údajov. Koreňový uzol rozdelí celú trénovaciu množinu. Každé nasledujúce rozhodnutie rozdelí podmnožinu údajov. Počet listov z uzla zavisí od otázky, ktorá vlasnosť by mala byť testovaná v uzle? Vo všeobecnosti je počet rozdelenia stanovený autorom rozhodovacieho stromu a môže sa líšiť v celom strome. Počet postupujúcich listov z uzla sa niekedy nazýva vetvený faktor, alebo vetvený pomer uzlov. Každé rozhodnutie a teda každý strom môže byť reprezentovaný len pomocou binárnych rozhodnutí [11].

B. Ktorá vlastnosť by mala byť testovaná v uzle?

Základnou zásadou tvorby stromov je jednoduchosť: uprednostňujeme rozhodnutia, ktoré vedú k jednoduchému, kompaktnému stromu s niekoľkými uzlami. Za týmto účelom hľadáme podrobný test T (test) na každom uzle N (počet uzlov), ktorý umožňuje, aby dáta dosiahli bezprostredne nadväzujúce uzly ako 'čisté'. Pri formalizácii tohto pojmu sa ukázalo, že je skôr vhodnejšie definovať nečistotu ako čistotu uzla. Existuje niekoľko rôznych matematických meraní nečistôt, ktoré majú v podstate rovnaké fungovanie. Najobľúbenejším meraním je entropická nečistota (entrópia). Tento vzťah už bol vysvetlený pri opise algoritmu ID3 2.7.

Existujú ďalšie matematické merania nečistôt ako odchýlka nečistôt (variance impurity) alebo Gini nečistota (Gini impurity).

Spôsob, ako nájsť optimálne rozhodnutie pre uzol, závisí od všeobecnej formy rozhodovania. Pretože rozhodovacie kritériá sú založené na extrémoch funkcií nečistôt, môžeme slobodne meniť takúto funkciu aditívnym, konštantným alebo celkovým faktorom merítka a toto neovplyvní rozdelenie.

Autori DT zvyčajne vyberajú funkcie, ktoré sa dajú ľahko vypočítať, napríklad na základe jedinej funkcie alebo atribútu, ktoré nám poskytnú monotetický strom [11].

C. Kedy sa uzol zmení na list?

Táto otázka vzniká pri rozhodnutí, kedy máme prestať s rozdeľovaním počas trénovania binárneho stromu. Ak bude strom pokračovať v raste úplne, až pokiaľ každý uzol listu nebude zodpovedať najnižšej nečistote, potom sa údaje začnú prekrývať. Naopak, ak ukončíme rozdelenie stromu príliš skoro, potom chyba v trénovacích údajoch nie je dostatočne nízka, čo môže viesť k skresľovaniu rozhodnutia.

Ako teda rozhodneme, kedy prestať rozdeľovať strom? Odpoveď je na základe tradičného prístupu križovej validácie (cross-validation). Pri princípe križovej validácie uplatňujeme nasledovné, strom je trénovaní pomocou podmnožiny údajov (90%), pričom zvyšných (10%) sa uchováva ako dáta na validáciu. Pokračujeme v rozdeľovaní uzlov v nasledujúcich vrstvách, pokiaľ chyba validačných údajoch je minimalizovaná [11].

D. Ak sa stane strom príliš veľkým, ako sa dá zmenšiť a zjednodušiť?

Niekedy, ak sa zastaví rozdeľovanie DT, môže to zapríčiniť nedostatok dostatočného pohľadu do budúcnosti, môže vzniknúť takzvaný fenomén 'horizontový efekt'. Určenie optimálneho rozdelenia v uzle N (počet uzlov) nie je ovplyvnené rozhodnutiami na N odvodených uzloch, t.j. na následných úrovniach. Pri zastavení rozdelenia môže byť uzol N vyhlásený za list, čím sa preruší možnosť výhodných rozdelení v nasledujúcich uzloch. Podmienka zastavenia

môže byť splnená "príliš skoro"pre celkovú optimálnu presnosť rozpoznávania. Hlavným alternatívnym prístupom k prerušeniu rozdelovania (splitting) je prerezávanie (prunning). Pri prerezávaní sa strom pestuje v plnom rozsahu, to znamená, kým listové uzly majú minimálnu nečistotu za akýkoľvek "horizont". Potom všetky dvojice susedných uzlov listov (t.j. tie, ktoré sú spojené s nadriadením uzlom, o jednu úroveň vyššie) sú zvažované pre elimináciu. Potom každá dvojica, ktorej eliminácia vedie k uspokojivému (malému) zvýšeniu nečistôt sa vylúči a spoločný nadriadený uzol sa vyhlási za list f. Takéto spájanie a zlučovanie je inverzné ku rozdeleniu.

Výhodou prerezávania je, že sa vyhýba účinkom fenoménu horizont. Ďalej, keďže neexistujú žiadne údaje o výcviku pre krížovú validáciu, priamo použijeme všetky informácie v trénovanej množine.

Existujú i ďalšie metódy, ktoré sú ale konceptuálne rozdielne od metódy prerezávania ako napríklad metóda založená na pravidlách (rules based method) [11].

E. Ak je listový uzol nečistý, ako by sa mu malo priradiť označenie kategórie?

Označenie kategórií uzlov listu je najjednoduchší krok pri konštrukcií rozhodovacieho stromu. Ak sú postupné uzly rozdelené čo do najväčšej hĺbky a každý uzlový list zodpovedá jednej kategórii vzoru(nulová nečistota), potom je tomuto uzlovému listu priradený tento názov kategórie. Vo viac typickejšom prípade, keď sme prestali rozdeľovať listový uzol a listové uzly majú pozitívnu nečistotu, tak by mal byť každý list označený podla kategórie, ktorá má najviac bodov. Extrémne malá nečistota nie je nevyhnutne žiadúca, pretože to môže naznačovať, že strom nadhodnocuje údaje o trénovaní [11].

F. Ako by sa malo zaobchádzať v prípade chýbajúcich údajov- atribútov?

Klasifikačné problémy môžu mať chýbajúce atribúty počas trénovania alebo počas klasifikácie, prípadne v oboch prípadoch. Aj napriek tomu, že niekedy môžu chýbať atribúty trénovacích údajov je potrebné zvážiť trénovanie klasifikátora rozhodovacieho stromu. V prípade chýbajúcich atribútov vypočítame nečistotu v uzle N iba pomocou prítomných informácii o atribútoch. Predpokladajme, že máme n trénovacích údajov v uzle N a každý z nich má tri atribúty, okrem jedného trénovacieho vzoru, v ktorom chýba atribút x3. Na nájdenie najlepšieho rozdelenia na uzle N, vypočítame možné rozdelenia pomocou všetkých n bodov pomocou atribútu x1, následne všetkých bodov pre atribút x2 a nakoniec pre n-1 bodov pre atribút x3. Každé takéto vypočítane rozdeleniam nám zníži nečistotu a potom je vybrané to najväčšie zníženie daných nečistôt.

Ako teda vytvoriť rozhodovacie stromy, ktoré dokážu klasifikovať vzory s chýbajúcimi atribútmi? Základný princíp počas klasifikácie je použiť tradičné rozhodnutie v uzle, vždy kedy je to možne (keď nechýbajú atribúty v testovacom vzore), alebo použiť alternatívne rozhodnutie (keď chýbajú atribúty v testovacom vzore).

Pri trénovaní rozhodovacieho stromu dostane každý neterminálny uzol N (uzol, ktorý má aspoň 1 a viac detí) usporiadanú množinu náhradných rozdelení pozostavujúcich z názvu atribútu a náhradného pravidla. Prvé také náhradné rozdelenie maximalizuje „prediktívne združenie“(predictive association) s primárnym rozdelením prediktívnej asociácie. Toto predikatívne združenie dvoch rozdelení s1 a s2 je iba číselný počet vzorov, ktoré sú zasielané „zľava“ obidvomi s1 a s2, plus počet vzorov zasielaných „doprava“ obidvoma rozdeleniami. Druhé náhradne rozdelenie je definované podobne a využíva inú funkciu a týmto spôsobom najlepšie aproximuje primárne rozdelenie.

Následne počas klasifikácie nekompletného testovacieho vzoru použijeme prvé náhradne rozdelenie, ktoré neobsahuje chýbajúci atribút testovaného vzoru. Táto stratégia na riešenie chýbajúceho atribútu v testovacom vzore je zodpovedá lineárnemu modelu, ktorý nahradzuje chýbajúce hodnoty vzoru hodnotou nechybajúceho atribútu, ktorý s nim najsilnejšie koreloval [11].

2.2.5 Zhrnutie rozhodovacích stromov

DT sa najbežnejšie vyskytujú pri analýze rozhodnutia (decision analysis), kde pomáhajú určiť stratégiu, ktorá s najväčšou pravdepodobnosťou bude úspešná. Existuje veľa DT algoritmov ako napríklad ID3, C4.5, C5.0, CART, CHAID alebo MARS.

ID3 algoritmus je vysvetlený v kapitole 2.2.2 a je implementovaný v rámci aplikačnej časti tejto bakalárskej práce. Algoritmy C4.5 a C5.0 sú vylepšenia algoritmu ID3. Algoritmus C4.5 na rozdiel od ID3 prešiel niekoľkými významnými zmenami, ktoré vylepšili rýchlosť algoritmu a odstránili niektoré obmedzenia ID3 ako napríklad, že vlastnosti musia byť dynamicky definované ako diskrétne vlastnosti. C4.5 akceptuje diskrétne aj spojité vlastnosti. Ďalej C4.5 dokáže spracovať aj neúplne údajové hodnoty, teda zvláda problém neúplných údajov veľmi dobre.

Autor ID3 a C4.5 Ross Quinlan pokračoval ďalej vo vylepšovaní algoritmu a vynašiel nový algoritmus C5.0. Vylepšenia v algoritme C5.0 sa hlavne zameriavajú na rýchlosť počítania, úsporu pamäti pri počítaní, zmenšenie rozhodovacích stromov pri C5.0 s rovnakým výsledok ako pri C4.5.

Výhody

DT majú viacero špecifických výhod, ktoré ich odlišujú od ostatných metód klasifikácie umelej inteligencie. Sú jednoduché na porozumenie a interpretovanie výsledkov vďaka možnosti grafického zobrazenia, ktoré umožňuje expertovi porozumieť pozorovaný problém. Pozitíva algoritmu sú napríklad rýchle klasifikovanie neznámych záznamov. Algoritmus pracuje veľmi dobre aj v prípade, ak rieši prípad, ktorý obsahuje veľa nepotrebných záznamov.

Nevýhody

Spoľahlivosť výstupných dát z DT a informácii získaných z DT závisí na tom, aké presné externé alebo interné údaje dostane DT. Aj malá zmena vstupných údajov, dokáže niekedy veľmi ovplyvniť zmeny v strome, a tým pádom aj zmeny výstupných údajov a informácii získaných z DT. Taktiež zmena premenných, vylúčenie informácii o duplikácii, alebo zmena sekvencie v strede, môže viesť k veľkým zmenám, ktoré budú vyžadovať prekreslenie DT. Ak existuje veľa nejasných hodnot, alebo mnohé výsledky sú prepojené, tak výpočty sa môžu stať veľmi komplexnými.

Ďalšiu zásadnou chybou pri analýze DT je, že rozhodnutia obsiahnuté v strome sú založené na očakávaní. Iracionálne očakávania môžu viesť k chybám v DT. Hoci DT sleduje prirodzený priebeh udalostí, sledovaním vzťahov medzi jednotlivými udalosťami, nie je možné naplánovať všetky nepredvídané udalosti, ktoré vyplývajú z rozhodnutia. Takéto prehliadnutia môžu viesť k zlým rozhodnutiam. DT sú taktiež náchylné na chyby v klasifikácii, v dôsledku rozdielov vo vnímaní a obmedzení pri použití štatistických nástrojov.

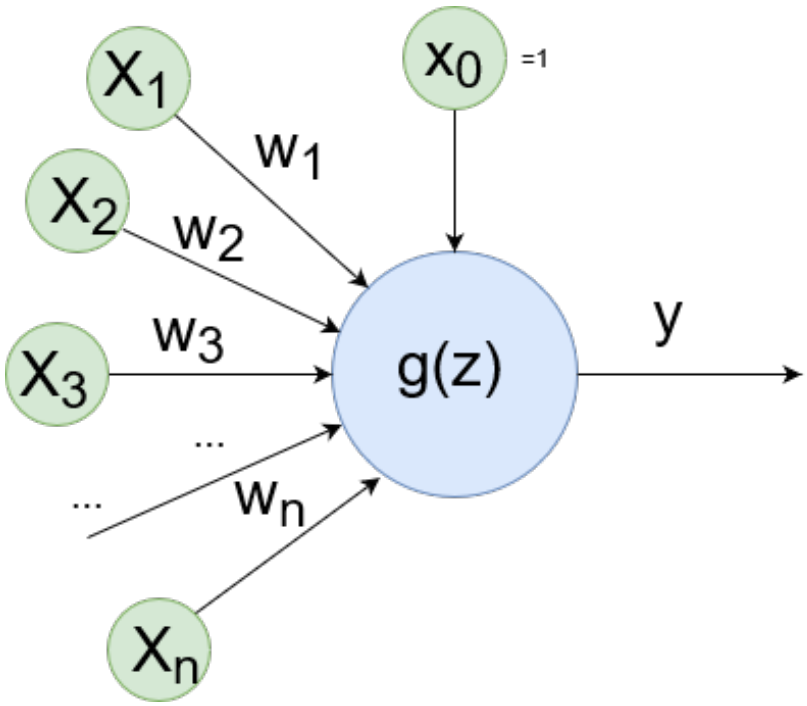
2.3 Neuronové siete

V tejto podkapitole si predstavíme Neuronové siete, základné prvky neurónových sieti ako neurón, modely neurónu, rôzne štruktúry sieti, prenosové funkcie a využitie neurónových sietí v praxi. Neurónová sieť (umelá neurónová sieť) je výpočtový model inšpirovaný ľudským mozgom, skladajúcich sa z jednotiek pomenovaných neuróny.

2.3.1 Neurón

Základná jednotka neurónovej siete je neurón. Neuróny v umelej inteligencii sú inšpirované biologickým neurónom.

Matematický model základného neurónu



Obr. 2.6: Model neurónu

- x - vstupy
- w - váhy neurónu
- g - aktivačná funkcia
- y - výstup

Umelý neurón ako je zobrazený na obrázku 2.6 vychádza z biologického neurónu. Premenné x_1,...,x_n predstavujú vstupné hodnoty, ktoré sa šíria do neurónu ako vstup. Váhy pre vstupné hodnoty označujeme ako w_1,...,w_n. Premenná x_0 a váha w_0 označujú konštantný vstup neurónu nazývaný bias, ktorý slúži ako hodnota pre ovplyvnenie aktivačnej funkcie neurónu. V umelom neuróne existuje vnútorný potenciál **z**, podobne ako prichádzajú do biologického neurónu ióny. Pre neuróny s lineárno bazovými funkciami (LBF) sa potenciál

počíta ako 2.9. Pre neuróny s radiálno bazovými funkciami (RDF) sa potenciál počíta ako 2.11.

Lineárna bazová funkcia (LBF) je definovaná ako 2.9.

$$z = w_0 x_0 + \sum_{i=1}^{n} w_i x_i = \vec{w}.\vec{x} \tag{2.9}$$

Vnútorný potenciál **z** je daný ako suma váh a vstupov plus bias. Výstup **y** z neurónu počítame na základe vnútorného potenciálu **z** a aktivačnej funkcie **g**.

$$y = g(z) \tag{2.10}$$

V najzákladnejšej forme je aktivačná funkcia binárna, teda funkcia sa spustí za hodnoty **1**, naopak sa neaktivuje pri hodnote **0**. Aktivačná (prenosová) funkcia **g** je vyberaná s cieľom zavedenia nelinearity do siete, táto nelinearita je potrebná pre riešenie lineárne neseparovateľných problémov. Aktivačná funkcia teda definuje výstup z neurónu vzhľadom na vstup. V kontexte biologického neurónu jedná sa o abstraktnú vlastnosť, ktorá reprezentuje pravdepodobnosť, že sa neurón aktivuje. Môže byť lineárna alebo nelineárna. Aktivačná funkcia priraďuje k potenciálu **z** výstup neurónu. V tabuľke 2.2 sú zobrazené rôzne aktivačné funkcie: prve tri patria pre LBF a Gaussová funkcia patri ku RDF.

Názov	Funkcia	Obor hodnôt
Logistická sigmoida	g(x)=$\frac{1}{(1+e^{-x})}$	(0,1)
Hyperbolický tangens	g(x)=$\frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$	(-1,1)
ReLu	g(x)=max(0,x)	<0,∞)
Gaussová funkcia	g(x)=(e^{x^2})	<0,1]

Tabuľka 2.2: Príklad vybraných aktivačných funkcií

Výpočet celého neurónu je možné opísať ako y=g(f(x,w)), kde **f** je bazová a **g** aktivačná funkcia. Tento princíp je zobrazený na obrázku 2.6. Bazové funkcie sa dajú ďalej rozdeliť na lineárne alebo radiálne.

Radiálna bazová funkcia (RBF) je definovaná ako 2.11.

$$z = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2}, \tag{2.11}$$

zodpovedá euklidovskej vzdialenosti vektorov **x** od vektoru **w**, dosahuje len nezáporne hodnoty.

V obidvoch bazových funkciách môžeme následne použiť ako aktivačné funkcie spojité i nespojité varianty aktivačných funkcií.

Neurónové siete s RBF neurónmi využívajú skokové funkcie alebo spojité (Gaussové funkcie) ako aktivačné funkcie. Sú koncepčne podobné modelom k-najbližší sused (k-nearest neighbor). Algoritmus vychádza z myšlienky, že podobné vstupy vytvárajú podobné výstupy. Výstupom siete je lineárna kombinácia radiálnych bázových funkcií vstupov a neurónových parametrov.

Výstup neurónu je analogický axón biologického neurónu. Táto výstupná hodnota sa ďalej šíri ako vstup ďalšej vrstvy pomocou synapsie (ak je neurón súčasťou viacvrstvovej siete), alebo opúšťa systém ako súčasť výstupného vektora.

2.3.2 Neurónové siete

Umelá neurónová sieť (ďalej ako NS) je jednou z hlavných výpočtových modelov v umelej inteligencii. Hlavnou inšpiráciou NS sú biologické NS. Podstatným základom NS je, že sieť sa učí realizovať úlohy zvažovaním pravidiel špecifických pre danú úlohu. Základná jednotka NS je neurón, ktorý sme si už predstavili.

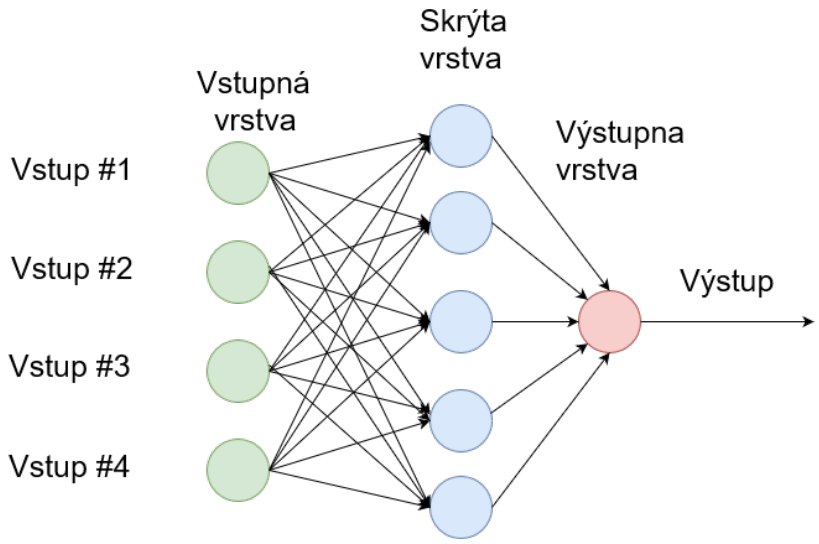
Model siete sa skladá z viacerých vrstiev: vstupnej vrstvy, kde prichádzajú počiatočné vstupné dáta, skrytej vrstvy (môže byť viacero skrytých vrstiev) tu sa vykoná výpočet a z výstupnej vrstvy, ktorá produkuje výsledok pre dané vstupné dáta. Siete môžu byť jednovrstvové alebo viacvrstvové.

Neurónové siete sú implementované na základe matematických operácií a súboru parametrov potrebných na určenie výstupu. Delíme ich podľa rôznych kritérií. Základné delenie NS je podľa topológie siete, podľa učenia siete a ďalšie delenie je napríklad podľa aplikácie.

Rozlišujeme dve základne typy podľa topológie:

- **Cyklická** (rekurentná z angl.reccurent). V cyklickej topológii skupiny neurónov vytvárajú kruh (cyklus). To vytvára problém, kedy nevieme určiť, ktorá vrstva je vstupná alebo výstupná. Jedna vrstva môže byť vstupná, ako aj výstupná. Typickým príkladom siete s cyklickou topológiou je Hopfieldová sieť. Cyklické siete sa pri klasifikáciach prakticky nepoužívajú.

- **Acyklická** (dopredná z angl. feed-forward). Naopak acyklické siete sa vyznačujú tým, že informácie sa v sieti preberajú iba jediným smerom od najnižšej vrstvy (vstupu) k najvyššej vrstve (výstupu) teda dopredu. Viacvrstvová sieť, kde neexistujú väzby medzi neurónmi rovnakej vrstvy, je typickým príkladom cyklickej NS. Potom sa neuróny dajú rozdeliť do jednotlivých vrstiev. Topológiu danej siete napríklad zapisujeme (3-6-3), kde 3 značí vstupnú vrstvu, 6 skrytú vrstvu a 3 výstupnú vrstvu. Na obrázku 2.7 je znázornený prípad acyklickej topológie, konkrétne viacvrstvovej neurónovej siete.



Obr. 2.7: Príklad viacvrstvovej neurónovej siete s jednou skrytou vrstvou.

Delenie podľa učenia NS. Učenie v kontexte neurónových sietí je proces, ktorým sa adaptujú voľné parametre neurónovej siete pomocou stimulácie prostredím, v ktorom je neuró-

nová sieť zahrnutá. Druh učenia je určený spôsobom, čím sa uskutočnia zmeny parametrov[3]. Učenie sa delí na viacero spôsobov:

- **Učenie s učiteľom**. V učení s učiteľom sa vytvára matematicky model zo súboru údajov, ktoré obsahujú vstupy aj výstupy, tento súbor údajov nazývame vzor. V NS je predložený vzor. Tento vzor sa porovná s výsledkom, ktorý získame na aktuálnom nastavení a určíme chybu, ktorá určuje rozdiel očakávaného výsledku podľa vzoru a reálneho výsledku. Aby sme zmenšili túto chybu spočítame nutnú korekciu (podľa typu sieti) a upravíme hodnotu váh, aby sme zmenšili túto chybu. Tento proces opakujeme až chyba dosiahne nami stanovený limit. Pri tomto učení sa používa spätná väzba.

- **Učenie bez učiteľa**. Pri tomto učení nevyhodnocujeme výstupné údaje, čo v praxi znamená, že výstup vôbec nepoznáme. NS dostáva na vstupe množinu vzorov, ktoré si sama zatriedi na konečnú množinu tried. Váhy sa menia iba na základe vstupu.

- **Posilňované učenie**. Učenie funguje na základe princípu odmeny. Princíp tohto učenia je, že udeľujeme pozitívne odmeny za chovanie, ktoré chceme podporiť a negatívne odmeny za chovanie, ktoré nechceme podporiť. Odmena môže prísť s oneskorením.

Existuje veľa rôznych NS, medzi základne typy NS môžu patriť perceptron, viacvrstvová sieť, rekurentná sieť, Hopfieldová sieť alebo Kohenová sieť. V tejto práci si zo všetkých NS bližšie priblížime perceptron, viacvrstvovú sieť a RCE sieť.

Štruktúra základného modelu neurónovej siete je zložená z nasledujúcich položiek: neurónov, prepojení, váh a prenosovej funkcie. NS sa skladá z umelých neurónov, ktoré si zachovali biologický koncept a sú vzájomne prepojené a predávajú si signály. Vstupné signály sú transformované pomocou prenosových funkcií a voliteľným prahom na výstupný signál. Neurón má ľubovoľný počet vstupov, ale iba jeden výstup.

Prepojenie sieťe sa skladá z prepojení medzi neurónmi. Každé prepojenie poskytuje výstup z jedného neurónu ako vstup do druhého neurónu.

Váhy pre každé prepojenie je priradená váha, ktorá prestavuje relatívnu silu, význam daného vstupu. Vo váhach sú teda uložené skúsenosti. K váham sa vzťahuje termín učenia neurónových sieti.

Prenosová funkcia existuje široký výber prenosových funkcií, ktoré sa vyberajú podľa typu neurónu a neurónovej siete. Príklady prenosových funkcií boli uvedené v podkapitole neurón.

Štruktúru modelu neurónových sieti sme si už rozobrali, ale ako taká neurónová sieť funguje? Do neurónu vstupujú signály, sila týchto signálov je rôzna niekedy silnejšia inokedy slabšia. Sila signálu sa určuje pomocou váh. Následne sa tieto signály sumujú a spracovávajú na výsledný signál pomocou prenosových funkcií. Tento výsledný signál sa šíri do ostatných neurónov v NS. Z posledných neurónov získame vektor výstupov.

2.3.3 Perceptrón

Perceptrón je jednovrstvová neurónová sieť s učením s učiteľom, viacvrstvový perceptrón nazývame NS. Perceptrón bol navrhnutý americkým vedcom Frankom Rosenblattom[4] a je určený na dichotomickú klasifikáciu rozdelenia do dvoch tried, pri ktorých sa predpokladá, že triedy sú lineárne separovateľné v príkladovom priestore. Funkcionalitu si vysvetlíme na nasledovnom príklade. Nech je daná množina vektorov x v n-rozmernom priestore. Každý

jeden z vektorov patrí do triedy1 alebo triedy2. Lineárna separovatelnosť dvoch tried predstavuje situáciu, keď rozdelíme objekty v priestore pomocou nadrovniny, v dvoj priestore pomocou priamky a v troj priestore pomocou roviny.

Perceptrón je rovnaký neurón ako je zobrazený na obrázku 2.6 s tím, že ma bipolárnu skokovú aktivačnú funkciu.

Celkový výstup perceptrónu o sa teda udáva ako vážený súčet:

$$o = g(z) = g(\vec{w}.\vec{x}) = g(\sum_{i=1}^{n} w_i x_i - \theta) \tag{2.12}$$

kde premenná z označuje váhovanú sumu vstupov, t.j. skalárny (zložkový) súčin váhového a vstupného vektora. Funkcia g je aktivačná funkcia perceptronu ide o bipolárnu skokovú aktivačnú funkciu.

Predpokladáme, že perceptrón má n+1 vstupov. Hodnota (n+1)-vého vstupu je vždy –1 a $w_{n+1} = \theta$, čo je hodnota prahu excitácie perceptrónu (angl. threshold).

Základné použitie perceptrónu: využíva sa pri lineárne separovateľných obrazov. Upravený perceptrón môžeme použiť i na klasifikovanie neseparovateľných obrazov.

2.3.4 Viacvrstvová neurónová sieť

Viacvrstvová neurónová sieť (MLP-Multilayer percepton)[6] ako už názov napovedá pozostáva z viac vrstiev ako jedna. Sieť s jednou vrstvou sa nazýva perceptrón. Viacvrstvová sieť sa skladá z vstupnej vrstvy, kde je m počet vstupných neurónov, skrytej vrstvy, ktorá môže byť n vrstvová a výstupnej vrstvy, ktorá obsahuje l výstupných neurónov.

Princíp fungovania viacvrstvovej siete je jednoduchý a odvodený od základných vlastností perceptrónu. Fungovanie si vysvetlíme na jednoduchej sieti, ktorá je znázornená na obrázku 2.7.

Na obrázku je zobrazená sieť s jednou skrytou vrstvou. Vektor X znázorňuje počet vstupných dát od $x_1, x_2, ..., x_m$, n značí počet neurónov v skrytej vrstve. Vstupné neuróny neuskutočňujú žiadny výpočet, iba poskytujú vstupné informácie do siete. Pre každý neurón v skrytej vrstve si vypočítame jeho hodnotu vnútorného potenciálu z pomocou vzorca.

$$z = W.X = w_1.x_1 + w_2.x_2 + ... + w_n.x_m = \sum_{i=1}^{n} w_i.x_i \tag{2.13}$$

Celkový vystúp každého neurónu potom vypočítame podla:

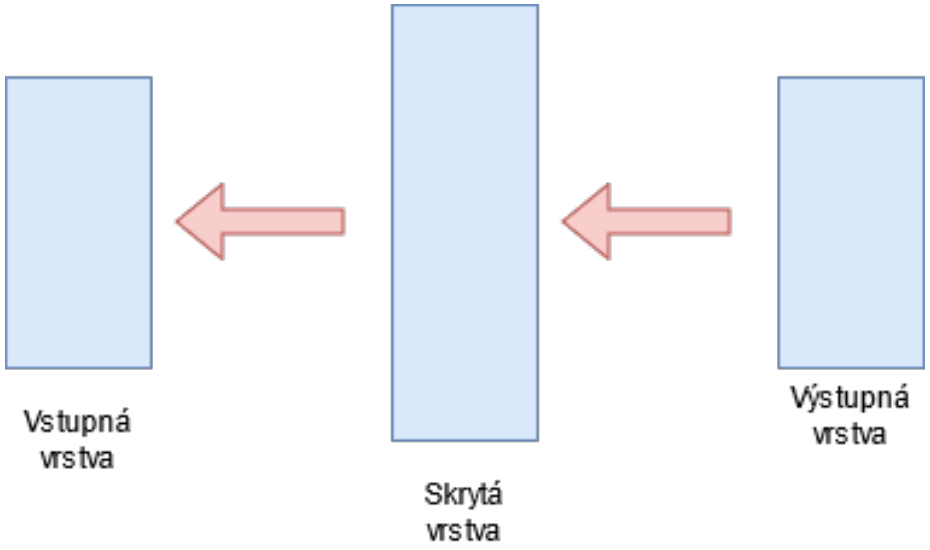
$$y = g(z) \tag{2.14}$$

kde g je aktivačná funkcia.

Príklad výpočtu prvého neurónu skrytej vrstvy z obrázka 2.7.

$$y_1^1 = f_1(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4) \tag{2.15}$$

V danom vzorci vektor tvorí váhu a predstavuje vstupné neuróny. Keď budeme mať spočítané všetky neuróny skrytej vrstvy, môžeme spočítať výstupné neuróny podľa rovnakého princípu. Na trénovanie viacvrstvovej NS môže byť použité učenie so spätným šírením chyby, ktoré je založené na vypočítaní chyby na výstupe NS a spätnej propagácii tejto chyby cez NS s úpravou váh.



Obr. 2.8: Spätné šírenie chyby u MLP

Využitie viacvrstvovej siete je širšie ako pri perceptróne. Hlavné využitie existuje v nasledovných oblastiach: pri riešení problematiky rozpoznávania hlasu, pri klasifikácii neseparovateľných obrazov a v oblasti strojového prekladu.

2.3.5 RCE (Restricted Coulomb Energy) neural network

Neurónová sieť RCE je trojvrstvová NS skladajúca sa zo vstupnej, skrytej a výstupnej vrstvy.

Vstupná vrstva slúži iba na premietanie vstupných signálov z vstupnej množiny na všetky neuróny v skrytej vrstve. Počet vstupných neurónov je priamoúmerný počtu vstupných signálov, ktoré sú priradené NS.

Skrytá vrstva pracuje s RBF neurónmi (neuróny s radiálnou bazovou funkciou), ktoré rozdeľujú vstupný priestor na menšie oblasti hypergule. U RBF neurónov počítame vnútorný potenciál, ktorý vyjadruje vzdialenosť medzi vstupným vektorom a vektorom, ktorý určuje stred RBF neurónu. Daný potenciál vypočítame pomocou euklidovskej metriky 2.16. Počet skrytých neurónov je presne daný a mení sa dynamicky.

$$u_k = ||\vec{i_p} - \vec{w_k}|| = \sqrt{\sum_{i=1}^{n}(i_i - w_{k_i})^2} \tag{2.16}$$

$$y_k = \begin{cases} 1 & \text{pre } u_k <= r_k \\ 0 & \text{pre } u_k > r_k \end{cases} \tag{2.17}$$

Parameter r_k určuje polomer hypergule. U_k je parameter určujúci vzdialenosť medzi vstupným vektorom a vektorom váh. Vektor $\vec{w_k}$ určuje stred hypergule reprezentovaným k-tým neurónom. Vektor $\vec{i}$ značí vstupný vektor.

Výstupná vrstva obsahuje neuróny s funkciou logického OR. Teda určuje počet klasifikovaných tried podľa počtu neurónov vo výstupnej vrstve. Neurón zo skrytej vrstvy môže byť klasifikovaný len do jedného výstupného neurónu.

U RBF 2.11 neurónu sa môžu používať viaceré prenosové funkcie napríklad Gaussová funkcia, ktorá počíta mieru toho, že vstupný vektor patrí ku stredu hypergule. V oblasti RCE siete budeme používať ako prenosovú funkciu, skokovú aktivačnú funkciu 2.17.

Hlavné využitie RCE siete je rozpoznávanie ako lineárnych, tak i nelineárnych oddelených oblasti. Hlavná výhoda RCE siete je oproti RBF siete v kratšom čase, ktorý je potrebný k trénovaniu. Ako predloha pri zostavovaní algoritmu bol použití popis učenia RCE siete v [17].

Popis učenia RCE siete

- 1. Nastaviť počiatočnú sieť, ktorá je prázdna, tj. q=0 počet neurónov skrytej vrstvy a m=0 počet neurónov výstupnej vrstvy.
- 2. Nastaviť indikátor zmeny v sieti modifl=false, nastaviť index p na prvý vektor trénovanej množiny p=1.
- 3. Nastaviť indikátor zásahu hit=false a index k (k-tého neurónu v skrytej vrstve na prvý neurón k=1).
- 4. Ak je k>q prejdite na bod 12, inak pokračujte.
- 5. Vypočítať vnútorný potenciál u_k (tj. euklidovskú vzdialenosť) k-tého skrytého neurónu medzi vstupným vektorom $\vec{i_p}$ a vektorom váh $\vec{w_k}$ k-tého neurónu.

$$u_k = ||\vec{i_p} - \vec{w_k}|| = \sqrt{\sum_{i=1}^{n}(i_{p_i} - w_{k_i})^2} \tag{2.18}$$

- 6. S použitím skokovo aktivačnej prenosovej funkcie zistite výstupnú hodnotu skrytého neurónu y_k.

$$y_k = \begin{cases} 1 & \text{pre } u_k <= r_k \\ 0 & \text{pre } u_k > r_k \end{cases} \tag{2.19}$$

  ak y_k=0 tak, skočte na bod 9, inak pokračujte.
- 7. Ak k-ty neurón reprezentuje rovnakú triedu ako je trieda aktuálneho vstupného vektora , tak nastavte hit=true a pokračujte bodom 9.
- 8. Zmenšiť polomer r_k tak, aby bol menší ako vzdialenosť u_k (obyčajné na polovicu) a nastavte modif =true.

$$r_k = \frac{\sqrt{\sum_{i=1}^{n}(i_{p_i} - w_{k_i})^2}}{2} \tag{2.20}$$

- 9. Inkrementujte index neurónu k a pokiaľ nie je väčší ako aktuálný počet neurónov skrytej vrstvy q, potom sa vrátte na bod 5 ,inak pokračujte.
- 10. Ak premenná hit=true, tak prejdite na bod 13 inak pokračujte.
- 11. Inkrementuje počet neuronov skrytej vrstvy q, vytvorte nový neurón s vektorom váh $\vec{w_q}$=$\vec{i_p}$ a polomer r=Rmax a nastavte premennú modit=true.

- 12. Ak trieda vstupného vektoru je už reprezentovaná nejakým výstupným vektorom, tak priraďte tento neurón tomuto výstupnému neurónu, inak vytvorte nový výstupný neurón reprezentujúci triedu aktuálneho vstupného vektora a pripojte nový neurón zo skrytej vrstvy k tomuto novému výstupnému neurónu.
- 13. Inkrementujte index p a pokiaľ nie je väčší ako počet vektorov trénovanej množiny P tak sa vráťte na bod 3.
- 14. Ak bola sieť modifikovaná modif =true vráťte sa na bod 2, inak sa učenie RCE siete ukončí.

Zhrnutie neurónových sietí. Vysvetlili sme si základnú štruktúru, matematický model neurónu a následne boli vymenované všeobecné neuronové siete. Detailnejšie sme si rozobrali popis jednovrstvovej siete-perceptrónu, viacvrstvovej siete a RCE siete.

Medzi výhody NS patrí rýchla predikcia v prípade dostupnej siete, ktorá už bola natrenovaná. NS je možné trénovať s ľubovoľným počtom vstupov a vrstiev. Na druhej strane medzi nevýhody patrí výpočtová a časová náročnosť, taktiež závislosť od trénovacích dát. Algoritmus NS sa využíva v rôznych aplikáciach, napríklad pri klasifikácii textu, strojovom preklade, rozpoznávaní reči a znakov.

2.4 Bayesové klasifikátory

V umelej inteligencii patria Bayesové klasifikátory do oblasti štatistického učenia algoritmov. Štatistický prístup sa odlišuje hlavne tým, že si ako jednoznačný podklad berie model pravdepodobnosti. Pravdepodobnostný model poskytuje pravdepodobnosť, v ktorej inštancia patrí do každej triedy na rozdiel od klasifikácie, ktorá rovno klasifikuje inštancie.

Medzi zástupcov štatistického učenia algoritmov patrí lineárna diskriminačná analýza (z angl. Linear discriminant analysis), ktorá je jednoduchá metóda používaná v strojovom učení a v štatistike na princípe lineárnej kombinácii prvkov, ktoré najlepšie oddeľujú dve alebo viacero tried objektov. Ďalším významným zástupcom je diskriminačná korešpondenčná analýza (z angl. Discriminant Correspondence Analysis) zameriavajúca sa na prácu s kategorickými premennými. Najznámejším zástupcom štatistického učenia a zároveň jedným z hlavných zameraním tejto bakalárskej práce sú Naivné bayesové klasifikátory.

2.4.1 Bayesovká teoréma

Metódy bayesovej klasifikácie vychádzajú z Bayesovej vety o podmienených pravdepodobnostiach. Bayesova veta je pomenovaná po anglickom štatistikovi Thomasovi Bayesovi(1701-1761) [16]. Ten riešil problém ako vypočítať rozdelenie pravdepodobnostného parametra binomického rozdelenia.

Bayesová veta (Bayesov Theorem) [15] sa zaoberá podmienenými pravdepodobnosťami javov. Formálny zápis podmienenej pravdepodobnosti javu H za predpokladu výskytu javu X je P(H|X). Ide o posteriórnu pravdepodobnosť hypotézy H za podmienky X. A naopak P(X|H) je pravdepodobnosť javu X podmienená výskytom javu H. Ide o posteriórnu pravdepodobnosť hypotézy X za podmienky H. P(H) je apriórna pravdepodobnosť hypotézy. P(X) je nepodmienená pravdepodobnosť dát, to znamená suma všetkých P(H),P(X|H) pre všetky hodnoty H.

Thomas Bayes zistil, že podmienená pravdepodobnosť P(H|X) súvisí s opačne podmienenou pravdepodobnosťou P(X|H) a to nasledovne: ak máme dve náhodné javy H a

X, ktorých pravdepodobnosti sú P(H) a P(X), tak pokiaľ platí P(X) > 0, potom platí aj nasledujúci vzťah.

$$\mathbf{P(H|X)} = \frac{\mathbf{P(H)} * \mathbf{P(X|H)}}{\mathbf{P(X)}} \tag{2.21}$$

P(**H**|**X**) je podmienená pravdepodobnosť javu H, za predpokladu výskytu javu X
P(**X**|**H**) je podmienená pravdepodobnosť javu X, za predpokladu výskytu javu H
P(**X**) a P(**H**) sú apriórne pravdepodobnosti výskytu jednotlivých javov nezávisle od seba.

Príklad použitia Bayesovej vety v praxi.

Príklad z oblasti finančnej analýzy v investičnej banke. Podľa prieskumu 60% obchodovaných spoločnosti na burze, ktoré v posledných troch rokoch zvýšili svoju cenu akcií o viac ako 5%, nahradilo svojich generálnych riaditeľov počas sledovaného obdobia.

Zároveň iba 35% spoločnosti nahradilo svojich generálnych riaditeľov, ktoré v rovnakom období nezvýšili svoju cenu akcií o viac ako 5%. Pravdepodobnosť, že ceny akcií porastú o viac ako 5% je 4%. Nájdite pravdepodobnosť, že akcie spoločnosti, ktorá prepusti svojho generálneho riaditeľa, sa zvýšia o viac ako 5%.

Pred samotným výpočtom pravdepodobnosti musíte najprv definovať zápis jednotlivých pravdepodobností.

P(**H**)-Pravdepodobnosť zvýšenia akcií o 5%.
P(**X**)-Pravdepodobnosť výmeny generálneho riaditeľa.
P(**H**|**X**)-Pravdepodobnosť zvýšenia ceny akcií o 5% vzhľadom na to, že generálny riaditeľ bol vymenený.
P(**X**|**H**)-Pravdepodobnosť výmeny generálneho riaditeľa vzhľadom na cenu akcií sa zvýšila o 5%.

Príklad reálneho použitia Bayesovej vety na nájdenie požadovanej pravdepodobnosti.

$$\mathbf{P(H|X)} = \frac{\mathbf{P(H)} * \mathbf{P(X|H)}}{\mathbf{P(X)}} = \frac{\mathbf{0.60} * \mathbf{0.04}}{\mathbf{0.60} * \mathbf{0.04} + \mathbf{0.35} * (\mathbf{1} - \mathbf{0.04})} = \mathbf{0.067}$$

Na základe výpočtu pravdepodobnosť skúmanej skutočnosti, že akcie obchodnej spoločnosti, ktorá nahradí generálneho riaditeľa, porastie o viac ako 5%, je 6.67%.

2.4.2 Naivný Bayesovský klasifikátor

Táto záverečná bakalárska práca bude bližšie venovaná popisu a aplikácii naivného bayesového klasifikáctora. Naivný bayesov klasifikator sa klasifikuje medzi štatistické klasifikátory, jedná sa o špeciálny prípad Bayesovej siete.

Naivný Bayesovský klasifikátor vychádza z predpokladu nezávislosti atribútov medzi sebou. To znamená, že efekt, ktorý má hodnota každého atribútu na danú triedu, nie je ovplyvnený hodnotami ostatných atribútov. Kvôli tomuto zjednodušeniu je tento klasifikátor nazývaný ako „naivný. Naivné Bayesové klasifikátory predstavujú klasifikačný algoritmus pre binárne (dvojtriedne) a viactriedné klasifikačné problémy. Ide o model pravdepodobností, ktorý využíva grafovú reprezentáciu pre zobrazenie pravdepodobnostných

vzťahov medzi jednotlivými javmi. Skladajú sa z riadených acyklických grafov, kde je iba jeden rodič uzol (predstavuje nepozorovaný uzol) a niekoľko synovských uzlov (predstavujú zodpovedajúce pozorované uzly) s predpokladom, že platí nezávislosť medzi detskými uzlami a ich rodičom [5].

Naivný Bayesovský klasifikátor pracuje nasledovne:[2]

- 1. Nech D je trénovacia množina príkladov a ich zaradení do tried. Každý príklad je reprezentovaný n-rozmerným vektorom vlastností X=(x_1, x_2, ..., x_n), patriacim n-atribútom A_1, A_2,..,A_n.
- 2. Majme m tried C_1, C_2, ..., C_m. Neznámy príklad X bude klasifikovaný do triedy s najväčšou posteriórnou pravdepodobnosťou.

$$\mathbf{P(C_i)|X) > P(C_j)|X) \quad pre \quad 1 <= j <= m, j \neq m} \tag{2.22}$$

- 3. Podľa Bayesovej teorémy P(C_i)|X)= $\frac{P(C_i)*P(X|C_i)}{P(X)}$. P(X) je konštantná pre všetky triedy C_i, stačí maximalizovať výraz P(X|C_i)*P(C_i). P(C_i) vyjadruje pravdepodobnosť s akou ľubovolne zvolený prvok patrí do triedy C_i. Používame pri tom vzťah P(C_i)=$\frac{S_i}{S}$, kde S_i predstavuje počet príkladov triedy C_i a S je počet všetkých príkladov. Maximalizujeme len člen P(X|C_i).
- 4. Vzhľadom na to, že pri veľkom počte atribútov by bolo výpočtovo náročne určovať P(X|C_i), vychádzame z predpokladu nezávislosti atribútov pri príslušnosti k danej cieľovej triede.

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) * P(x_2|C_i) * ... * P(x_n|C_i) \tag{2.23}$$

Pravdepodobnosť P(x_1|C_i)*P(x_2|C_i)*...*P(x_n|C_i) vieme určiť z trénovanej množiny, pričom

a) Ak A_k je kategorický atribút, potom P(x_k|C_i)=$\frac{Si}{S}$, kde s_{ik} je počet príkladov z trénovanej množiny patriacich triede C_i, pre ktoré A_k=x_k.S_i je počet vzoriek z trénovanej množiny patriacich triede C_i.

b) Pre spojité atribúty A_k sa zvyčajne predpokladá Gaussovo normálne rozdelenie hodnôt, a potom

$$P(X_k|C_i) = g(x_k, \mu_c i, \sigma_c i) = \frac{1}{\sqrt{2\pi\sigma c_i}}.e^{\frac{-(x-\mu c_i)^2}{2\sigma^2 c_i}} \tag{2.24}$$

μ_{ci} je stredná hodnota a σ_{ci} rozptyl hodnôt atribútu A_k trénovaných vzoriek z triedy C_i.

- 5. Pri klasifikácií neznámeho príkladu X je vypočítaný člen P(X|C_i)P(C_i) pre každú triedu C_i. Príkladu X je priradená trieda C_i, vtedy a len vtedy ak

$$\mathbf{P(C_i)|X) > P(C_j)|X) \quad pre \quad 1 <= j <= m, j \neq m} \tag{2.25}$$

Inými slovami, príklad X je zaradený do tej triedy, pre ktorú je P(X|C_i)P(C_i) maximálne.

Laplaceová korekcia [2]

Pri použití Bayesového modelu môže nastať problém, kedy podmienená pravdepodobnosť P($x_k|C_i$) je rovná 0 (v množine dát sa nenachádza žiaden príklad patriaci do danej triedy), pretože kvôli násobeniu dôjde k P(X|C_i)=0, čo znehodnotí celú klasifikáciu.To znamená, že v množine dát sa nenachádza žiaden príklad do danej triedy. Tomuto výskytu sa dá vyhnúť použitým Laplaceovým odhadom (Laplace estimator) alebo m-odhadom (m-estimator).

Príkladom riešenia je Laplaceova korekcia, čo znamená pridanie jedného prvku do všetkých množín. Vychádzame pri tom z toho, že naša množina príkladov je tak veľká, že pridanie ďalšieho príkladu ovplyvní pravdepodobnosť len zanedbateľne, pričom sa ale vyhneme nulovej hodnote pravdepodobnosti [1].

Bayesové klasifikátory sú menej úspešne pri klasifikácii vo svojich výpočtov ako ostatné viac sofistikované klasifikačné algoritmy, ako napríklad Neuronové siete. Dôvodom tejto nepresnosti je predpoklad nezávislosti medzi detskými uzlami [5].

Na základe týchto poznatkov niekto môže predpokladať, že výpočtová sila Bayesovských klasifikátorov je slabá a klasifikátory si nedokážu poradiť s veľkými dátovými množinami, ktoré by mali medzi sebou veľkú závislosť. Výsledky ale dokazujú, že klasifikátor dosahuje dobré výsledky i pre súbor dát, kde sa vyskytuje veľká závislosť medzi prvkami [5].

Základný nezávislý Bayesov model bol časom mnohokrát upravený rôznymi spôsobmi s cieľom vylepšenia jeho výkonu. Časť pokusov na prekonanie predpokladu o nezávislosti bolo založených na pridaní ďalších hrán k niektorým závislým prvkov. V tomto prípade má sieť také obmedzenie, že každá vlastnosť môže súvisieť iba s jednou ďalšou vlastnosťou [5].

Po predstavení základného Bayesovho modelu a jeho hlavného nedostatku môžeme v krátkosti diskutovať o ďalších rôznych rozličných Bayesovských klasifikátorov. Zameriame sa na klasifikátory, ktoré sa rozlišujú hlavne v predpokladoch, ktoré sa robia v súvislosti s distribúciou P(X_i|y).

Príklady typov distribúcie Naivného Baysovského klasifikátora:

- **Gaussovský Naivný Bayess**. Najľahší Bayessov klasifikátor na porozumenie je Gaussov. Tento klasifikátor predpokladá, že dáta od každého prvku vychádzajú z jednoduchého Gausovho rozdelenia. Keď predikátory zaujmú stálu hodnotu a nie sú diskrétné, predpokladáme, že tieto hodnoty sú vzorkované z gaussovského rozdelenia.

- **Multinomický Naivný Bayess**. Tento druh klasifikátoru sa väčšinou používa na riešenie problému klasifikácie dokumentov. Použitie môže byť vysvetlené na príklade, keď potrebujeme určiť, či dokument patrí do kategórie politiky, športu, umenia atď. Charakteristiky predikátory používané kvalifikátorom sú frekvencia slov výskytu pritomných v dokumente.

- **Bernoulliho Naivný Bayess**. Je podobný multinomickým Naivným Bayessom s tým rozdielom, že predikátory sú boolovské premenné. Parametre, ktoré používame na predpovedanie premennej triedy, prijímajú iba hodnoty áno alebo nie, príklad ak sa slovo nachádza v texte.

Zhrnutie Hlavnou výhodou naivného bayesovského klasifikátora je jednoduchosť, zrozumiteľnosť a efektívnosť. Štúdie preukázali, že účinnosť algoritmu je porovnateľná s účinnosťou ostatných zložitejších algoritmov (ako napríklad niektoré neuronové siete alebo rozhodovacie stromy [5]).

Naivné Bayesové klasifikátory sa väčšinou používajú pri analýze sentimentu, filtrovania spamu, doručovacích systémov atď. Ich najväčšia nevýhoda je v požiadavke na to, aby boli

predikátory nezávisle. V skutočnosti sú vo väčšine prípadov predikátory závislé, čo bráni optimálnemu výkonu klasifikátora.

Kapitola 3

Implementované klasifikačné algoritmy

Pre praktickú implementáciu záverečnej bakalárskej práce sme navrhli a vytvorili program Classify, ktorý pomocou implementovaných algoritmov rieši zadané klasifikačné problémy. Klasifikačné problémy Cancer, Card, Diabetes a Mushrooms sú bližšie opísané v kapitole č.5 .

Ako prvý krok v aplikačnom procese sme si určili, ktoré klasifikačné algoritmy budú implementované. Teoretická časť práce bola venovaná priblíženiu a opisu klasikačných algoritmov, s ktorými budeme pracovať ďalej. Z hľadiska dostupnosti domácej a zahraničnej literatúry a našich praktických skúseností s klasifikačnými algoritmami, pre účel aplikačnej časti záverečnej bakalárskej práce boli vybrané tri nasledovné klasifikačné algoritmy:

- Rozhodovacie stromy - algoritmus ID3
- Neuronové siete - RCE neurónová sieť
- Bayesové klasifikátory - Naivný Bayesov klasifikátor

Druhý krok v aplikačnom procese tvorí jednotlivá implementácia vybraných klasifikačných algoritmov, ktorá je bližšie popísaná v podkapitolách nižšie.

Posledný, tretí krok v aplikačnom procese je samostatné riešenie klasifikačných problémov v kapitole č.5 .

3.1 Implementácia algoritmu ID3

Implementácia algoritmu ID3 v aplikácii nami vytvoreného programu Classify. Na uskutočnenie klasifikácie pomocou algoritmu ID3 je vytvorená trieda Tree. Triedu Tree tvoria atribút, level stromu, hodnota výslednej triedy a HashMap[1] obsahujúci hodnotu, ktorá reprezentuje atribút a triedu Tree. V kontexte implementácie je atribút jeden z atribútov zo vstupnej dátovej množiny. Ak napríklad problém obsahuje 9 údajov, atribút predstavuje jeden údaj. Pre atribút bola vytvorená samostatná trieda z anglického názvu Attribute, ktorá obsahuje nasledovné prvky index, meno, hodnotu, ktorá určuje koľko rôznych hodnôt môže atribút nadobúdať, hodnoty, ktoré atribút nadobúda a v neposlednom rade aj arraylist s konkrétnymi dátami pre daný atribút. K určovaniu vhodného atribútu pre vetvenie

[1] Hašovacia tabuľka je údajová štruktúra, ktorá asociuje kľúče s hodnotami.

stromu je použitý výpočet entrópie 2.7 s informačným ziskom 2.6.Pre bližšiu ilustráciu implementovanej metódy Information Gain uvádzame obrázok 3.1. Výsledkom implementácie algoritmu ID3 je pripravenosť algoritmu riešiť klasifikačné problémy.

```
private double Information_Gain_calculate( Entropy[] ent, Entropy entropy )
{
    double Information_gain = entropy.value;
    double total = entropy.Number_of_count_entropy_one + entropy.Number_of_count_entropy_zero;
    double count_of_ones = 0;
    double count_of_zeros = 0;

    for ( int i = 0; i < ent.length; i++ )
    {
        count_of_ones = ent[ i ].Number_of_count_entropy_one;
        count_of_zeros = ent[ i ].Number_of_count_entropy_zero;
        Information_gain -= (((count_of_ones + count_of_zeros) / total) * ent[ i ].value);
    }

    return Information_gain;
}
```

Obr. 3.1: Implementácia metódy na počítanie informačného zisku atribútu

3.2 Implementácia neurónovej RCE siete

Podla princípu učenia RCE siete 2.3.5 je učenie siete implementované následovne v programe Classify, ako je znázornené na obrázku 3.2. Implementácia RCE siete sa skladá z troch tried z anglických názvov RCE neural network, HiddenNeuron a Output neuron.

Hlavná trieda RCE neural network sa zaoberá trénovaním a testovaním RCE siete. Hlavná metóda train slúži na trénovanie siete. Táto trieda obsahuje tiež implementovanú metódu predikcie na predikciu výslednej triedy a metódu na pridanie neurónu do skrytej vrstvy siete.

Trieda Outputneuron implementuje výstupný neurón siete RCE, ktorý obsahuje výslednú triedu a zoznam skrytých neurónov, ktoré sú k nemu pripojené. Trieda HiddenNeuron reprezentuje skrytý neurón a jeho vlastnosti. V implementácií trieda obsahuje vektor váh a vektor výstupnej triedy a parameter Rmax, ktorý určuje polomer hypergule. Metódy implementované v HiddenNeuron triede slúžia hlavne na nastavenie polomeru hypergule, na vypočítanie vnútorného potenciálu k-tého neurónu v skrytej vrstve a nastavuje aj ostatné náležitosti skrytej vrstvy neurónovej siete. Výsledkom implementácie algoritmu RCE siete je pripravenosť algoritmu riešiť klasifikačné problémy.

3.3 Implementácia Naivného Bayesovho klasifikátora

Implementácia Naivného Bayesovho klasifikátora v programe Classify. Naivný Bayesov klasifikátor pracuje na princípe, detailnejšie opísanom v teoretickej časti tejto práce 2.4.2. Pretože riešené klasifikačné problémy (Cancer, Card, Diabetes, Mushrooms) majú všetky dva výstupy, ide o jednoduchú implementáciu filtra na dva výstupy.

Pre klasifikácie pomocou Naivného Bayesovského klasifikátora bola vytvorená trieda Bayes.Táto trieda obsahuje metódy na načítanie dát, metódy pre trénovanie klasifikátora a metódu pre klasifikáciu. Klasifikátor sa skladá z vypočetných pravdepodobností pre jednotlivé typy tried. Pre odstránenie nulových aposteriórnych pravdepodobností je použitá Laplacova korekcia 2.4.2. Samotná klasifikácia prebieha vzájomným vynásobením pravde-

```
while(true){
   this.modif=false; this.p=0;
   while(true){
      this.hit = false;this.k = 1;
      if(this.p>=results.size()){
          return;      }
      ArrayList<Double> input_vector = input.get(this.p);
      ArrayList<Double> Result_class = results.get(this.p);
      if(this.k<=this.Hidden_neuron_count){
          while(true){
               Hidden_neuron Hiddden_neuron=neuronsHidden.get(this.k-1);
               Double InnerPotential = Hiddden_neuron.Get_InnerPotential(input_vector);
               if(InnerPotential>Hiddden_neuron.getRadius()){  }
               else if(Hiddden_neuron.getResultClass().equals(Result_class)){
                   this.hit=true;      }
               else{
                    Hiddden_neuron.Reduce_Ratio_of_hyperball(Reduce_Ratio_of_hyperball);
                    this.modif = true;     }
               this.k+=1;
               if(this.k> this.neuronsHidden.size())
                   break;
          }
          if(!this.hit){
               Add_Hidden_neuron(input_vector, Result_class);
               this.modif=true;
          }
      }
      else{
          this.modif=true;
           Add_Hidden_neuron(input_vector, Result_class);

      }
       this.p++;//increment index of input vector
       if(this.p > results.size()){
           break;
       }
   }
  if(this.modif==false){
      return;
      }
}
```

Obr. 3.2: Implementácia učenia v RCE sieti

podobností jednotlivých meraní pre všetky triedy. Výsledkom výpočtu Bayesovho klasifikátora je trieda s najvyššou pravdepodobnosťou. Pre ilustráciu implementovanej trénovacej metódy train v triede Bayes uvádzame obrázok 3.3 a v prílohe je zvyšok trénovacej funkcie 4.

```
public  void train(ArrayList<ArrayList<Double>> input,ArrayList<ArrayList<Double>> results) {
   Hash_Class1_words = new HashMap<Double, Integer>();Hash_Class2_words = new HashMap<Double, Integer>();int index=0
   for (ArrayList<Double> inputVector : input) {
       ArrayList<Double> Result_Vector_Output=results.get(index);
       //if output class1
       if((Result_Vector_Output.get(this.boolOut+this.realOut-2).equals(1.0))
               && (Result_Vector_Output.get(this.boolOut+this.realOut-1).equals(0.0)) ) {
           for (Double sw : inputVector) {
               if (!(sw.equals(null)))
               {
                   if (Hash_Class1_words.containsKey(sw)) {

                       Hash_Class1_words.put(sw, Hash_Class1_words.get(sw) + 1); // increasing value
                   } else {
                       Hash_Class1_words.put(sw, 1);
                   }
               }
               Number_of_words_OutClass1++;
           }
           Number_of_messages_OutClass1++;
       }
       OutClass1_prior_probabbility = Number_of_messages_OutClass1 / (float) Count_of_all_messages;
   }
```

Obr. 3.3: Implementácia trénovacej metódy v Naivnom Bayesovom klasifikátorovy časť 1

Podobne ako pri vyššie spomenutých klasifikačných algoritmov, výsledkom implementácie algoritmu naivného bayesovho klasifikatora je pripravenosť algoritmu riešiť klasifikačné problémy.

Kapitola 4

Implementácia programu

Táto kapitola popisuje implementáciu aplikácie Classify, ktorá implementuje preštudované základné klasifikačné algoritmy a funkčnosť aplikácie pri riešení konkrétnych problémov. Prvé dva podkapitoly sa zaoberajú štruktúrou a hierarchiou tried. Podkapitola dva obsahuje obrázok hierarchie tried. Tretia podkapitola sa zameriava na technickú časť aplikácie, aké nástroje, jazyky, knižnice boli použité pri tvorbe aplikácie. Posledná kapitola popisuje užívateľské rozhranie.

4.1 Štruktúra programu

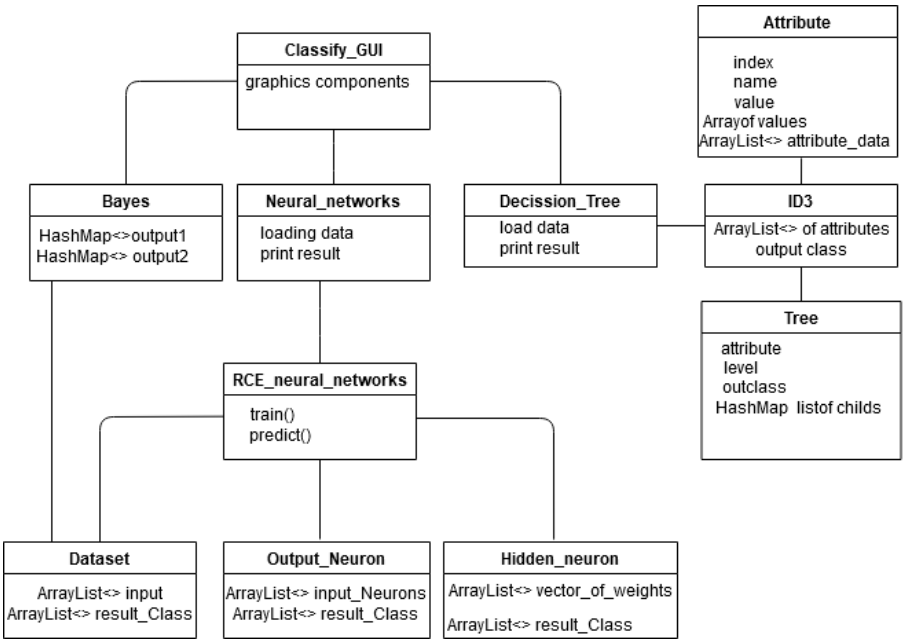
Program Classify obsahuje viacero súborov, v ktorom každý z nich vykonáva svoj špecifický účel. Súbory sú rozdelené na kategórie s implementáciou GUI (Graphical user interface), ktoré riešia vzhľad aplikácie a implementácie jednotlivých algoritmov. Do prvej kategórie patrí súbor **ClassifyGUI.java**, v ktorom sú implementované GUI aplikácie. Do druhej kategórie spadá napríklad súbor **DecissionTree.java**, v ktorom sa načítavajú vstupné údaje, ktoré sú predané do metódy ID3 zo súboru **ID3.java**, ktorý implementuje ID3 algoritmus.

ID3.java tvorí základný pilier implementácie algoritmu ID3, spolupracuje spolu s **Tree.java** a **Attribute.java**. Súbor **Tree.java** je jadrom ID3 algoritmu, pretože definuje stromovú štruktúru. Súbor **Attribute.java** obsahuje informácie pre atribúty ako meno, hodnota a index.

V súbore **Dataset.java** je vytvorená štruktúra Dataset pre uloženie údajov, táto štruktúra sa používa pre RCE sieť ako aj pre Bayesov klasifikátor. Súbor **Neuralnetworks.java** slúži na načítanie vstupných dát a ich úpravu do štruktúry Dataset. Z triedy **NeuralNetworks** sa volá trieda **RCEneuralnetworks**, ktorá implementuje učenie a predikciu RCE siete. Súbory **Hiddenneuron.java** a **OutputNeuron.java** definujú skryté neuróny, výstupné neuróny a ich vlastnosti . Implementácia Bayesovho klasifikátora je v súbore **Bayes.java**, ktorý obsahuje metódy na trénovanie a testovanie. Komunikuje hlavne so súborom **Dataset.java**, ktorý mu posiela dáta v štruktúre Dataset. Okrem vyššie spomenutých súborov máme aj súbor **pom.xml**, ktorý slúži na zostavenie programu Classify a vygenerovanie javodoc dokumentácie.

4.2 Hierarchia tried

Na obrázku 4.1 sú graficky zobrazené jednotlivé triedy aplikácie Classify a prepojenia medzi jeho triedami. Algoritmus ID3 využíva triedy DecissionTree, ID3, Tree a Attribute. RCE sieť používa triedy Neuralnetworks, Dataset, OutputNeuron a HiddenNeuron. Bayesov klasifikátor využíva iba jednu triedu Bayes. Grafické rozhranie sa nachádza v triede ClassifyGUI, ktoré obsahuje všetky grafické komponenty pre aplikáciu Classify.



Obr. 4.1: Hierarchia tried pre program Classify

4.3 Použité technológie

Implementácia programu Classify bola napísaná v programovacom jazyku **Java**. Programovací jazyk **Java** je vyvíjaný spoločnosťou Oracle, predstavuje objektovo orientovaný programovací jazyk, ktorý je podobný jazyku C++, nakoľko aj on vychádza z rodiny jazykov C a C++. Hlavná zmena na rozdiel od C++ v Jave je, že zdrojový text sa kompiluje do strojovo nezávislého, bajtového kódu tzv 'byte code', ktorý je nezávislý od konkrétnej platformy. Konkrétny 'byte code' sa neskôr spracováva pomocou interpretu Java Virtual Machine.

Pri tvorbe programu bolo použité vývojové prostredie **Netbeans IDE**. Jedná sa o prostredie pomocou ktorého môžu programátori písať a ladiť programy v jazyku Java. Vývojové prostredie bolo vyvinuté spoločnosťou Sun Microsystems. Netbeans IDE bol použitý za účelom zjednodušenia práce najmä v oblasti grafického rozhrania. Ďalšie vývojové prostredie, ktoré bolo použité je **IntelliJ IDEA**. Je to integrované vývojové prostredie, napísané v Jave na vývoj počítačového softvéru. Vývojové prostredie bolo vyvinuté spoločnosťou JetBrains. Okrem týchto dvoch prostredí bol použitý aj nastroj na zostavovanie **Maven project**, ktorý sa používa na spravu a zostavovanie projektov, rieši taktiež závislosti v projekte a dokumentáciu. Je založený na modele POM (projektový objektový model).

Grafické rozhranie (ďalej ako GUI) bolo implementované pomocou frameworku[1] JavaFX, ktorý vytvára MVC štruktúru aplikácie Classify. Ide o framework na platforme Java pre implementáciu grafických a užívateľských prvkov. Aplikácia je založená na modely MVC. Model MVC rozdeľuje logiku programu na tri vzájomne prepojené prvky Model-View-Controller.

- **Model** je nezávislá dynamická dátová štruktúra aplikácie, ktorá priamo riadi údaje a logiku aplikácie.
- **View** (pohlaď) poskytuje používateľovi zobrazenie dát z aplikácie formou vhodnou k interakcií.
- **Controller** (riadič) reaguje na vstupné udalosti vykonané užívateľom.

Ďalej bola použitá knižnica Swing. Knižnica Swing je odľahčená sada nástrojov grafického rozhrania Java (GUI), ktorá obsahuje bohatú sadu widgetov[2] a obsahuje balíky na vývoj desktopových aplikácii v Jave.

4.4 Užívateľské rozhranie

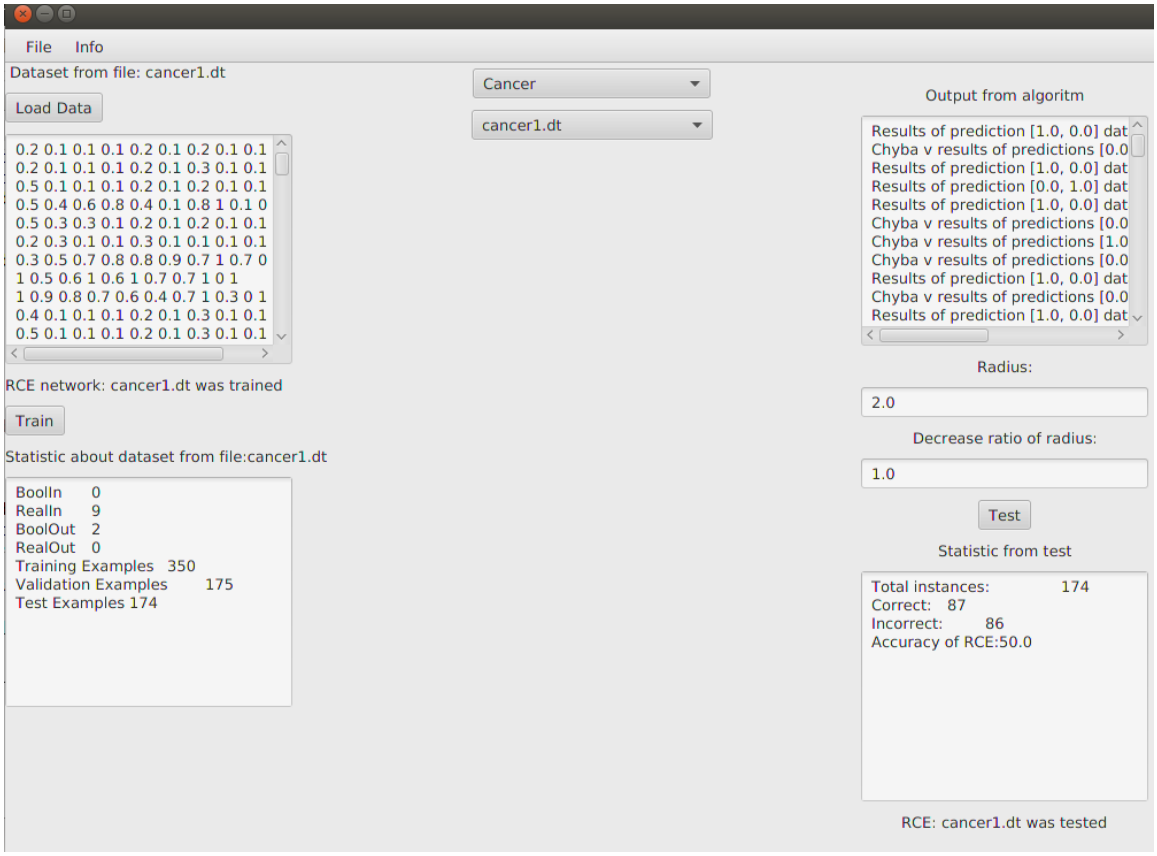
Ako už bolo spomenuté užívateľské rozhranie bolo navrhnuté vo vývojovom prostredí Netbeans IDE pomocou frameworku **JavaFX** a využíva tiež komponenty z knižnice **Swing**. Rozhranie je prispôsobené jednotlivým klasifikačným algoritmom, v ktorom každý algoritmus má vlastný pohlaď (scénu). Na začiatku spustenia aplikácie sa zobrazí prvý pohľad, menu. Menu pozostáva z textovej navigácie a štyroch tlačidiel. Prvé tri tlačidlá tvoria riešené klasifikačné algoritmy v nasledovnom poradí Decission Tree (ID3), RCE (Neural Network) a Bayes classifier. Posledné tlačidlo (Exit program) slúži na ukončenie programu. Po výbere konkrétneho algoritmu sa zmení starý pohlaď na konkrétny jedinečný, nový pohlaď pre daný algoritmus.

V každom ďalšom pohľade je implementované menu zobrazené v hornej časti aplikácie. Menubar pozostáva z dvoch záložiek File a Info, v ktorých je možnosť ďalšieho výberu. V prvej záložke File sa dá prepnúť na štartovaciu scénu a opakovaný výber konkrétneho algoritmu. V druhej záložke Info sa nachádza popis k jednotlivým algoritmom. V samotnom pohľade v hornej časti je na výber z comboBoxu, najprv konkrétny problém (cancer, diabetes, card alebo mushrooms). Následne konkrétny súbor s dátami k danému problému na trénovanie a testovanie. Na ľavej strane pohľadu sú dve textové oblasti na vstup dát zo súboru a informácie o vstupnom súbore. Na pravej strane pohľadu sú nasledovné ďalšie dve textové oblasti na vystúp výsledkov z algoritmu a štatistická informácie o výstupe konkrétneho problému riešeného pomocou vybraného algoritmu.

Na obrázku 4.2 zobrazeného na ďalšej strane sa nachádza príklad grafického užívateľského rozhrania pre RCE sieť.

[1] Framework je softwarová štruktúrá, ktorá slúži na podporu pri programovaní a organizácii iných softwarových projektoch

[2] Widget je všeobecné označenie pre ovládací prvok.



Obr. 4.2: Užívateľské rozhranie

Kapitola 5

Testovanie a ďalší vývoj

Piata kapitola je venovaná testovaniu a experimentovaniu s programom **Classify**. Všetky experimenty boli testované na implementovaných klasifikačných algoritmoch pomocou aplikácie Classify, ktorá bola vytvorená ako súčasť aplikačnej časti tejto záverečnej bakalárskej práce. Každý jeden algoritmus z implementovaných algoritmov pracuje s vlastnými špecifickými parametrami, podľa ktorých sa nadobúdajú rôzne výsledky. Uskutočnené experimenty a ich výsledky pre danú kombináciu parametrov sú zhrnuté nižšie v jednotlivých podkapitolách. Účelom týchto experimentov je poukázať na diferenciáciu jednotlivých parametrov algoritmov na výsledok. Všetky riešené problémy pochádzajú z benchmarku **PROBEN1** [10].

5.1 Klasifikačný dataset PROBEN1

Hlavným cieľom záverečnej bakalárskej práce je implementácia klasifikačných algoritmov a následne experimenty s špecifickou dátovou množinou. Na uskutočnenie tohto cieľa potrebujeme konkrétne klasifikačné algoritmy a konkrétny dataset[1]. Pojmom dataset rozumieme množinu reálnych klasifikačných problémov. Pri výbere konkrétneho datasetu sme zvážili niekoľko parametrov napríklad, aká je dostupná veľká množina reálnych dát, aby boli problémy riešiteľné pomocou techniky učenia s učiteľom, teda aby boli dostupné separované vstupy a výstupy. Ďalší parameter, ktorý je zohľadnený pri výbere datasetu je statickosť problémov. Potrebujeme vybrať dataset, ktorého všetky problémy sú statické, čo znamená vstupné dáta sa nemenia pri učení s učiteľom.

Na výber existuje celá rada datasetov, ktoré spĺňajú uvedené požiadavky so zameraním, či už od rozpoznávania obrazu, audio rozpoznávania alebo datasety, ktoré sa zaoberajú diagnostickými úlohami. Pri výberovom procese do úvahy pripadali nasledovné datasety: **MNIST** [8], **CIFAR-10** [7] a **PROBEN1** [10]. **MNIST** je veľká databáza ručne písaných čísel vhodných pre algoritmy učenia s učiteľom. **CIFAR-10** je zbierka obrázkov, ktoré sa používajú na trénovanie algoritmov strojového učenia na rozpoznávanie obrazov, obsahuje veľkú kolekciu 60000 obrazov. **PROBEN1** obsahuje dáta reálnych problémov pre neuronové siete. Kvôli praktickosti a experimentov z minulosti, tento dataset bol vybraný na naše experimentovanie a testovanie s implementovanými algoritmani.

Dataset PROBEN1 tvorí súbor problémov pre učenie sa neurónových sietí v oblasti klasifikácie vzorov a aproximácie funkcií plus súbor pravidiel a konvencií na vykonávanie testov porovnávania s týmito alebo podobnými problémami [10]. Autor daného súboru je

[1] Dataset - kolekcia dat

nemecký, softwarový inžinier Lutz Prechelt. Dataset pozostáva z kolekcie reálnych problémov pre učenie neurónových sieti z kategórie klasifikačných algoritmov. PROBEN1 obsahuje 15 problémov, bližšie si popíšeme konkrétny problém a postup ako daný problém riešime s pomocou neurónovej siete. PROBEN1 definuje aj sadu pravidiel ako uskutočňovať a porovnávať pomocou benchmarkingu neurónových sietí. Cieľ datasetu PROBEN1 je poskytnúť výskumníkovi dáta pre skontrolovanie správnej implementácie ich algoritmu a možnosť rýchleho a jednoduchého porovnania výsledkov.

Zo sady problémov PROBEN1, ktorý obsahuje 10 klasifikačných problémov, boli vybrané nasledovné štyri problémy: **Cancer**, **Card**, **Diabetes** a **Mushrooms**. Pomocou algoritmu ID3 bolo možné riešiť iba prípad Cancer. Ostatné algoritmy RCE siete a naivný Bayesov klasifikátor sa dajú použiť na všetky vybrané štyri problémy. Na testovanie algoritmov boli použité dáta zo súborov s príponou .dt, ktoré obsahujú hlavičku, kde je daný počet trénovaných a testovaných príkladov plus ďalšie informácie o dátovej množine.

5.2 Testovanie dát zo sady problémov PROBEN1

Proces testovania a experimentovania dát prebiehal v programe Classify v časovej súslednosti na základe nasledovných krokov a postupov:

- 1. Zapnutie aplikácie Classify, výber konkrétneho klasifikačného algoritmu.
- 2. Výber konkrétneho problému, ktorý sa bude testovať.
- 3. Výber dát z dostupných súborov problému, ktoré obsahujú vstupné množiny dát pre daný problém.
- 4. Proces trénovania algoritmu na vstupnej množine dát. Prípadné nastavenie parametrov pre konkrétny algoritmus, napríklad nastavenie parametrov u RCE siete ako Radius(polomer hypergule) a Decreas ratio of radius(redukcia hypergule).
- 5. Testovanie už natrénovaného algoritmu pomocou množiny dát určených na testovanie a overenie ich konečného výsledku predikcie daného algoritmu so skutočnými výsledkami.
- 6.Finálne výsledky testovania vybraného algoritmu v kombinácii s datasetom sú zobrazené v poslednom okne programu.

Všetky testované problémy sú otestované podľa vyššie uvedeného postupu.

5.3 Problém Cancer

Klasifikačný problém Cancer (rakovina) sa zaoberá diagnostikovaním rakoviny prs. Snaží sa klasifikovať problém ako nezhubný(benígny) alebo zhubný(malígny) nádor. Údaje boli získane na základe mikroskopického vyšetrovania. Problém obsahuje 9 vstupných atribútov, ktoré sú diskrétne. Tieto atribúty sú rovnomernosť veľkosti buniek, tvar buniek a ďalšie lekársky odborne terminologické faktory. Okrem vstupných atribútov problém obsahuje dve výstupné atribúty, ktoré sú binárne. Príklad z databázy môže vyzerať nasledovne: 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0, príklad pochádza zo súboru cancer1.dt. Testovanie a experimentovanie s problémom Cancer prebiehalo s dátami zo súborov cancer1.dt až cancer3.dt. Tieto súbory obsahujú spolu 350 príkladov na trénovanie a 175 príkladov na

testovanie a validáciu. Súbor cancer4.dt obsahuje 500 príkladov na trénovanie a 750 na testovanie.

5.3.1 Algoritmus ID3 (Iterative Dichotomiser 3)

Algoritmus ID3 bol kvôli náročnosti na implementáciu primárne testovaný iba na probléme Cancer. Tabuľka 5.1 zobrazuje výsledky získané z testovania.

	Iterative Dichotomiser 3
cancer1.dt	0.90229
cancer2.dt	0.8620689
cancer3.dt	0.8793103

Tabuľka 5.1: Zobrazujúca výsledky z testovania problému Cancer pomocou ID3 algoritmu

Zhrnutie. Z dosiahnutých výsledkov môžeme vidieť, že algoritmus ID3 dokáže riešiť klasifikovanie problému Cancer na vysokej úrovni úspešnosti. Priemerné výsledky z 3 testov dosahovali 88.122 percent úspešnosti odhadu výslednej triedy.

5.3.2 RCE

RCE je druhý algoritmus, ktorý bol testovaný na probléme Cancer. Pri testovaní je dôležité nastavenie parametrov Radius(Rmax) a Decrease ratio of radius(Rdc). Ako prvé sme testovali s hodnotami parametrov Rmax=1.25 a Rdc=0.625. V nasledujúcom testovaní sme zvýšili hodnoty parametrov na Rmax=2 a Rdc=1. Získané výsledky boli trochu lepšie ako pri prvom testovaní. V poslednom testovaní sme definovali parametre Rmax=1.5 a Rdc=0.75. Získané výsledky dosiahli najlepšie hodnoty pri druhom testovaní, pretože úspešnosť správnej predikcie bola najvyššia. Tabuľka 5.2 zobrazuje testované hodnoty parametrov, ako aj výsledky.

	Rmax=1.25 Rdc=0.625	Rmax=2 Rdc=1	Rmax=1.5 Rdc=0.75
cancer1.dt	0.47701	0,60919	0,61494
cancer2.dt	0.59770	0,51724	0,54597
cancer3.dt	0.52298	0,57471	0,54022
cancer4.dt	0.59915	0,61462	0,56540

Tabuľka 5.2: Zobrazujúca výsledky z testovania problému Cancer

5.3.3 Naivný Bayesov klasifikátor

Naivný Bayesov klasifikátor je posledný algoritmus testovaní na probléme Cancer. Výsledky z tabuľky 5.3 ukazujú, že Naivný Bayesov klasifikátor dosiahol najpresnejšie hodnoty pri testovaní problému Cancer zo všetkých nami testovaných algoritmov.

5.4 Próblem Card

Klasifikačný problém Card (kreditné karty) sa zaoberá predikciou schválenia alebo neschválenia udelenia kreditnej karty zákazníkovi v banke. Ide o skutočné údaje z banky a

	Bayes classifier
cancer1.dt	0,98275
cancer2.dt	0.96551
cancer3.dt	0.95597
cancer4.dt	0.97327

Tabuľka 5.3: Zobrazujúca výsledky z testovania problému Cancer

výstup predstavuje, či banka udelila alebo neudelila kreditnú kartu zákazníkovi. Problém Card bol testovaný pomocou algoritmu RCE a Naivného Bayesovho klasifikátora. Databáza príkladov obsahuje 51 vstupných údajov a 2 výstupné údaje. Pre diskrétnosť nie je uvedený význam jednotlivých atribútov. Ide o dosť zložitejší problém ako u Cancer, pretože vstupné hodnoty atribútov môžu byť nominálne alebo spojité (continuous), výstupné hodnoty sú binárne. Príklad z databázy môže vyzerať nasledovne: 1 0 0 0 1 0.178571 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0.298246 1 0 0 0 1 0 0 0 0 0 0 1. Testovanie problému Card prebiehalo s dátami zo súborov card1.dt až card3.dt. Tieto súbory obsahujú 345 príkladov na trénovanie a 172 na testovanie a validáciu. Súbor card4.dt obsahuje 500 príkladov na trénovanie a 500 na testovanie.

5.4.1 RCE

Pri testovaní algoritmu RCE je dôležité nastavenie parametrov Radius(Rmax) a Decrease ratio of radius(Rdc). Pri probléme Card boli v prvom testovaní hodnoty parametrov Rmax=2 a Rdc=1. V nasledujúcom testovaní boli hodnoty parametrov Rmax=2.75 a Rdc=1.325. Zistili sme, že výsledky boli horšie ako pri prvom testovaní. Tretí test bol realizovaní pri hodnotách parametrov Rmax=1.5 a Rdc=0.75. Výsledky dosiahli horšie hodnoty ako pri prvom testovaní. Tabuľka 5.4 zobrazuje testované hodnoty parametrov, ako aj výsledky.

	Rmax=2 Rdc=1	Rmax=2.75 Rdc=1.325	Rmax=1.5 Rdc=0.75
card1.dt	0.56976	0.51744	0.52906
card2.dt	0.54651	0.49418	0.51162
card3.dt	0.48837	0.50581	0.53488
card4.dt	0.49	0.496	0.488

Tabuľka 5.4: Zobrazujúca výsledky z testovania problému Card

5.4.2 Naivný Bayesov klasifikátor

Problém Card bol testovaný aj pomocou Naivného Bayesovho klasifikátora. Výsledky získané z testovania sú zobrazené v tabuľke 5.5.

	Bayes classifier
card1.dt	0.53488374
card2.dt	0.5755814
card3.dt	0.622093
card4.dt	0.794

Tabuľka 5.5: Zobrazujúca výsledky z testovania problému Card

Výsledky oboch testovaní nám ukázali, že menenie parametrov pri RCE nedosahuje vplyvné zmeny v úspešnosti správnej predikcii výsledku. Na druhej strane pri NBC, testovania u súborov card1.dt až card3.dt dosiahli trochu lepšie výsledky. Za použitia súboru card4.dt, ktorý obsahuje väčšiu dátovú množinu pre trénovanie a testovanie, sme dosiahli významnejšiu úspešnosť správnej predikcii výsledku.

5.5 Problém Diabetes

Klasifikačný problém Diabetes (cukrovka) sa zaoberá diagnostikovaním cukrovky ľudí v uzatvorenej indiánskej skupine - Pima indians. Predikcia je o tom, či jedinec má cukrovku alebo nemá. Je založená na vstupných atribútoch, medzi ktoré patria osobné údaje a lekárske testy. Databáza obsahuje 8 vstupných údajov, ktoré sú spojité (continuous) a 2 výstupy, ktoré sú binárne. Príklad vstupu z databázy môže vyzerať nasledovne 0.117647 0.875 0.721311 0 0 0.341282 0.105892 0.0166667 0 1. Testovanie problému Diabetes prebiehalo s dátami zo súboru diabetes1.dt až diabetes3.dt. Tieto súbory obsahujú 384 príkladov na trénovanie a 192 na testovanie a validáciu. Súbor diabetes4.dt obsahuje 700 príkladov na trénovanie a 900 na testovanie.

5.5.1 RCE

Pri testovaní algoritmu RCE je dôležité nastavenie parametrov Radius(Rmax) a Decrease ratio of radius(Rdc). Pri probléme Diabetes boli v prvom testovaní nastavené hodnoty parametrov Rmax=0.9 a Rdc=0.45. V nasledujúcom testovaní boli hodnoty parametrov znížené na Rmax=0.8 a Rdc=0.4. Zistili sme, že výsledky boli trochu horšie ako pri prvom testovaní, preto sme pokračovali ďalej v testovaní s rôznymi hodnotami parametrov. Tretí test bol uskutočnený pri hodnotách parametrov Rmax=1.4 a Rdc=0.7. Štvrté testovanie prebehlo s hodnotami parametrov Rmax=1.2 a Rdc=0.6. Posledné testovanie bolo uskutočnené pri hodnotách parametrov Rmax=1.0 a Rdc=0.5. Výsledky získané z testovania sú zobrazené v tabuľke 5.6.

Tabuľka 5.6: Zobrazujúca výsledky z testovania problému Diabetes

	Rmax=0.9 Rdc=0.45	Rmax=0.8 Rdc=0.4	Rmax=1.4 Rdc=0.7
diabetes1.dt	0.61979	0,57291	0,59375
diabetes2.dt	0.5625	0,47395	0,4375
diabetes3.dt	0.58333	0,60416	0,64062
diabetes4.dt	0.59097	0,58323	0,60808

	Rmax=1.2 Rdc=0.6	Rmax=1.0 Rdc=0.5
diabetes1.dt	0,58854	0,60416
diabetes2.dt	0,45833	0,4375
diabetes3.dt	0,64062	0,59375
diabetes4.dt	0,58631	0,6094

5.5.2 Naivný Bayesov klasifikátor

Problém Diabetes bol testovaný aj pomocou Naivného Bayesovho klasifikátora. Výsledky získané z testovania sú zobrazené v tabuľke 5.7.

	Bayes classifier
diabetes1.dt	6614583
diabetes2.dt	0.6458333
diabetes3.dt	0.6822917
diabetes4.dt	0.8600311

Tabuľka 5.7: Zobrazujúca výsledky z testovania problému Diabetes

Výsledky oboch testovaní nám ukázali, že menenie parametrov pri RCE nedosahuje vplyvné zmeny v úspešnosti správnej predikcii výsledku. Na druhej strane pri NBC, testovania v rámci súborov diabetes1.dt až diabetes3.dt dosiahli trochu lepšie výsledky. Za použitia súboru diabetes4.dt, ktorý obsahuje väčšiu dátovú množinu pre trénovanie a testovanie, sme dosiahli významnejšiu úspešnosť správnej predikcii výsledku na úrovní až 86%. Hypoteticky, s ešte väčšou množinou dát na testovanie a trénovanie mali by sme dosiahnuť ešte lepšie výsledky.

5.6 Problém Mushrooms

Klasifikačný problém Mushrooms (huby) sa zaoberá rozlišovaním, či huby sú jedlé alebo nejedlé. Podľa opisu huby, farby a jej vône sa prijíma toto rozhodnutie. Databáza obsahuje 125 vstupov, ktoré sú nominálne a 2 binárne výstupy. Príklad vstupu z databázy môže vyzeraťnasledovne 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0. Testovanie problému Mushroom prebiehalo s dátami zo súborov mushroom1.dt až mushroom3.dt. Tieto súbory obsahujú 4062 príkladov na trénovanie a 2031 na testovanie a validáciu. Súbor mushroom4.dt obsahuje 8124 príkladov na trénovanie a 4062 na testovanie. Ide o posledný testovaní problém v programe Classify.

5.6.1 RCE

Pri testovaní algoritmu RCE je dôležité nastavenie parametrov Radius(Rmax) a Decrease ratio of radius(Rdc) Pri probléme Mushrooms boli v prvom testovaní nastavené hodnoty parametrov Rmax=2.0 a Rdc=1.0. V druhom testovaní boli hodnoty parametrov Rmax=1.5 a Rdc=0.75. Získané výsledky po druhom testovaní boli v podstate rovnaké, ale pokračovali sme ďalej v testovaní. Tretí test bol uskutočnený pri hodnotách parametrov Rmax=8.0 a Rdc=4.0. Z troch uskutočnených testovaní nám vyšli najlepšie výsledky pri druhom teste. Výsledky získané z testovania sú zobrazené v tabuľke 5.8.

	Rmax=2.0 Rdc=1.0	Rmax=1.5 Rdc=0.75	Rmax=8.0 Rdc=4.0
mushroom1.dt	0,49581	0.49926	0.49433
mushroom2.dt	0.49532	0.48941	0.48202
mushroom3.dt	0.49187	0.50362	0.49876

Tabuľka 5.8: Zobrazujúca výsledky z testovania problému Mushrooms

5.6.2 Naivný Bayesov klasifikátor

Problém Diabetes bol testovaný aj pomocou Naivného Bayesovho klasifikátora. Výsledky získané z testovania sú zobrazené v tabuľke 5.9.

	Bayes classifier
mushroom1.dt	0,51550
mushroom2.dt	0.51058
mushroom3.dt	0.49039
mushroom4.dt	0.51403

Tabuľka 5.9: Zobrazujúca výsledky z testovania problému Mushrooms

Výsledky oboch testovaní nám ukázali, že menenie parametrov pri RCE nedosahuje vplyvné zmeny v úspešnosti správnej predikcii výsledku. Pozorujeme, že výsledky pri oboch algoritmov neboli značne diferenciálne. Riešenie problému Mushrooms bolo časovo náročnejšie ako u predchádzajúcich problémov.

5.7 Zhrnutie výsledkov a ďalší vývoj aplikácie

Zhrnutie ID3 z prevedených experimentov s použitím algoritmu ID3 na probléme Cancer sme zistili, že úspešnosť odhadu výslednej triedy bola značne vysoká, 88%.

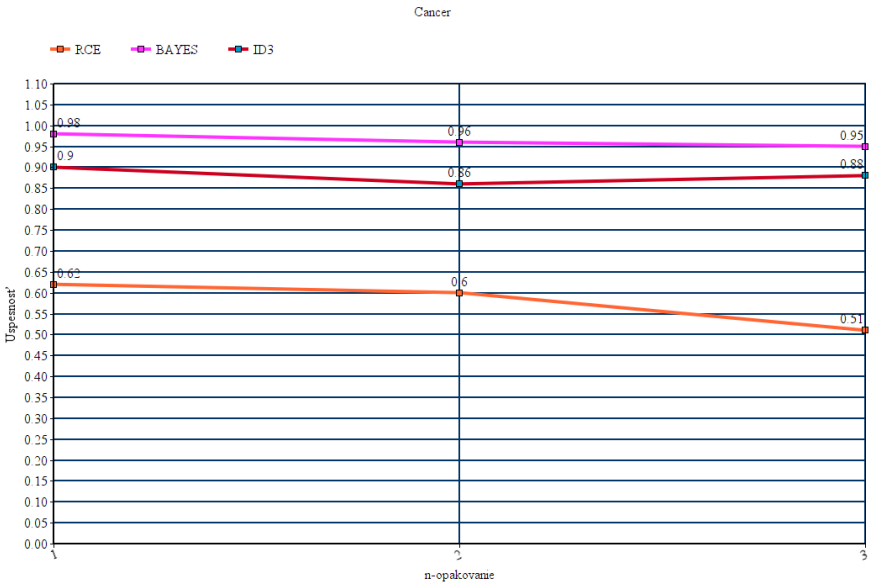
Zhrnutie RCE zo všetkých prevedených experimentov s použitím RCE siete bol výsledok pri probléme Cancer, keď parametre mali hodnoty Rmax=1.5 a Rdc=0.75. Pri týchto parametrov dosahovala RCE priemerne úspešnosť odhadu výslednej triedy 57%. Výsledky z testovania problému Cancer pri rôznych parametroch Rmax a Rdc nedosahovali uspokojivé výsledky. Pri testovaní ďalšieho problému Diabetes sme dosiahli najlepšie výsledky pri hodnote parametrov Rmax=1.25 a Rdc=0.625. Priemerná úspešnosť odhadu výslednej triedy bola 55.5%.

Testovanie problémov Cancer, Card a Diabetes odhalilo veľký vplyv hodnôt parametrov Rmax a Rdc na výsledok predikcie pomocou RCE siete. V probléme Mushrooms vplyv zmeny hodnôt paramentrov nebol výrazný. Celkové výsledky zo všetkých experimentov ukazujú na slabú úspešnosť odhadu výslednej triedy implementovanej pomocou RCE siete.

Zhrnutie Naivného Bayesovho klasifikátora z prevedených experimentov sme získali najlepší výsledok pri prvom probléme Cancer, pri ktorom úspešnosť predikcie správneho výsledku dosahovala takmer 100%. Pri probléme Card dosiahla úspešnosť odhadu riešenia v priemere okolo 58%. Pozorujeme, že rovnako ako v prípade Diabetes úspešnosť odhadu riešenia sa zvyšuje, keď trénujeme klasifikátor na väčšej množine dát. Následne môžeme pozorovať, že výsledky pri probléme Diabetes dosahovali úspešnosť priemerne okolo 66%. Pri väčšej trénovacej množine 700 príkladov na trénovanie a 960 na testovanie bola úspešnosť predikcie väčšia až okolo 87%. V prípade problému Mushrooms, ktorý mal najviac vstupov a bol časovo zložitejší oproti ostatným, dosiahla úspešnosť riešenia iba v priemere okolo 50%. Celkovo môžeme pozorovať a vyvodzovať, že pri dostatočne veľkej trénovacej sade, úspešnosť odhadu riešenia stúpa. Toto tvrdenie, ale neplatilo pri riešení problému Mushrooms, kde pri väčšej trénovacej sade, úspešnosť správnej predikcie výsledku nestúpala.

Obrázok 5.1 zobrazuje graf úspešnosti všetkých jednotlivých algoritmov pri riešení problému Cancer na troch rôznych dátových množinách. Na osi X sú zobrazené vstupné dátové

množiny 1 až 3. Na osi Y je zobrazená percentuálna úspešnosť správnej predikcie pre konkrétny algoritmus. Toto označenie os platí u všetkých grafov.



Obr. 5.1: Graf porovnania ID3,RCE a Bayesa pre Cancer

Porovnanie výsledkov problému Card pre RCE a Naivný Bayesov klasifikator je zobrazené v prílohe na obrázku 1. Môžeme pozorovať, že obe algoritmy mali pri prvých troch testovaniach približne rovnaké výsledky. Pri poslednom testovaní číslo 4 na väčšej testovacej množine dochádza k výrazným odlišným výsledkom. Zatiaľ čo úspešnosť predikcie NBC sa zvýšila na 79%, úspešnosť RCE siete výrazne klesla na 49%.

Porovnanie výsledkov problému Diabetes pre algoritmy RCE a NBC je zobrazené v prílohe na obrázku 2. Z grafu pozorujeme približne rovnakú úspešnosť predikcie odhadu riešenia pri NBC z prvých troch testovaní. Pri väčšej dátovej množine pre trénovanie a testovanie úspešnosť predikcie výslednej triedy dosahuje vyššie hodnoty.

Porovnanie výsledkov problému Diabetes pre algoritmy RCE a NBC je zobrazené v prílohe na obrázku 3.

Testovanie RCE siete pri všetkých problémov zavisí od zadanej hodnoty polomera hypergule a pomeru zmenšenia polomera (zvyčajne sa odporúča na polovicu). Pri experimentovaní s problémom Cancer bola najvyššia priemerná úspešnosť 57%.

Ďalší vývoj aplikácie Classify môže sa uberať viacerými smermi. Do úvahy pripadá implementácia nejakého ďalšieho klasifikačného algoritmu z vybraných smerov z tejto bakalárskej práce. Príkladom by mohli byť algoritmy C4.5 a C5.0 z kategórie rozhodovacích stromov, ktoré pracujú na rovnakom princípe ako opísaný ID3. Samostatný, tematický okruh neurónových sieti predstavuje veľa možností, z ktorých sa dajú čerpať nové algoritmy pre riešenia rôznych problémov. Vývoj aplikácie sa môže uberať aj ďalším novým tematickým smerom klasifikačných algoritmov, ktoré neboli detailnejšie spomenuté ako napríklad Support vector machines (SVM) alebo Random Forest (RF).

Kapitola 6

Záver

Cieľom záverečnej bakalárskej práce je teoretické priblíženie súčasných prístupov ku klasifikáciam v umelej inteligencii, implementácia vybraných klasifikačných algoritmov a následné experimenty so špecifickými problémami. V úvodnej časti bakalárskej práce (kapitola dva) definujeme a popisujeme teoretické poznatky o jednotlivých klasifikačných smeroch. Popisujeme jednotlivú terminológiu, modely, funkčnosť a ich využitia v praxi. V tejto časti vysvetľujeme aj vybrané klasifikačné smery: Rozhodovacie stromy, Neurónové siete a Bayesové klasifikátory. V tretej kapitole popisujeme konkrétnu implementáciu nasledovných algoritmov, algoritmus ID3 z rozhodovacích stromov, Restricted Coulomb Energy (RCE) neurónová sieť a Naivný Bayesov klasifikátor vo vytvorenom programe Classify.

Aplikačná časť práce je venovaná implementácii programu Classify, ktorý sme vytvorili a jeho primárnou úlohou bola demonštrácia jednotlivých klasifikačných algoritmov ID3, RCE a Naivného Bayesovho klasifikátora. V aplikačnom programe sme riešili štyri problémy, problém Cancer, Card, Diabetes a Mushrooms. Všetky riešené problémy pochádzajú z benchmarku PROBEN1, ktorý obsahuje dáta reálnych problémov pre neurónové siete. Z dôvodu praktickosti a experimentov z minulosti daný dataset bol vybraný na naše experimentovanie a testovanie s implementovanými algoritmani. Každý spomenutý problém bol podrobený testovaniu a experimentovaniu na daných algoritmoch. Prvý problém Cancer bol testovaní na všetkých troch algoritmoch, ďalšie problémy Card, Diabetes a Mushrooms boli testované pomocou algoritmov RCE siete a Naivný Bayesov klasifikátor. Každé testovanie jednotlivých problémov bolo uskutočnené na štyroch množinách dát získaných zo súborov. Proces trénovania a testovania algoritmov v programe Classify prebiehal nasledovne: výber algoritmu a problému, výber množiny vstupných dát, prípadne nastavenie parametrov pri RCE sieti, trénovanie algoritmu, testovanie už natrénovaného algoritmu pomocou množiny dát určených na testovanie a overenie výsledkov.

V piata kapitola informuje o zhodnoteniach a zhrnutiach získaných výsledkov. Na základe všetkých uskutočnených testovaní sme zistili, že Naivný Bayesov klasifikátor, ktorý bol testovaný na všetkých spomenutých problémov, dosiahol pri riešení najlepšie hodnoty úspešnosti predikcie odhadu výslednej triedy. Z uskutočnených testovaní algoritmus ID3 bol testovaný iba na probléme Cancer. Výsledky tohto testovania dosiahli vysokú úspešnosť. Algoritmus Reduced Coulomb Energy (RCE) neurónová sieť, ktorý bol testovaný na všetkých problémov, dosiahol na základe našich výsledkov najhoršiu úspešnosť predikcie. Záver piatej kapitoly je venovaný úvahe o ďalšom možnom vývoji aplikácie Classify.

Literatúra

[1] HALAŠ, M.: *Analýza vybraných metód a algoritmov dolovania v dátach.* . 2009, [Online; navštíveno 20.06.2020].
URL http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0910/halas/index.html

[2] Han, J.; Kamber, M.; Pei, J.: *Data Mining: Concepts and Techniques* . Morgan Kaufmann Publishers Inc., 2011, ISBN 0123814790.

[3] Haykin, S.: *Neural Networks: A Comprehensive Foundation* . Prentice Hall PTR, 1998, ISBN 978-0-13-273350-2.

[4] Kishan Mehrotra, S. R., Chilukuri K. Mohan: *Elements of Artificial Neural Networks.* 1996, ISBN 978-0-262-13328-9.

[5] Kotsiantis, S.: *Supervised Machine Learning: A Review of Classification Techniques.* 2007.
URL https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf

[6] Krawczak, M.: *Multilayer Neural Networks* . Springer, 2013, ISBN 978-3-319-03390-7.

[7] Krizhevskya, A.: Learning multiple layers of features from tiny images. 2009.
URL https://www.cs.toronto.edu/%7Ekriz/cifar.html

[8] LeCun, Y.; Cortes, C.: handwritten digit database. 2010.
URL http://yann.lecun.com/exdb/mnist/

[9] Machová, K.: *Strojové učenie - Princípy a algoritmy.* 2002.

[10] Prechelt: *PROBEN1- a Set of Neural Network Benchmark Problems and Benchmarking Rules.* 1994.

[11] Richard O.Duda, D. G., Peter E.Hart: *Pattern Classification* . Wiley, 2001, ISBN 0471056693.

[12] Schapire, R.: *Machine Learning Algorithms for Classification.*
URL http://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf

[13] Wikipedia: *Decision tree — Wikipedia, The Free Encyclopedia.* 2020, [Online; accessed 10.6.2020].
URL https://en.wikipedia.org/wiki/Decision_tree

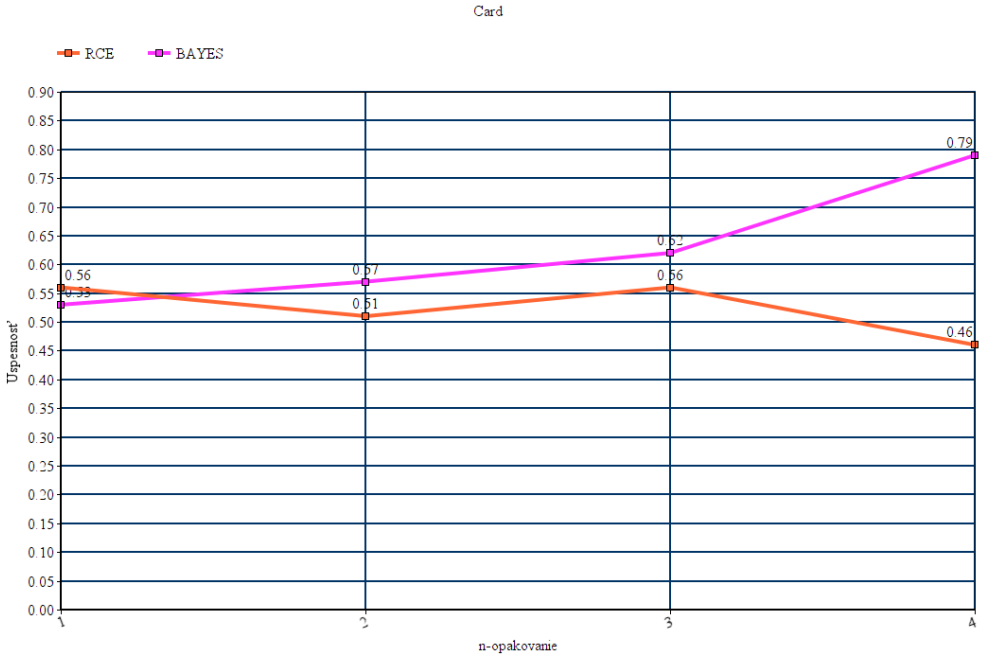
[14] Wikipedia: *ID3 algortihm — Wikipedia, The Free Encyclopedia.* 2020, [Online; accessed 5.5.2020].
URL https://en.wikipedia.org/wiki/Decision_tree

[15] Wikipedia: *Bayes' theorem — Wikipedia, The Free Encyclopedia*. 2021, [Online; accessed 10.4.2021].
URL https://en.wikipedia.org/wiki/Bayes%27_theorem

[16] Wikipedia: *Thomas Bayes — Wikipedia, The Free Encyclopedia*. 2021, [Online; accessed 10.4.2021].
URL https://en.wikipedia.org/wiki/Thomas_Bayes

[17] ZBOŘIL, F.: *Neuronové sítě s RBF neurony*. 2020, [Prednáška předmetu Soft computing vedená na Vysokém učení technickém v Brne na Fakulte informatiky].

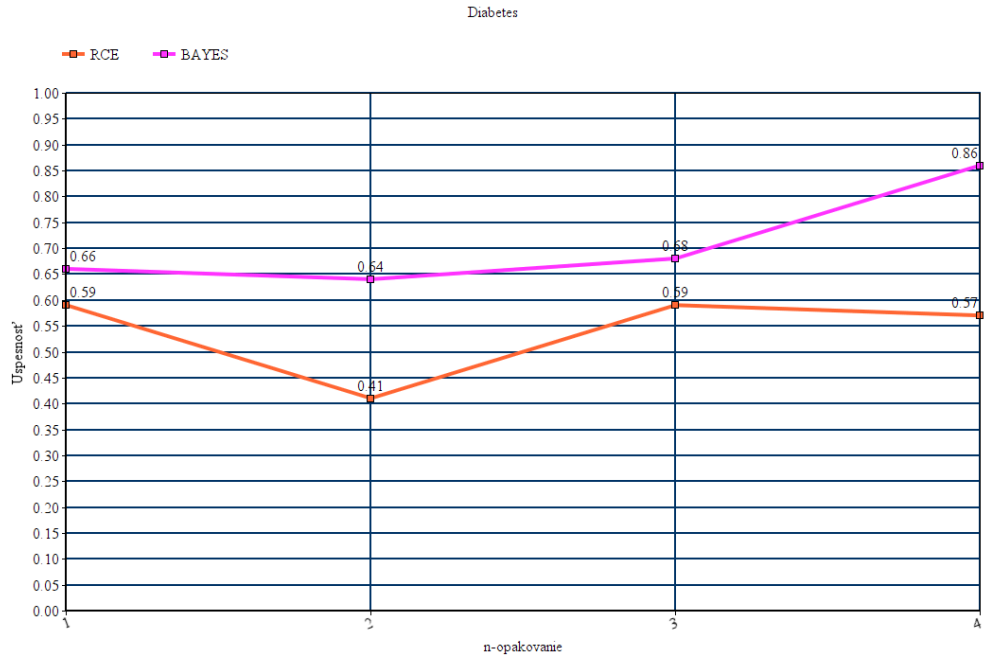
Prílohy



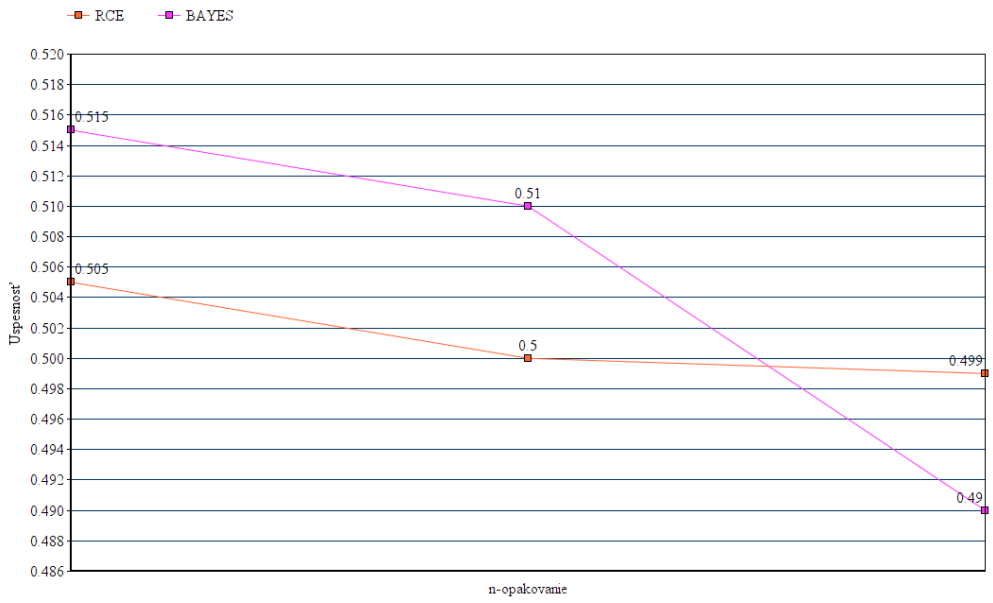
Obr. 1: Graf porovnania výsledkov pre RCE a Naivného Bayesa pri probléme Card

```
//if output class 2
if((Result_Vector_Output.get(this.boolOut+this.realOut-2).equals(0.0))
                    && (Result_Vector_Output.get(this.boolOut+this.realOut-1).equals(1.0)) ) {
                    for (Double hw : inputVector) {
                        if (!(hw.equals(null)))
                        {
                            if (Hash_Class2_words.containsKey(hw)) {
                                Hash_Class2_words.put(hw, Hash_Class2_words.get(hw) + 1);
                            } else {
                                Hash_Class2_words.put(hw, 1);
                            }
                        }
                        Number_of_words_OutClass2++;
                    }
                    Number_of_messages_OutClass2++;
                }
                Count_of_all_messages++;
                index++;
            }
        OutClass1_prior_probabbility = Number_of_messages_OutClass1 / (float) Count_of_all_messages;
        OutClass2_prior_probability = Number_of_messages_OutClass2 / (float) Count_of_all_messages;
}
```

Obr. 4: Implementácia trénovacej metódy v Naivnom Bayesovom klasifikátorovy časť 2



Obr. 2: Graf porovnania výsledkov pre RCE a Naivného Bayesa pri probléme Diabetes



Obr. 3: Graf porovnania výsledkov pre RCE a Naivného Bayesa pri probléme Mushroom