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 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

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:

- Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, Wiley & Sons, 2000
- Kotsiantis, S.B.: Supervised Machine Learning: A Review of ClassificationTechniques, Informatica 31, 2007
- 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

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 2019
Datum odevzdání: 31. července 2020
Datum schválení: 31. října 2019

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, 2020. 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
29. júla 2020

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

Cieľom tejto bakalárskej práce bolo si naštudovať problematiku prístupov ku klasifikáciám v umelej inteligencii. Po preskúmaní danej problematiky, vybrať niekoľko klasifikačných algoritmov, navrhnúť a implementovať program, ktorý demonštruje činnosti daných klasifikačných algoritmov. S použitím tohto programu následne urobiť experimenty o rôznych praktických klasifikačných problémoch, riešených danými algoritmami a zhodnotiť výsledky daných experimentov.

V tejto práci 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 uvedene v danej kapitole slúžia ako podklad pre ďalšiu kapitolu, ktorá je venovaná teoretickej aj praktickej časti, kde sú popísané konkrétne vybrané algoritmy a následne jednotlivé implementácie vybraných algoritmov.

V tretej kapitole je popísaný výber jednotlivých klasifikačných algoritmov, ktoré sa budú implementovať. Pre dané algoritmy je popísaný proces ich učenia a je uvedene ako boli dane algoritmy implementované.

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

Piata kapitola sa zaoberá zhrnutím vykonaných experimentov pre dané problémy a algoritmy, ktoré dané problémy riešili. Problém Cancer bol riešený všetkými implementovanými algoritmami. Problémy Card, Diabetes, Mushrooms boli riešené RCE neuronovou sieťou a bayesovým klasifikátorom. V tejto kapitole sú zhrnuté výsledky experimentovania. Na záver kapitoly je úvaha ako ďalej pokračovať s možným vývojom aplikácie.

Záver práce obsahuje zhrnutie nadobudnutých poznatkoch o klasifikačných algoritmoch, ktoré sa získali ako výsledky počas experimentovania.

Kapitola 2

Prehľad prístupov ku klasifikáciám

Táto kapitola predstavuje klasifikačné algoritmy. Opisuje rôzne vetvy smerovania klasifikácie od Rozhodovacích stromov, cez Neuronové siete až po Bayessovské klasifikátory. Na začiatku kapitoly je teoretický definovaný význam 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 sub-algoritmy, miesta využitia daných jednotlivých algoritmov. V poslednej časti sú rozobrané výhody a nevýhody vybraných algoritmov.

2.1 Klasifikácia

2.1.1 Informácie

Pretože v tejto bakalárskej práce sa často vyskytuje pojem klasifikácia, na začiatku si definujeme jej vyznám v našom kontexte.

Klasifikácia je v obore umelej inteligencie druh problému, ktorý určuje ako rozdeliť dáta do skupín na základe logického rozhodovania, tak aby čo najlepšie zodpovedala skutočnému roztriedeniu. Klasifikácia je teda príklad rozpoznávania vzorov.

V terminológii strojového učenia sa klasifikácia považuje za metódu učenia s učiteľom, kde máme k dispozícii tréningovú množinu správne klasifikovaných príkladov. 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 patri nový prípad na pozorovanie. Tato klasifikácie sa robí na základe trénovanej sady dat, obsahujúcej pozorovania, pre ktoré už je známa ich kategória. Táto klasifikácia dat prebieha pri tréningovej časti. Druha tréningová časť je keď sa klasifikátor snaží klasifikovať nové tréningové dáta.[15]

2.1.2 Výber algoritmu klasifikácie

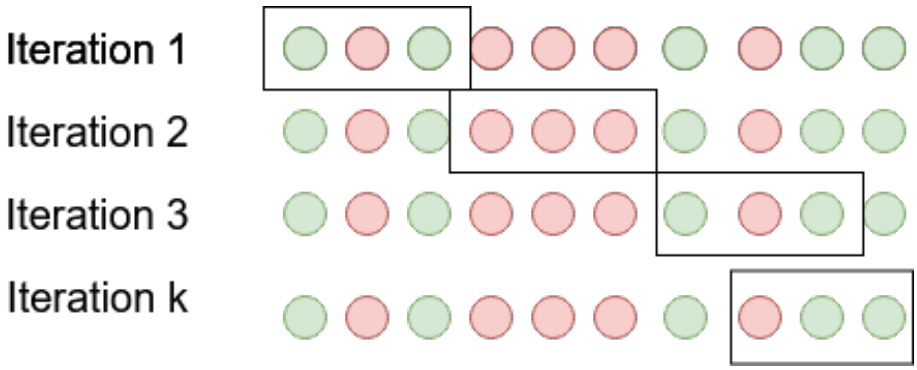
Výber algoritmu, ktorý použijeme na implementáciu klasifikácie je dôležitý krok. Akonáhle je predbežne testovanie hotové a sme spokojní s výsledkami, tak klasifikátor je pripravený k dispozícii na použitie. Ohodnotenie klasifikátora je často založené na presnosti predikcie

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

(počet správnych predikcii / celkový počet predikcii). Existujú aspoň 2 techniky, ktoré sa používajú na počítanie presnosti predikcie [7].

Prvá technika delí tréningovú množinu použitím dvoch tretín na tréning a poslednú tretinu na odhadovaný výkon.

Druhá technika je cross validation (krížová validácia). Pri tejto technike prichádza k rozdeleniu dátovej množiny, ktorá bude rozdelená na 2 časti, testovaciu a tréningovú. Pričom ale testovacia množina bude pri každej iterácii iná. Vždy sa vyberie iný subset. Pre každý subset sa spočíta chybovosť. Z týchto sa potom vypočíta priemerná chybovosť. Priemerná chybovosť celej dátovej množiny je odhadovaná ako priemerná chybovosť klasifikátora. Princíp križovej validácie je zobrazený na obrázku 2.1.



Obr. 2.1: Zobrazenie ako funguje cross validation

Ak ohodnotenie klasifikátora nie je uspokojivé, tak sa vraciame do predošlého kroku a preskúmame rôzne faktory napríklad relevantné vlastnosti problému, ktoré neboli predtým skúmané, komplexnosť problému je príliš vysoká, alebo vybraný algoritmus je nevyhovujúci.

Bežnou metódou na porovnávanie algoritmov je štatistické porovnanie presnosti trénovaných klasifikátorov na špecifickej dátovej skupine. Ak máme dostatok dát, môžeme vytvoriť niekoľko tréningových množín o veľkosti N. Na tieto množiny spustíme 2 algoritmy a pre každého z nich odhadujeme rozdiel v presnosti pre každý pár klasifikátorov na testovacej množine. Priemer týchto rozdielov je odhadovanou očakavannou rozdielovou chybou cez všetky možné tréningové množiny o veľkosti N. Ich variancia, rozptyl je odhadovaním rozptylom všetkých klasifikátorov v celej tréningovej množine.[7]

2.1.3 Druhy klasifikácie

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

- Logistic Regression
- Naive Bayes Classifier

- Support Vector Machines
- Decision Trees
- Random Forest
- Neural Networks

Logistic Regression, (logistická regresia), je štatistická metóda pre analýzu dátovej množiny, v ktorej je jedna, alebo viac nezávislých premenných, ktoré určujú výsledok. Výsledok sa meria pomocou dichotomickej premennej. V dichotomickej premennej (alebo tiež v tzv. binárnej premennej) existujú len 2 možné výsledky. Cieľom logistickej regresie je získať najvhodnejší model na opis vzťahu medzi dichotomickou charakteristikou a skupinou nezávislých premenných. Ide o lepší algoritmus 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 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 akékoľvek iného prvku. 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 i iné vysoko sofistikované klasifikačné metódy. Primárne použitie NBC je v oblasti spracovania prirodzeného jazyka.

Support Vector Machines (ďalej ako SVM) známy tiež ako metóda podporných vektorov. SVM je metóda slúžiaca na klasifikáciu a regresiu. Cieľom metódy je nájsť optimálnu nadrovinu, ktorá priestor príznakov rozdeľuje tak, že dáta z tréningovej metódy pátracie k odlišným triedam, ležia v opačných polpriestoroch. Optimálna nadrovina je taká, že hodnota minima vzdialenosti bodu od roviny je čo najväčšia. Uplatnenie SVM je hlavné pre klasifikáciu, reálna aplikácia je 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 pri data miningu a pri strojom učení.

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

Neural Networks (ďalej ako NS) známe tiež ako Neurónové siete. NS je to 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 na ňu napríklad nelineárnu funkciu a potom 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

Ciel tejto podkapitoly zoznámiť sa s podstatou rozhodovacích stromov. Popísať ich vlastnosti, komponenty, typy, heuristicke funkcie a algoritmy. Predstavuje framework CART.

2.2.1 Informácie

Rozhodovacie stromy známe tiež ako Decission Tree (DT) od teraz len DT, slúžia pre binárnu klasifikáciu objektov na základe ich vlastnosti. 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, kde pomáhajú určiť stratégiu, ktorá najpravdepodobnejšie dosiahne cieľ. Ďalšie a pre nás primárne použitie je v nástrojoch pri strojovom učení a data mining.

Pri DT je prirodzené a intuitívne klasifikovať vzor 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[14].

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 bud spojené s ďalšími vetvami alebo koncovým uzlom.

Klasifikácia určitého vzoru, vstupu začína v koreňovom uzle, ktorý sa pýta na konkrétnu vlastnosť zo vzoru. V každom kroku je vzor otestovaný podla otázky v aktuálnom uzle DT a pokračuje po uzle ktorý je shodny s konkrétnym výsledkom otázky. Na základe odpovede nasledujeme príslušný odkaz na nasledujúci, alebo nadradený uzol. Listy musia byť navzájom odlišné to znamená, iba jeden list bude nasledovaný. Ak vzor dôjde až ku koncovému uzlu je klasifikovaný triedou pre daný koncový uzol DT.

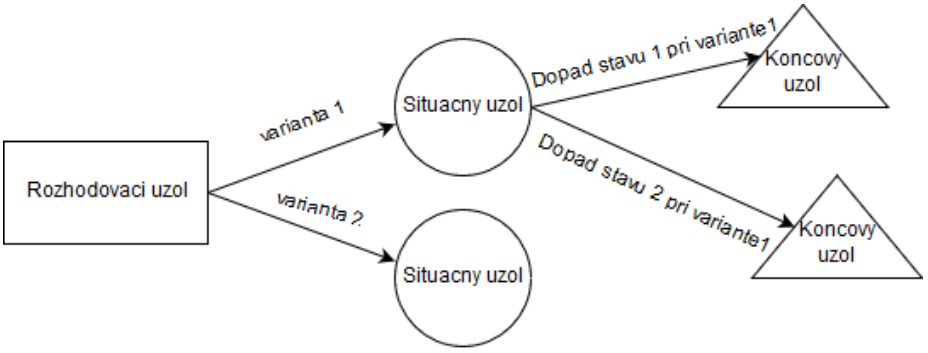
Existujú 3 typy uzlov [16], zobrazené na obrázku 2.2

- **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

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 máme klasifikačný problém.

Regresívne stromy sa využívajú v prípade, keď je premenná spojitá, alebo číselná, nie kategorická. Príkladom použitia je odhad ceny domu, ide o spojitú premennú, ktorej výsledok zaleží od ďalších spojitých premenných, napríklad rozloha domu a záhrady. Tento typ teda používame, keď máme problém odhadu (predikcie).



Obr. 2.2: Obrázok znázornujucí štruktúru DT

Pre konštrukciu optimálneho DT hľadáme efektívnu heuristiku. Čo znamená heuristika? Ide o metódu a spôsoby riešenia problémov, ktoré skracujú a zjednodušujú tradičné spôsoby riešenia a objavovania problémov. Funkcia, ktorá najlepšie rozdeľuje tréningovú množinu, je znázornená vo forme koreňového uzlu stromu. Poznáme niekoľko metód na nájdenie tejto funkcie pre rozdelenie DT napríklad : **Information gain**, **Gini index**, **Variance reduction**.

Entropia Aby sme dobre stanovili informačný zisk potrebuje zadefinovať entropiu. Tá charakterizuje (ne)čistoty v ľubovolnej skupine príkladov. Matematický vzorec pre entropiu je 2.1.

$$\mathbf{E(H)} = \sum_{\mathbf{i=1}}^{\mathbf{K}} -\mathbf{p_i} \log_{\mathbf{2}} \mathbf{p_i} \tag{2.1}$$

kde p_i je relatívna početnosť triedy i na určitej množine a K je počet tried. Pre vetvenie stromu sa vyberie atribút s najmenšou entropiou.

Information gain (ďalej ako IG) je množstvo informácii získaných o náhodnej premennej, alebo signálu z pozorovania inej náhodnej premennej. Pomocou IG, náhodnú premennú X získame z pozorovania náhodnej veličiny A, ktorá prijíma hodnotu A=a. Očakávaná hodnota IG je vzájomná informácia **I(X,A)** z X a A, takzvané zníženie entropie X dosiahnuté naučením stavu náhodnej premennej A. **Definícia**: Očakávaný IG je zmena entropie informácii H z predchádzajúceho stavu do stavu, ktorý berie určité informácie tak, ako je uvedené 2.2.

$$\mathbf{IG(T, a)} = \mathbf{H(T)} - \mathbf{H(T|a)} \tag{2.2}$$

kde H(T|a) je podmienená entropia vzhľadom na hodnotu premennej a.

Gini index meria mieru, alebo pravdepodobnosť nesprávnej klasifikácie konkrétnej premennej, ak je daná premenná vybraná. Tuto metriku využíva CART. Ak poznáme, že stupeň indexu Gini je 0, na základe toho vieme vydedukovať, že všetky prvky patria do danej konkrétnej triedy. Tento jav sa nazýva, že trieda je čistá. Naopak index Gini 1 označuje, že prvky sú náhodne rozdelené do rôznych tried. Definícia 2.3.

$$\mathbf{Gini(N)} = \sum_{\mathbf{i \neq j}} \mathbf{P(w_i)P(w_j)} = \mathbf{1} - \sum_{\mathbf{j}} \mathbf{P^2(w_j)} \tag{2.3}$$

, kde P je pravdepodobnosť, že objekt je priradený do konkrétnej triedy.

2.2.2 CART

CART je anglická skratka pre Classification and Regresion Trees. CART poskytuje užívateľom všeobecný framework [2], ktorý môže byť použití rôznymi spôsobmi na vytvorenie rôznych rozhodovacích stromov. Inými slovami termín CART označuje algoritmus rozhodovacieho stromu, ktorý môžeme použiť na klasifikáciu, ale i na regresívne problémy predikatívneho modelovania. Pri použití CART metódy sa kladie 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 mení na list?
- D. Ak sa stane strom 'príliš veľkým', ako sa dá zmenšiť a zjednodušiť?

A. Koľko listov bude vychádzať z jedného uzla ?

Počet listov z uzlu zavisí od druhej otázky. Vo všeobecnosti je počet rozdelenia stanovený autorom DT 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 binarných rozhodnutí.

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 na každom uzle N, 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.

$$\textbf{Entropy } \mathbf{H(X)} = -\sum_{\mathbf{j}} \mathbf{P(X_j)} \log \mathbf{P(X_j)} \tag{2.4}$$

kde $P(X_j)$ je zlomok vzoriek v uzle N, ktoré sú v kategórii j známe vlastnosti entropie, ak sú všetky vzory rovnakej kategórie,nečistota je 0; inak je to pozitívna, s najväčšou hodnotou vyskytujúcou sa, keď sú rôzne triedy rovnako pravdepodobné.

Existujú ďalšie matematické merania nečistôt ako odchýlka nečistôt (variance impurity) alebo Gini nečistota (Gini impurity) 2.3.

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 to neovplyvní rozdelenie. Autori DT zvyčajne vyberajú funkcie, ktoré sa dajú ľahko vypočítať, napríklad na základe jedinej funkcie alebo atribútu, ktoré nám poskytnú monotetický strom [14].

[2] Framework je štrukturá, ktorá slúži na podporu pri zostavovaní

C. Kedy sa uzol mení na list ?

Táto otázka vzniká pri rozhodnutí, keď máme prestať rozdeľovaním počas tréningu 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 na tréningových údajoch nie je dostatočne nízka, a preto môže výkon trpieť. Ako teda rozhodneme, kedy prestať rozdeľovať strom ? Na základe tradičného prístupu križovej validácie (cross-validation), princíp križovej validácie uplatňuje nasledovné, strom je vyškolený pomocou podmnožiny údajov (90%), pričom zvyšných (10%) sa uchováva ako dáta na validáciu. Pokračujeme v rozdeľovaní uzlov v nasledujúcich vrstvách, až pokiaľ chyba na právoplatných údajov je minimálna [14].

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 dopredu, vznik takzvaného fenoménu 'horizontový efekt'. Určenie optimálneho rozdelenia v uzle N nie je ovplyvnené rozhodnutiami na N odvodených uzloch, t.j. na následných úrovniach. Pri zastavení rozdelení môže byť uzol N vyhlásený za list, čím sa preruší možnosť výhodných rozdelení v nasledujúcich uzloch; ako taký, 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 tréningovej množine.

Existujú i ďalšie metódy, ktoré sú ale konceptuálne rozdielne od metódy prerezávania ako napr. metoda založená na pravidlách (rules based method) [14].

2.2.3 Algoritmy a zhrnutie rozhodovacích stromov

DT sa najbežnejšie vyskytujú pri analýze rozhodnutia (decesion 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 3.3 a je implementovaný v rámci tejto bakalárskej prace. 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 body, teda zvláda problém neúplných údajov veľmi dobre, rieši problém presahovania a môže použiť rôzne hmotnosti na vlastnosti z tréningových dát. Ako metriku používa zisk informácii 2.2 pri generovaní DT. Dôležitá vlastnosť je, že C4.5 má v sebe zahrnuté prerezávanie, na zmiernenie preplnenia stromu. Pre tento proces sa používa prerezávanie jedným priechodom.

Autor ID3 a C4.5 Ross Quinlan pokračoval ďalej vo vylepšovaní algoritmu a vynašiel nový algoritmus C5.0, ktorý ponúka ďalšie vylepšenia od C4.5. Vylepšenia sa hlavne zameriavajú na rýchlosť počítania, úsporu pamäti pri počítaní, zmenšenie rozhodovacích stromov pri C5.0 s rovnakým výsledok ako pri C4.5. Medzi ďalšie vylepšenia patrí zosilňovanie (boosting), ktorý ďalej vylepšuje rozhodovacie stromy a dáva im väčšiu presnosť.

Výhody.

DT majú viacero špecifickych 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 napr. klasifikovanie neznámych záznamov veľmi rýchlo, algoritmus pracuje veľmi dobre aj ak máme prípad, ktorí obsahuje veľa nepotrebných záznamov. DT sú veľmi silné, ak v vstupnej dátovej množine je veľa zhluku (noise), pripadne ak sú k dispozícii metódy ako nadmerné vybavenie (overfitting). Nepresnosť komplexného rozhodnutia je minimalizovaná, čo vedie k priradeniu presných hodnôt k výsledku rôznych akcií. Kategorizuje prvky z dátových množín bez väčšieho výpočtu.

Nevýhody.

Spoľahlivosť výstupných dát z DT a informácii získane z DT zavisí 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 v 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 hodnoť, 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 a iracionálne očakávania môžu viesť k chybám v DT. Hoci DT sleduje prirodzený priebeh udalostí, sledovaním vzťahov medzi 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.

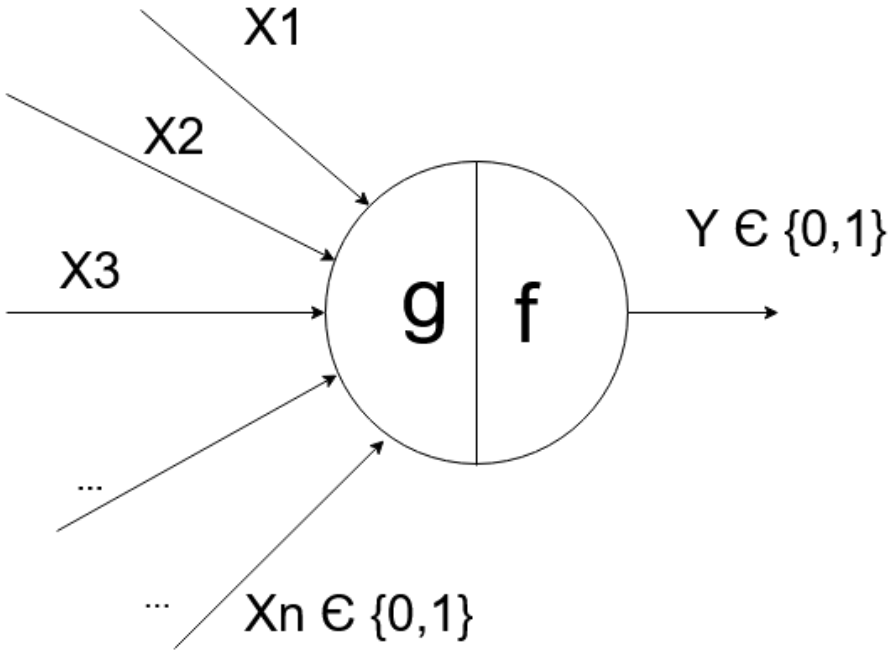
2.3 Neuronové siete

V tejto podkapitole si predstavíme Neuronové siete, základné prvky neurónových sieti ako neurón, modely neurónu, rôzne štruktúry sieti, prenosové funkcie a využitie NS 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 neuronom. Existujú rôzne modely umelého neurónu, od základného (ktorý si uvedieme v ďalšej časti) cez model semi-lineárnej jednotky, Nv neuróny po McCulloch-Pitts(MCP).

Počiatky neurónových sieti siahajú až do obdobia druhej svetovej vojny, keď Warren MuCulloch a Walter Pitts navrhli prvý matematicky model biologického neurónu v roku 1943.[11] Tento model zobrazený na obrázku 2.3 sa nazýva McCulloch-Pitts neurón. Tento neurón sa dá rozdeliť na dve časti, kde prvá vstupná časť **g** urobí agregáciu a podľa tejto agregovanej hodnoty druhá časť **f** urobí rozhodnutie.



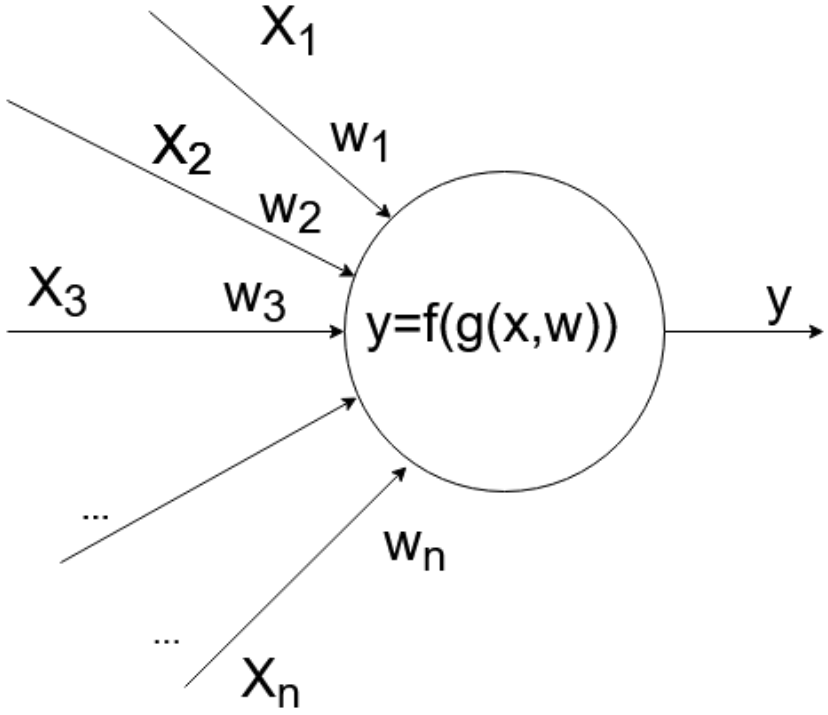
Obr. 2.3: Model McCulloch-Pitts neurónu

Vstupné hodnoty sú typu boolean 0,1 rovnako aj výstupné hodnoty sú boolean. Pri vstupných hodnotách ešte rozlišujeme, či sa jedná o recitačné alebo inhibičné. Inhibičné majú hlavný účinok na rozhodovanie bez ohľadu na druhé vstupy, teda ovplyvňujú spustenie neurónu. Excitačné vstupy sami o sebe neovplyvňujú spustenie neurónu, musia byť viaceré dokopy. Model využíva Booleovské funkcie ako AND, OR, NOT na rozhodnutia. Tento model mal veľa nedostatkov, hlavným problémom bolo, ak sme chceli použiť iné ako booleanovské vstupy napríklad reálne hodnoty.

Dnes sa skôr používa všeobecný základný model umelého neurónu , ktorý si predstavíme a bude stredobodom tejto štúdie o neurónových sieťach. Všeobecný neurón má n vstupov x_i

a jeden výstup y, ktorý je bud napojený na ďalší neurón, alebo môže byť priamo výstupom neurónovej siete.

Matematický model základného neurónu



Obr. 2.4: Schéma umelého neurónu

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

Výstup neurónu je analogický axónu biologického neurónu. Táto výstupná hodnota sa ďalej buď šíri ako vstup ďalšej vrstvy pomocou synapsie (ak je neurón súčasťou viacvrstvovej siete), alebo opúšťa systém ako súčasť výstupného vektora. Neurón sám o sebe nemá žiadny vzdelávací proces. Váhy majú stanovenú hodnotu, ktorá určuje citlivosť s akou vstupy pôsobia na výstupy a pomocou prenosovej funkcie sa vypočíta výstup. Bázová f funkcia priradí reálne číslo u, potenciál neurónu, vstupnému vektoru $\mathbf{x}$ a váhovému vektoru $\mathbf{w}$.

Aktivačná (prenosová) funkcia g je vyberaná s cieľom vylepšiť, alebo zjednodušiť sieť, ktorá obsahuje neurón. Definuje výstup z neurónu vzhľadom na vstup. V kontexte biologického neurónu ide o abstraktnú vlastnosť, ktorá reprezentuje pravdepodobnosť, že sa neurón aktivuje. Môže byť lineárna alebo nelineárna. Aktivačná funkcia priraďuje k potenciálu u vystúp neurón. V najzákladnejšej forme je funkcia binárna, teda funkcia sa spusti za hodnoty **1**, naopak sa neaktivuje pri hodnote **0**. Príklad niekoľko aktivačných funkcií

- **Skoková prenosová funkcia**, známa ako tiež binárna funkcia: vracia pre vstup, ktorý je menší ako určená hranica nulu, pre väčší vstup vracia jedna, f(x)=0 pre x $<$ 0, f(x)=1 pre x $\geqslant$ 1.
- **Sigmoidná prenosová funkcia**, ktorá funguje nasledovne: ak sa blížia hodnoty k mínus nekonečno, tak funkcia vracia hodnotu 0. Naopak ak sa hodnoty blížia k plus nekonečno tak funkcia vráti hodnotu 1.
- **Funkcia identity**, ktorá vždy vracia rovnakú hodnotu ako bol vstup.

Výpočet celého neurónu je teda možne opísať ako y=g(f(x,w)), kde f je bazová a g aktivačná funkcia.Tento princíp je zobrazený na obrázku 2.4. Bazové funkcie sa dajú ďalej rozdeliť na lineárne alebo radiálne.

Lineárna bazová funkcia (LBF) je definovaná ako 2.5.

$$u = \sum_{i=1}^{n} w_i.x_i, \tag{2.5}$$

teda zodpovedá skalárnemu súčinu vektorov **x** a **w**.

Radiálna bazová funkcia (RBF) je definovaná ako 2.6.

$$u = \sqrt{\sum_{i=1}^{n}(x_i - w_i)^2}, \tag{2.6}$$

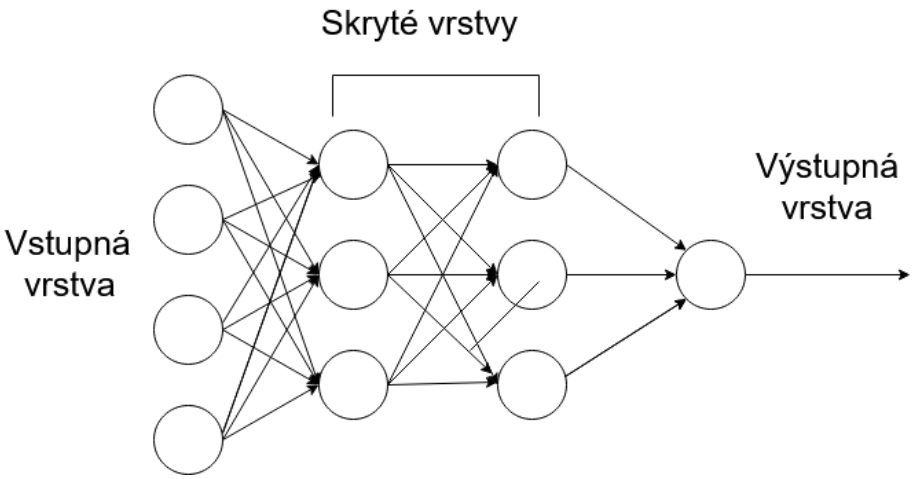
teda zodpovedá euklidovskej vzdialenosti vektorov **x** od vektoru **w**, dosahuje len nezáporne hodnoty.

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

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 viacvrstvovej NS 2.5.



Obr. 2.5: Model viacvrstvovej neurónovej sieti

Model siete sa skladá z viacerých vrstiev: vstupnej vrstvy, kde prichádzajú počiatočné vstupné dáta, skrytá vrstva (môže byť viacero skrytých vrstiev) tu sa vykoná výpočet a výstupná vrstva, 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) a acyklická (dopredná z angl. feed-forward). 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 aj výstupná.

Typickým príkladom siete s cyklickou topológiou je Hopfieldová sieť. 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.

Delenie podľa učenia NS. Učenie v kontexte neurónových sietí je proces ,ktorým sa adaptujú voľné parametre neurónovej siete pomocou stimulácie prostredím, v ktorom je neurónová sieť zahrnutá . Druh učenia je určený spôsobom, čim sa uskutočnia zmeny parametrov[5]. 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 teda 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.

Medzi základne typy NS patria napríklad perceptron, viacvrstvová sieť, rekurentná sieť, Hopfieldová sieť alebo Kohenová sieť. V tejto práci si bližšie popíšeme 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 ľubovolný 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

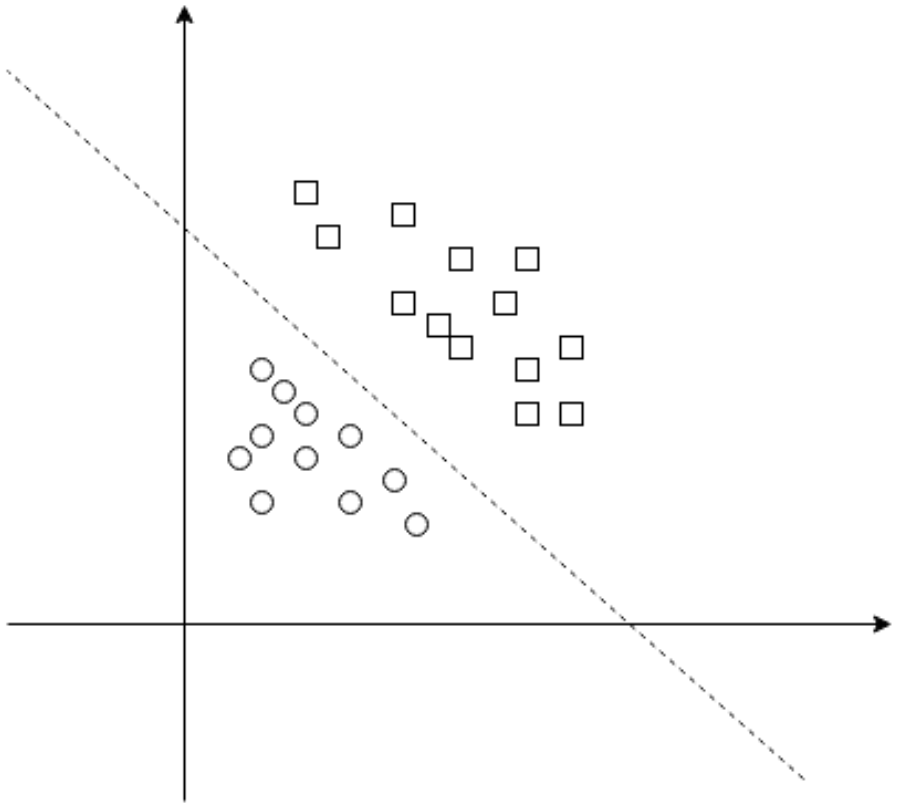
Perceptrón je typom neurónu, ktorý predstavuje všeobecný výpočtový prvok NS. Vnútorný potenciál perceptronu je počítaný ako vážený súčet vstupov ξ.

Aktivačná funkcia je Sigmoidná prenosová funkcia.

$$f(\xi) = \frac{1}{1 + e^{-\lambda\xi}} \tag{2.7}$$

Perceptrón je jednovrstvová neurónová sieť s učením s učiteľom a viacvrstvový perceptrón nazývame NS. Bol navrhnutý americkým vedcom Frankom Rosenblattom[6] a je určený na dichotomickú klasifikáciu, tj. rozdelenie do dvoch tried, pri ktorých sa predpokladá, že triedy sú lineárne separovateľné v príkladovom priestore. Ako to teda funguje ? Zoberme si príklad, 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.

Príklad lineárnej separovateľnosti v rovine2.6.



Obr. 2.6: Príklad lineárnej separovateľnosti

Matematická definícia perceptronu je uvedená nasledovne : Perceptron je binárny klasifikátor, ktorý mapuje vektor vstupov $x = [x1, x2, ..., xn]$ na výstupné hodnoty f(x). Kde w je vektor váh a b je konštanta.

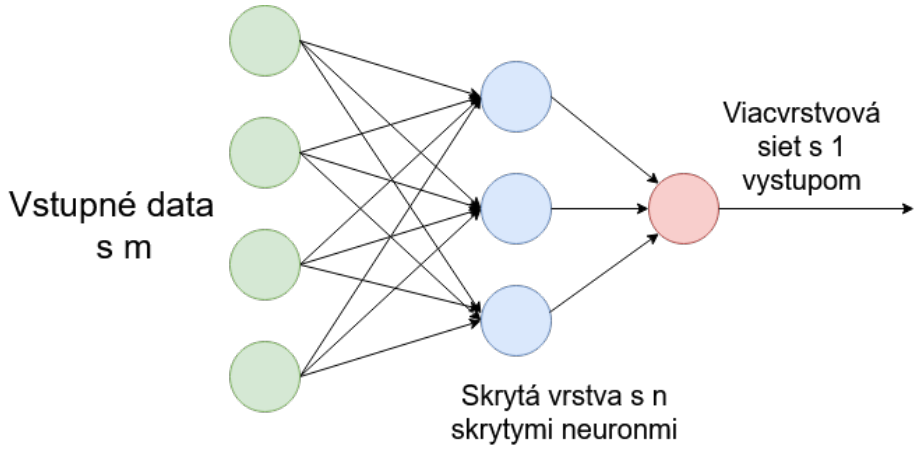
$$f(x) = \begin{cases} 0 & \text{otherwise} \\ 1 & \text{if w.x+b>0} \end{cases} \tag{2.8}$$

Základné použitie perceptrónu je pre lineárne separovatelné obrazy. Upravený perceptrón môžeme použiť i na klasifikovanie neseparovateľných obrazov.

2.3.4 Viacvrstvová neurónová sieť

Viacvrstvová neurónová sieť[8] 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 neurónov, skrytej vrstvy, ktorá môže byť n vrstvová a výstupnej vrstvy, ktorá obsahuje l výstupný neurón.

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



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

Na obrázku je zobrazená sieť s jednou skrytou vrstvou. Hodnota M znázorňuje počet vstupných dát od $m_1, m_2, ..., m_n$, N je počet neurónov v skrytej vrstve. Vstupne neuróny neuskutočňujú žiadny výpočet, iba poskytujú vstupné informácie do siete. Pre každý neurón v skrytej vrstve si vypočítame jeho hodnotu pomocou vzorca $W.X = w_1.x_1 + w_2.x_2 + ... + w_m.x_m = \sum_{i=1}^{m} w_i.x_i$ w je váha. Keď budeme mať spočítané všetky neuróny skrytej vrstvy môžeme spočítať výstupný neurón podľa rovnakého princípu.

Ako sme si uviedli viacvrstvová sieť je veľmi podobná perceptrónu, v podstate je to len rozšírenie perceptrónu. Využitie viacvrstvovej siete je širšie ako pri perceptróne. Hlavné využitie je nasledovných oblastiach: pri problematike rozpoznávania hlasu, klasifikovania neseparovatelných obrazov a v oblasti strojového prekladu.

2.3.5 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.

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 patria napríklad: 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 štatického učenia a zároveň jedným z hlavných zameraním tejto bakalárskej práce sú Bayessové klasifikátory/siete.

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). Ten riešil problém ako vypočítať rozdelenie pravdepodobnostného parametra binomického rozdelenia.

Bayesová veta (Bayesov Theorem) sa zaoberá podmienenými pravdepodobnosťami javov. Formálny zápis podmienenej pravdepodobnosti javu A za predpokladu výskytu javu B používame zápis P(A|B), a naopak P(B|A) je pravdepodobnosť javu B podmienená výskytom javu A. Thomas Bayes zistil, že podmienená pravdepodobnosť P(A|B) súvisi s opačne podmienenou pravdepodobnosťou P(B|A) a to nasledovne: ak máme dve náhodné javy A a B, ktorých pravdepodobnosti sú P(A) a P(B) a pokiaľ platí $P(B) > 0$, potom platí nasledujúci vzťah.

$$\mathbf{P}(\mathbf{A}|\mathbf{B}) = \frac{\mathbf{P}(\mathbf{A}) * \mathbf{P}(\mathbf{B}|\mathbf{A})}{\mathbf{P}(\mathbf{B})} \tag{2.9}$$

P($\mathbf{A}$|$\mathbf{B}$) je podmienená pravdepodobnosť javu A, za predpokladu výskytu javu B
P($\mathbf{B}$|$\mathbf{A}$) je podmienená pravdepodobnosť javu B, za predpokladu výskytu javu A
P($\mathbf{B}$) a P($\mathbf{A}$) sú 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($\mathbf{A}$)-Pravdepodobnosť zvýšenia akcií o 5%.
P($\mathbf{B}$)-Pravdepodobnosť výmeny generálneho riaditeľa.
P($\mathbf{A}$|$\mathbf{B}$)-Pravdepodobnosť zvýšenia ceny akcií o 5% vzhľadom na to, že generálny riaditeľ bol vymenený.

P(**B**|**A**)-Pravdepodobnosť výmeny generálneho riaditeľa vzhľadom na cenu akcií sa zvýšila o 5%.

Príklad reálneho použitia Bayesovej vety na nájdenie požadovanej pravdepodobnosti.

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

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

Naivná Bayesovká klasifikácia je jednoduchá na implementáciu a dosahuje nízkú chybovosť, keď porovnávame výsledky z iných algoritmov na princípe učenia s učiteľom. Nepotrebuje veľké množstvo dat pri vytváraní modelu[4]. Naivné Bayesové klasifikátory predstavujú klasifikačný algoritmus pre binarné (dvojtriedne) a viactriedné klasifikačné problémy. Ide o model pravdepodobnostní, ktorý využíva grafovú reprezentáciu pre zobrazenie pravdepodobnostných vzťahov medzi jednotlivými javmi. Skladajú sa z riadených acyklických grafov, kde je iba jeden rodič uzol (ktorý predstavuje nepozorovaný uzol a niekoľko synovských uzlov (ktoré predstavujú zodpovedajúce pozorované uzly) s predpokladom, že platí nezávislosť medzi detskými uzlami a ich rodičom.[7] Názov Naivné Bayesove klasifikátory je odvodený podľa toho, že výpočty pravdepodobnosti pre každú triedu sú zjednodušené, čo umožňuje sledovať ich výpočty.

Vzťah medzi Naivnými Bayesovskými klasifikatormi je vyjadrený nasledujúcim matematickým vzorcom 2.10. Naivní bayesovský klasifikátor vychádza z predpokladu, ze jednotlive operácie E_1,...,E_k sú podmienene nezávisle pri platnosti hypotézy H[1].

$$\mathbf{P(H|E1}, ..., \mathbf{EK}) = \frac{\mathbf{P(E1}, ..., \mathbf{EK|H}) \times \mathbf{P(H)}}{\mathbf{P(E1}, ..., \mathbf{EK})} \tag{2.10}$$

Pri používaní tohto modelu ale môže nastať problém, kedy podmienená pravdepodobnosť P($X_k|C_i$) je rovná 0. 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). Pri Laplacelovej korekcie dôjde k pridaniu jedného prvku 1 do všetkých množín. Korekcia vychádza z teorie, že pridanie iba jedného príkladu neovplyvní pravdepodobnosť výrazne. Vyhneme sa ale nulovej hodnote pravdepodobnosti[3].

Bayesové klasifikátory sú menej úspešne pri klasifikacii vo svojich výpočtov ako ostatné viac sofistikované klasifikačné algoritmy, ako napríklad Neuronové siete. Dôvodom tejto nepresnosti je predpoklad nezávislosti medzi detskými uzlami[7].

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 dobre výsledky i pre súbor dat, kde sa vyskytuje veľká závislosť medzi prvkami[7].

Základný nezávislý Bayesov model bol časom mnohokrát upravený rôznymi spôsobmi s cieľom vylepšenia jeho výkonu. Časť pokusov na prekonanie predpokladu o nezávislosti boli založené 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[7].

Po predstavení základného Bayesovho modelu a jeho hlavného nedostatku môžeme v krátkosti diskutovať o ďalších rôznych rozličných Bayesovských klasifikatorov. Zameriame sa na klasifikátory, ktoré sa rozlišujú hlavne v predpokladoch, ktoré robia v súvislosti s distribúciou P(Xi|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-predikatory používané kvalifikátorom sú frekvencia slov výskytu pritomných v dokumente.

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

Zhrnutie Naivné Bayesové klasifikátory sa väčšinou používajú pri analýze sentimentu, filtrovania spamu, doručovacích systémov atď. Implementujú sa jednoducho a rýchlo, ale ich najväčšia nevýhoda je v požiadavke na to, aby boli predikatory nezávisle. V skutočnosti sú vo väčšine prípadov predikatory závislé, čo bráni optimálnemu výkonu klasifikátora.

2.4.3 Bayesové siete

Bayesová sieť (Bayesian networks) predstavuje pravdepodobnostný model, ktorý používa grafickú reprezentáciu pre zobrazenie pravdepodobnostných vzťahov medzi jednotlivými javmi. Siete sa využívajú na určenie pravdepodobnosti určitých javov, pričom siete vychádzajú zo základov teórie pravdepodobnosti. Štruktúra Bayesovské siete **S** pozostáva z acyklického riadeného orientovaného grafu(DAG), kde uzly v **S** sú v vzájomnej korešpondencii s vlastnosťami **X** [7]. Teda definícia pre graf je **G=(S,X)**, kde S predstavuje konečnú množinu vrcholov (uzlov) a X je množina hrán (vlastnosti). Každý uzol odpovedá jednej náhodnej premennej, kde každý graf typicky obsahuje niekoľko premenných-uzlov. Orientované hrany medzi uzlami predstavujú vzťahy podmienenej závislosti medzi premennými, zakresľujeme ich pomocou šipiek $\leftarrow, \rightarrow$. Usporiadaná dvojica (u,v), ktorá patrí do X vyjadruje orientovanú hranu z vrcholu u do vrcholu v, kde u sa nazýva rodič (parent) vrcholu v a v je potom potomok (child) vrcholu u.

Oblúky predstavujú neformálne vplyvy medzi vlastnosťami, zatiaľ čo nedostatok možných oblúkov v sieti S zakodováva podmienené nezávislosti. Funkcia(uzol) je podmienečne nezávislá od svojich ne-potomkov vzhľadom na svojich rodičov(X1 je podmienečne nezavisly od X2).

Všetky premenné v grafe sa vzťahujú k neznamým javom, kde každá premenná je reprezentovaná jedným uzlom a hrany (vzťahy) medzi uzlami zobrazujú pravdepodobnostnú závislosť medzi vybranými veličinami. Tieto závislosti počítame na základe štatistických metód.

Definícia Bayesovskej sieťe N je daná ako trojica (V, A, P), kde [2]:

- 1. V je množina premenných
- 2. A je množina orientovaných hrán, ktoré spolu s V tvoria orientovaný acyklický graf G = (V, A)
- 3.P=P(v|π_v);v ϵV ,π_v je množina rodičov v. Teda **P** je množina podmienených pravdepodobností všetkých premenných, podmienených ich rodičmi.

Združená pravdepodobnosť P_n Bayesovej siete N je definovaná ako 2.11.

$$P_n(V) = \prod P(v|\pi_v) \tag{2.11}$$

Vzorec 2.11 vyjadruje spoločnú funkciu združenej pravdepodobnosti s ohľadom na mieru produktu, ktorá je ako súčin jednotlivých združených pravdepodobnosti, podmienených na ich rodičovských premenných.

Definícia pravdepodobnostnej inferencie je znázornená vzorcom 2.12

$$P(x|e) = \alpha \sum P(x, e, Y) \tag{2.12}$$

Na základe vzorca je možné definovať pravdepodobnosť priradenia podmnožiny premenných X, ku priradeniu iných premenných.

Učenie sa štruktúry Bayesovskej siete je problém, ktorému sa venuje oblasť strojového učenia sa. Samotné učenie Bayesovej siete môže byť rozdelene do dvoch podkategórii: určenie štruktúry siete a následne určenie jeho parametrov. Najprv je potrebné definovať štruktúru-topológiu grafu a potom nastaviť jeho parametre rozdelenia podmienenej pravdepodobnosti pre jednotlivé vrcholy v grafe. Určenie parametrov je jednoduchšie ako určenie štruktúry. Rovnako jednoduchšie je keď, poznáme všetky vrcholy, ako keď niektoré by boli

skryté, alebo keď chýbajú dáta k vrcholom. V tabuľke 2.4.3 [2] sú zobrazene 4 metódy učenia Bayesovských sietí.

Štruktúra	**Pozorovateľnosť**	**Metóda**
Známa	Úplná	Maximálny odhad vierohodnosti
Známa	Čiastočná	EM alebo rast gradientu
Neznáma	Úplná	Prehľadávanie oblasti modelu
Neznáma	Čiastočná	EM + prehľadávanie oblasti modelu

Tab. 1: Metódy učenia bayesovských sietí v závislosti od požiadaviek

Teoretické využitie Bayesových sietí je v modelovaní situácii, kde sa vyskytuje určitý faktor neurčitosti, a kde je cieľom získať inteligentné a kvantifikované rozhodnutie, ktoré bude maximalizovať šancu na želaný výsledok. Príklad všeobecného použitia z bežnej praxe môže byť uplatnený analýza rizík, použitie v oblasti spracovania textu alebo obrazu, v aplikáciach v expertných systémoch. Medzi exotickejšie použitie môžeme brať testovanie predpovedania futbalových výsledkov[13].

2.4.4 Zhrnutie Bayesových klasifikátorov a sieti

Hlavná výhoda je jednoduchosť a zrozumiteľnosť, takže i obyčajný používateľ môže vďaka pomerne jasnej štruktúre sieti a vzťahom medzi premennými dokáže pochopiť konkrétny reprezentovaný model pravdepodobnostní. Nevýhoda je, že klasifikátor sa nehodí na množiny s veľkými počtami atribútov a taktiež naivne predpokladá nezávislosť jednotlivých atribútov.

Kapitola 3

Implementované klasifikačné algoritmy

V tejto kapitole sa najprv rozoberá konkrétny výber klasifikačných algoritmov na riešenie klasifikačných problémov. Následne je vysvetlený popis implementácii konkrétneho algoritmu v aplikácii Classify.

Pre praktickú implementáciu programu Classify je najprv potreba určiť klasifikačné algoritmy, ktoré sa budú implementovať. Algoritmy boli vybrané z pred tým skúmaných kategórii klasifikačných algoritmov. Pre účel tejto bakalárskej práce boli vybrané RCE neurónová sieť, algoritmus ID3 a Bayesov klasifikátor.

3.1 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 3.1. Počet skrytých neurónov je presne daný a mení sa dynamicky. Parameterr_k určuje polomer hypergule. U_k je parameter určujúci vzdialenosť medzi vstupným vektorom a vektorom váh.

$$u_k = \sqrt{\sum (i - w_k)^2} \tag{3.1}$$

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 2.6 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 3.2.

$$y_k = \begin{cases} 1 & \text{pre } u_k < r_k \\ 0 & \text{pre } u_k > r_k \end{cases} \tag{3.2}$$

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 tréningu. Ako predloha pri zostavovaní algoritmu bol použití popis učenia RCE siete v [18].

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 prví vektor tréningovej množiny p=1.
- 3. nastaviť indikátor zásahu hit=false a index k (k-tého neurónu v skrytej vrstve na prví 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 i_p a vektorom váh w_k k-tého neurónu.

$$u_k = \sqrt{\sum (i_p - w_k)^2} \tag{3.3}$$

- 6. s použitím skokovo aktivačnej prenosovej funkcie zistime výstupnú hodnotu skrytého neurónu y_k.

$$y_k = \begin{cases} 1 & \text{pre } u_k <= r_k \\ 0 & \text{pre } u_k > r_k \end{cases} \tag{3.4}$$

  ak y_k=0 tak,skočte bodom 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.

$$r_k = \frac{\sqrt{\sum (i_p - w_k)^2}}{2} \tag{3.5}$$

- 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 w=i a polomer rq=Ŕmax 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 vektoru a pripojte nový neurón z skrytej vrstvy k tomuto novému výstupnému neurónu.

- 13. inkrementujte index p a pokiaľ nie je väčší ako počet vektorov tréningovej 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 ukonči.

Implementácia neurónovej RCE siete

Podla princípu učenia RCE siete je učenie siete implementované následovne v programe ako je uvedene na obrázku 3.1 v triede RCEneuralnetwork. Táto trieda obsahuje tiež implementované metódy predikcie na predikciu výslednej triedy a metódu na pridanie neurónu do skrytej vrstvy.

Trieda Outputneuron implementuje výstupný neurón siete RCE, ktorí obsahuje výslednú triedu a zoznam skrytých neurónov, ktoré sú k nemu pripojene. Trieda HiddenNeuron reprezentuje skryty neurón a jeho vlastnosti. V implementácie obsahuje vektor váh, vektor výstupnej triedy a parameter Rmax, ktorí určuje polomer hypergule.

```
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. 3.1: Implementácia učenia v RCE sieti

3.2 Bayesov klasifikátor

Program obsahuje Bayesov klasifikátor, ktorí pracuje na princípe ako bolo uvedené v teoretickej časti 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 klasifikátor bola vytvorená trieda Bayes.

Táto trieda obsahuje metódy na načítanie dát, pre vytvorenie klasifikátora a pre klasifikáciu. Klasifikátor sa skladá z vypočetných pravdepodobností pre jednotlivé typy tried. Pre odstránenie nulových aposteriórnych pravdepodobnosti je použitá Laplacova korekcia 2.4.2. Samotná klasifikácia prebieha vzájomným vynásobením pravdepodobností jednotlivých meraní pre všetky triedy. Trieda s najvyššou pravdepodobnosťou je výsledkom.

3.3 ID3

Posledný algoritmus, ktorí program obsahuje je z kategórie rozhodovacích stromov a ide o ID3. ID3 (Iterative Dichotomiser 3) je algoritmus, objavený austrálskym inžinierom Roosom Quinlan, ktorý sa používa na generovanie DT z dátovej množiny. Jeho použitie je časté v strojovom učení, ale používa sa napríklad aj v doménach pri spracovaní prirodzeného jazyka. Je určený len na použitie s nominálnymi(neurčenými) vstupmi. Ak vstupný problém zahŕňa premenné s reálnou hodnotou, tak sa tieto hodnoty prenesú do intervalov, pričom každý interval je považovaný za neusporiadaný, nominálny (neurčený) atribút.

Rekurzia v podskupine sa môže zastaviť v niekoľkých prípadoch [17].

- Ak každý prvok v podskupine patrí do rovnakej triedy, potom sa uzol zmení na listový uzol a označí sa triedou príkladov
- Ak už nie je možné vybrať žiadne ďalšie atribúty, príklady však stále nepatria do rovnakej triedy, potom v tomto prípade sa následne z uzlu vytvorí listový uzol s najbežnejšou triedou príkladov v podskupine.
- Ak v podskupine nie sú žiadne príklady, čo môže nastať v prípade, keď sa nenájde žiaden príklad v nadradenej množine, ktorý by zodpovedal konkrétnej hodnote vybraného atribútu. Následne sa vytvorí listový uzol, ktorý sa označí najbežnejšou triedou príkladov v množine rodičovského uzla.

Rozhodovací strom je konštruovaný cez celý algoritmus tak, že každý nekoncový uzol (vnútorný uzol) predstavuje vybraný atribút, na ktorom boli dáta rozdelené a terminálne uzly (listové uzly), ktoré predstavujú označenie triedy konečnej podmnožiny tejto vetvy. ID3 využíva entropiu 2.1 a získavanie informácii 2.2 ako metriku. Hlavné výhody tohto algoritmu sú, že ID3 vytvára krátky a zároveň najrýchlejší strom. Testovanie je potrebné iba pri dostatku atribútov, pokiaľ všetky dáta nie si klasifikované. Nájdenie listových uzlov umožňuje orezanie testovacích údajov, čo následne ovplyvňuje, že sa zníži počet testov. Naopak hlavná nevýhoda algoritmu vzniká pri testovaní menšej vstupnej vzorkovej množine. Pri danom scenári môžeme dojsť k výsledku, že údaje budú preplnené, alebo nadmerne klasifikované. Ďalšie problémy ID3 pri rozhodovaní môžu byť nasledovné: v danom čase je vždy testovaný jeden atribút a výpočtovo nákladná klasifikácia nepretržitých údajov.

Zhrnutie procesu fungovania algoritmu ID3 [17].

- Vypočítaj entropiu každého atribútu **a** z dátovej množiny **S**.
- Rozdelenie množiny **S** na podmnožiny pomocou atribútu, pre ktorý je výsledná entropia po rozdelení najmenšia alebo je zisk informácii najväčší
- Vytvor uzol rozhodovacieho stromu obsahujúci tento atribút.
- Aplikovanie rekurzie na podmnožiny pomocou zvyšných atribútov.

Implementácia algoritmu ID3 v aplikácii Classify. Pre klasifikáciu pomocou algoritmu ID3 je vytvorená trieda Tree. Triedu tvoria atribute, level stromu, hodnota výslednej triedy a HashMap[1] ktorí obsahuje hodnotu, ktorá reprezentuje atribute a triedu Tree. V kontexte implementácie je atribute jeden z atributov zo vstupnej dátovej množiny. Ak napríklad problém obsahuje 9 údajov, atribute predstavuje jeden údaj. Pre atribute bola vytvorená samostatná trieda Attribute, ktorá obsahuje index, meno, hodnotu ktorá určuje koľko rôznych hodnoť môže atribute nadobúdať, hodnoty ktoré atribute nadobúda a arraylist s dátami pre daný atribute. K určovaniu vhodného atributu pre vetvenie stromu je použitý výpočet entropie 2.1 s informačným ziskom 2.2.

3.4 Klasifikačný dataset PROBEN1

Hlavnou úlohou bakalárskej prace je implementácia klasifikačných algoritmov a následne experimenty s špecifickou dátovou množinou. Na toto potrebujeme nielen konkrétne klasifikačné algoritmy ale i konkrétny dataset[2]. V tejto BP sa jedna o množinu reálnych klasifikačných problémov. Pri výbere konkrétneho datasetu sme zvážili niekoľko parametrov ako: dostupná veľká množina reálnych dat, aby boli problémy riešiteľné pomocou techniky učenia s učiteľom, teda aby boli dostupne separované vstupy a výstupy. Ďalší parameter, ktorý je zohľadnený pri výbere je, aby všetky problémy boli statické, teda nemenili sa dáta pri učení neurónovej siete.

Na výber existuje celá rada datasetov, ktoré spĺňajú uvedené požiadavky s zameraním, či už od rozpoznávania obrazu, audio rozpoznávania alebo tie datasety, ktoré sa zaoberajú diagnostickými úlohami. Do úvahy pripadali **MNIST**[10] ,**MEHROTRA**, **CIFAR-10**[9] a **PROBEN1**[12]. **MNIST** je dataset 60000 ručne písaných čísel vhodných pre algoritmy učenie s učiteľom. **MEHROTRA** je dataset reálnych problémov vhodných pre neuronové siete. **CIFAR-10** je dataset na rozpoznávanie obrazov, obsahuje veľkú kolekciu 60000 obrazov. **PROBEN1** obsahuje dáta reálnych problémov pre neuronové siete. Tento dataset bol vybraný experimentovanie a testovanie s implementovanými algoritmani.

PROBEN1 je sada problémov s neurónov sieťou a pravidlami porovnávania. Autor danej sady je nemecký, softwarový inžinier Lutz Prechelt. Ide o kolekciu reálnych problémov pre učenie neurálnych sieti z kategórie klasifikačných algoritmov. PROBEN1 obsahuje 15 problémov, popíšeme si konkrétny problém a postup ako daný problém riešim s pomocou neurnovej siete. PROBEN1 definuje aj sadu pravidiel ako uskutočňovať a porovnávať benchmarking neurónových sieti. Ciel datasetu PROBEN1 je poskytnúť vyskumníkový data pre skontrolovanie správnej implementácii ich algoritmu a možnosť rýchleho a jednoduchého porovnania výsledkov.

Zo sady problémov PROBEN1 boli vybrané štyri problémy: Cancer, Card, Diabetes a Mushrooms. Algoritmus ID3 je implementovaný iba pre prípad Cancer. Ostatné algoritmy sa dajú použiť na všetky štyri problémy. Na testovanie algoritmov boli použité dáta z súborov .dt, ktoré obsahujú hlavičku, kde je daný počet trénovaných a testovaných príkladov plus ďalšie informácie o dátovej množine.

[1] Hašovacia tabuľka je údajová štruktúra, ktorá asociuje kľúče s hodnotami.
[2] Dataset - kolekcia dat

Kapitola 4

Implementácia programu

Táto kapitola popisuje implementáciu aplikácie Classification, ktorá implementuje preštudované základne klasifikačné algoritmy a funkčnosť aplikácie pri riešení konkrétnych problémov. Prve dva časti kapitoly sa zaoberajú štruktúrou a hierarchiou tried. Podkapitola dva obsahuje obrazok hierarchie tried. Tretia podkapitola sa zameriava na technickú časť aplikácie teda aké nástroje, jazyky, knižnice boli použité pri tvorbe aplikácie. Posledná kapitola popisuje užívateľské rozhranie.

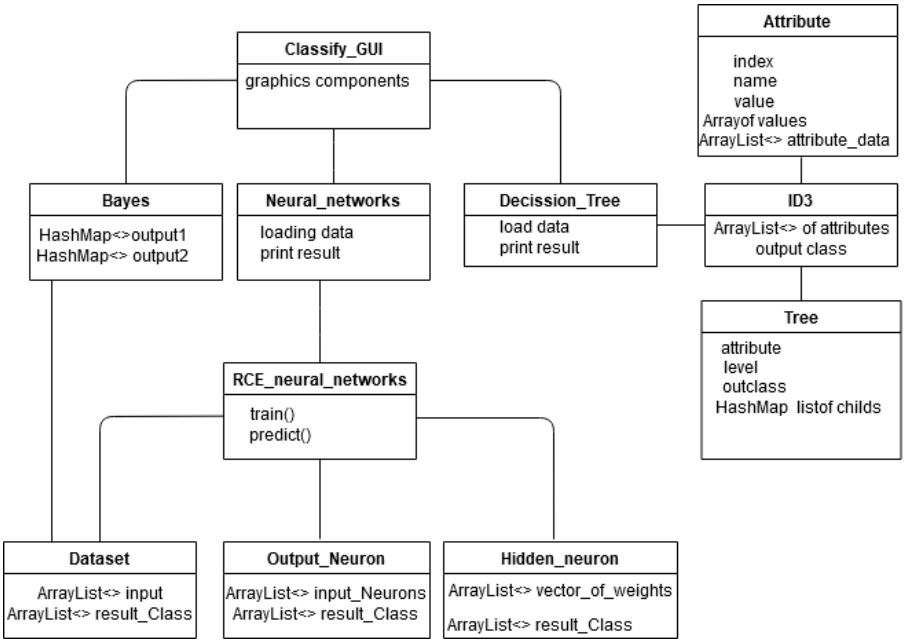
4.1 Štruktúra programu

Projekt obsahuje viacero súborov, kde každý z nich vykonáva svoj špecifický účel. Sú rozdelene na kategórie s implementáciou GUI, ktoré riešia vzhľad aplikácie a implementácie jednotlivých algoritmov. Do prvej kategorie patri subor **ClassifyGUI.java**, kde je implementovane GUI aplikacie. Do druhej kategórie spadá napríklad súbor **DecissionTree.java**, kde sa načítavajú vstupne údaje a volá sa **ID3.java**, ktorý implementuje ID3 algoritmus.

ID3.java pracuje spolu s **Tree.java** a **Attribute.java**, okrem toho obsahuje metódy na výpočet entropie a information gain. Súbor **Tree.java** je jadrom ID3 algoritmu, pretože definuje stromovú štruktúru. **Attribute.java** obsahuje informácie pre atribúte ako meno,hodnota,index... V **Dataset.java** je vytvorená štruktúra pre uloženie údajov, tato štruktúra sa používa pre RCE ako aj pre Bayesov klasifikátor. Súbor **Neuralnetworks.java** slúži na načítanie vstupných dat 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 a výstupné neuróny a ich vlastnosti . Implementácia Bayesovho klasifikátora je v súbore **Bayes.java**, ktorý obsahuje metódy na trénovanie a testovanie.

4.2 Hierarchia tried

Na obrázku 4.1 môžete vidieť 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, HiddenNeuron. Bayesov klasifikator využíva iba jednu triedu Bayes. Grafické rozhranie je v triede ClassifyGUI.



Obr. 4.1: Hierarchia tried pre program Classify

4.3 Použité technológie

Implementácia programu Classification bola napísaná v programovacom jazyku **Java**. Jazyk **Java** je vyvíjaný spoločnosťou Oracle. Ide o objektovo orientovaný programovací jazyk, je podobný jazyku C++, nakoľko 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 neskor spracováva pomocou interpretu Java Virtual Machine.

Pri tvorbe programu bolo použitá vývojové prostredie **Netbeans IDE**. Ide o prostredie pomocou ktorého môžu programátori písať a ladiť programy v jazyku Java. Vyvinuté spoločnosťou Sun Microsystems. Netbeans IDE bol použitý za účelom zjednodušenia práce najmä v oblasti grafického rozhrania.

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, užívateľských prvkov. Aplikácia je založená na modely MVC. MVC rozdeľuje logiku programu na tri vzájomne prepojene prvky Model-view-controller.

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

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

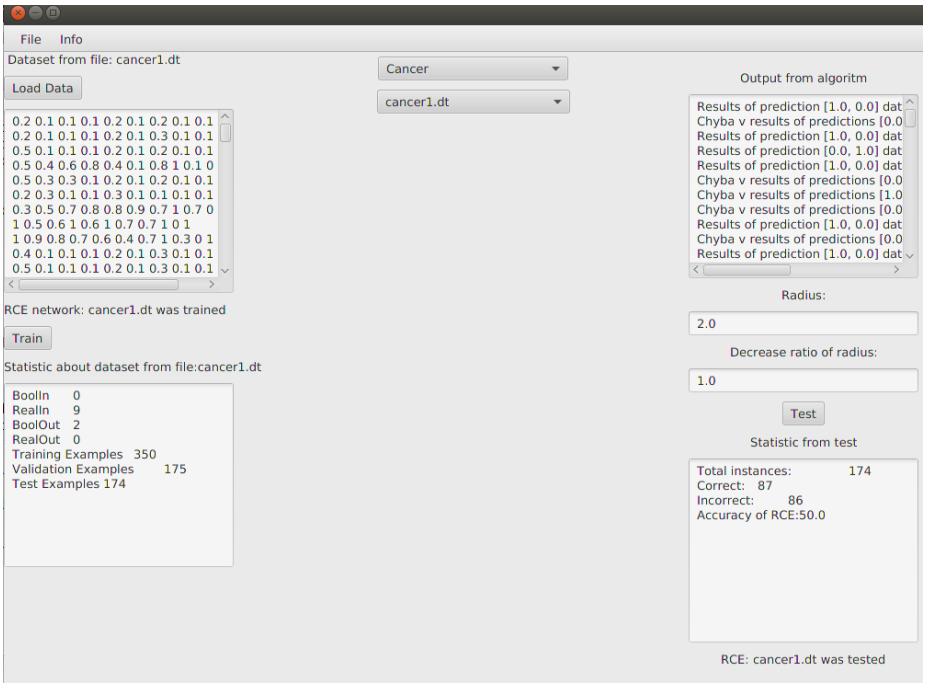
Ďalej bola použitá knižnica Swing. Knižnica Swing je odľahčená sada nástrojov grafického rozhrania Java (GUI), ktorá obsahuje bohatú sadu widgetov a obsahuje balíky na vývoj desktopových aplikácii v Jave.

4.4 Užívateľské rozhranie

Ako už bolo spomenuté užívateľské rozhranie bolo navrhnuté v Netbeans IDE pomocou frameworku **JavaFX** a využíva tiež komponenty z knižnice **Swing** Rozhranie je prispôsobene jednotlivým klasifikačným algoritmom, kde každý ma vlastný pohlaď (scénu). Na začiatku spustenia aplikácie sa zobrazí menu, kde je na výber Bayes classifier, RCE alebo ID3. Následne sa po výbere konkrétneho algoritmu prepne starý pohlaď na konkrétny jedinečný, nový pohlaď.

V každom pohľade je implementované menu v hornej časti. V menu sa dá prepnúť na štartovaciu scénu na opakovaný vyber konkrétneho algoritmu. Ďalej existuje v menu popis k jednotlivým algoritmom. V samotnom pohľade je v hornej časti 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 z súboru a informácie o vstupnom súbore. Na pravej strane pohľadu sú nasledovne ďalšie dve textové oblasti na vystúp výsledkov z algoritmu a štatistiku o výstupe.

Na obrázku 4.2 môžete vidieť grafické užívateľské rozhranie pre RCE.



Obr. 4.2: Užívateľské rozhranie

Kapitola 5

Testovanie a ďalší vývoj

V tejto kapitole je popísane testovanie a experimentovanie s programom **Classify**. Všetky experimenty boli testované na implementovaných klasifikačných algoritmoch pomocou aplikácie Classify, ktorá bola vytvorená ako súčasť tejto práce. Každý jeden algoritmus z implementovaných algoritmov pracuje s vlastnými špecifickými parametrami, podla ktorých sa nadobúdajú rôzne výsledky. Uskutočnené experimenty a ich výsledky pre danú kombináciu parametrov sú zhrnuté nižšie. Účel týchto experimentov je ukázať silu jednotlivých parametrov algoritmov na výsledok. Všetky riešene problémy sú z benchmarku **PROBEN1**[12].

5.1 Algoritmus ID3 (Iterative Dichotomiser 3)

Algoritmus ID3 bol testovaný na problému Cancer.

5.1.1 Problém Cancer

Klasifikačný problém Cancer (rakovina) sa zaoberá diagnostikovaním rakoviny pŕs. Snaží sa klasifikovať problém ako benign alebo malignat. Údaje boli získane na základe mikroskopického vyšetrovania. Obsahuje 9 vstupných atribútov, ktoré sú diskrétne, ako napr. rovnomernosť veľkosti a tvaru buniek alebo a 2 výstupné atribúty, ktoré sú binárne. Príklad z databázy 0.1 0.1 0.1 0.1 0.2 0.1 0.2 0.1 0.1 1 0 .

Testovanie problému Cancer prebiehalo s dátami z cancer1.dt až cancer3.dt. Tieto súbory obsahujú 350 príkladov na trénovanie a 175 na testovanie a validáciu.

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

Tabuľka 5.1: Zobrazujúca výsledky z testovania problému Cancer ID3 algoritmom

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

5.2 Bayesov klasifikátor

Tento algoritmus bol testovaný na viacerých problémoch a to konkrétne Card, Diabetes, Mushrooms a Cancer.

5.2.1 Problém Card

Klasifikačný problém Card sa zaoberá predikciou schválenia alebo neschválenia udelenia kreditnej karty zákazníkovi v banke. Ide o skutočné údaje z banky a výstup predstavuje, či banka kartu udelí alebo neudelí. 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 nepretržite (continuous), výstupy sú binárne. Príklad vstupu z databázy 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 1 0 0 1 0 0 0 0 0 0 0 0 0 0.298246 1 0 0 0 1 0 0 0 0 0 1. Testovanie problému Card prebiehalo s dátami z 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.

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

Tabuľka 5.2: Zobrazujúca výsledky z testovania problému Card

5.2.2 Problém Diabetes

Klasifikačný problém Diabetes (cukrovka) sa zaoberá diagnostikovaním cukrovky medzi skupinou ludi Pima indians. Predikcia je o tom, či jedinec má cukrovku alebo nemá. Je založená na vstupných atribútov, medzi ktoré patria osobné údaje a lekárske testy. Databáza obsahuje 8 vstupných údajov, ktoré sú nepretržite (continuous) a 2 výstupy, ktoré sú binárne. Príklad vstupu z databázy 0.117647 0.875 0.721311 0 0 0.341282 0.105892 0.0166667 0 1. Testovanie problému Diabetes prebiehalo s dátami z 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.

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

Tabuľka 5.3: Zobrazujúca výsledky z testovania problému Diabetes

5.2.3 Problém Mushroom

Klasifikačný problém huby sa zaoberá rozlišovaním či sú huby jedlé alebo nie. Podla opisu huby, farby, voňe atd. sa prijíma toto rozhodnutie. Databáza obsahuje 125 vstupov, ktoré

sú nominálne a 2 binárne výstupy. Príklad vstupu z databázy 0 0 1 0 0 0 0 0 1 0 1 0 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 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 0 0 1 0 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 0 1 0. Testovanie problému Mushroom prebiehalo s dátami z mushroom1.dt až mushroom3.dt. Tieto súbory obsahujú 4062 príkladov na trénovanie a 2031 na testovanie a validáciu.

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

Tabuľka 5.4: Zobrazujúca výsledky z testovania problému Mushroom

5.2.4 Problém Cancer

Princíp problému Cancer už bol vysvetlený pri ID3 algoritme 5.1.Testovanie problému Cancer prebiehalo s dátami z cancer1.dt až cancer3.dt. Tieto súbory obsahujú 350 príkladov na trénovanie a 175 na testovanie.

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

Tabuľka 5.5: Zobrazujúca výsledky z testovania problému Cancer

5.2.5 Zhrnutie Bayesovho klasifikátora

Z prevedených experimentov bol najlepší výsledok získaný pri probléme Cancer, kde úspešnosť dosahovala skoro sto percent. Následne môžme pozorovať, že výsledky pri probléme Diabetes dosahovali úspešnosť priemerne okolo 66 percent pri 350 príkladov na trénovanie a 175 na testovanie. Pri väčšej trénovacej množine 700 príkladov na trénovanie a 960 na testovanie bola úspešnosť predikcie okolo 87 percent. V prípade problému Mushroom, ktorý mal najviac vstupov a aj bol časovo najzložitejší bola úspešnosť riešenia iba v priemere okolo 50 percent. Pri probléme card bola znova úspešnosť pri 350 príkladov na trénovanie a 175 na testovanie v priemere okolo 58 percent úspešnosti odhadu riešenia. Môžeme ale rovnako ako v prípade diabetes pozorovať, že úspešnosť odhadu riešenia sa zvyšuje, keď trénujeme klasifikátor pri väčšej množine dat. Celkovo môžeme pozorovať, že pri dostatočne veľkej tréningovej sady, úspešnosť stúpa.

5.3 RCE (Restricted Coulomb Energy) neural network

Algoritmus RCE bol testovaný na viacerých problémoch z PROBEN1. Konkrétne Cancer, Card, Diabetes a Mushrooms. Každý problém obsahuje 3 alebo datasety na trénovanie a testovanie. Pre súbory použite v testovaní platia rovnaké hodnoty príkladov na treninng a testovanie ako pri Bayesovom klasifikatore. Jednotlive problémy už boli vysvetlené pri

experimentovaní s ID3 alebo u experimentovaní s Bayesovím klasifikátorom. Pri testovaní u RCE neurónovej siete zaleží na hodnotách parametrov Radius(Rmax) a Decreas ratio of radius(Rdc). Tieto parametre predstavujú polomer hypergule a jeho zmenšenie o polovicu. Experimenty boli uskutočnené s rôznymi hodnotami týchto parametrov.

5.3.1 Problém Cancer

Pri probléme Cancer boli najprv hodnoty Rmax=1.25 a Rdc=0.625. V nasledujúcom testovaní boli hodnoty Rmax=2 a Rdc=1. V poslednom testovaní boli hodnoty Rmax=1.5 a Rdc=0.75.

	Rmax=1.25 Rdc=0.625	Rmax=2 Rdc=1	Rmax=1.5 Rdc=0.75
cancer1.dt	0.50574	0,50	0,6206
cancer2.dt	0.49425	0,42	0,6091
cancer3.dt	0.63793	0,51	0,5114
cancer4.dt	0.57524	0,48	0,5597

Tabuľka 5.6: Zobrazujúca výsledky z testovania problému Cancer

5.3.2 Problém Diabetes

Pri probléme Cancer boli v prvom testovaní hodnoty Rmax=0.9 a Rdc=0.45. V nasledujúcom testovaní boli hodnoty Rmax=0.8 a Rdc=0.4. Tretí test bol pri hodnotách Rmax=1.4 a Rdc=0.7. V poslednom testovaní boli hodnoty Rmax=1.2 a Rdc=0.6.

Tabuľka 5.7: 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	0.58333	0,5781	0,5468
diabetes2.dt	0.50	0,4687	0,4739
diabetes3.dt	0.61458	0,6093	0,5416
diabetes4.dt	0.53654	0,5412	0,5489

	Rmax=1.2 Rdc=0.6	Rmax=1.0 Rdc=0.5
diabetes1.dt	0,5989	0,6145
diabetes2.dt	0,4114	0,5104
diabetes3.dt	0,5937	0,4270
diabetes4.dt	0,5738	0,5225

5.3.3 Problém Card

Pri probléme Cancer boli v prvom testovaní hodnoty Rmax=3.5 a Rdc=1.75. V nasledujúcom testovaní boli hodnoty Rmax=2.75 a Rdc=1.325. Tretí test bol pri hodnotách Rmax=3.0 a Rdc=1.5 .

	Rmax=3.5 Rdc=1.75	Rmax=2.75 Rdc=1.325	Rmax=3.0 Rdc=1.5
card1.dt	0.5639	0.5232	0.5639
card2.dt	0.5116	0.50	0.5116
card3.dt	0.4941	0.5639	0.5639
card4.dt	0.496	0.46	0.46

Tabuľka 5.8: Zobrazujúca výsledky z testovania problému Card

5.3.4 Zhrnutie RCE

Z všetkých prevedených experimentov s použitím RCE siete bol najlepší 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 percent. Výsledky z testovania problému Cancer pri rôznych parametroch Rmax a Rdc nedosahovali uspokojivých výsledkov. Testovanie ďalšieho problému diabetes dosiahli najlepšie výsledky pri Rmax=1.25 a Rdc=0.625, kedy priemerná úspešnosť odhadu výslednej triedy bola 55.5 percent.

Testovanie problému Cancer a ďalších problémov odhalilo veľký vplyv parametrov Rmax Rdc na výsledok predikcie z RCE. Celkové výsledky zo všetkých experimentov ukazujú na slabú úspešnosť odhadu výslednej triedy implementovanej RCE sieť.

5.4 Zhrnutie výsledkov a ďalší vývoj

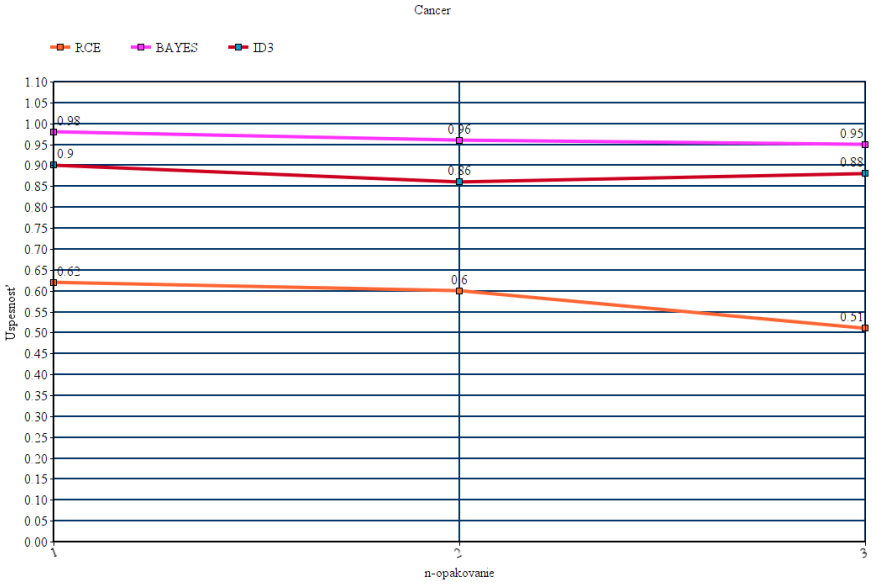
Testovanie implementácie algoritmu ID3 pri probléme Cancer bolo uspokojivé. Výsledný priemer úspešnosti 88.122 percent považujem za dobrý. Testovanie s bayesov klasifikatorom prinieslo pri probléme Cancer prinieslo najúspešnejšie výsledky to v priemere 97 percent úspešnosť odhadu. Obrázok 5.1 zobrazuje úspešnosť všetkých jednotlivých algoritmov pri riešení problému Cancer na troch rôznych dátových množinách.

Ďalšie testovanie klasifikátora odhalilo pri diabetes úspešnosť 66 percentu úspešnosť odhadu výstupnej triedy. Môžeme ale pozorovať, že ďalšie testovanie pri zväčšení množstva príkladov malo výslednú úspešnosť 86 percent. Porovnanie výsledkov u problému Card pre RCE a Bayes je na obrázku 5.2. Môžeme pozorovať, že obe algoritmy mali pri prvých troch testovaní približne rovnaké výsledky. Ale pre testovanie číslo 4 na väčšej testovacej množine dochádza k výrazným odlišným výsledkom. Zatiaľ čo úspešnosť predikcie klasifikátora sa vyšplhala na 79 percent, úspešnosť RCE siete klesla výrazne na 49 percent.

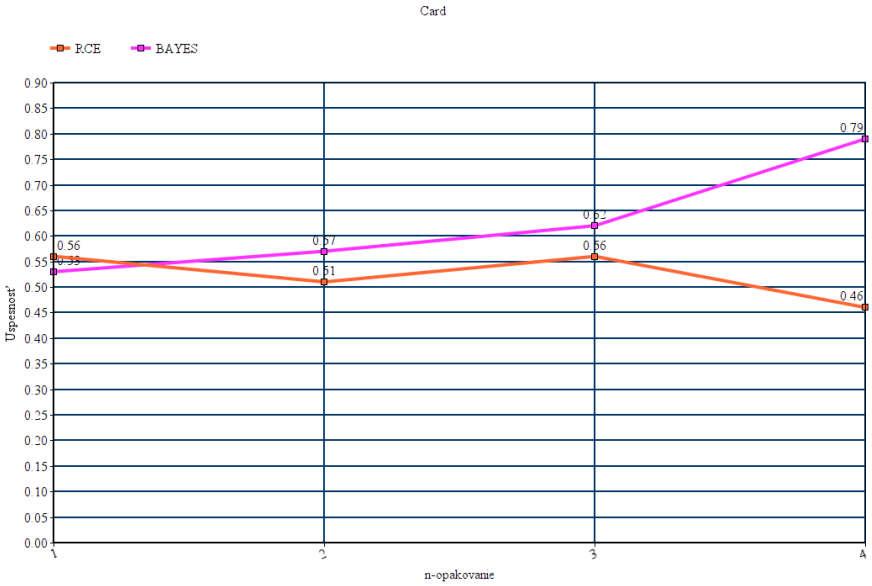
Porovnanie výsledkov u problému Diabetes pre algoritmy RCE a Bayes je na obrázku 5.3. Kde pozorujeme približne rovnakú úspešnosť predikcie pre bayesovho klasifikátora u prvých troch testovaní, pri väčšej dátovej množiny pre trénovanie a testovanie znova je úspešnosť predikcie výslednej triedy vyššia.

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

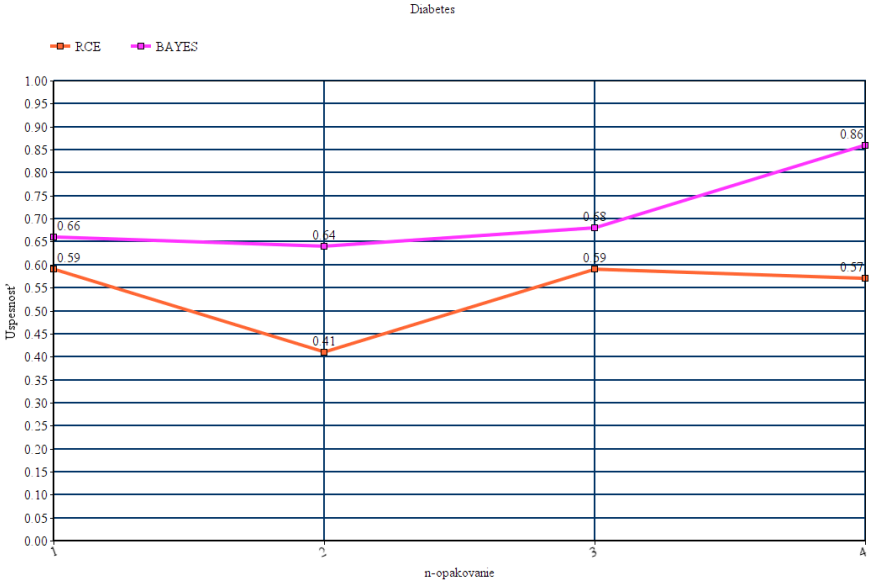
Ďalší vývoj aplikácie Classify môže smerovať viacerými smermi a to implementáciu nejakého ďalšieho klasifikačného algoritmu z vybraných smerov v tejto bakalárskej práce. Príklad algoritmus C4.5 pre rozhodovacie stromy. Tematický okruh neurónových sieti predstavuje veľa možností s ktorých sa dajú čerpať nové algoritmy pre riešenie daných problémov. Pre vývoj aplikácie sa môže taktiež isť ďalším novým tematickým smerom klasifikačných algoritmov ako napríklad Support vector machines (SVM) alebo Random Forest (RF).



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



Obr. 5.2: Graf porovnania ID3 a Bayesa pre Card



Obr. 5.3: Graf porovnania ID3 a Bayesa pre Diabetes

Kapitola 6

Záver

V úvodnej časti bakalárskej práce (kapitola dva) popisujem teoretické vedomosti o jednotlivých klasifikačných smeroch. Popisujem tu jednotlivú terminológiu, modely, funkčnosť a ich možné použitie v praxi. V tejto časti vysvetľujem aj klasifikačné smery: Neurónové siete, Bayesové klasifikátory a Rozhodovacie stromy. V tretej kapitole zaznamenávam konkrétne implementované algoritmy v programe Classify: Restricted Coulomb Energy (RCE) neurónovú sieť, Bayesov klasifikátor a rozhodovací strom podla algoritmu ID3. Táto kapitola okrem popisu daných algoritmov vysvetľuje ako boli konkrétne dané algoritmy implementované.

V praktickej časti som popísal implementáciu programu Classify, ktorý slúži na demonštráciu jednotlivých klasifikačných algoritmov RCE, ID3 a Bayesov klasifikátor. Pomocou programu Classify boli riešené problémy z benchmarku PROBEN1. Tento program slúži na riešenie štyroch problémov. Každý problém bol podrobený experimentovaniu na daných algoritmoch. Na probléme Cancer bolo porovnaná efektivita všetkých troch algoritmov. Na záver experimentovania s každou úlohou boli zhodnotené získané výsledky z daného programu. Získané výsledky závisia od rôznych vstupných parametrov, ktoré nedokážem zabezpečiť, aby získané riešenie bolo najlepším možným riešením konkrétneho problému. Najlepšiu úspešnosť klasifikácie dosiahol pri úlohe Cancer Naivný Bayesovský algoritmus. Dobrú úspešnosť dosiahol i rozhodovací strom implementovaný algoritmom ID3. Najhorších výsledkov dosiahla Reduced Coulomb Energy (RCE) neuronová sieť.

Literatúra

[1] Bekra: *Bayesovská klasifikace*. [Online; navštíveno 25.05.2020].
URL https://sorry.vse.cz/~berka/docs/izi456/kap_5.6.pdf

[2] Darula, M.: *Bayesovské siete.* . 2007, [Online; navštíveno 15.06.2020].
URL http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0607/darula/index.html

[3] 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

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

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

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

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

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

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

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

[11] McCulloch, P.: *A Logical Calculus of the Ideas Immanent in Nervous Activity* . 1943.

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

[13] Razali, N.; Mustapha, A.; Yatim, F. A.; aj.: *Predicting Football Matches Results using Bayesian Networks for English Premier League (EPL)*. 2017.

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

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

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

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

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