ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

Ján Rabčan

Aplikácia na vyhodnocovanie dotazníkov pomocou hĺbkovej analýzy dát

Vedúci práce: Ing. Jozef Kostolný, PhD.

Registračné číslo: 74/2015

Žilina, 2016

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIPLOMOVÁ PRÁCA

INFORMAČNÉ SYSTÉMY

Ján Rabčan

Aplikácia na vyhodnocovanie dotazníkov pomocou hĺbkovej analýzy dát

Žilinská univerzita v Žiline
Fakulta riadenia informatiky
Katedra informatiky
Žilina, 2016

POĎAKOVANIE

Touto cestou by som sa chcel poďakovať pánovi Ing. Jozefovi Kostolnému, PhD. za pomoc, odborné vedenie, cenné rady a pripomienky, ktorými ma usmerňoval pri písaní mojej diplomovej práce.

Ďakujem aj svojej rodine, ktorá ma počas celej doby podporovala a vytvárala mi vhodné podmienky na písanie.

PREHLÁSENIE

Čestne prehlasuje, že diplomovú prácu som vypracoval samostatne a taktiež, že som uviedol všetky zdroje, z ktorých som pri tvorbe diplomovej práce čerpal.

V Žiline, dňa 15. 04. 2016

_____

ABSTRAKT

RABČAN, Ján: *Aplikácia na vyhodnocovanie dotazníkov pomocou hĺbkovej analýzy dát*. [Diplomová práca] – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra informatiky. –Vedúci: Ing. Jozef Kostolný, PhD. – Stupeň odbornej kvalifikácie: inžinier. – Žilina: FRI UNIZA, 2016. – 84 s.

V práci sa zaoberám tvorbou a návrhom aplikácie pre vyhodnocovanie dotazníkov definovaných pomocou fuzzy premenných. Práca obsahuje teoretický popis vybraných algoritmov, ktoré sú v praktickej časti implementované do knižnice. Implementované algoritmy sú porovnané podľa chyby prijatého rozhodnutia. Vytvorená knižnica je využitá vo výslednej aplikácii. V práci sa venujem aj použiteľnosti a preukázaniu funkčnosti vzniknutej aplikácie.

Kľúčové slová: hĺbková analýza dát, fuzzy logika, fuzzy množiny, rozhodovacie stromy

ABSTRACT

RABČAN, Ján: *The application for the evaluation of the questionnaires using data mining*. [Diploma thesis] – University of Žilina. Faculty of Management Science and Informatics; Department of informatics. – Supervisor: Ing. Jozef Kostolný, PhD. – Qualification level: Engineer. – Žilina: FRI UNIZA, 2016. – 84 p

The main aim of the work was to create the application for the evaluation of the questionnaires by using data mining. The work contains description of chosen algorithms of data mining. To achieve creation of powerful application I created the library consists of described algorithms. In work I provided a basic comparison of implemented algorithms. I made the analysis of functionality of application for evaluation of questionnaires. In final, I have shown applicability of the developed application.

Key words: data mining, fuzzy logics, fuzzy sets, decision trees

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM DIAGRAMOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

ÚVOD

V práci sa zaoberám tvorbou a návrhom aplikácie pre vyhodnocovanie dotazníkov definovaných pomocou fuzzy premenných. Práca obsahuje teoretický popis vybraných algoritmov, ktoré sú v praktickej časti implementované do knižnice. Implementované algoritmy sú porovnané podľa chyby prijatého rozhodnutia. Vytvorená knižnica je využitá vo výslednej aplikácii. V práci sa venujem aj použiteľnosti a preukázaniu funkčnosti vzniknutej aplikácie.

Práca sa člení na tri kapitoly. V prvej kapitole sa venujem teoretickým pojmom z hĺbkovej analýzy dát a opisu algoritmov ID3 a C4.5, ale aj spracovaniu dotazníkov. Druhá kapitola je venovaná fuzzy logike. Objasňuje problematiku fuzzy množín a fuzzy dát. V tejto kapitole je popísaný postup transformácie numerických hodnôt na fuzzy kategorické hodnoty, postup tvorby klasifikačných pravidiel a aj vysvetlenie fuzzy rozhodovacích stromov. Fuzzy rozhodovacie stromy sú vysvetlené aj za pomoci príkladov. V tretej kapitole sa venujem praktickému využitiu teoretických vedomostí z prvých dvoch kapitol. Ako prvé je popísaný inžiniersky projekt, z ktorého diplomová práca vychádza. Nasleduje popis vyvinutého fuzzy systému pre spracovanie fuzzy dát a budovanie fuzzy rozhodovacích stromov. Ďalej sa v tejto kapitole venujem tvorbe a popisu vytvorenej aplikácie, ktorá vychádzala zo vzniknutého fuzzy systému. V závere tretej kapitoly som sa zameriavam na preukázanie funkčnosti a použiteľnosti aplikácie.

1 DOTAZNÍK A HĹBKOVÁ ANALÝZA DÁT

1.1 Dotazník

Dotazník je nástroj, ktorý sa zameriava na zber dát a je neoddeliteľnou súčasťou rôznych vedeckých výskumov. Jeho využite je takmer všade, či už na zozbieranie preferencií, overenie znalostí, názorov, hodnôt a mnoho ďalších informácií. Dotazníky sú výhodné predovšetkým z ekonomického hľadiska, pretože s ich pomocou môžeme nazbierať veľa informácií za krátky čas bez toho, aby bol výskumník fyzicky prítomný pri osobe, ktorá dotazník vypĺňa. Veľkou výhodou je možnosť anonymného zberu dát. To samozrejme nie je podmienka a dotazník je možné vytvoriť aj neanonymnou formou. S overovaním, či dotazník bol vyplnený podpísanou osobou, môžu byť problémy. Dotazník ponúka respondentom priestor na dôkladné prečítanie otázky a premyslenie si odpovedí, ako aj ľubovoľné poradie otázok. Pokiaľ sú otázky kladené slovne, tak to ide väčšinou rad po rade. Na druhej strane, pri nepochopenej otázke nie je možné jej dodatočné vysvetlenie. Pri overovaní výslednej aplikácie budeme pracovať s databázou uzavretých otázok. Uzavretá otázka má vopred známu množinu odpovedí, z ktorých si respondent vyberá odpoveď. Vyhodnocovanie takýchto otázok je jednoduchšie na automatizáciu. Tento fakt je z informatického hľadiska nesporná výhoda, no pre respondenta to predstavuje obmedzenú množinu možných odpovedí. Podľa Dvorskej [1] sa dotazník označí za validný, ak sme pomocou dotazníka zisťovali to, čo sme mali zámer zisťovať. Na validitu dotazníkov vplýva veľa ťažko ovplyvniteľných aspektov. Nemusí sa jednať o štruktúru dotazníka ani kvalitu otázok. Hlavný faktor sú respondenti. Pokiaľ dotazník nie je anonymný, tak mnoho ľudí má tendenciu odpovede prikrášliť. Naopak pri anonymnom dotazníku môže nastať situácia, že človek jednoducho označí odpovede na otázky bez čítania a premýšľania. Pre takéto prípady sa pridávajú L-otázky. Takáto otázka môže byť nasledovná: „Čítal si túto otázku?“ s možnosťami áno, nie. Existuje mnoho prístupov k vyhodnocovaniu dotazníkov. Môže sa jednať o štatistické analýzy, alebo aj hĺbkovú analýzu dát, ktorou sa budem v práci ďalej zaoberať.

Dotazníky a ich vyhodnocovanie:

1. **Definícia problému** – stanoviť cieľ. Dotazník bude formulovaný na zber dát, ktoré nám poskytnú požadované informácie k danému problému. Ideálny stav je,

keď vieme definovať, čo chceme zistiť. Napríklad, či nás zaujímajú štatistické údaje, alebo budeme hľadať rôzne závislosti pomocou hĺbkovej analýzy dát.

2. **Zber dát do elektrickej podoby** – existuje mnoho spôsobov, ako zbierať dáta v elektronickej podobe. Veľmi rozšírené sú dotazníky. Pre hĺbkovú analýzu je dôležité nazbierať dostatočné množstvo dát.

3. **Analýza dát –** po zozbieraní dostatočného počtu dát, môžeme vykonať analýzu. Existuje množstvo spôsobov, ako sa dáta analyzujú. Ja sa v práci venujem hĺbkovej analýze dát.

4. **Overenie správnosti postupov a výsledkov –** je vhodné overiť napríklad, či analyzovaná množina dát bola dostatočne veľká.

5. **Vytvorenie reportov –** po ukončení analýzy sa vygeneruje množina reportov, prípadne jeden veľký report, ktorý obsahuje požadované informácie a výsledky analýzy.

Dotazníky sú len jedna z foriem na zber dát. Vo všeobecnosti môžeme tvrdiť, že počet dát prudko narastá. Dáta prichádzajú odvšadiaľ. Nárast počtu dát súvisí s vysokou dostupnosťou prostriedkov, ktoré umožňujú dáta ľahko zaznamenať. V súčasnosti sú veľmi rozšírené osobné počítače, mobilné telefóny a iné zariadenia, ponúkajúce možnosti, ako jednoducho zaznamenávať dáta. Celkový počet dát vo svete rastie každým dňom. Koniec tohto rastúceho trendu je v nedohľadne [2]. Dáta v informatike predstavujú informácie spracovávané pomocou informačných systémov. Popisujú javy, alebo vlastnosti pozorovaných entít. Úlohou dát je vyjadriť skutočnosť pomocou formálneho zápisu. Požadované vlastnosti sú prenositeľnosť a spracovateľnosť. Získať dáta je možné pomocou rôznych techník. Môžeme sledovať, ako sa zmenšuje pomer medzi populáciou dát, čiže ich počtom a ich porozumením. S nárastom počtu dát klesá ich kvalita a stávajú sa bezvýznamné. Metódy typu „pozriem sa a vidím“ sú pri veľkom množstve dát nepraktizovateľné. Preto je potrebné dáta dôkladne analyzovať. Analýza dát môže priniesť nové postrehy a v komerčnom prostredí konkurenčnú výhodu. Hĺbková analýza dát je o riešení problémov na základe analýzy dát, ktoré sú prítomné v rôznych dátových zdrojoch. Je to interdisciplinárny vedný odbor, ktorý je súčasťou objavovania znalostí v dátach. V slovenskom jazyku sa zvykne označovať aj termínom *dátokopectvo* alebo dolovanie dát. Často sa stretávame aj s anglickým názvom *datamining*. Ekonómovia, štatistici, meteorológovia a komunikační inžinieri dlho pracovali s myšlienkou, že vzory v dátach môžeme

automaticky vyhľadať, identifikovať, validovať a využiť ich na predikciu, klasifikáciu či zhlukovanie. Proces analýzy dát je výpočtový proces. Tento proces zhromažďuje dáta a následne ich analyzuje. Dáta sú uložené v elektrickej forme a analyzovanie je automatizované. Objavuje vzory vo veľkých dátových množinách, pričom využíva aj metódy umelej inteligencie, štatistické metódy a metódy strojového učenia. Hlavným cieľom procesu hĺbkovej analýzy dát je získať informácie z dátovej množiny a transformovať ich na zrozumiteľnú štruktúru pre ďalšie používanie. Používanie niektorých dát, najmä dát o ľuďoch, má pre hĺbkovú analýzu rôzne etnické aspekty. Tieto aspekty by mal dátový analytik pred vytvorením modelu zvážiť. Pokiaľ sa aplikuje dátová analýza na dáta, ktoré zhromažďujú informácie o ľuďoch, môže to napríklad viesť k aplikácii, ktorá určí ľudí, ktorí dostanú výhodnejšiu ponuku. Určité spôsoby diskriminácie ako rasová, sexuálna, náboženská a mnohé ďalšie, sú nielen neetické, ale v mnohých prípadoch aj protizákonné. Ako ďalšie by bolo treba premyslieť, čo chceme v dátach hľadať. Napríklad, ak by sa skúmal vplyv banánov na dĺžku ľudského života, tak pri použití určitej dátovej množiny môžeme dostať skreslené výsledky. Máme rozsiahle historické dáta, pričom vieme, že banány sa na území Slovenska začali jesť v minulom storočí. Je vysoko pravdepodobné, že výsledok by hovoril, že banány majú výrazný vplyv na dĺžku života, pretože priemerná dĺžka života sa predlžuje. Je zrejmé, že za zmenou dĺžky života stojí technický a medicínsky pokrok a konzumácia banánov na to zásadný vplyv nemá.

1.2 Základy hĺbkovej analýzy

V tejto kapitole sa stručne vyjadrím k niektorým základným technikám hĺbkovej analýzy dát. Všetky tieto techniky budú aspoň z časti využité v praktickej časti práce. Algoritmy v práci narábajú s dátovými množinami, ktoré obsahujú inštancie dát. Inštancia dát je v práci označovaná ako inštancia, príklad alebo vzorka. Inštancie dát môžu byť popísané pomocou lingvistických a numerických atribútov. Numerické hodnoty sú číselné hodnoty a sú spojité. Lingvistické (ináč kategorické) hodnoty sú diskrétne. Aj kategorické hodnoty môžu byť vyjadrené číslom. Napríklad v klasifikačnej úlohe by toto číslo udávalo informáciu o príslušnosti inštancie k danej triede výstupného atribútu.

Všetky algoritmy v práci vykonávajú pred svojou činnosťou predspracovanie dát. Týmto predspracovaním sa rozumie čistenie alebo normalizácia dátovej množiny.

Čistenie dátovej množiny vykonávajú algoritmy pri odstraňovaní inštancií, ktoré obsahujú napríklad chýbajúce hodnoty atribútov. Normalizáciu som vykonával pri úlohe zhlukovania. Cieľom normalizácie v tomto prípade bolo previesť hodnoty číselného atribútu do intervalu hodnôt $\langle 0,1 \rangle$. Na tento prevod som využíval nasledujúci algoritmus:

Algoritmus normalizácie numerických dát

Nech A je numerický atribút. V dátovej množine sa nachádza *n* inštancií. Jednotlivé hodnoty x_i, kde $i = 1, \dots, n$ vyjadrujú hodnotu i-tej inštancie atribútu A a nadobúdajú hodnoty v intervale $\langle min, max \rangle$, kde $min, max \in R$. Počas normalizácie sa každá hodnota x_i nahradí hodnotou $x_{i_{new}}$, ktorá sa vypočíta podľa nasledujúceho vzťahu [3]:

$$x_{i_{new}} = \frac{x_i - min}{max - min}$$

1.2.1 Zhlukovanie

Anglicky sa označuje ako Clustering. Úlohou tejto techniky je vytvárať zhluky dát (cluser). Do týchto zhlukov postupne algoritmus rozdelí prvky dátovej množiny na základe podobnosti, tak že do jednotlivých zhlukov vyberá objekty, ktoré sú na seba v nejakom zmysle podobné. V práci som využíval nasledujúci algoritmus zhlukovania:

Algoritmus zhlukovania reálnych hodnôt

- Vstupné dáta:
  - Zoznam numerických hodnôt
  - Prirodzené číslo Q, ktoré udáva počet výstupných intervalov.
- Výstupné dáta
  - Q výstupných intervalov
- Postup:
  1. Zoznam numerických hodnôt $X = \{x_1, \dots, x_i\}$ sa pomocou Algoritmus normalizácie numerických dát prevedie na zoznam numerických hodnôt v intervale $\langle 0,1 \rangle$
  2. Algoritmus vytvoríme Q intervalov. Interval Q_q obsahuje centrum C_q vyjadrené podľa pravidla $C_q = \frac{q-1}{Q-1}$, kde $q = 1, \dots, Q$.

3. Každý prvok x_i pridáme do intervalu, v ktorom je hodnota euklidovskej vzdialenosti x_i a centra C_q intervalu Q_q minimálna.

$$q_{min} = \min_{q=1,\dots,Q} |x_i - C_q|$$

4. Pre každý interval Q_q vyrátame novú hodnotu centra C_q ako aritmetický priemer hodnôt x_i patriacich do toho intervalu:

$$C_q = \frac{\sum_{i=1}^{N_q} x_i}{N_q}$$

kde N_q je počet hodnôt x_i patriacich do intervalu C_q. Ak sa nezmenilo žiadne centrum, tak algoritmus končí, ináč sa opakuje od kroku 3.

1.2.2 Klasifikačné úlohy

Klasifikačné úlohy riešia problém v rôznych vedných odboroch ako štatistika, hĺbková analýza dát, či strojové učenie. Cieľom tejto problematiky je zaradiť nové pozorovanie do konkrétnej podmnožiny, respektíve triedy. Pozorovania predstavujú inštancie (mnohokrát označované ako príklady alebo vzorky) popísane pomocou atribútov. Klasifikačné triedy je možné definovať pomocou numerických atribútov, alebo ich môžeme zadať v slovnej podobe, čiže lingvisticky. Inštancie dát dátových množín v práci sú popísané množinou vstupných atribútov a jedným výstupným atribútom. Inštancie v množine musia mať rovnakú štruktúru, to znamená, že všetky inštancie trénovacej množiny sú popísané rovnakými atribútmi, aj rovnakým počtom atribútov. Cieľom algoritmov je klasifikovať inštanciu do tried popísaných výstupným atribútom na základe hodnôt vstupných atribútov. Tieto algoritmy fungujú v dvoch krokoch. V prvom kroku musia vybudovať klasifikačný mechanizmus, respektíve model. Na budovanie využijú tréningovú množinu inštancií. V druhom kroku využívajú vybudovaný model na klasifikovanie nových inštancií. Počas klasifikácie sa model pokúša vyjadriť hodnotu výstupného atribútu klasifikovanej inštancie. Formálne sa dá zadefinovať klasifikačná úloha podľa [4] Algoritmus funguje v dvoch krokoch. Nech T je tréningová množina v klasifikačnej úlohe, ktorá obsahuje k inštancií na natrénovanie algoritmu. Inštancie množiny T sú popísané n atribútmi. Atribút množiny je vyjadrovaný v tvare $X_j, j = 1, \dots, n$. Každý atribút nadobúda viac ako jednu hodnotu, ináč by nemal zmysel. Pokiaľ atribút X_j nadobúda numerické hodnoty, týchto hodnôt môže byť nekonečne veľa. Kategorické atribúty nadobúdajú lingvistické hodnoty.

Kategorický atribút X_j môže byť vyjadrený pomocou m_j rôznych hodnôt $X_{j,m_1}, X_{j,m_2}, \dots, X_{j,m_j}; m_j \geq 2$. Inštancie v tréningovej množine T sú popísané aj výstupným atribútom. Pomocou výstupného atribút sa určuje informácia o príslušnosti ku klasifikačnej triede. Výstupný atribút sa označuje ako Y a nadobúda y lingvistických hodnôt $Y_1, Y_2, \dots, Y_y$. Taktiež platí, že musí mať minimálne dve hodnoty. Pokiaľ by nadobúdal len jednu hodnotu, tak úloha by nebola klasifikačná. Všetky inštancie by boli priradené do jednej triedy a v takom prípade nie je dôvod klasifikovať. Výsledok by bol známy ešte pred samotnou klasifikáciou. Algoritmus z trénovacej množiny T vytvorí klasifikačný model, pomocou ktorého bude klasifikovať. Spôsob tvorby modelu je určený výberom algoritmu. Proces klasifikácie je druhý krok algoritmov. V tomto procese algoritmy klasifikujú inštancie, ktoré nepatria do trénovacej množiny. Klasifikované inštancie nemajú známu hodnotu výstupného atribútu. To spôsobuje, že nevieme určiť ich príslušnosť k triede. Cieľom klasifikácie je priradiť objekt do triedy, ktorá vznikla v prvej časti algoritmu poľa hodnôt výstupného atribútu jednotlivých prvkov trénovacej množiny T.

Atribút	Hodnoty, ktoré atribút môže nadobúdať
Sklon podpisu (X_1)	Rovný $(X_{1,1})$, Naklonený $(X_{1,2})$
Má podpis slučku? (X_2)	Áno $(X_{2,1})$, Nie $(X_{2,2})$
Hrúbka podpisu (X_3)	Tenký $(X_{3,1})$, Normálny $(X_{3,2})$, Hrubý $(X_{3,3})$
Podčiarknutý podpis (X_4)	Áno $(X_{4,1})$, Nie $(X_{4,2})$
Povaha autora podpisu (výstupný $-Y$)	Pokojný (Y_1), Agresívny (Y_2)

Tabuľka 1. Príklad klasifikačnej úlohy

1.3 Rozhodovacie stromy

Rozhodovacie stromy predstavujú efektívny nástroj na podporu rozhodovania. Sú to klasifikátory, ktoré majú stromovú štruktúru. Rozdeľujú objekty popísané atribútmi do tried. Objekty sa označujú aj ako inštancie. Klasifikácia sa vykonáva tak, že inštancia objektu prechádza strom zhora nadol resp. od vrcholu k listom. Tieto stromy sa skladajú z vnútorných a vonkajších uzlov. Každý vnútorný uzol takéhoto stromu sa označuje ako rozhodovací. Rozhodovací uzol predstavuje test nad atribútom daného

objektu. Vetvy stromu znamenajú výsledky týchto testov, alebo inak povedané rozhodnutia. Vnútorný uzol sa asociuje s atribútom, ktorý je vstupný a vykoná previerku hodnoty tohto atribútu. Všetky vonkajšie uzly resp. listy stromu vyjadrujú konečné prijaté rozhodnutie pre daný objekt. Udávajú hodnotu cieľovej vlastnosti a radia objekt do konkrétnej triedy. Vonkajšie uzly sa asociujú s hodnotou výstupného atribútu. Výhodou rozhodovacích stromov je, že sú jednoduché na pochopenie i interpretáciu. Výsledná interpretácia stromu umožňuje rýchle a jednoduché spracovanie výsledkov. [5] [6]

Orezávanie stromov: Tento proces sa označuje aj anglickým výrazom *tree pruning*. Listy často obsahujú jeden trénovací príklad, alebo nedostatočný počet príkladov, vzhľadom na veľkosť dátovej množiny. Takýto rozhodovací strom správne klasifikuje trénovacie príklady, no s vysokou pravdepodobnosťou klasifikácia testovacích príkladov nebude správna. Orezávanie sa rozdeľuje na *prepruning* a *postpruning*. Metóda *prepruning* sa vykonáva už počas konštrukcie stromu, a to tak, že rozhoduje podľa kritéria zastavenia (prahových hodnôt), či vykoná rozdelenie dátovej množiny S podľa atribútu asociovaného s vrcholom, alebo rozdelenie nevykoná a vrchol prehlási za list. Metóda vychádza väčšinou zo štatistického testu, napríklad χ^2 – test. Pri metóde *postpruning* sa retrospektívne odstránia niektoré podstromy a každý odstránený podstrom sa nahradí listom.

Entropia je miera z teórie informácie, ktorá vyjadruje mieru neistoty. Pri hĺbkovej analýze dát entropia $\mathrm{H}(S)$ vyjadruje mieru nečistoty v dátovej množine *S*.

$$H(S) = -\sum_{x \in X} p(x) \times \log_2 p(x)$$

Kde:

- *S* je konkrétna dátová množina, pre ktorú počítame entropiu.
- *X* je množina tried patriacich do *S*.
- $p(x)$ je pomer počtu prvkov v triede x a počtu prvkov v dátovej množine *S*.

Logaritmus má základ 2, pretože entropia je miera očakávanej dĺžky kódovania vyjadrovanej v bitoch. Keď $\mathrm{H}(S) = 0$, tak dátová množina S je perfektne klasifikovaná. Všetky elementy *S* patria do rovnakej triedy.

Príklad: Nech dátová množina *S* obsahuje 14 inštancií. Nech 6 inštancií patrí do triedy + a 8 inštancií patrí do triedy -. Potom sa entropia $H(\mathrm{S})$ vyjadruje ako:

$$H(S) = H([6+,8-]) = -\left(\frac{6}{14}\right)\log_2\left(\frac{6}{14}\right) - \left(\frac{8}{14}\right)\log_2\left(\frac{8}{14}\right) = 0.985$$

Informačný zisk: Mnohokrát sa používa označenie anglickým pojmom *Information Gain*. Informačný zisk vyjadruje rozdiel entropií pred rozdelením dátovej množiny S a po rozdelení dátovej množiny S podľa atribútu A. Inými slovami, udáva množstvo neistoty, ktoré ubudlo z dátovej množiny S po jej rozdelení podľa atribútu A.

$$IG(A,S) = \mathrm{H}(S) - \sum_{x \in X} \frac{|S_x|}{|S|} \mathrm{H}(S_x)$$

Kde:

- $\mathrm{H}(S)$ je entropia množiny S.
- X je množina všetkých možných hodnôt atribútu A.
- S_x sú podmnožiny S, na ktorých atribút A nadobúda hodnotu x.

$$S = \bigcup_{x \in X} x$$

- $\mathrm{H}(S_x)$ je entropia podmnožiny S_x.

Informačný zisk môžeme využiť na ohodnotenie atribútov a skonštruovanie rozhodovacieho stromu, kde v každom vrchole bude umiestnený atribút s najväčším informačným ziskom. Atribúty, ktoré boli umiestnené do niektorého vrcholu, sa už viackrát nevyberajú. Každý atribút sa preto vyskytne na ceste od koreňa k niektorému z listov maximálne jedenkrát.

1.3.1 ID3 algoritmus

Tento algoritmus predstavil Ross Quinlan [7] ako vhodnú metódu na generovanie rozhodovacích stromov z dátových množín. Algoritmus buduje rozhodovacie stromy z množiny tréningových dát. Využíva koncept informačnej entropie. Tréningové dáta sú množina $S = s_1, s_2, \ldots$ už klasifikovaných prípadov. Každá vzorka s_i pozostáva z p-dimenzionálneho vektora $(x_{1,i}, x_{2,i}, \cdots, x_{p,i})$, kde x_j predstavuje hodnoty vlastností, ako aj triedu s_i, do ktorej prípad padne. V každom vrchole rozhodovacieho stromu algoritmus vyberá atribút A, ktorý najefektívnejšie rozdelí jeho množinu

vzoriek na podmnožiny obsiahnuté v prvej alebo druhej triede. V každej iterácií prechádza nepoužité atribúty množiny S a vypočíta entropiu H(S) alebo informačný zisk IG(A) atribútu A. Algoritmus vyberá atribút s najnižšou entropiou alebo najväčším informačným ziskom. Dátovú množinu S rozdelí na podmnožiny podľa zvoleného atribútu a tak vytvorí nové dátové podmnožiny z množiny S. Algoritmus rovnakým spôsobom spracováva všetky podmnožiny s tým, že vyberá atribúty, ktoré ešte predtým neboli v danej vetve stromu použité [6].

Rekurzívne spracovanie podmnožiny množiny S končí, ak nastane aspoň jedna z nasledujúcich podmienok:

- Každý element množiny vo vrchole patrí do tej istej triedy (+ alebo -), potom je uzol prehlásený za list a označený triedou príkladov**.**
- Príklady stále nepatria do jednej triedy (niektoré sú + a niektoré -), ale nie je možné vybrať ďalší atribút. Potom je uzol prehlásený za list a označený najčastejšou triedou príkladov v podmnožine.
- V podmnožine nie sú žiadne príklady. Tento prípad nastane, ak nebol nájdený žiadny príklad v rodičovskej množine (v strome predok aktuálnej množiny), ktorý by zodpovedal konkrétnej hodnote vybraného atribútu. Napríklad, neexistuje prípad pre $plat \leq 50$. Potom je takýto uzol prehlásený za list a označí sa najčastejšou triedou rodičovskej množiny.

Činnosť algoritmu sa dá zhrnúť nasledovne:

1. Vypočítaj entropiu každého atribútu použitého v množine S.
2. Rozdeľ množinu S na podmnožiny podľa atribútu, ktorý má najmenšiu hodnotu entropie, alebo najväčšiu hodnotu informačného zisku.
3. Algoritmus pokračuje rekurzívne na podmnožiny s tým, že vyberá nepoužité atribúty.

Pseudokód [7]**:**

ID3 (Examples, Target_Attribute, Attributes)

- Vytvor vrchol stromu;
- Ak všetky príklady patria do tej istej triedy (+), tak vráť list s označením +;
- Ak všetky príklady patria do tej istej triedy (-), tak vráť list s označením -;

- Ak je množina prediktívnych atribútov prázdna, tak vráť list označený najpočetnejším cieľovým atribútom príkladov.
- Ináč Begin
    - A ← Atribút, ktorý najlepšie klasifikuje príklady
    - Rozhodovací atribút pre vrchol = A
    - Pre každú možnú hodnotu v_i atribútu A
        - Pridaj pod vrchol vetvu korešpondujúcu s testom $A = v_i$
        - Nech Examples (v_i) je podmnožina príkladov, ktoré majú hodnotu v_i pre A
        - Ak Examples (v_i) je prázdna
            - Pod touto vetvou pridaj list označený najpočetnejšou cieľovou hodnotou v príkladoch.
        - Ináč pod túto vetvu pridaj podstrom ID3 (Examples(vi), Target_Attribute, Attributes – {A})

Príklad algoritmu ID3

V nasledujúcom príklade ukážem konštrukciu ID3 rozhodovacieho stromu. Dáta sú zadané v Tabuľka 2. Trieda - vyjadruje agresívnu povahu a trieda + predstavuje pokojnú povahu.

P. č.	SKLON	HRÚBKA	SLUČKA	TRIEDA (POVAHA)
1.	Rovný	Tenký	Áno	-
2.	Naklonený	Normálny	Áno	-
3.	Naklonený	Tenký	Nie	+
4.	Naklonený	Normálny	Nie	-
5.	Rovný	Normálny	Nie	-
6.	Naklonený	Hrubý	Nie	+
7.	Naklonený	Tenký	Áno	-
8.	Rovný	Tenký	Nie	+

Tabuľka 2. Zadané dáta pre príklad ID3

Ako prvý krok je potrebné určiť atribút, ktorý sa asociuje s vrcholom rozhodovacieho stromu ID3. S vrcholom stromu sa asociuje atribút s maximálnou

hodnotou informačného zisku IG. V nasledujúcej časti sa vypočítajú informačné zisky pre jednotlivé atribúty. Vždy sa berú do úvahy len doposiaľ nezaradené atribúty, no keďže vyberáme atribút pre vrchol, tak množina nezaradených atribútov obsahuje všetky atribúty.

$$H(S) = -\sum_{x \in X} p(x)\log_2 p(x) = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8} = 0.9544$$

- **Výpočet informačného zisku pre atribút SKLON**

Rovný	Naklonený	Spolu
2-	3-	5-
1+	2+	3+

$$H(rovný) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.9183$$

$$H(naklonený) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.9710$$

$$H(S, SKLON) = \frac{3}{8}\cdot 0.9183 + \frac{5}{8}\cdot 0.9170 = 0.9512$$

$$\boldsymbol{IG(\text{SKLON}) = H(S) - H(S, \text{SKLON}) = 0.9544 - 0.9512 = 0.0032}$$

- **Výpočet informačného zisku pre atribút HRÚBKA**

Tenký	Normálny	Hrubý	Spolu
2-	3-	0-	5-
2+	0+	1+	3+

$$H(tenký) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1$$

$$H(normálny) = -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3} \to 0$$

$$H(hrubý) = -\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1} \to 0$$

$$H(S, HRÚBKA) = \frac{4}{8}\cdot 1 + \frac{3}{8}\cdot 0 + \frac{1}{8}\cdot 0 = 0.5$$

$$\boldsymbol{IG(HRÚBKA) = H(S) - H\left(S, HRÚBKA\right) = 0.9544 - 0.5 = 0.4544}$$

- **Výpočet informačného zisku pre atribút SLUČKA**

Nie	Áno	Spolu
2-	3-	5-
3+	0+	3+

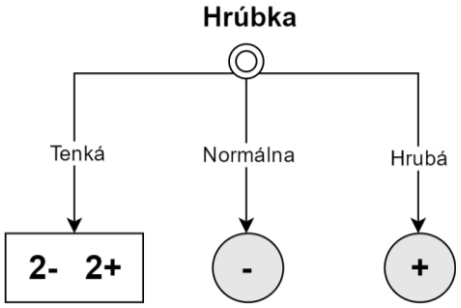
$$H(nie) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.9710$$

$$H(áno\ ) = -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3} \to 0$$

$$H(S, SLUČKA) = \frac{5}{8}\cdot 0.9710 + \frac{3}{8}\cdot 0 = 0.6069$$

$$\boldsymbol{IG(SLUČKA) = H(S) - H\left(S, SLUČKA\right) = 0.9544 - 0.6069 = 0.3475}$$

Maximálny informačný zisk *IG* nadobúda atribút *Hrúbka*, a preto tento atribút asociujeme s koreňom rozhodovacieho stromu. Atribút *Hrúbka* nadobúda tri hodnoty. To znamená, že z tohto vrcholu pôjdu tri vetvy, pričom každá vetva bude pre inú hodnotu atribútu.



Obrázok 1. Rozhodovací strom ID3 - prvá úroveň

Pre *hrubú* a *normálnu* Hrúbku existuje len jeden príklad, to spôsobí, že všetky prípady pre *hrubú* a *normálnu* Hrúbku padnú do jednej triedy. *Normálna* Hrúbka padne do triedy -, hrubá do triedy +. Pokiaľ padnú všetky prípady do jednej triedy, tak v takom vrchole sa už ďalej nebude vetviť a prehlási sa za list. V takýchto vrcholoch sa rast stromu zastaví. Pri *tenkej* Hrúbke padnú dva prípady do triedy + a dva prípady do triedy -. V takom vrchole sa bude vo vetvení pokračovať ďalej a rast stromu sa v tomto vrchole nezastaví. V novom vrchole bude nasledovná dátová množina:

Hrúbka tenká:

P. č.	SKLON	SLUČKA	TRIEDA (POVAHA)
1.	Rovný	Áno	-
3.	Naklonený	Nie	+
7.	Naklonený	Áno	-
8.	Rovný	Nie	+

$$H(S) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1$$

- **Výpočet informačného zisku pre atribút Sklon**

Rovný	Naklonený	Spolu
1-	1-	2-
1+	1+	2+

$$H(rovný) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

$$H(naklonený) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

$$H(S, Sklon) = \frac{2}{4}\cdot 1 + \frac{2}{4}\cdot 1 = 1$$

$$\boldsymbol{IG(Sklon) = H(S) - H(S, Sklon) = 1 - 1 = 0}$$

- **Výpočet informačného zisku pre atribút Slučka**

Nie	Áno	Spolu
0-	2-	2-
2+	0+	2+

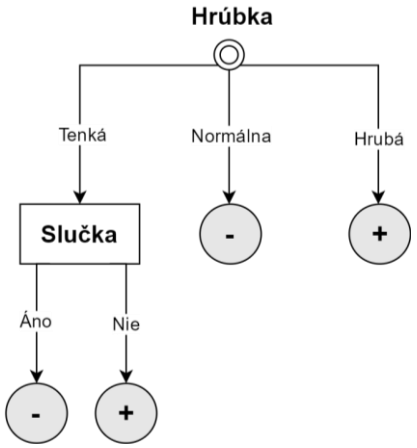
$$H(nie) = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} \to 0$$

$$H(áno) = -\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2} \to 0$$

$$H(S, OČI) = \frac{2}{4}\cdot 0 + \frac{2}{4}\cdot 0 = 0$$

$$\boldsymbol{IG(Slučka) = H(S) - H(S, Slučka) = 1 - 0 = 1}$$

Maximálny informačný zisk IG má atribút *Slučka*. Preto do vrcholu asociujeme práve tento atribút. V tomto prípade atribút nadobúda dve hodnoty. Tento fakt spôsobí, že z tohto vnútorného vrcholu rozhodovacieho stromu budú vychádzať dve vetvy.



Obrázok 2. Rozhodovací strom ID3 - druhá úroveň

Všetky prípady, pri ktorých hodnota atribútu *Slučka* je rovná *áno* padnú do triedy -. Preto sa tento vrchol prehlási za list a označí sa triedou -. Tak isto všetky prípady pre hodnotu atribútu *slučka* rovnajúce sa hodnote *nie* padnú do jednej triedy a tak tiež sa tento vrchol prehlási za list a označí sa triedou +. Z dôvodu, že všetky vonkajšie vrcholy sú prehlásené za list, už nebudeme ďalej vetviť a strom na **Chyba! Nenašiel sa žiaden zdroj odkazov.** je výsledný rozhodovací strom.

1.3.2 C4.5 algoritmus

Tento algoritmus predstavil taktiež Ross Quinlan v [7]. Jedná sa o vylepšenie algoritmu ID3, pričom C4.5 sa označuje aj ako nástupca algoritmu ID3. Taktiež sa využíva na budovanie rozhodovacích stromov a klasifikáciu. Označuje sa aj ako štatistický klasifikátor. Popularita algoritmu vzrástla, keď početná skupina odborníkov vytvorila rebríček algoritmov, ktoré riešia úlohy hĺbkovej analýzy. Publikovali ho v *Top 10 algorithms in data mining* [8]. Tento algoritmus zaradili na prvé miesto rebríčka.

C4.5 algoritmus buduje rozhodovacie stromy z množiny tréningových dát rovnakým spôsobom ako ID3 algoritmus. Rozdiel je v tom, že atribúty vyberá podľa normalizovaného informačného zisku:

$$nIG(A,S) = \frac{IG(A,S)}{V(A)}$$

Kde $V(A)$ je počet synov vrcholu, v ktorom sme rozdelili dátovú množinu *S* podľa atribútu A.

Oproti algoritmu ID3 dokáže pracovať s diskrétnymi aj spojitými atribútmi. ID3 pracuje len s diskrétnymi. Dokáže sa vysporiadať aj s chýbajúcimi hodnotami atribútov v trénovacej dátovej množine. Počas výpočtu entropií berie do úvahy len známe hodnoty atribútov. Chýbajúce hodnoty atribútov sa nevyužijú pri výpočte informačného zisku. Orezávanie realizuje metódou *postpruning*. Orezávanie v algoritme C4.5 využíva štatistické odhady spoľahlivosti. Táto technika má výhodu v tom, že všetky označené dáta môžu byť použité na trénovanie. Základ je vypočítať interval spoľahlivosti pre chybovosť. Stručne povedané, vychádza z pozorovania výberovej chybovosti f pozorovanej nad množinou *N* trénovacích prípadov. Výberová chybovosť je analogická pomeru hláv v *N* hodoch mincou. Úmyslom je odhadnúť skutočnú chybovosť p, ktorú by sme získali, ak by test prešiel cez všetky možné príklady dát, ktoré budú vždy k dispozícii. Skutočná chybovosť p je analogická ku skutočnej pravdepodobnosti, že pri hode mincou padne hlava. Rozdiel medzi f a p je ten, že p je populačná chybovosť, zatiaľ čo f je chybovosť daná pomerom chýb a veľkosti dátovej trénovacej množiny, ktorá je k disponujem. Populačná chybovosť je daná pomerom chýb, ktoré sú zaznamenané v nekonečne veľkej dátovej trénovacej množine ku veľkosti tejto množiny.

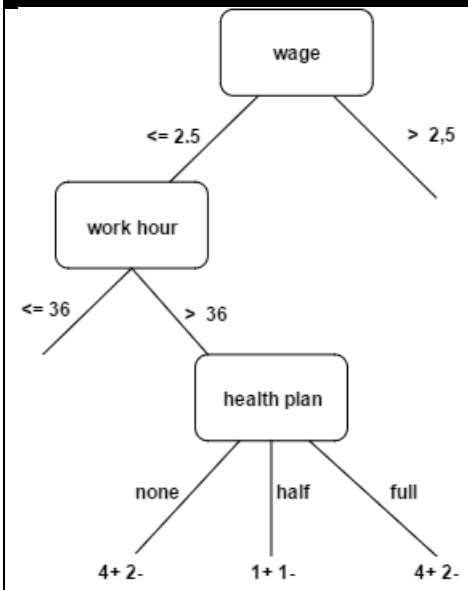
Interval spoľahlivosti sa vypočíta nasledovne:

$$p = f \pm z * \sqrt{\frac{f * (1 - f)}{N}}$$

Kde z je faktor, ktorý závisí na požadovanej úrovni spoľahlivosti a p, f, N sú popísané vyššie. Aby bolo možné rozhodnúť, či algoritmus nahradí podstrom určitého vrcholu jedným listom, tak C4.5 porovná hornú hranicu intervalu spoľahlivosti pre chybovosť dvoch stromov. Pre neorezaný strom sa horná hranica chybovosti stanový ako vážený priemer jeho podradených vrcholov. Algoritmus určí, aká bude horná hranica chybovosti po orezaní a následne zvolí strom, ktorý má hornú hranicu chybovosti menšiu.

Príklad:

Nech hodnota $z = 0{,}69$.

Neorezaný strom	
	health plan: none $f = \frac{2}{6}, N = 6$ horná hranica $p = 0.46$ health plan: half $f = \frac{1}{2}, N = 2$ horná hranica $p = 0.74$ health plan: full $f = \frac{2}{6}, N = 6$ horná hranica $p = 0.46$ **Priemer: 0.55**

Pokiaľ by sme nahradili uzol *health plan* listom (9+, 5-), tak výpočet intervalu spoľahlivosti by bol nasledovný:

$$f = \frac{5}{14}, N = 14$$

$$\text{horná hranica } p = 0.49$$

Pretože orezaný strom má nižšiu hornú hranicu horného odhadu chybovosti, tak by k orezaniu skutočne došlo.

2 FUZZY A HÍLBKOVÁ ANALÝZA DÁT

Nosná časť diplomovej práce sa zaoberá vytvorením aplikácie na vyhodnocovanie dotazníkov, v ktorých sú otázky vyhodnocované pomocou fuzzy premenných. Preto uvediem aj informácie o fuzzy logike [9].

Teória množín vyjadruje členstvo, a to, či daný prvok do množiny jednoznačne patrí alebo jednoznačne nepatrí. Môže tam patriť alebo nepatriť, iná možnosť neexistuje. Napríklad, stanovená optimálna hmotnosť je 60 kg a viac. Prvok s hmotnosťou 71 kg do množiny jednoznačne patrí, zatiaľ čo prvok s váhou 59.5 kg, ale aj s váhou 12 kg do množiny jednoznačne nepatria. Pri fuzzy logike sa na prvky pozeráme tak, že vyjadríme, či je daný prvok typickým prvkom množiny. Robí sa to tak, že stanovíme stupeň príslušnosti pre daný prvok. Tento stupeň nám poskytne funkcia príslušnosti, ktorá je pre každú fuzzy množinu definovaná.

Nech U je univerzum rozpravy (angl. Universe of discourse). Univerzum je množina, ktorá zahŕňa súbor skúmaných prvkov. V množine U zadefinujeme fuzzy množinu A. Množina A je charakterizovaná funkciou príslušnosti (angl. Membership function) nasledovne [9]:

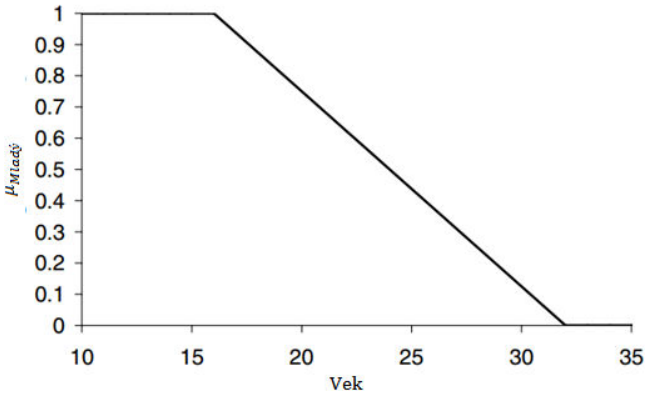
$$\mu_A(u) = 1 \text{ vtedy a len vtedy, ak } x \text{ je určite prvkom množiny } A$$
$$\mu_A(u) = 0 \text{ vtedy a len vtedy, ak } x \text{ určite nie je prvkom množiny } A$$

Funkcia $\mu_A(u)$ sa označuje aj ako *funkcia príslušnosti.* Táto funkcia pre všetky prvky u univerzálnej množiny U priradí hodnotu z intervalu $\langle 0,1 \rangle$. Fuzzy množina je zadefinovaná ako usporiadaná množina dvojíc:

$$A = \{(u, \mu_A(u)), u \in A\}$$

Predchádzajúcu definíciu môžeme ilustrovať na nasledujúcom príklade. V tomto prípade je množina U množinou mladých ľudí, ďalej označovaná *Mladí*. V príklade sa pokúsime vyjadriť, či je prvok typický člen množiny tým, že udáme stupeň príslušnosti človeka u k fuzzy množine *Mladý*. Funkcia príslušnosti bude odpovedať na otázku „do akej miery je človek u v skupine mladých ľudí?“. Najjednoduchší spôsob ako to urobiť, je použiť členskú funkciu, ktorá vychádza z veku osoby. V príklade použijeme nasledovnú členskú funkciu:

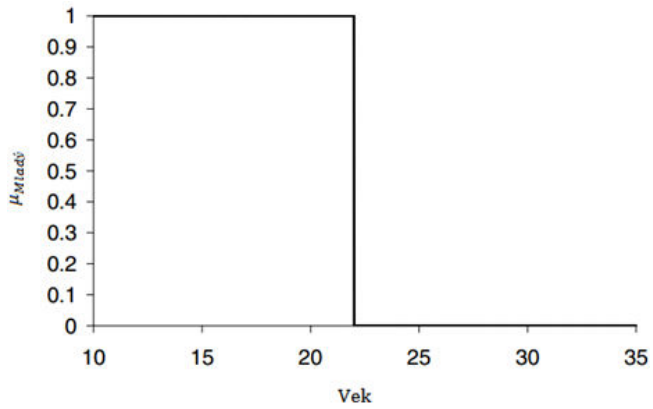
$$\mu_{Mladý}(u) = \begin{cases} 0 & vek(u) > 32 \\ 1 & vek(u) < 16 \\ \dfrac{32 - vek(u)}{16} & inák \end{cases}$$



Obrázok 3. Graf funkcie príslušnosti $\mu_{Mladý}$

Zvolená funkcia príslušnosti by priradila 18-ročnému jedincovi stupeň príslušnosti rovný 0.875. Pre 20 ročnú osobu by stupeň príslušnosti zodpovedal hodnote 0.75. Na rozdiel od teórie pravdepodobnosti, súčet všetkých stupňov členstva prvkov množiny nemusí dávať súčet 1. Preto v množine môže existovať veľa, ale aj málo objektov s vysokým stupňom členstva. Avšak objekty, ktoré patria do množiny (napr. *Mladý*) s doplnkom (komplementom) tejto množiny sa musia v súčte rovnať číslu 1. Ako som spomínal vyššie, hlavný rozdiel medzi klasickou teóriou množí a fuzzy teóriou množín je čiastočné pripustenie členstva. Klasická množina A sa môže považovať za fuzzy množinu *Mladý*, ktorej funkcia príslušnosti $\mu_{Mladý}(u)$ obmedzuje hodnoty členstva na $\{0,1\}$. Pre príklad, ktorý radí do množina A mladých ľudí by funkcia príslušnosti vyzerala nasledovne:

$$\mu_{Mladý}(u) = \begin{cases} 0, & vek(u) > 22 \\ 1, & vek(u) \leq 22 \end{cases}$$



Obrázok 4. Graf funkcie príslušnosti $\mu_{Mladý}$

V typických klasifikačných problémoch predpokladáme, že pre každý atribút jednej inštancie existuje jedna hodnota a každá inštancia je klasifikovaná do jednej z vopred zvolených tried. Pre ilustráciu ako môže fuzzy logika pomôcť pri úlohách hĺbkovej analýzy dát ukážem na príklade, ktorý modeluje problém preferencií televíznych divákov. Tento problém je popísaný pomocou troch vstupných atribútov:

$$A = \{\text{Denný čas}, \text{Veková kupina}, \text{Nálada}\}$$

Atribúty môžu nadobúdať nasledujúce hodnoty:

- $dom\,(\text{Denný čas}) = \{\text{Ráno}, \text{Obed}, \text{Večer}, \text{Noc}\}$
- $dom\,(\text{Veková kupina}) = \{\text{Mladý}, \text{Starý}\}$
- $dom\,(\text{Nálada}) = \{\text{Šťastný}, \text{Znudený}, \text{Smutný}, \text{Otrávený}, \text{Nahnevaný}\}$

Klasifikovať môžeme na základe filmového žánru, ktorý diváci pozerajú najradšej. Klasifikačné triedy pre danú úlohu budú nasledovné:

$$C = \{\text{Akcia}, \text{Komédia}, \text{Dráma}\}$$

Všetky tieto atribúty sú definované vágne, neurčito. Napríklad ľudia cítia šťastie, znudenosť, smútok či hnev neurčito bez toho, aby tieto emócie oddeľovali nejaké jasné hranice. Samozrejme pri atribútoch ako veková skupina alebo denný čas sa môžeme vyhnúť vágnym hodnotám definovaním exaktných hodnôt pre vek alebo čas. Takéto definovanie pravidiel by ale spôsobilo v rozhodovacích stromoch prísne ohraničenie hodnôt atribútov, ako napríklad „AK VEK < 16 POTOM akcia“. Ale čo ak by sme klasifikovali niekoho, kto ma 17 rokov? Mohli by sme tvrdiť, že nepozerá akčné filmy? Aj divákov preferovaný žáner môže byť stále neurčitý atribút. Pokiaľ má divák náladu

pozerať drámu aj komédiu súčasne. Navyše, priradenie žánrov k jednotlivým filmom je tiež vágne. Napríklad film "*Lethal Weapon*" je označovaný aj za komédiu a aj za akčný film.

Fuzzy logika sa môže využiť v klasifikačných úlohách v prípade, ak aspoň jeden vstupný atribút je fuzzy, alebo cieľový atribút je fuzzy. V príklade vyššie sú všetky vstupné a aj výstupný atribút fuzzy atribútmi.

Tento problém zadefinovali v roku 1995 Yuan a Shaw v [10] nasledovne:

Každá trieda c_j je definovaná ako fuzzy množina na univerze objektov U. Funkcia príslušnosti $\mu_{c_j}(u)$ indikuje stupeň príslušnosti, ktorý udáva do akej miery patrí objekt u do triedy c_j. Každý atribút a_i je definovaný ako lingvistický atribút, ktorý nadobúda lingvistické hodnoty z $dom(a_i) = \{v_{i,1}, v_{i,2}, \cdots, v_{i,|dom(ai)|}\}$. Každá lingvistická hodnota $v_{i,k}$ je taktiež fuzzy množina definovaná na U. Funkcia príslušnosti $\mu_{v_{i,k}}(u)$ špecifikuje stupeň, s ktorým atribút a_i objektu u nadobúda hodnotu $v_{i,k}$. Lingvistické hodnoty členských funkcií môžu byť subjektívne priradené alebo transformované z numerických hodnôt členskej funkcie definovaných ako rozsah numerických hodnôt.

2.1 Operácie na fuzzy množinách

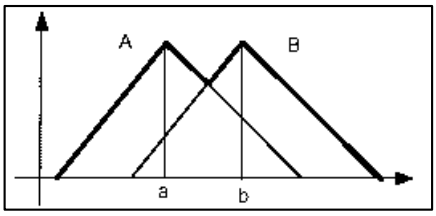
Tak ako klasická teória množín, aj teória fuzzy množín zahŕňa operácie ako zjednotenie, doplnok, prienik a inklúziu. Teória fuzzy množín zahŕňa aj také operácie, ktoré klasická teória množín neobsahuje napríklad náprotivok (angl. *counterpart*).

Nech fuzzy množiny A s členskou funkciou $\mu_A(u)$ a B s členskou funkciou $\mu_B(u)$sú definované na univerze U.

Funkcia príslušnosti zjednotenia dvoch fuzzy množín A a B s funkciami príslušnosti $\mu_A(u)$ a $\mu_B(u)$ je definovaná ako maximum z individuálnych funkcií príslušnosti množín A a B.

Funkcia príslušnosti prieniku množín A a B je definovaná ako:

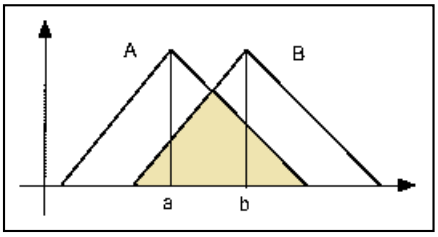
$$\mu_{A\cap B}(u) = min\{\mu_A, \mu_B\}$$



Obrázok 5. $\mu_{A\cap B}$

Funkcia príslušnosti zjednotenia množín A a B je definovaná ako:

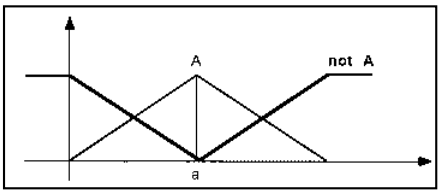
$$\mu_{A\cup B}(u) = max\{\mu_A, \mu_B\}$$



Obrázok 6. $\mu_{A\cup B}$

Funkcia príslušnosti doplnku množiny A je definovaná ako:

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u)$$



Obrázok 7. $\mu_{\bar{A}}$

Pre ilustráciu fuzzy operácií uvediem nasledujúci príklad. Funkcia príslušnosti hovorí, že osoba patrí do množiny *Mladý* so stupňom príslušnosti s hodnotou 0.875. Do množiny *Múdry* patrí so stupňom členstva 0.125.

$$Mladý \cap Múdry = max\{0.875\ , 0.125\} = 0.875.$$

Operácie na fuzzy množinách umožňujú reťazenie operátorov, napríklad $\overline{Mladý} \cap Múdry$.

Fuzzy *subsethood* $S(A,B)$ meria stupeň, s ktorým je množina A podmnožinou množiny B [9].

$$S(A,B) = \frac{M(A \cap B)}{M(A)}$$

kde $M(A)$ vyjadruje mohutnosť množiny A definovanú nasledovne:

$$M(A) = \sum_{u \in U} \mu_A(u)$$

Subsethood sa využíva na meranie pravdivostnej úrovne pravidla klasifikačných pravidiel [9].

2.2 Klasifikačné úlohy fuzzy dát

Klasifikačná úloha definovaná pomocou fuzzy dát sa popisuje podobne ako klasifikačné úlohy v kapitole Klasifikačné úlohy. Pri opise klasifikačných úloh, pri ktorých sú dáta zadané pomocou fuzzy logiky vychádzam z [4]. Univerzum $U = \{u\}$ je tvorené N trénovacimi inštanciami a n vstupnými atribútmi. Každý vstupný atribút A_i $1 \leq i \leq n$ vyjadruje určitú dôležitú vlastnosť a je prezentovaný skupinou diskrétnych lingvistických hodnôt. Predpokladá sa, že každá táto skupina je súbor hodnôt m_i $(m \geq 2)$ hodnôt fuzzy podmnožín $\{A_{i,1}, \dots, A_{i,j}, \dots, A_{j,m_i}\}$. Atribút A_i v tomto type úloh býva často ohodnotený nákladmi $Cost_i$. Tieto náklady vyjadrujú materiálne a časové náklady na zmeranie hodnoty atribútu A_i. V týchto úlohách, je každý objekt u z univerza klasifikovaný do množiny tried $\{B_1, \dots, B_{m_b}\}$. Táto množina tried popisuje výstupný atribút B. Zásadný rozdiel oproti kapitole Klasifikačné úlohy je v tom, že pri fuzzy dátach nie sú klasifikované inštancie radené len do jednej triedy. Algoritmy, ktoré sú v práci popísané udávajú úroveň vierohodnosti primanej alternatívy rozhodnutia. Inými slovami, na konci klasifikácie dostaneme informáciu, s akou vierohodnosťou je inštancia klasifikovaná do každej triedy. Vierohodnosť je vyjadrená stupňom príslušnosti novej inštancie k danej triede.

2.3 Fuzzy dáta

Vhodná reprezentácia fuzzy dát pre potreby hĺbkovej analýzy dát je pomocou fuzzy tabuľky. Fuzzy tabuľka je v práci [3] definovaná ako šestica $\{CS, CV, AS, AV, C, P\}$ popísaná nasledovne:

$CS = \{A_1, \dots, A_n\}$ je množina vstupných atribútov,

$CV = \{A_{i,j}\}$ pre $i = 1, \dots, n; j = 1, \dots, m_i$ je množina hodnôt vstupných atribútov,

AS = množina výstupných atribútov,

$AV = \{B_1, \dots, A_{m_b}\}$ je množina hodnôt výstupného fuzzy atribútu,

$C = \{Cost(A_i)\}$ sú náklady na stanovenie hodnôt výstupného fuzzy atribútu,

P je tabuľka, opisujúca množinu N situácií.

V nasledujúcich príkladoch budem ako vstupnú fuzzy tabuľku využívať tabuľku so štyrmi vstupnými atribútmi $\{A_1, A_2, A_3, A_4\}$ a jedným výstupným atribútom $\{B\}$. V tabuľke je zaznamenaných $N = 16$ výsledkov pozorovania. Každý vstupný atribút A_i nadobúda hodnoty A_{i,q_m}. Jednotlivé riadky v tabuľke vyjadrujú stupeň príslušnosti k týmto hodnotám. Tabuľka v príklade je prevzatá z prác [3], [10] a [11].

NO	A_1			A_2			A_3		A_4		B		
	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{3,1}$	$A_{3,2}$	$A_{4,1}$	$A_{4,2}$	B_1	B_2	B_3
1.	0.9	0.1	0.0	1.0	0.0	0.0	0.8	0.2	0.4	0.6	0.0	0.8	0.2
2.	0.8	0.2	0.0	0.6	0.4	0.0	0.0	1.0	0.0	1.0	0.6	0.4	0.0
3.	0.0	0.7	0.3	0.8	0.2	0.0	0.1	0.9	0.2	0.8	0.3	0.6	0.1
4.	0.2	0.7	0.1	0.3	0.7	0.0	0.2	0.8	0.3	0.7	0.9	0.1	0.0
5.	0.0	0.1	0.9	0.7	0.3	0.0	0.5	0.5	0.5	0.5	0.0	0.0	1.0
6.	0.0	0.7	0.3	0.0	0.3	0.7	0.7	0.3	0.4	0.6	0.2	0.0	0.8
7.	0.0	0.3	0.7	0. 0	0.0	1.0	0.0	1.0	0.1	0.9	0.0	0.0	1.0
8.	0.0	1.0	0.0	0.0	0.2	0.8	0.2	0.8	0.0	1.0	0.7	0.0	0.3
9.	1.0	0.0	0.0	1.0	0.0	0.0	0.6	0.4	0.7	0.3	0.2	0.8	0.0
10.	0.9	0.1	0.0	0.0	0.3	0.7	0.0	1.0	0.9	0.1	0.0	0.3	0.7
11.	0.7	0.3	0.0	1.0	0.0	0.0	1.0	0.0	0.2	0.8	0.3	0.7	0.0
12.	0.2	0.6	0.2	0.0	1.0	0.0	0.3	0.7	0.3	0.7	0.7	0.2	0.1
13.	0.9	0.1	0.0	0.2	0.8	0.0	0.1	0.9	1.0	0.0	0.0	0.0	1.0
14.	0.0	0.9	0.1	0.0	0.9	0.1	0.1	0.9	0.7	0.3	0.0	0.0	1.0
15.	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.8	0.2	0.0	0.0	1.0
16.	1.0	0.0	0.0	0.5	0.5	0.0	0.0	1.0	0.0	1.0	0.5	0.5	0.0
Cost(A_i)	2.5			1.7			2.0		1.8				

Tabuľka 3. Fuzzy tabuľka

Prvý riadok tabuľky možno vysvetliť nasledovne. Z tabuľky je zrejmé, že atribút A_1 nadobúda hodnoty $\{A_{1,1}, A_{1,2}, A_{1,3}\}$. Číselné hodnoty v tabuľke udávajú stupne príslušnosti do jednotlivých hodnôt $\{A_{1,1}, A_{1,2}, A_{1,3}\}$ atribútu A_1. Každá hodnota atribútu $\{A_{1,1}, A_{1,2}, A_{1,3}\}$ predstavuje jednu fuzzy množinu. Prvý riadok v Tabuľka 3 pre atribút A_1 udáva nasledujúce stupne príslušnosti $\{\mu_{A_{1,1}}(x) = 0.9, \mu_{A_{1,2}}(x) = 0.1, \mu_{A_{1,3}}(x) = 0.0\}$. V tomto prípade atribút A_1 nadobúda hodnotu $A_{1,1}$ so stupňom príslušnosti 0,9. Pripúšťa sa aj hodnota $A_{1,2}$, kde je ale stupeň príslušnosti len 0,1.

Atribút A_1 hodnotu $A_{1,3}$ určite nenadobudne, pretože stupeň príslušnosti pre túto hodnotu je rovný 0. Atribút A_2 v prvom riadku nadobúda jednoznačne hodnotu $A_{2,1}$, pretože stupeň príslušnosti je $\mu_{A_{2,1}}(x) = 1$. Celá tabuľka sa dá opísať analogickým postupom.

2.4 Transformácia reálnych hodnôt atribútu na lingvistickú premennú

V práci predpokladám, že dátová množina môže okrem fuzzy atribútov popísaných lingvistickými premennými (atribúty ako v Tabuľka 3. Fuzzy tabuľka) obsahovať aj numerické atribúty. Tieto atribúty je potrebné transformovať na lingvistické premenné. Proces transformácie sa vykonáva manuálne odborníkom v danej oblasti, alebo môže využiť automatizovaný algoritmus. Algoritmus, ktorý som v práci využíval vychádza z popisu tejto transformácie v dielach [12] a [3]. Ja som postup popísaný v týchto prácach rozdelil na dva algoritmy.

Prvý algoritmus transformuje každú hodnotu x_i numerického atribút A na lingvistickú premennú vyjadrenú pomocou Q hodnôt funkcie príslušnosti

$$\mu_{A_1}(x_i), \dots, \mu_{A_q}(x_i), \dots \mu_{A_Q}(x_i), \text{ kde } \mu_{A_1}(x_i) + \mu_{A_q}(x_i) + \mu_{A_Q}(x_i) = 1.$$

Druhý algoritmus cyklicky vykonáva prvý algoritmus. Začína rozkladom atribútu A na dva intervaly $Q = 2$. Počet intervalov sa zvyšuje v každej iterácii cyklu, pokiaľ klesá fuzzy entropia v dátovej množine.

Transformácia reálnych hodnôt atribútu na lingvistickú premennú

- Dáta
  - Vstupné dáta: počet intervalov Q. Tento počet udáva počet funkcií príslušnosti. Ďalší vstupný parameter je atribút A definovaný reálnymi hodnotami.
  - Výstupné dáta: Fuzzy matica U s rozmermi $N \times Q$, kde N je počet hodnôt x_i atribútu A.
- Postup:

1. Hodnoty x_i atribútu A rozdelíme do Q intervalov $\{Q_1, \dots, Q_q, \dots, Q_Q\}$ podľa Algoritmus zhlukovania reálnych hodnôt popísaného v kapitole Zhlukovanie. Pripomeniem, že každý interval Q_q obsahuje centrum C_q.
2. Následne sa sformulujú funkcie príslušnosti podľa vzťahov.
   - Pre prvý lingvistický term $x_{i,1}$ sa použije nasledujúca funkcia príslušnosti:

$$\mu_{x_{i,1}}(x) = \begin{cases} 1 & x \le C_1 \\ \dfrac{C_2 - x}{C_2 - C_1} & C_1 < x < C_2 \\ 0 & x \ge C_2 \end{cases}$$

   - Každý lingvistický term $x_{i,q}$, $q = 2, \dots, N-1$ má nasledujúcu trojuholníkovú funkcia príslušnosti:

$$\mu_{x_{i,q}}(x) \begin{cases} 0 & x \le C_{j-1} \\ \dfrac{x - C_{j-1}}{C_j - C_{j-1}} & C_{j-1} < x \le C_j \\ \dfrac{C_{j+1} - x}{C_{j+1} - C_j} & C_j < x \le C_{j+1} \\ 0 & x \ge C_{j+1} \end{cases}$$

   - Funkcia príslušnosti pre posledný lingvistický term $x_{i,Q}$ je:

$$\mu_{x_{i,Q}}(x) = \begin{cases} 0 & x \le C_{k-1} \\ \dfrac{x - C_{k-1}}{C_k - C_{k-1}} & C_{k-1} < x \le C_k \\ 1 & x \ge C_k \end{cases}$$

Transformácia reálnych hodnôt atribútu na lingvistickú premennú s optimálnym počtom lingvistických termov [3].

V tomto algoritme sa cyklicky opakuje vyššie popísaný algoritmus. Počet fuzzy termov sa zvyšuje v každej iterácií algoritmu až kým nevstupné vážená fuzzy entropia. Fuzzy entropia $FE_{\langle d_1, d_2 \rangle}$ intervalu $\langle d_1, d_2 \rangle$ je vyjadrená nasledujúcim vzorcom:

$$FE_{\langle d_1, d_2 \rangle}(\mathrm{A_q}) = -\sum_{k=1}^{K} D_{A_q}^k \times \log_2 D_{A_q}^k, \text{ kde } D_{A_q}^k = \frac{\sum_{x_{i \in b_k}} \mu_{A_q}(x_i)}{\sum_{x_i} \mu_{A_q}(x_i)}$$

$$FE_{\langle d_1, d_2 \rangle}(\mathrm{A}) = \sum_{\mathrm{q}=1}^{0} \mathrm{FE}(\mathrm{A_q})$$

Algoritmus vo svojich výpočtoch využíva fuzzy váženú entropiu, ktorá prihliada aj na počet prvkov v jednotlivých intervaloch. Fuzzy vážená entropia sa vypočíta nasledovne:

$$wFE_{\langle d_1,d_2\rangle} = \frac{N_q}{N} \times FE_{\langle d_1,d_2\rangle} \text{ (A)}$$

- Dáta
  - Vstupné dáta sú parametre výstupný fuzzy atribút a vstupný numerický atribút A definovaný reálnymi hodnotami.
  - Výstupné dáta: Fuzzy matica U s rozmermi $N \times Q$, kde N je počet hodnôt x_i atribútu A a Q udáva počet termov.
- Postup
  1. Stanov počiatočný počet centier na $Q = 2$
  2. $wFE(A)_{Q-1}$ nastav na nekonečno
  3. Vykonaj transformáciu reálnych hodnôt atribútu na lingvistickú premennú
  4. Vypočítaj $wFE(A)_Q$. Ak je $wFE(A)_{Q-1} < wFE(A)_Q$, tak algoritmus končí, lebo našiel optimálny počet intervalov. Ináč zväčši $Q = Q + 1$, priraď $wFE(A)_{Q-1} = wFE(A)_Q$ a pokračuj krokom 3.

2.5 Informačné charakteristiky

Pre potreby nasledujúcich algoritmov zadefinujem informačné charakteristiky podľa [3].

Nech A_i je fuzzy atribút, ktorý nadobúda hodnoty $A_{i,j}$. *Sumárna vlastná entropia* atribútu A_i je aritmetický priemer (vážená suma) hodnôt $A_{i,j}$.

$$H_f(A_i) = \sum_{j=1}^{m_i} M(A_i) \times I\left(A_{i,j}\right), \text{ kde } I\left(A_{i,j}\right)$$

sa označuje ako *vlastná informácia* a vypočíta sa zo vzťahu

$$I\left(A_{i,j}\right) = \log_2 N - \log_2 M\left(A_{i,j}\right).$$

N vyjadruje počet dát vo fuzzy tabuľke. Funkcia $M\left(A_{i,j}\right)$ udáva mohutnosť fuzzy množiny $A_{i,j}$ a vypočíta sa z nasledujúceho vzťahu

$$M\left(A_{i,j}\right) = \sum_{k=1}^{N} \mu_{Aj}(x_k),$$

kde $\mu_{Aj}(x_k)$ sú hodnoty príslušnosti. Nech A_{i2} a A_{i1} sú fuzzy atribúty a $A_{i_2 j_2}$ a $A_{i_1 j_1}$ sú

hodnoty, ktoré nadobúdajú. *Spoločná informácia* hodnôt $A_{i_2j_2}$ a $A_{i_1j_1}$ je vyjadrená podľa nasledujúceho vzorca:

$$\mathbf{I}(A_{i_1j_1}, A_{i_2j_2}) = \log_2 N - \log_2 M(A_{i_1j_1} \times A_{i_2j_2}).$$

Táto funkcia je symetrická a nezávislá od poradia previerky týchto atribútov

$$\mathbf{I}(A_{i_1j_1}, A_{i_2j_2}) = I(A_{i_2j_2}, A_{i_1j_1}).$$

Nech A_{i2} a A_{i1} sú fuzzy atribúty a $A_{i_2j_2}$ a $A_{i_1j_1}$ sú hodnoty, ktoré nadobúdajú. *Podmienená informácia* medzi hodnotami $A_{i_2j_2}$ a $A_{i_1j_1}$ je zadefinovaná ako

$$\mathbf{I}(A_{i_1j_1}|A_{i_2j_2}) = \mathbf{I}(A_{i_1j_1}, A_{i_2j_2}) - \mathbf{I}(A_{i_1j_1}) = \log_2 M(A_{i_1j_1}) - \log_2 M(A_{i_2j_2} \times A_{i_1j_1})$$

bitov.

Vzájomnú informáciu medzi hodnotami $A_{i_2j_2}$ a $A_{i_1j_1}$ udáva vzťah

$$I(A_{i_2j_2}; A_{i_1j_1}) = I(A_{i_2j_2}) - I(A_{i_1j_1}|A_{i_2j_2}).$$

2.6 Fuzzy rozhodovacie stromy

Algoritmy fuzzy rozhodovacích stromov rozširujú klasické algoritmy rozhodovacích stromov. Na reprezentáciu dátových množín využívajú fuzzy logiku. Na rozdiel od klasických rozhodovacích stromov pripúšťajú neurčitosť hodnôt vstupných aj výstupných atribútov. Neurčitosť spôsobujú odchýlky merania, nepresnosti spôsobené ľudským faktorom a mnohé iné nepresnosti. Odstrániť tieto nedostatky je možné pomocou fuzzy dát [3]. Existuje niekoľko algoritmov, ktoré spájajú fuzzy logiku a rozhodovacie stromy. Napríklad UR-ID3 [9] vybuduje striktne rozhodovací strom. Následne zavedie fuzzy na pravidlá v strome. Roger Jang predstavil v [13] metódu *Fuzzy-CART*. Existujú stratégie, ktoré využívajú pri konštrukcií fuzzy rozhodovacieho stromu fuzzy rozhodovaciu tabuľku [3].

2.6.1 Fuzzy ID3

Vzorka dát s fuzzy reprezentáciou je taktiež popísaná pomocou atribútov. Všeobecne existujú dva rozdielne typy atribútov a to diskrétne a spojité. Veľa algoritmov vyžaduje dáta s diskrétnymi hodnotami. Nie je jednoduché nahradiť spojitú dátovú doménu diskrétnou [14]. Pri takom procese je potrebné spraviť nejaký rozklad a zhlukovanie. Taktiež je náročné určiť hranice spojitých atribútov. Napríklad – problém je definovať, či je dopravná zápcha dlhá alebo krátka. Je otázne, či môžeme

tvrdiť, že zápcha, čo má 2.9 km je krátka a zápcha dlhá 3 km je dlhá. Preto v algoritme ID3 boli nahradené vzorky dát fuzzy výrazmi a sformovali sa fuzzy ID3 metódy. Vzorce na výpočet entropie atribútov a informačného zisku sú iné, pretože dáta sú vyjadrené pomocou fuzzy logiky. Sú definované nasledovne. Nech $S = \{x_1, x_1, \dots, x_j\}$ je dátová množina. Potom

$$H_f(S, A) = -\sum_{i=1}^{C} \frac{\sum_j^N \mu_{ij}}{S} \log_2 \frac{\sum_j^N \mu_{ij}}{S}$$

$$G_f(S, A) = H_f(S) - \sum_{v \subseteq A}^{N} \frac{|S_v|}{|S|} * H_{fs}(S_v, A)$$

Kde μ_{ij} sú funkcie príslušnosti *j*-tého prípadu patriaceho do *i*-tej triedy. $H_f(S)$ predstavuje entropiu dátovej množiny S vo vrchole f. S_v je veľkosť podmnožiny $S_v \subseteq S$ trénovacích prípadov x_j s v atribútom a $|S|$ je veľkosť množiny S. [14]

Definovanie prahových hodnôt pre ID3 [14]

- Prahová hodnota θ_r [14]
    - Ak pomer veľkosti dátovej množiny C_k je väčší alebo rovný θ_r, koniec expanzie (rastu stromu) stromu. Napríklad, ak pre triedu 1 je pomer 90% a pre triedu 2 je pomer 10%, a $\theta_r = 85\%$, tak expanzia sa zastaví.
- Prahová hodnota θ_n [14]
    - Ak veľkosť dátovej množiny vo vrchole je menšia ako θ_n, expanzia sa zastaví. Napríklad ak dátová množina má 600 príkladov, kde $\theta_n = 2\%$. Ak počet príkladov v niektorom uzle je menší ako 12 $(2\% \; zo \; 600)$, tak expanzia sa v tomto uzle zastaví.

Hladina týchto prahových hodnôt má veľký vplyv na výsledok stromu. Definujú sa v rôznych úrovniach počas experimentu, aby sa našli optimálne hodnoty. Navyše, ak nie sú k dispozícii žiadne ďalšie atribúty pre klasifikáciu, algoritmus nevytvára nový uzol.

Algoritmus tvorby fuzzy ID3 [14]

1. Vytvor koreňový vrchol
2. Ak vrchol t s fuzzy dátovou množinou D splní nasledujúcu podmienku, potom ho prehlás za list a priraď mu meno príslušnej triedy
   - Pomer triedy C_k je väčší alebo rovný prahovej hodnote θ_r, $\frac{|D^{C_i}|}{|D|} \geq 0$
   - Veľkosť množiny údajov je menšia ako prahová hodnota θ_n
   - Nie sú k dispozícii žiadne atribúty pre ďalšie klasifikovanie
3. Ak vrchol t s D nesplní podmienky v kroku 2, potom nebude list a vygeneruje sa mu potomok nasledovne:
   - Pre každý atribút $A_i, i = 1, \dots, L$ vypočítaj informačný zisk G a vyber atribút A_{max}, ktorý má hodnotu G maximálnu
   - Rozdeľ D na fuzzy podmnožiny $D_1, \dots, D_m$ podľa A_{max}, kde stupeň príslušnosti dát v D_j je súčin stupňa príslušnosti v D a hodnoty $F_{max,j}$ z A_{max} v D.
   - Vytvor nové vrcholy $t_1, \dots, t_m$ pre fuzzy podmnožiny $D_1, \dots, D_m$ pričom vrchol t_j a vrchol t spoj vetvou, ktorá je označená $F_{max,j}$
   - Rekurzívne pokračuj od kroku 2 pre všetky $D_1, \dots, D_m$

2.6.2 Neusporiadaný fuzzy rozhodovací strom

V anglickej literatúre sa označuje ako *unordered fuzzy decision tree*. Tento algoritmu počas procesu budovania rozhodovacieho stromu využíva fuzzy rozhodovaciu tabuľku. Tabuľka je vopred zadaná. V tabuľke sú vyjadrené sumárne informačné charakteristiky dát. Informačné charakteristiky sú vysvetlené v kapitole Informačné charakteristiky.

Rovnako ako pri klasických rozhodovacích stromoch, aj pri fuzzy rozhodovacích stromoch je potrebné stanoviť metódu pre správny výber atribútu asociovaného s jednotlivými vrcholmi a taktiež zastavenie rastu alebo orezanie stromu. Ako prvé popíšem kritéria výberu atribútu. V algoritme vyberáme atribút, ktorý nám poskytne najväčšie množstvo informácie v kombinácií s najnižšími nákladmi na zmeranie tohto atribútu $Cost\left(A_{i_q}\right)$. Množstvo takejto informácie udáva *vzájomná sumárna informácia*

medzi výstupným atribútom B a výstupným fuzzy atribútom A_{i_q}. Náklady $Cost\left(A_{i_q}\right)$ sú vyjadrené reálnym číslom a vyjadrujú časový i materiálny odhad na zmeranie vstupného atribútu. Tieto náklady bývajú známe už pred začatím budovania fuzzy rozhodovacieho stromu. Postupnosť hodnôt $U_{i_{q-1}j_{q-1}} = (A_{i_1 j_1}, \dots, A_{i_{q-1}j_{q-1}})$ predchádzajúcich vstupných fuzzy atribútov $U_{i_{q-1}} = (A_{i_1}, \dots, A_{i_{q-1}})$ je pre každý vrchol známa vopred, nakoľko postupnosť vzniká v procese budovania stromu. Pokiaľ zohľadníme aj minimálnu hodnotu nákladov $Cost\left(A_{i_q}\right)$, potom atribút vyberáme podľa nasledujúceho kritéria:

$$q = argmax\left(\frac{\mathbf{I}\left(B; A_{i,j}, \dots, A_{i_{q-1}j_{q-1}}, A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right)$$

Druhá, už spomínaná úloha pri tvorbe takéhoto stromu je ohraničenie rastu. Ohraničenie je nevyhnutné, aby došlo k odstráneniu málo perspektívnych resp. málo produktívnych hrán. Tento algoritmus využíva definovanie prahových hodnôt α a β. Vnútorný vrchol stromu prehlásime za list, ak sa splní aspoň jedna z nasledujúcich podmienok:

- $f\left(U_{i_q j_q}\right) = \frac{M\left(A_{i_1 j_1} \times \dots \times A_{i_2 j_2}\right)}{N} \leq \alpha$
- $2^{-\mathbf{I}\left(B|A_{i_1 j_1}, \dots, A_{i_q j_q}\right)} \geq \beta$

Zmena prahových hodnôt α a β ovplyvní aj veľkosť výsledného rozhodovacieho stromu. Ak zámerne zväčšíme hodnotu β a zmenšíme hodnotu α je očakávané, že vznikne strom s väčšou hĺbkou. Ak hodnotu β zmenšíme a hodnotu α zväčšíme, tak je očakávaný výsledný strom s menšou hĺbkou.

Algoritmus tvorby neusporiadaného fuzzy rozhodovacieho stromu [3]

Ako vstupné dáta sú do algoritmu poslané parametre ohraničenia α a β a tabuľka s fuzzy dátami $\{CS, CV, AS, AV, C, P\}$ popísaná nasledovne:

- $CS = \{A_1, \dots, A_n\}$ - množina vstupných atribútov
- $CV = \{A_{i,j}\}$ pre $i = 1, \dots, n; j = 1, \dots, m_i$ - množina hodnôt atribútov
- $AS = \{B\}$ - výstupný fuzzy atribút

- $AV = \{B_1, \dots, A_{m_b}\}$ - množina hodnôt výstupného fuzzy atribútu
- $C = \{Cost(A_i)\}$ - náklady na stanovenie hodnôt výstupného fuzzy atribútu
- P – tabuľka, opisujúca množinu N situácií

Výstupom algoritmu je neusporiadaný fuzzy rozhodovací strom. Algoritmus prebieha v dvoch fázach. V prvej fáze prebehnú nasledujúce tri kroky:

1. $Free\_attr = CS$; V tomto kroku priradí do množiny $Free\_attr$ všetky atribúty, ktoré môžeme asociovať s vrcholom, ináč povedané do množiny $Free\_attr$ pridá atribúty, ktoré ešte neboli asociované s iným vrcholom.
2. $q = 1$; Vytvorí prvú úroveň fuzzy rozhodovacieho stromu.
3. $U_{i_{q-1}j_{q-1}} = \emptyset$

Po ukončení prvej fázy nasleduje druhá fáza. V druhej fáze algoritmus spustí rekurzívnu funkciu $\boldsymbol{buildFDT}\left(q, U_{i_{q-1}j_{q-1}}, Free\_attr\right)$. Táto funkcia prebieha v nasledujúcich krokoch.

1. Pre všetky fuzzy atribúty v $Free\_attr$ vypočíta sumárnu vzájomnú informáciu $\mathbf{I}\left(B; U_{i_{q-1}j_{q-1}}, A_{i_q}\right)$ a následne vyberie atribút s maximálnou hodnotou výverového kritéria.
2. Asociovať vybraný atribút s vrcholom fuzzy neusporiadaného stromu a hodnoty tohto atribútu s hranami tohto vrcholu.
3. Vybraný fuzzy atribút sa musí odstrániť z $Free\_attr$.
4. V poslednom kroku je potrebné preveriť, či na konci hrán nevznikol list. Táto kontrola sa robí vzhľadom na zadané α a β hraničné hodnoty. Pokiaľ výsledok testu, či vznikol list, je záporný, tak sa zavolá rekurzívne funkcia $\boldsymbol{buildFDT}\left(q++, U_{i_{q-1}j_{q-1}}, Free\_attr\right)$. Pokiaľ by test na list dopadol pozitívne, v danom vrchole expanzia stromu končí a je prehlásený za list.

Pre znázornenie konštrukcie fuzzy neusporiadaného stromu využijem fuzzy Tabuľka 3. Ako prahové hodnoty sú zadané $\alpha = 0.16$ a $\beta = 0.75$. Príklad bude spracovaný podľa zdroja [15] a [3].

Algoritmus začína prvou fázou. V nej buduje prvú úroveň ($q = 1$). Vyberá atribút, ktorý bude asociovaný s vrcholom fuzzy rozhodovacieho stromu. Celú množinu vstupných atribútov CS priradí do zoznamu $Free\_attr$. Z pomedzi atribútov v zozname $Free\_attr$ vyberieme atribút s maximálnou hodnotou výberového kritéria.

$$i_q = argmax\left(\frac{\mathbf{I}\left(B; A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right) pre\ \forall\ A_i\ \epsilon\ Free\_attr$$

$$i_q = \left\{\frac{3.052}{2.5}; \frac{3.752}{1.7}; \frac{0.257}{2.0}; \frac{1,402}{1.8}\right\} = 2$$

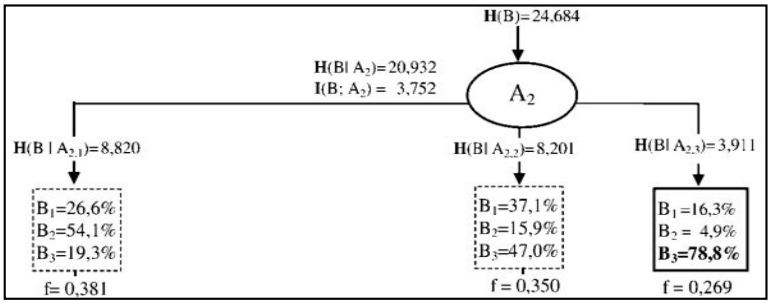
Je zrejmé, že najväčšiu hodnotu výberového kritéria nadobúda atribút A_2. Tento atribút preto budeme asociovať s vrcholom rozhodovacieho stromu. Hodnoty $A_{2,1}, A_{2,2}, A_{2,3}$, ktoré atribút A_2 nadobúda algoritmus asociuje s hranami vychádzajúcimi práve z tohto vrcholu. Následne odstráni z $Free\_attr$ atribút A_2. Po odstránení atribútu A_2 z $Free\_attr$ algoritmus preverí existenciu listov na konci hrán. Využíva k tomu kritéria pre prahové hodnoty definované vyššie. $f\left(A_{2,1}\right) = \frac{6.1}{16} = 0.381$; $f\left(A_{2,2}\right) = \frac{5.6}{16} = 0.350$; $f\left(A_{2,3}\right) = \frac{4.3}{16} = 0.296$. Všetky tieto hodnoty sú väčšie ako prahová hodnota $\alpha = 0.16$, to znamená, že toto kritérium nespôsobí vznik listu na prvej úrovni. Potrebné je overiť aj druhé prahové kritérium.

Hrana $min_{\text{jb}=1,\ldots,3}\ \mathbf{I}\left(B_{jb}\,|A_{2,1}\right) = \mathbf{I}\left(B_1|A_{2,1}\right) = 0.886\,;\quad 2^{-0.886} = 0.541$

hrana $min_{\text{jb}=1,\ldots,3}\ \mathbf{I}\left(B_{jb}\,|A_{2,2}\right) = \mathbf{I}\left(B_2|A_{2,2}\right) = 1.090;\quad 2^{-1.090} = 0.470$

hrana $min_{\text{jb}=1,\ldots,3}\ \mathbf{I}\left(B_{jb}\,|A_{2,3}\right) = \mathbf{I}\left(B_3|A_{2,3}\right) = 0.343;\quad 2^{-0.343} = 0.778$

Prahová hodnota $\beta = 0.75$ resp. 75% je menšia ako hodnota tohto kritéria pre hranu asociovanú s hodnotou $A_{2,3}$. Táto prahová hodnota predstavuje maximálnu úroveň vierohodnosti. Preto na konci tejto hrany vznikne list.



Obrázok 8. Prvá úroveň fuzzy neusporiadaný rozhodovacieho stromu

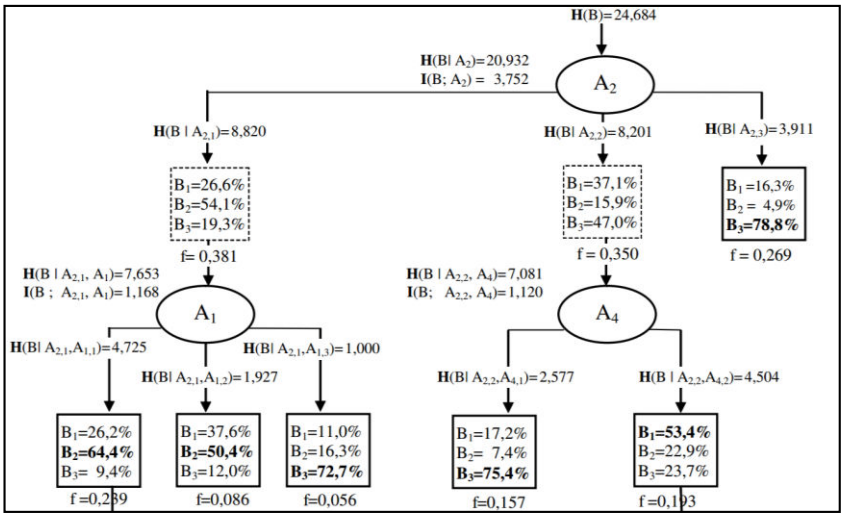
Pri rozhodnutí v tomto liste treba zvoliť rozhodnutie B_3, pretože má najvyššiu hodnotu vierohodnosti. Pri koreňoch asociovaných s prvou $A_{2,1}$ a druhou hranou $A_{2,2}$ je taktiež odporúčané vyberať rozhodnutia s najvyššou vierohodnosťou. Vzťah pre overenie prahového kritéria α udáva aj frekvencie výskytov situácií $A_{2,1}, A_{2,2}, A_{2,3}$. Tieto frekvencie sú vypočítané už vyššie, preto uvediem len ich hodnoty $f(A_{2,1}) = 0.381$; $f(A_{2,2}) = 0.350$; $f(A_{2,3}) = 0.296$. Algoritmus následne vytvorí druhú úroveň ($q = 2$) stromu. Na konci hrán $A_{2,1}$ a $A_{2,2}$ vznikli vrcholy, ktoré momentálne nemajú asociovaný žiadny atribút. Preto zo zoznamu $Free\_attr$ voľných vstupných atribútov fuzzy rozhodovacieho stromu vyberieme pre tieto hrany vhodné atribúty. Spôsob výberu atribútov popíšem pre hranu $A_{2,1}$. Postup je podobný ako pri prvej úrovni rozhodovacieho stromu. Pri výpočte *sumárnej vzájomnej informácie* pre atribúty v zozname $Free\_attr = \{A_1, A_3, A_4\}$ je potrebné zohľadniť postupnosť už vykonaných previerok. Preto bude výberové kritérium pre danú vetvu $A_{2,1}$ nasledovné :

$$i_q = argmax\left(\frac{\mathbf{I}\left(B; A_{2,1}, A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right) pre\ \forall\ A_i \in Free\_attr$$

$$i_q = \left\{\frac{1.168}{2.5}; \frac{0.219}{2.0}; \frac{0.259}{1.8}\right\} = 1$$

Maximálnu hodnotu výberového kritéria dosiahol atribút A_1. Tento atribút sa preto musí odstrániť zo zoznamu nepoužitých vstupných atribútov $Free\_attr$. Atribút A_1 asociujeme s koncovým vrcholom hrany $A_{2,1}$. Hodnoty tohto atribútu sa asociujú s hranami, ktoré budú z vrcholu asociovaného s atribútom $A_{2,1}$ vychádzať. Tak isto je potrebné preveriť existenciu listov na konci týchto hrán. Jednotlivé frekvencie podľa kritéria pre výpočet prahových hodnôt α sú $f(A_{2,1}, A_{1,1}) = 0.239$; $f(A_{2,1}, A_{1,2}) = 0.086$; $f(A_{2,1}, A_{1,3}) = 0.056$. Hrany $A_{1,2}$ a $A_{1,3}$ budú ukončené listom, pretože hodnoty ich frekvencií sú menšie ako hodnota prahového kritéria $\alpha = 0.16$. Po vypočítaní vierohodností, zistíme že viac listov nepribudne. Pre hranu $A_{2,2}$ by sme algoritmus analogickým postupom asocioval atribút A_4, ktorý nadobúda dve hodnoty, tak z vrcholu by vychádzali dve hrany. Ďalej by overil existenciu listov. Frekvencie nadobudnú hodnoty $f(A_{2,2}, A_{4,1}) = 0.157$; $f(A_{2,2}, A_{4,2}) = 0.193$. To znamená existenciu listu na konci hrany $A_{4,1}$, pretože hodnota frekvencie je menšia ako prahová hodnota α. Po vyjadrení vierohodnosti nevznikne nový list, nakoľko táto podmienka by

platila práve pre vrchol, ktorý je na konci hrany $A_{4,1}$ a vrchol na konci hrany $A_{4,1}$ už listom je. Druhá úroveň stromu je znázornená na obrázku Obrázok 9.

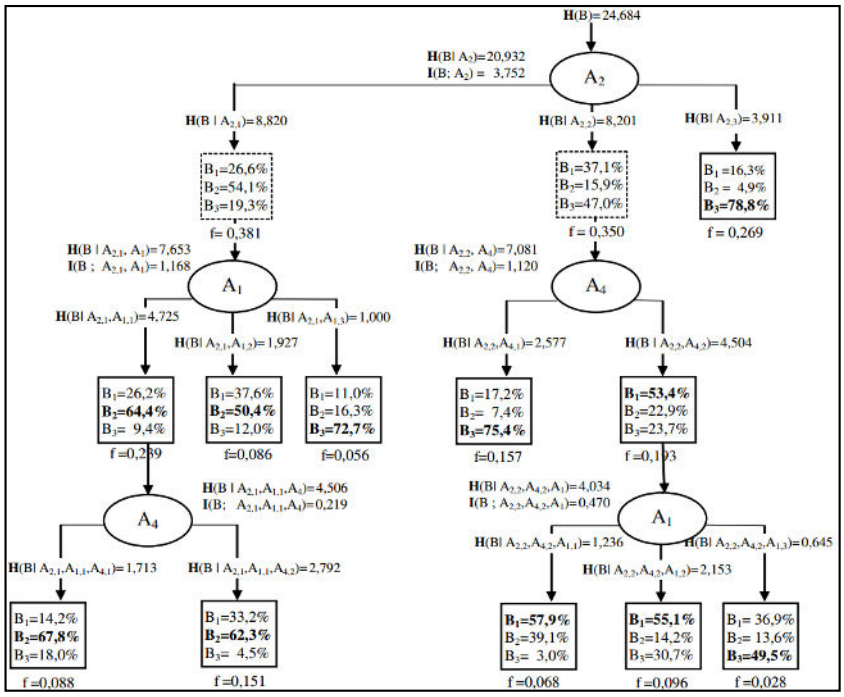


Obrázok 9. Druhá úroveň fuzzy neusporiadaný rozhodovacieho stromu

Aj pre tretiu úroveň sa zopakuje postupnosť krokov, ako v predchádzajúcich dvoch úrovniach. Pre hranu $(A_{2,1}, A_{1,1})$ algoritmus vyberie atribút A_{i_q} zo zoznamu voľných vstupných atribútov $Free\_attr = \{A_3, A_4\}$, ktorý asociuje s vrcholom fuzzy rozhodovacieho neusporiadaného stromu. Výberové kritérium pre túto hranu bude

$$i_q = argmax\left(\frac{\mathbf{I}\left(B; A_{2,1}, A_{1,1}, A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right) pre\ \forall\ A_i \in Free\_attr$$

Najvhodnejší atribút A_{i_q} podľa tohto výberového kritéria je vstupný atribút A_4 pre ktorý výberové kritérium nadobúda hodnotu 0.622. Táto hodnota je z pomedzi dvoch atribútov v zozname $Free\_attr$ maximálna. Frekvencie vzniku situácii $(A_{2,1}, A_{1,1}, A_{4,1})$ a $(A_{2,1}, A_{1,1}, A_{4,2})$ hrán vychádzajúcich z tohto vrcholu sú $f(A_{2,1}, A_{1,1}, A_{4,1}) = 0.088$ a $f(A_{2,1}, A_{1,1}, A_{4,2}) = 0.151$. To znamená, že hrany $A_{4,1}$ a $A_{4,2}$ sa budú končiť listami, pretože prahová hodnota $\alpha = 0.16$. Rovnaký postup algoritmus vykoná aj pre hranu $U_{i_{q-1}, j_{q-1}} = (A_{2,2})$. V tomto prípade vyberá zo zoznamu voľných vstupných atribútov $Free\_attr$ atribúty A_a a A_1. Výsledný neusporiadaný fuzzy rozhodovací strom z príkladu je na obrázku Obrázok 10.



Obrázok 10. Neusporiadaný fuzzy rozhodovacieho stromu

2.6.3 Usporiadaný fuzzy rozhodovací strom

Tento algoritmus sa líši od predchádzajúceho v tom, že vyberá vstupný fuzzy atribút na previerku bez ohľadu na výsledky previerok, ktoré sa už počas behu algoritmu uskutočnili. Algoritmus umožňuje paralelizované vykonávanie previerok vstupných fuzzy atribútov. Aj v tomto algoritme sa do vrcholu asociuje atribút A_{i_q}, ktorý nám poskytne najväčšie množstvo novej informácie o výstupnom atribúte, lenže výber atribútu A_{i_q} nemôže byť závislý od previerok predošlých atribútov $A_{i_1}, \dots, A_{i_{q-1}}$. Algoritmus pri výbere atribútu, ktorý sa podrobí previerke využíva ako výberové kritérium sumárnu vzájomnú informáciu:

$$\Delta\mathbf{I}\left(B; A_{i_1}, \dots, A_{i_{q-1}}, A_{i_q}\right) = \mathbf{I}\left(B; A_{i_1}, \dots, A_{i_{q-1}}, A_{i_q}\right) - \mathbf{I}\left(B; A_{i_1}, \dots, A_{i_{q-1}}\right)$$

Táto informačná charakteristika je vhodná práve preto, že na q-tej úrovni rozhodovacieho stromu sa všetky vrcholy asociujú práve s jedným atribútom. Pokiaľ chceme zohľadniť aj minimálne náklady na spracovanie jednotlivých atribútov, tak výberové kritérium sa modifikuje nasledovne:

$$q = argmax\left(\frac{\Delta \mathbf{I}\left(B; A_{i_1}, \ldots, A_{i_{q-1}}, A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right)$$

Ako už bolo spomenuté, zásadný rozdiel oproti algoritmu na konštrukciu neusporiadaného fuzzy rozhodovacieho stromu je ten, že toto kritérium nezohľadní predchádzajúce hodnoty $A_{i_1,j_1}, \ldots, A_{i_{q-1},j_{q-1}}$ spracovaných vstupných atribútov $A_{i_1}, \ldots, A_{i_{q-1}}$. Kritérium využíva ich priemerné hodnoty. Vybudovanie fuzzy usporiadaného rozhodovacieho stromu má menšiu výpočtovú zložitosť, ako budovanie neusporiadaného fuzzy rozhodovacieho stromu, nakoľko nie je potrebné vyjadriť sumárne informačné charakteristiky pre každú hranu stromu samostatne.

Algoritmus tvorby neusporiadaného fuzzy rozhodovacieho stromu [3]

Vstupné dáta sú rovnaké ako v predchádzajúcom algoritme. Do algoritmu je potrebné poslať parametre ohraničenia α a β a tabuľku s fuzzy dátami $\{CS, CV, AS, AV, C, P\}$ popísaná nasledovne:

- $CS = \{A_1, \ldots, A_n\}$ - množina vstupných atribútov
- $CV = \{A_{i,j}\}$ pre $i = 1, \ldots, n; j = 1, \ldots, m_i$ - množina hodnôt atribútov
- $AS = \{B\}$ - výstupný fuzzy atribút
- $AV = \{B_1, \ldots, A_{m_b}\}$ - množina hodnôt výstupného fuzzy atribútu
- $C = \{Cost(A_i)\}$ - náklady na stanovenie hodnôt výstupného fuzzy atribútu
- P – tabuľka, opisujúca množinu N situácií

Výstupom algoritmu je usporiadaný fuzzy rozhodovací strom. Aj tento algoritmus je popísaný v dvoch fázach. Kroky prvej fázy sú vysvetlené nižšie:

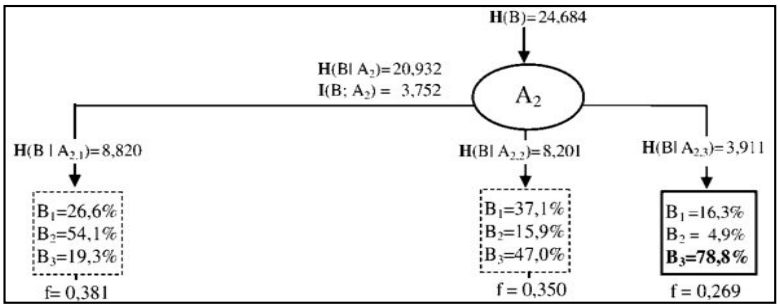
1. $Free\_attr = CS$; V tomto kroku priradí do množiny $Free\_attr$ všetky atribúty, ktoré môžeme asociovať s vrcholom, ináč povedané do množiny $Free\_attr$ pridá atribúty, ktoré ešte neboli asociované so žiadnym iným vrcholom.
2. $q = 1$; Vytvorí prvú úroveň fuzzy rozhodovacieho stromu.
3. $U_{i_{q-1}} = \emptyset$; zoznam už asociovaných atribútov.

V druhej fáze vykoná tieto kroky:

1. V prvom kroku algoritmus vyčísli sumárnu vzájomnú informáciu všetkých atribútov v zozname $Free\_attr$ a vyberá z nich atribút s najvyššou hodnotou daného výberového kritéria.
2. Vybraný atribút A_{i_q} je asociovaný s každým vrcholom na q-tej úrovni a hodnoty atribútu A_{i_q} asociujeme s hranami, ktoré vychádzajú z vrcholov na q-tej úrovni. Algoritmus následne odstráni atribút A_{i_q} zo zoznamu neasociovaných atribútov $Free\_attr$ a atribút A_{i_q} pridá do zoznamu U_{i_q} asociovaných atribútov.
3. V tomto kroku algoritmus preverí každú vychádzajúcu hranu z vrcholu A_{i_q}, či nie je ukončená listom.
4. Ak všetky hrany na q-tej úrovni končia listom, alebo je zoznam neasociovaných atribútov $Free\_attr$ prázdny, tak algoritmus skončí. Ináč zväčší úroveň $q = q + 1$ a rekurzívne pokračuje od kroku 1 druhej fázy.

Aj pre znázornenie konštrukcie fuzzy usporiadaného stromu využijem fuzzy Tabuľka 3. Ako prahové hodnoty sú aj v tomto príklade zadané $\alpha = 0.16$ a $\beta = 0.75$. Príklad bude spracovaný podľa [16] a [3].

Algoritmus začína prvou fázou. V nej buduje prvú úroveň ($q = 1$). Algoritmus vyberá atribút, ktorý bude asociovaný s vrcholom fuzzy rozhodovacieho stromu. Aj v tomto algoritme na začiatku celú množinu vstupných atribútov CS priradí do zoznamu $Free\_attr$. Z pomedzi atribútov v zozname $Free\_attr$ vyberieme atribút s maximálnou hodnotou výberového kritéria. Tento atribút sa z množiny $Free\_attr$ odstráni a algoritmus ho musí pridať do množiny už asociovaných atribútov U_{i_q}. Na prvej úrovni usporiadaného fuzzy stromu a neusporiadaného fuzzy rozhodovacieho stromu sú výberové kritéria zhodné. Celý postup pre prvú úroveň stromu ($q = 1$) je zhodný s postupom v predchádzajúcom príklade. Preto aj prvá úroveň stromu bude rovnaká, nakoľko obidva príklady využívajú rovnakú fuzzy Tabuľka 3.



Obrázok 11. Prvá úroveň fuzzy usporiadaný rozhodovacieho stromu

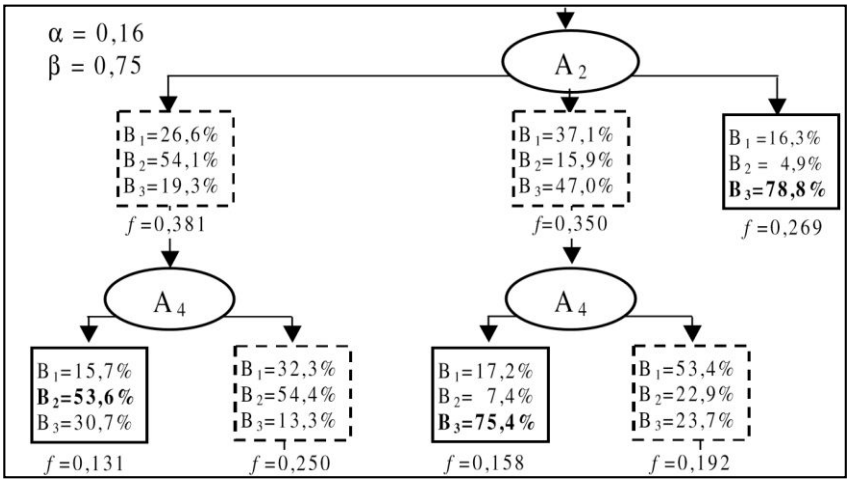
Na druhej úrovni $q = 2$ algoritmus vyberie z množiny $Free\_attr = \{A_1, A_3, A_4\}$ podľa výberového kritéria najvhodnejší atribút pre asociáciu s vrcholmi, ktoré sú na konci hrán $A_{2,1}$ a $A_{2,2}$ usporiadaného fuzzy rozhodovacieho stromu. Tieto hrany $A_{2,1}$ a $A_{2,2}$ zodpovedajú hodnotám atribútu A_2. Algoritmus vypočíta sumárnu vzájomnú informáciu pre všetky atribúty v množine $Free\_attr$. Následne vyberie atribút A_{i_q} s maximálnou hodnotou výberového kritéria

$$i_q = argmax\left(\frac{\Delta \mathbf{I}\left(B; A_2, A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right), kde\ \ A_{i_q} \epsilon\ Free\_attr$$

$$i_q = argmax\left\{\frac{2.556}{2.5}; \frac{0.455}{2.0}; \frac{1.864}{1.8}\right\} = 4$$

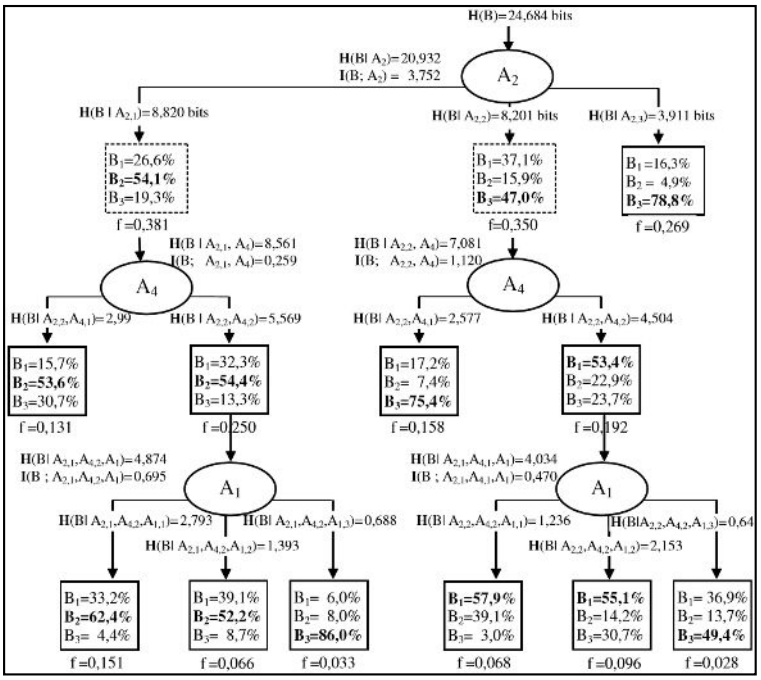
Maximálnu hodnotu výberového kritéria nadobúda atribút A_4. Tento atribút preto bude algoritmus asociovať s vrcholmi usporiadaného fuzzy rozhodovacieho stromu na druhej úrovni $(q = 2)$. Algoritmus taktiež asociuje hodnoty atribútu A_4 s hranami vychádzajúcimi z týchto vrcholov. Vstupný atribút fuzzy rozhodovacieho stromu A_4 algoritmus odstráni z množiny $Free\_attr$ a pridá ho do množiny už asociovaných atribútov U_{i_q}. Množina U_{i_q} bude teda obsahovať atribúty A_2 a A_4 a množina voľných, ešte neasociovaných $Free\_attr$ atribútov obsahuje A_1 a A_3. Algoritmus následne preverí existenciu listov na danej úrovni stromu. Preverí existenciu listov na konci každej hrany vychádzajúcej z vrcholov asociovaných s atribútom A_4. Po overení existencie listov algoritmus zistí, že na druhej úrovni vzniknú dva listy, pretože frekvencie hodnôt týchto hrán sú $f(A_{2,1}, A_{4,1}) = 0.131$ a $f(A_{2,2}, A_{4,1}) = 0.158$. Tieto hodnoty sú menšie ako prahová hodnota $\alpha = 0.16$. Podľa druhého prahového kritéria

algoritmus na tejto úrovni nevytvorí žiadny ďalší list. Druhá úroveň usporiadaného fuzzy rozhodovacieho stromu je zobrazená na nasledujúcom obrázku.



Obrázok 12. Druhá úroveň fuzzy usporiadaný rozhodovacieho stromu

Na tretej úrovni usporiadaného fuzzy rozhodovacieho stromu musí algoritmus spracovať dve hrany $(A_{2,1}, A_{4,2})$ a $(A_{2,2}, A_{4,2})$. Množina $Free\_attr$ obsahuje atribúty A_1 a A_3. Algoritmus vyberie z tejto množiny, ktorý ma maximálnu hodnotu výberového kritéria $argmax\left(\frac{\Delta \mathbf{I}\left(B;A_2,A_4,A_{i_q}\right)}{Cost\left(A_{i_q}\right)}\right), kde\ A_{i_q} \epsilon\, Free\_attr$. Podľa tohto výberového kritéria algoritmus zvolí pre tretiu úroveň usporiadaného fuzzy rozhodovacieho stromu atribút, ktorý sa bude asociovať s vrcholmi na tejto úrovni atribút A_1. Aj v tomto prípade bude z každého vrcholu asociovaného s týmto atribútom vychádzať toľko hrán, koľko atribút nadobúda hodnôt. Atribút A_1 nadobúda tri hodnoty, takže algoritmus vytvorí pre každý vrchol tri hrany. Následne musí preveriť, či na niektorej z týchto hrán nevznikol list. V tomto prípade budú všetky tieto hrany ukončené listom, pretože frekvencia hodnôt týchto hrán je menšia ako prahová hodnota $\alpha = 0.16$. V tomto momente sú už všetky vonkajšie vrcholy usporiadaného fuzzy rozhodovacieho stromu ukončené listom a rast tohto stromu sa zastavil. Výsledný usporiadaný fuzzy rozhodovací strom je znázornený na Obrázok 13.



Obrázok 13. Fuzzy usporiadaný rozhodovací strom

2.6.4 Produkčné pravidlá a ich konštrukcia

Rozhodovacie stromy sú jednou z foriem vyjadrenia znalostí. Táto forma je vhodná predovšetkým pre vizualizáciu dát. Vhodnejší spôsob ako reprezentovať znalosti v pamäti počítača predstavujú produkčné pravidlá. Podstata produkčných pravidiel je určiť hodnotu výstupného atribútu po zmeraní jednej alebo viac hodnôt vstupných atribútov novej inštancie. Konštrukcia produkčných pravidiel vychádza z fuzzy rozhodovacieho stromu, ktorý bol vybudovaný podľa fuzzy tabuľky popísanej v kapitole Fuzzy dáta. Pripomeniem, že tabuľka sa skladá z n vstupných atribútov A_i, kde $i = 1, \dots n$ a každý atribút A_i nadobúda m_i hodnôt $A_{i,1}, A_{i,2}, \dots, A_{i,m_i}$. Pokiaľ atribút A_i nadobúda m_i hodnôt, tak z vnútorného vrcholu fuzzy rozhodovacieho stromu asociovaného s týmto atribútom bude vychádzať m_i hrán. Každá hodnota atribútu A_i bude asociovaná práve s jednou touto hranou. Každý vonkajší vrchol (list) udáva úroveň vierohodnosti pre hodnoty $B_1, \dots, B_{j,b}, \dots, B_{m_b}$ výstupného atribútu B.

Nech fuzzy rozhodovací strom obsahuje R listov $L = \{l_1, \dots, l_r, \dots l_R\}$ a vrchol K je koreň stromu. Cesta od vrcholu K do každého listu v množine listov L určuje

postupnosť vrcholov a hrán. Vo vrcholoch sú asociované atribúty a v hranách, ktoré z nich vychádzajú sú asociované hodnoty týchto atribútov. Konštrukcia produkčného pravidla spočíva v zapisovaní tejto cesty. Cesta sa zapisuje pomocou výrazov v tvare „vstupný atribút – jeho hodnota“ pre každý vrchol na ceste od koreňa k listu. Pokiaľ je na ceste od koreňa k listu l_r viac ako jeden vrchol, potom sa výrazy spájajú logickou operáciou AND. Pre skonštruovanie všetkých pravidiel je potrebné prejsť všetky cesty od koreňa ku každému listu rozhodovacieho stromu. Počet týchto ciest sa bude rovnať počtu listov v strome. Pravidlo je ukončené vektorom $\mathbf{B}^r = \left[B_1^r \dots B_{m_b}^r\right]$. Tento vektor zodpovedá hodnotám vierohodnosti v liste l_r.

Formálne sa dá zapísať produkčné pravidlo nasledovne [3]. Nech cesta od vrcholu rozhodovacieho stromu obsahuje S vrcholov a S hrán. Počet hrán sa musí rovnať počtu vrcholov. Vrcholy sú asociované s atribútmi $A_{i_1}, \dots, A_{i_s}, \dots, A_{i_S}$. Hrany sú asociované s hodnotami týchto atribútov $A_{i_1 j_1}, \dots, A_{i_s j_s}, \dots, A_{i_S j_S}$. Potom produkčné pravidlo pre túto cestu má nasledujúci tvar.

$$if\left(A_{i_1}\ is\ A_{i_1 j_1}\right) \,\&\, \dots \,\&\, if\left(A_{i_s}\ is\ A_{i_s j_s}\right) \,\&\, \dots \,\&\, if\left(A_{i_S}\ is\ A_{i_S j_S}\right)\ then\ B\ (vierohodnosť\ \mathbf{B}^r)$$

2.6.5 Klasifikácia pomocou produkčných pravidiel

Produkčné pravidlá umožňujú výpočet hodnôt výstupného atribútu pre novú inštanciu, pokiaľ poznáme hodnoty jej vstupných atribútov. Nech z fuzzy rozhodovacieho stromu vznikne r produkčných pravidiel p_1 až p_R. Pre každé pravidlo p_r , $r = 1, \dots, R$ je možné vyjdriť váhu w_r podľa vzťahu:

$$w_r = \prod_{s=1}^{S} \left[\mu_{A_{i_s j_s}}(x)\right]^r$$

Kde S je počet hrán fuzzy rozhodovacieho stromu, $c = A_{i_1 j_1}, \dots, A_{i_s j_s}, \dots, A_{i_S j_S}$ je postupnosť vstupných hodnôt atribútov. Táto postupnosť prechádza od koreňa stromu k r-tému listu. Váha produkčného pravidla sa dá slovne opísať ako súčin stupňov príslušnosti hodnôt vstupných atribútov postupnosti c. Pokiaľ poznáme váhy produkčných pravidiel, tak môžeme vyjadriť stupeň príslušnosti pre každú hodnotu B_j výstupného atribútu B.

$$\mu_{B_j} = \sum_{r=1}^{R} w_r \times B_j^r,$$

kde B_j^r je j-ta hodnota vektora $\mathbf{B}^r$ v liste produkčného pravidla.

Príklad výpočtu váhy produkčného pravidla a vierohodnosti rozhodnutia

Z fuzzy rozhodovacieho stromu sme získali nasledujúce produkčné pravidlá. V príklade ukážem postup pre výpočet váh týchto pravidiel a hodnôt výstupného atribútu. Nová inštancia je zadaná tabuľkou

A_1		A_2				B	
$A_{1,1}$	$A_{1,2}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$	B_1	B_3
0.5	0.5	0.1	0.0	0.9	0.0	?	?

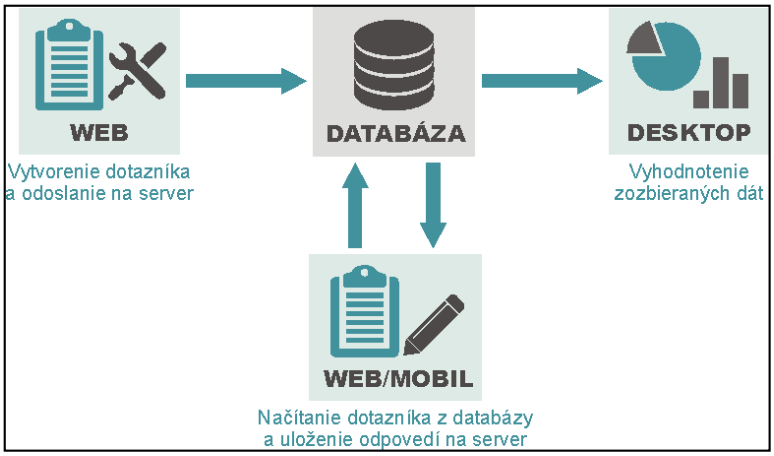
- **if**(A1 = A11) & **if**(A2 = A21) then B = [0.332,0.668]
- **if**(A1 = A11) & **if**(A2 = A22) then B = [0.335,0.665]
- **if**(A1 = A11) & **if**(A2 = A23) then B = [0.33,0.67]
- **if**(A1 = A11) & **if**(A2 = A24) then B = [0.346,0.654]
- **if**(A1 = A12) then B = [0.322,0.678]

$$w_1 = \mu_{1,1}(x) \times \mu_{2,1}(x) = 0.5 \times 0.1 = 0.05$$
$$w_2 = \mu_{1,1}(x) \times \mu_{2,2}(x) = 0.5 \times 0.0 = 0.00$$
$$w_3 = \mu_{1,1}(x) \times \mu_{2,3}(x) = 0.5 \times 0.9 = 0.45$$
$$w_4 = \mu_{1,1}(x) \times \mu_{2,4}(x) = 0.5 \times 0.0 = 0.00$$
$$w_5 = \mu_{1,2}(x) \qquad = 0.5 \qquad = 0.40$$

$$B_1 = 0.332 \times 0.05 + 0.33 \times 0.45 + 0.322 \times 0.50 = 0.2939$$
$$B_2 = 0.668 \times 0.05 + 0.67 \times 0.45 + 0.678 \times 0.50 = 0.6061$$

3 PRAKTICKÁ ČASŤ

Praktická časť práce z veľkej miery vychádza z inžinierskeho projektu. Cieľom projektu bolo vytvárať a vyhodnocovať dotazníky zamerané na psychologické vlastnosti človeka a dotazníky zamerané na podpis osoby. Pre účely projektu vzniklo niekoľko aplikácií, či už na zber dát, alebo aj samotné vyhodnotenie. Pre spomínané vyhodnocovanie boli zvolené práve metódy hĺbkovej analýzy dát. Hĺbková analýza dát zahŕňa širokú škálu algoritmov, v ktorej sme sa zamerali na rozhodovacie stromy a ich implementáciu. Schéma ako projekt fungoval je na obrázku Obrázok 14

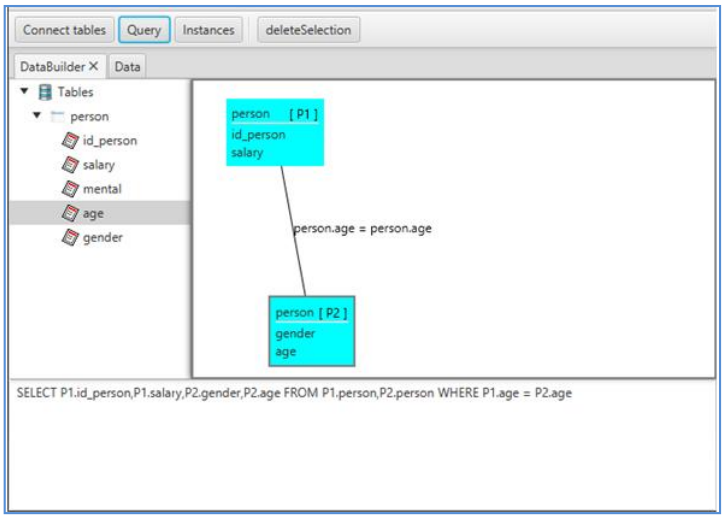


Obrázok 14. Schéma inžinierskeho projektu

Na obrázku sú znázornené spôsoby komunikácie s databázou. Prvá aplikácia slúžila na vytváranie dotazníkov a aj ich modifikáciu. Ďalšie aplikácie boli zamerané na zber dát. Tie boli vyvinuté pre mobilnú platformu *android*, ale aj webová stránka. V tejto kapitole sa budem venovať časti desktop. Pôvodne dáta neboli ukladané pomocou fuzzy logiky. Aplikácia je z toho dôvodu členená na dve časti. V prvej časti sú algoritmy pre spracovanie klasických dátových množín. Druhá časť spracováva fuzzy dátové množiny. V tejto práci rozšírim aplikáciu z inžinierskeho projektu tak, aby bolo schopná vyhodnocovať dotazníky definované pomocou fuzzy premenných.

3.1 Aplikácia z inžinierskeho projektu

Logická časť programu je napísaná v jazyku *Java.* Vizuálna časť aplikácie je vytvorená pomocou *JavaFx. JavaFx* je skupina grafických a mediálnych balíčkov, ktoré umožňujú softvérovým vývojárom navrhovať, vytvárať, testovať, ladiť a nasadzovať bohaté klientske aplikácie, ktoré konzistentne a spoľahlivo fungujú na rôznych platformách [17]. Na písanie *JavaFX* aplikácií nepostačuje klasické *Java Development Kit*. Pred začatím vývoja aplikácie je nevyhnutné pridať do počítača *JavaFx runtime* knižnice. *Java development kit* je taktiež nevyhnutný. Odporúčaná verzia je 7 a viac. Pôvodne sa predpokladalo, že aplikácia sa bude pripájať na databázu, z ktorej bude preberať dáta. Pripojenie k databáze sa konfiguruje v externom súbore. Užívateľ musí vyplniť prihlasovacie údaje k databáze. Program sa potom k tejto databáze dokáže pripojiť. Keďže sa databáza počas tvorby aplikácie taktiež vyvíjala a menila, bolo vhodné aplikáciu navrhnúť tak, aby bolo možné vyhodnocované dáta v aplikácií namodelovať podľa potreby. Zmenu štruktúry databázy spôsobovali zo začiatku predovšetkým psychologické testy, ktoré sú pomerne zložito stavané a podliehajú prísnym legislatívnym pravidlám. Zaobstarať platný a validný test bolo pomerne zložité. V aplikácií som preto vytvoril niečo ako dátový modeler. Hlavná idea je v tom, že na ľavej strane užívateľ vidí databázové tabuľky s príslušnými stĺpcami a pomocou *drag and drop* funkcionality si ich môžem pretiahnuť na plátno. Na plátne môže pridávať potrebné stĺpce tabuliek, alebo vytvárať spojenia medzi tabuľkami. Takýmto spôsobom je možné namodelovať to, čo chceme vyhodnotiť. Pokiaľ ukončíme proces modelovania, tak klikneme na príslušné tlačidlo a vygenerujeme si databázový dotaz. Tento dotaz je následne možné ešte modifikovať. Modifikácia je umožnená pre potreby doplnenia podmienok, ktoré modeler neumožňuje vytvoriť. Aplikácia využíva algoritmy pre podporu rozhodovania na základe neurčitých, ale aj určitých dát.



Obrázok 15. Ukážka aplikácie (príprava dát)

3.1.1 Vyhodnocovanie určitých dát

Začnem popisom prvej časti. Vyberal som algoritmy, ktoré sú vhodné na prácu s dátami v lingvistickej forme. Algoritmy C4.5 a ID3 sú implementované pomocou *weka* knižnice. *Weka* je knižnica, ktorá ponúka rozsiahlu kolekciu algoritmov riešiacich problémy hĺbkovej analýzy dát. Tento balíček je vydávaný pod GNU licenciou. Ponúka aj isté, už spracované softwarové riešenie, ktoré sme nevyužívali. *Weku* sme využili priamo v *Java* kóde. Výhodou takéhoto použitia je väčšia miera prispôsobiteľnosti kódu. Ovládanie vytvorenej aplikácie je možné cez grafické rozhranie. Určité dáta sa načítavajú priamo z databázy. Na načítanie dát je potrebné vytvoriť dotaz. Tento dotaz je možné spracovať manuálne alebo za pomoci knižnice *weka*. Pri využití algoritmov z tejto knižnice, potrebujeme vytvoriť z dát objekt, ktorý sa volá *Instances*. Objekt *instances* obsahuje jednotlivé inštancie dát a tak isto obsahuje aj informácie o atribútoch týchto objektov.

Veľa z algoritmov nedokáže pracovať s numerickými hodnotami atribútov. Numerické hodnoty treba previesť pomocou známych postupov na lingvistické hodnoty. Aj nami vybrané algoritmy C4.5 a ID3 vyžadujú v implementácií *weki* striktne lingvistické hodnoty. Prevod hodnôt je taktiež možné urobiť automatizovane za pomoci *weka* knižnice. Využíva sa na to inštancia objektu *Filter*. Objekt *Filter*

obsahuje statickú metódu *useFilter,* do ktorej sa ako vstupný parameter pošle objekt dediaci od predka *SimpleBatchFilter* a druhý parameter je pôvodná množina prípadov.

Pokiaľ máme v dátovej množine len lingvistické atribúty, tak je možné z tejto množiny vytvoriť rozhodovací strom. Pred vytvorením stromu je potrebné vedieť, ktorý atribút z objektu *Instances* bude klasifikačný, respektíve výstupný. Podľa klasifikačného atribútu sa vytvárajú klasifikačné triedy. Pri volaní metódy na vytvorenie rozhodovacieho stromu sa ako parameter tejto metódy zadáva *confidenceFactor.* Tento parameter sa využíva v procese orezávania stromu. V práci som zvolil algoritmus C4.5. V knižnici *weka* je tento algoritmus označený ako *J48.* Zodpovedá tomu aj meno triedy, ktorá tento algoritmus implementuje.

Celý proces tvorby modelu určeného na vyhodnocovania určitých dát v aplikácií je možné znázorniť nasledujúcim diagramom.

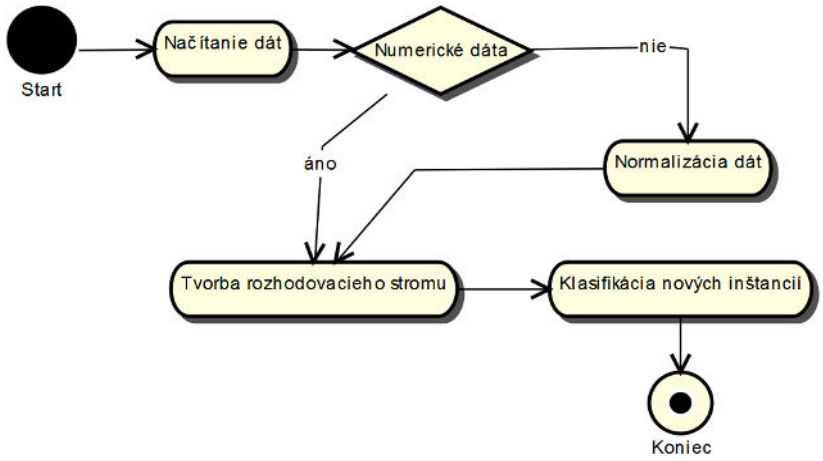


Diagram 1. Proces tvorby modelu a klasifikácie

3.1.2 Implementácia algoritmov pre neurčité dáta

Algoritmy pre podporu rozhodovania na základe neurčitých (fuzzy) dát som implementoval podľa [3]. Validitu implementovaných algoritmov som testoval pomocou fuzzy tabuľky v kapitole Fuzzy dáta. Výstup implementovaných algoritmov bol zhodný s výsledkami príkladov v [3]. Preto predpokladám validitu a správnosť algoritmov.

Projekt, v ktorom sú implementované popísané algoritmy je oddelený od samotnej aplikácie a je využívaný ako externá knižnica. Túto knižnicu som navrhoval tak, aby sa s jej pomocou dalo urobiť potrebné vyhodnotenie. Je rozdelená do troch základných balíčkov. Prvý balíček určený je pre prácu s dátami, druhý je zameraný na konštrukciu fuzzy rozhodovacích stromov a produkčných pravidiel a tretí balíček rieši pomerne jednoduchú klasifikačnú úlohu.

Balíček pre prácu s dátami. V tomto balíčku je implementovaná časť, ktorá zabezpečuje vhodnú reprezentáciu dát pre algoritmy celej knižnice. Dáta sú reprezentované pomocou troch typov atribútov a to lingvistický jednohodnotový, fuzzy lingvistický atribút a numerický atribút. Spôsob aké dáta popisujú jednotlivé atribúty vysvetlím pomocou nasledovnej tabuľky

Fuzzy lingvistický atribút			Jednohodnotový lingvistický atribút	Numerický atribút
Výška			***Výška***	***Výška***
veľký	***priemerný***	***malý***	veľký	190
0.9	0.1	0.0	*priemerný*	169
0.8	0.2	0.0	*malý*	182
0.0	0.7	0.3	*priemerný*	185
			malý	162

Tabuľka 4. Príklad atribútov z knižnice pre fuzzy dáta

Numerické a lingvistické jednohodnotové som v práci využíval predovšetkým v procese transformácie reálnych hodnôt atribútu na fuzzy lingvistickú premennú. Z atribútov som vytváral dátové množiny. Každá dátová množina knižnice môže obsahovať ľubovoľný počet atribútov. V knižnici sú implementované dva typy dátových množín. Prvá množina je univerzálna a môže obsahovať ktorýkoľvek typ atribútu. Druhá dátová množina sa zameriava na prácu s fuzzy atribútmi. V tejto časti knižnice sú umiestnené triedy, ktoré zabezpečujú transformácie atribútov na fuzzy kategorické atribúty. Transformovať je možné fuzzy lingvistický aj jednohodnotový lingvistický atribút. Prevod medzi jednohodnotovým lingvistický atribút a fuzzy lingvistický atribút je implementovaný pomerne jednoducho. Ako prvé vytvorím množinu hodnôt, ktoré jednohodnotový lingvistický atribút nadobúda. V ďalšom kroku vyjadrím pre každú možnú hodnotu stupeň príslušnosti, tak že pôvodná hodnota inštancie zodpovedá stupňu príslušnosti s hodnotou 1.

Jednohodnotový lingvistický atribút

Fuzzy lingvistický atribút

Výška
veľký
priemerný
malý
priemerný
malý

Výška		
veľký	***priemerný***	***malý***
1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0
0.0	1.0	0.0
0.0	0.0	1.0

Tabuľka 5. Transformácia Jednohodnotový lingvistický atribút na Fuzzy lingvistický atribút

Transformácia numerického atribútu na Fuzzy lingvistický atribút je implementovaná podľa kapitoly 2.4. Atribút je možné transformovať dvoma spôsobmi. Pri prvom spôsobe je potrebné zadať počet termov. Pri druhom spôsobe sa automaticky určí optimálny počet termov na základe fuzzy váženej entropie. Dôvod ponechania aj prvého spôsobu bol ten, aby bolo možné transformovať atribút aj bez známeho výstupného atribútu.

Dáta je možné načítať z *csv* súboru. Na začiatok súboru je potrebné zadať zoznam atribútov a ich hodnôt. Následne sa v súbore vypĺňajú už len riadky pre jednotlivé inštancie dát. Uvediem príklad spomenutého *csv* súboru. V tomto súbore načítam dátovú množinu popísanú atribútmi *rýchlosť*, *výška*, *intelekt* a *pohlavie*. Atribút rýchlosť nadobúda tri hodnoty a to *rýchly*, *pomalý* a *priemerný*. Výška nadobúda hodnoty *malý* a *veľký*. *Intelekt* a *pohlavie* taktiež nadobúdajú dve hodnoty. Pre intelekt *múdry* a *priemerný* a *pohlavie* nadobúda *muž* alebo *žena*. Súbor, ktorý popisuje takúto dátovú množinu vyzerá nasledovne.

```
Rýchlosť, Rýchly, Priemerný, Pomaly,
Výška, Veľký, Malý,
Intelekt, Múdry, Priemerný,
Pohlavie, Žena, Chlap,
#VAL#
  0.0900,  0.0790,    0.8310,  0.9022,  0.0978,  0.9993,  0.0007,  0.7263,  0.2737,
  0.0907,  0.0043,    0.9051,  0.0061,  0.9939,  0.8450,  0.1550,  0.6113,  0.3887,
  0.2743,  0.2534,    0.4723,  0.5388,  0.4612,  0.2545,  0.7455,  0.9868,  0.0132,
  0.1261,  0.2483,    0.6256,  0.1556,  0.8444,  0.3293,  0.6707,  0.4324,  0.5676,
  0.3303,  0.0033,    0.6664,  0.4560,  0.5440,  0.1832,  0.8168,  0.9270,  0.0730,
```

V prvých štyroch riadkoch súboru sú zapísané názvy atribútov a ich hodnoty. V každom riadku je prvá položka názov atribútu. Za názvom nasledujú hodnoty atribútu až do konca riadku. V súbore je dôležitý riadok, ktorý obsahuje text *#VAL#*. Pod týmto riadkom sa nachádzajú riadky, ktoré obsahujú hodnoty jednotlivých dátových inštancií. Pre prvú inštanciu zodpovedá prvý riadok, pre druhú inštanciu zodpovedá druhý riadok, až pre n-tú inštanciu zodpovedá n-tý riadok. Prvému atribútu *rýchlosť* zodpovedajú prvé tri stĺpce. Pre každú hodnotu jeden. Pre hodnotu *rýchly* zodpovedá prvý stĺpec, pre hodnotu *priemerný* zodpovedá druhý a pre hodnotu *pomalý* tretí stĺpec. Analogickým postupom je možné popísať všetky ďalšie atribúty.

Dátové množiny je možné rozdeľovať a spájať. Implementácie dátových umožňujú pridávať a odoberať atribúty z existujúcich dátových množín. Ďalej umožňujú rozdelenie inštancií dátovej množiny na dve dátové množiny a to trénovaciu a testovaciu v zadanom pomere.

Na nasledujúcom obrázku znázorním diagram tried balíčku *Data.*

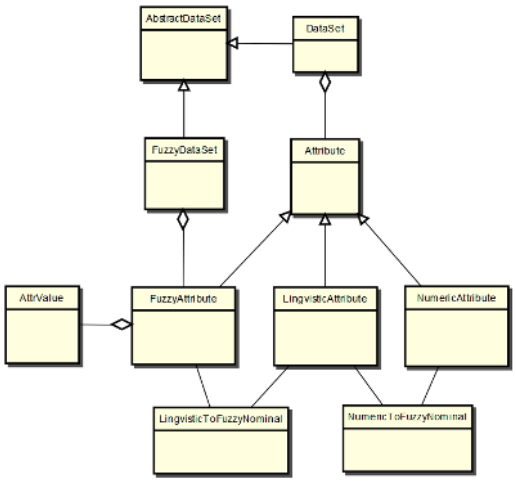


Diagram 2. Diagram tried dátového balíčka

Balíček FDT: tento balíček predstavuje nosnú časť celej knižnice. Celú túto knižnicu som začal vytvárať z dôvodu budovania fuzzy rozhodovacích stromov. Táto funkcionalita je zabezpečená práve týmto balíčkom, ktorý umožňuje vytvoriť dva takéto stromy a to fuzzy usporiadaný a neusporiadaný strom. V tomto balíčku sú implementované aj produkčné pravidlá, ktoré sa vytvárajú z už vybudovaných fuzzy

rozhodovacích stromov. Celý postup tvorby stromu pomocou tohto projektu je pomerne jednoduchý. Z pohľadu užívateľa ho je možné vyjadriť nasledujúcim diagramom činností.

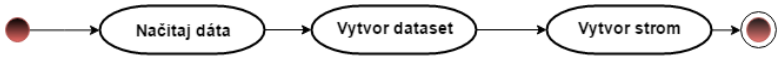


Diagram 3. Činnosti pre vytvorenie fuzzy rozhodovacieho stromu

Implementované riešenia umožňujú vykonávať klasifikáciu nových inštancií. Nová inštancia je taká inštancia, ktorá sa nenachádzala v trénovacej množine počas konštrukcie fuzzy rozhodovacieho stromu. Pomocou fuzzy rozhodovacieho stromu som vykonával klasifikáciu, bez toho aby sa vyčísľovali všetky hodnoty výstupného atribútu novej inštancie. Funguje to tak, že nová inštancia prechádza od koreňa až k listu fuzzy rozhodovacieho stromu. Pokiaľ sa nachádza už v liste, tak proces klasifikácie vráti triedu s najväčšou vierohodnosťou v danom liste, alebo všetky stupne vierohodnosti v liste. Táto návratová hodnota záleží od zvolenej funkcie.

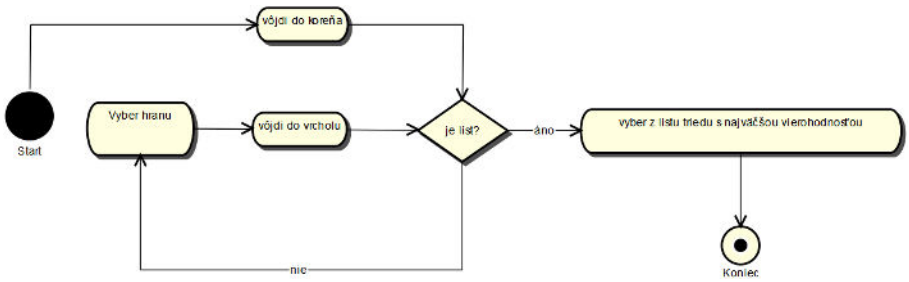


Diagram 4. Diagram popisujúci klasifikáciu

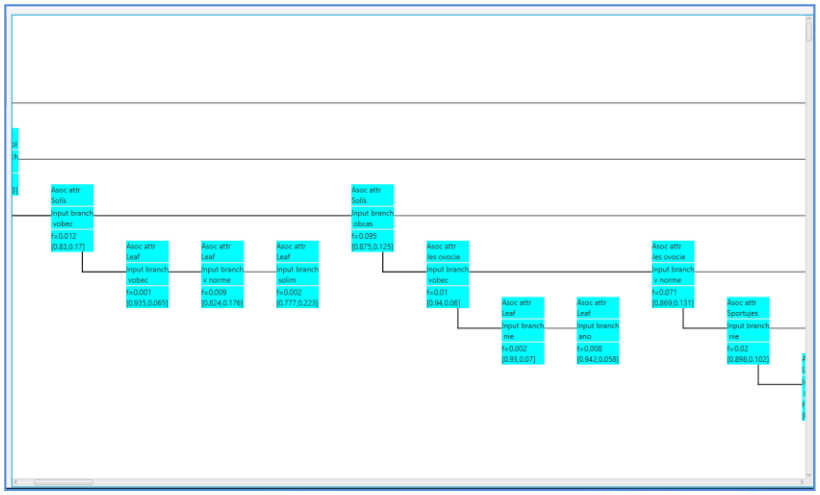
V projekte je naprogramovaná funkcia, ktorá pre novú inštanciu vyjadrí stupne príslušnosti pre hodnoty výstupného atribútu. Pre vyčíslenie všetkých neznámych hodnôt výstupného atribútu je potrebné vytvoriť produkčné pravidlá. V mojej implementácií vytváram produkčné pravidlá pomocou rekurzívnej funkcie, ktorá prejde postupne všetky cesty od koreňa ku každému listu fuzzy rozhodovacieho stromu, a pre každú takúto cestu vytvorí produkčné pravidlo podľa kapitoly Produkčné pravidlá a ich konštrukcia. Produkčné pravidlá vygenerované pomocou tejto knižnice pre fuzzy dátovú tabuľku zadanú v kapitole Fuzzy dáta vyzerajú nasledovné:

```
if(A2 = A21) & if(A1 = A11) & if(A4 = A41) then B = [0.142,0.678,0.180]
if(A2 = A21) & if(A1 = A11) & if(A4 = A42) then B = [0.332,0.624,0.045]
if(A2 = A21) & if(A1 = A12) then B = [0.376,0.504,0.120]
if(A2 = A21) & if(A1 = A12) then B = [0.110,0.163,0.727]
if(A2 = A22) & if(A4 = A41) then B = [0.173,0.074,0.754]
if(A2 = A22) & if(A4 = A42) & if(A1 = A11) then B = [0.579,0.391,0.030]
if(A2 = A22) & if(A4 = A42) & if(A1 = A12) then B = [0.551,0.142,0.307]
if(A2 = A22) & if(A4 = A42) & if(A1 = A12) then B = [0.369,0.136,0.494]
if(A2 = A23) then B = [0.163,0.049,0.788]
```

Tieto pravidlá zodpovedajú usporiadanému fuzzy rozhodovaciemu stromu. Strom je zobrazený na Obrázok 10.

V balíčku FDT je umožnené vyjadriť presnosť a chybu prijatého rozhodnutia. Tieto pojmy sú vysvetlené v kapitole 3.2. Uvedenú funkcionalitu som pridal z dôvodu, aby bolo možné algoritmy porovnať, ale aj z dôvodu možnosti číselne vyjadriť kvalitu rozhodovacieho modelu.

V knižnici je implementovaná trieda, ktorá umožňuje vizuálne zobrazenie výsledných fuzzy rozhodovacích stromov. Pri tvorbe zobrazovacej triedy som vychádzal z prehliadky *preorder*. Vrcholy stromu som vykreslil v poradí tejto prehliadky a každý vrchol bol pred vykreslením odsadený podľa úrovne, na ktorej sa nachádza. Vykresliť výsledný strom je možné vertikálne aj horizontálne. Na nasledujúcom obrázku znázorním horizontálne vykreslený fuzzy usporiadaný rozhodovací strom pre fuzzy dátovú tabuľku zadanú v kapitole Fuzzy dáta.



Obrázok 16. Vykreslený usporiadaný fuzzy rozhodovací strom

Balíček **Clustering** je posledný nepopísaný balíček knižnice. Tento balíček obsahuje len jeden algoritmus zabezpečujúci zhlukovanie. Vstupným parametrom metódy je buď zoznam reálnych čísel alebo numerický atribút a počet výsledných intervalov. Algoritmus potom rozdelí tieto čísla do intervalov podľa algoritmu popísaného v kapitole 1.2.1.

3.2 Porovnanie implementovaných algoritmov

Implementované algoritmy porovnám na vygenerovaných dátových množinách, ktoré sú priložené v prílohe C. Dátové množiny sú popísané pomocou vstupných atribútov a jedeného výstupného. Počet vstupných atribútov je uvedený v tabuľke Tabuľka 6. Žiadna dátová množina neobsahovala počas testov chýbajúce hodnoty atribútov. V spomínanej tabuľke sa nachádza stĺpec s hlavičkou *Dátová množina,* ktorý obsahuje názov súboru, z ktorého je dátovú množinu možné načítať. Boli uskutočnené dva typy porovnania.

Prvý spôsob je, že dátovú množinu rozdelím na dve časti a to množinu na natrénovanie a množinu na klasifikovanie, resp. testovanie. Rozdelenie spravím v pomere 80% prípadov padne do trénovacej množiny a 20% prípadov do testovacej množiny. Prípady sú do množín vyberané náhodne podľa rovnomerného rozdelenia pravdepodobnosti. Následne pomocou implementovaných algoritmov vytvorím fuzzy rozhodovacie stromy z trénovacej množiny. Po vytvorení stromu klasifikujem prípady v testovacej množine. Nakoniec percentuálne vyjadrím, koľko prípadov bolo nesprávne klasifikovaných. Tento postup vyjadrí presnosť prijatého rozhodnutia pre daný model. V tabuľke sú výsledky tohto porovnania v stĺpci PPR.

Druhý spôsob porovnania algoritmov nerozdelí dátovú množinu na dve časti. Pracuje len s tréningovou množinou prípadov. Najskôr z tejto množiny vytvorí fuzzy rozhodovací strom a následne klasifikuje všetky prípady v trénovacej množine pomocou vytvoreného fuzzy rozhodovacieho stromu. Pri tomto spôsobe testovania je dôležitý pomer medzi počtom chybne klasifikovaných prípadov a celkovým počtom prípadov. Tento pomer vyjadruje chybu prijatého rozhodnutia, v tabuľke pod stĺpcom CPR. V tabuľke je táto hodnota vyjadrená v percentách.

Algoritmus	Dátová množina	Počet atribútov v dátovej množine	Počet inštancií v dátovej množine	PPR	CPR
FDTu	G1	8	872	08.944 %	11,494 %
FDTo	G1	8	872	08.944 %	11,111 %
FDTu	G2	5	1000	10.800 %	12,500 %
FDTo	G2	5	1000	10.800 %	12,767 %
FDTu	G3	5	1081	39,000 %	33.500 %
FDTo	G3	5	1081	39,000 %	33.500 %
FDTu	G4	8	600	15.333 %	15,832 %
FDTo	G4	8	600	15.333 %	15,841 %
FDTu	G5	11	600	10.833 %	15,833 %
FDTo	G5	11	600	10.833 %	15,833 %

Tabuľka 6. Výsledky porovnania algoritmov *FDTu* a *FTDo*

Pre použité dátové množiny sú výsledky porovnania pre chybu a presnosť prijatého rozhodnutia veľmi podobné. Môže to byť spôsobené metódou generovania dátovej množiny.

3.3 Systém pre spracovanie dotazníkov

Aplikácia vytvorená v tejto práci predstavuje systém na vyhodnocovanie dotazníkov definovaných pomocou fuzzy logiky. V procese vyhodnocovania dotazníkov som použil rozhodovacie stromy popísané v predchádzajúcich kapitolách. Aplikácia v prvom kroku vytvára dátovú množinu zo zadaných dát. Táto množina má štruktúru, s ktorou dokážu pracovať implementované algoritmy. Následne sa z dátovej množiny vytvorí rozhodovací strom. Strom sa ďalej použije na analýzu, prípadne doplnenie funkcionality dotazníka, ktorý bude podľa vetvy stromu vyberať už iba vhodné otázky. Ďalšia implementovaná funkcionalita je vznik klasifikačného modelu. Z vytvoreného stromu je možné odvodiť produkčné pravidlá a na ich základe je implementovaná funkcia, ktorá klasifikuje nové inštancie do príslušnej triedy. Základné kroky aplikácie sú zobrazené v nasledujúcom *flowchart* diagrame.

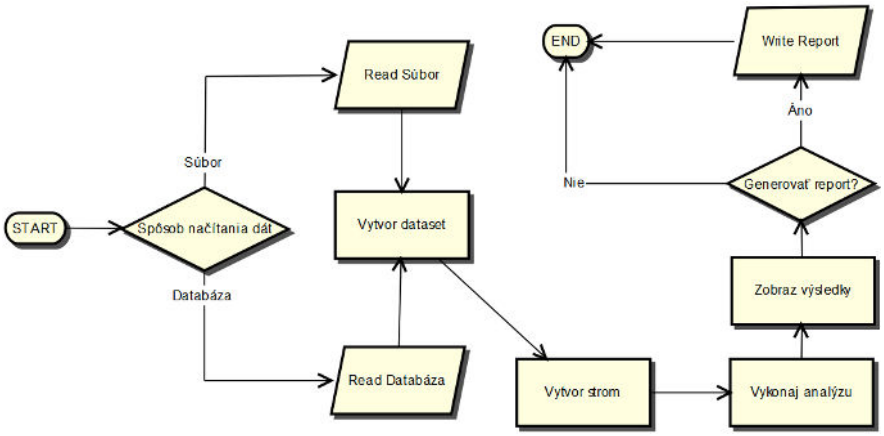


Diagram 5. Základné kroky aplikácie

Systém vytvorený v tejto práci predpokladá úplne vyplnenie testov. Aj implementované modely algoritmov predpokladajú, že dáta v dátovej množine neobsahujú chýbajúce hodnoty. Pokiaľ sa vyskytne v dátovej množine prípad, ktorý by obsahoval chýbajúce hodnoty, tak takýto prípad je pred konštrukciou fuzzy rozhodovacieho stromu z dátovej množiny odstránený. Samotná dátová množina je podobná fuzzy tabuľke vysvetlenej v kapitole Fuzzy dáta. Testová otázka predstavuje atribút a možnosti otázky predstavujú hodnoty tohto atribútu. Uvediem príklad pre takúto dátovú množinu. V Príklade dátová množina popisuje test s troma otázkami. Príklad je len na ilustráciu, nakoľko reálne testy obsahujú zvyčajne viac ako tri otázky. V dátovej množine sa nachádzajú dva vypracované testy. V takomto prípade je štruktúra dátovej množiny popísaná Tabuľka 7.

Atribút	Obsahuje podpis slučku?		Aká je hrúbka tvojho podpisu?			Je tvoj podpis naklonený?		Povaha	
Hodnoty atribútu	áno	nie	tenká	stredná	hrubá	rovný	naklonený	pokojný	agresívny
	0.1	0.9	0.9	0.1	0.0	1.0	0.0	0.2	0.8
	1.0	0.0	0.8	0.1	0.1	0.6	0.4	0.6	0.4

Komentár [N1]: skor: ano, nie

Tabuľka 7. Popis dátovej množiny pre vyhodnocovanie testov

Systém ponúka funkcionalitu na vyhodnotenie dotazníkov a tvorbu reportov, ktorú zhrniem v nasledujúcich bodoch:

- Systém umožňuje načítať dáta zo súboru, alebo databázy. Dáta je počas načítavania možné normalizovať. Normalizácia v tomto prípade znamená odstránenie inštancií, ktoré obsahujú chýbajúce hodnoty. Následne dokáže zobraziť načítaný test, ale aj fuzzy tabuľku.
- Vytvorenie fuzzy rozhodovacieho stromu podľa voľby užívateľa. Na výber má z dvoch možností a to fuzzy usporiadaný alebo neusporiadaný strom. Užívateľ môže strom zobraziť a vyhodnotiť aj opticky.
- Strom je po vytvorení možné uložiť a načítať. Pri uložení stromu sa neukladá dátová množina. Uloží sa len fuzzy rozhodovací strom. Pri načítaní uloženého stromu môže užívateľ ušetriť čas, ktorý je potrebný na vybudovanie fuzzy rozhodovacieho stromu.
- Systém na základe rozhodovacieho stromu identifikuje dôležité otázky. Pokiaľ by sa v teste vyskytovala otázka s nízkou vierohodnosťou, táto otázka nebude vybratá do žiadneho vrcholu ako asociačný atribút počas konštrukcie fuzzy rozhodovacieho stromu a systém odporučí takúto otázku odstrániť.
- Systém na základe vetiev stromu vytvára produkčné pravidlá. Tieto pravidlá dokážu identifikovať odpovede s otázkami pre konkrétnu triedu výstupného atribútu. Systém dokáže navrhnúť test podľa konkrétnej vetvy stromu, ktorý po vyhodnotení poskytne informáciu, či testovaná osoba padne do danej triedy. Vetva stromu, ktorá radí objekt do triedy výstupného atribútu poskytuje informáciu o najčastejších odpovediach, ktoré ľudia patriaci do tejto triedy s daným stupňom príslušnosti odpovedali. Na základe tejto analýzy systém vytvorí príslušný report.
- Systém umožňuje vytvárať produkčné pravidlá a pomocou nich dokáže vypočítať hodnoty výstupného atribútu pre nové inštancie. Pokiaľ test nebol vyhodnotený, systém dokáže vyhodnotenie spraviť.
- Systém umožňuje klasifikovať nové nevyhodnotené testy. Tiež sa to môže považovať za vyhodnotenie. Je očakávané, že takéto vyhodnotenie nemusí byť úplne správne. Preto systém umožňuje vyjadriť aj predpokladanú chybu a presnosť prijatého rozhodnutia.

- Systém ponúka prehľad základnej štatistiky pre dátovú množinu. Je možné zobrazovať priemerné hodnoty stupňov príslušnosti pre jednotlivé odpovede na otázky. Zobrazený je aj počet testov v dátovej množine.
- Systém umožní spojiť trénovaciu množinu prípadov prvého testu a hodnotu výstupného atribútu druhého testu. Pokiaľ sú testy mapované tak, že osoba vyplnila vždy súčasne prvý aj druhý test, tak na základe takejto trénovacej množiny je možné vyjadriť hodnotu stupňov príslušnosti výstupného atribútu pre druhý test. Pre lepšie pochopenie ukážem vysvetlenie pomocou Tabuľka 8 a Tabuľka 9. Tabuľky znázorňujú štruktúru prípadov pre testovaciu a trénovaciu množinu.

Vstupné atribúty prvý test	Výstupný atribút prvý test	Výstupný atribút druhý test

Tabuľka 8. Trénovacia množina

Vstupné atribúty prvý test	Výstupný atribút prvý test	Neznáma hodnota Výstupný atribút 2. test

Tabuľka 9. Testovacia množina – nový prípad sa nenachádza v trénovacej množine

Známe hodnoty
Neznáme hodnoty

- Taktiež je možné spojiť dve dátové množiny. Po spojení dvoch dátových množín je nutné znova vybrať výstupný atribút. Preddefinovane sa výstupný atribút nastavuje na posledný atribút dátovej množiny.
- Systém predstavuje doplnenie už existujúcej aplikácie z inžinierskeho projektu, ktorá umožňuje základné vyhodnocovanie určitých dát pomocou algoritmov ID3 a C4.5

Základnú funkcionalitu systému vyjadrím aj diagramom prípadov použitia. V diagrame sa nachádza niekoľko prípadov použitia, pričom každý z nich predstavuje inú funkcionalitu, ktorú fuzzy systém aplikácie ponúka pre užívateľa.

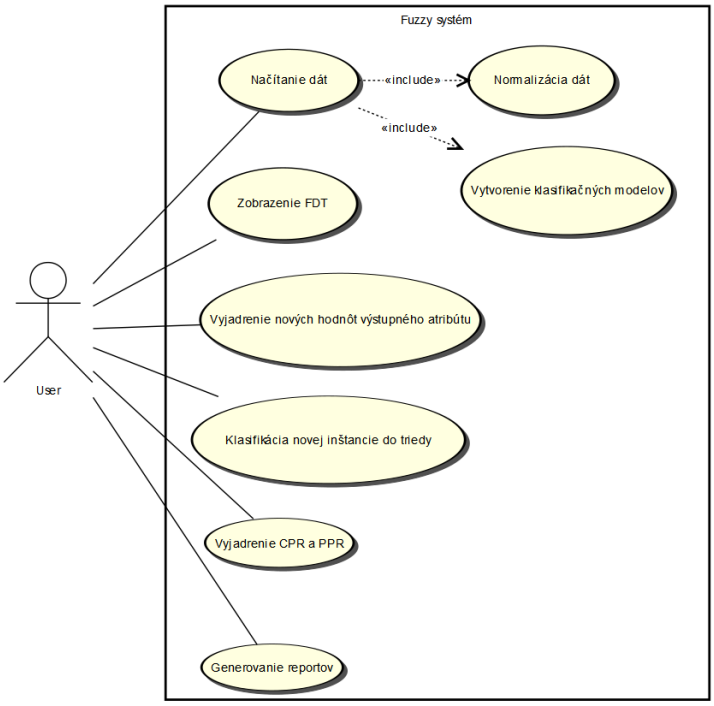
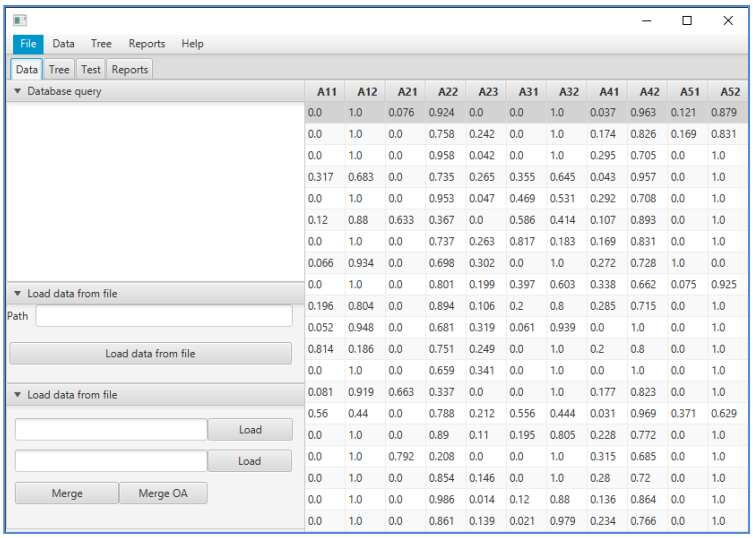


Diagram 6. Diagram prípadov použitia fuzzy systému aplikácie

3.3.1 Ovládanie aplikácie

Aplikáciu som rozčlenil do štyroch záložiek. Prvá časť je zameraná na načítanie dát. V tejto časti je možné zobraziť fuzzy tabuľku. Táto časť je v záložke *Data.* Druhá záložka sa venuje tvorbe rozhodovacieho stromu. Užívateľ si môže zadať minimálnu frekvenciu výskytov a maximálnu vierohodnosť pre ohraničenie rastu stromu. Túto funkciu som ponechal, aby bolo možné ovplyvniť dĺžku jednotlivých vetiev. Tým sa ovplyvní aj množstvo odporúčaných otázok do testu po vyhodnotení. V tejto záložke sa vytvorený strom aj vykreslí. Nasleduje záložka *Test.* V tejto záložke môže užívateľ zistiť chybu a presnosť prijatého rozhodnutia, klasifikovať nové inštancie, či vyjadriť kompletne hodnoty stupňov príslušnosti výstupného atribútu pre novú inštanciu. Ďalej môže zobraziť produkčné pravidlá. V tejto záložke sa taktiež zobrazí celý vypísaný test. Forma vypísaného testu je dostatočne prehľadná aj pre dlhé testové otázky. Posledná záložka je záložka *Reports.* V tejto záložke si užívateľ vyberie resp. vytvorí štruktúru reportu, ktorý chce vygenerovať.



Obrázok 17. Snímka obrazovky aplikácie z modulu pre fuzzy systém

3.3.2 Tvorba výstupných reportov

Report v mojej práci predstavuje automatizované vytvorenie výstupného dokumentu. Na vytváranie reportov v práci som využil knižnicu *iText*. Táto knižnica umožňuje pohodlnú a rýchlu tvorbu *pdf* dokumentov. *iText* knižnicu je možné voľne využívať len v neziskových aplikáciách na základe licencie *AGPL (Affero General Public License)*. Nepríjemná podmienka je, že aplikácia nesmie mať uzavretý zdrojový kód. S týmto som sa vysporiadal oddelením algoritmickej časti systému do iného projektu, ktorý som využil ako externú knižnicu. Pokiaľ by bola aplikácia komerčná, tak treba zakúpiť komerčnú licenciu pre *iTex*. Knižnica *iTex* umožňuje vytvoriť *pdf* dokument podľa zadaných príkazov HTML, preto som pred reportom vytvoril HTML výstup reportovaných dát. Na základe HTML výstupu som pomocou knižnice *iTex* vytvoril *pdf* dokument. Pred vytvorením HTML výstupu sa vždy kontroluje, či je report na základe dát možné vytvoriť. Ak to možné nie je, tak aplikácia zobrazí chybové hlásenie. Postup tvorby reportu je znázornený pomocou Diagram 7. Diagram aktivít tvorby reportu

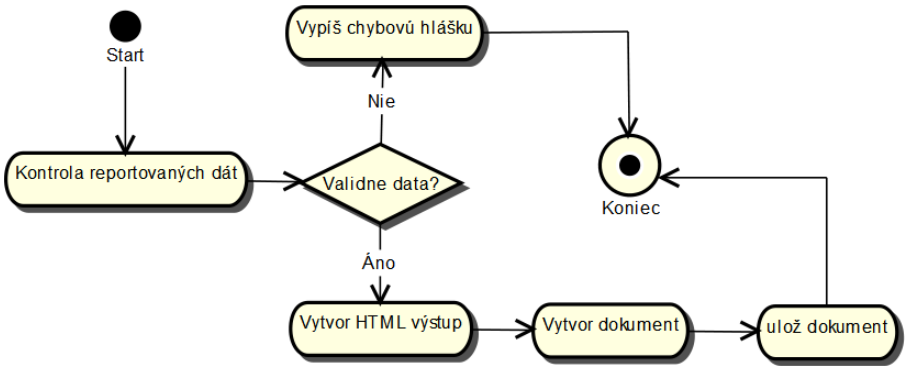


Diagram 7. Diagram aktivít tvorby reportu

Reporty si užívateľ môže vytvoriť v záložke *Reports* alebo aj pomocou príslušného menu. Aplikácia umožňuje tvorbu niekoľkých reportov. Jednotlivé reporty popíšem v nasledujúcich bodoch:

- Informačný report – v aplikácii je označený ako *Base report*. Tento report obsahuje základné informácie o dátovej množine ako počet otázok a množstvo vyhodnotených testov. V reporte je aj informácia o parametroch ohraničujúcich rast stromu. Ďalej report obsahuje údaje o presnosti a chybe prijatého rozhodnutia.
- Report odporúčaných otázok – aplikácia na základe rozhodovacieho stromu vyberie len potrebné otázky. Potrebné otázky sú také, ktoré boli použité ako asociačný atribút s niektorým vrcholom fuzzy rozhodovacieho stromu. Tieto otázky systém exportuje do *pdf* reportu súboru.
- Tvorba výberových testov – užívateľ zadá stupne vierohodnosti, podľa ktorých systém nájde najpodobnejší list fuzzy rozhodovacieho stromu. Podľa cesty, ktorá vedie z nájdeného listu až do vrcholu stromu sa vygeneruje nový test, ktorý preverí či respondent, ktorý test vyplnil spadá do danej triedy.
- Tvorba výberových testov – užívateľ zadá interval pre každú triedu výstupného atribútu. Systém cyklicky prejde všetky listy stromu a listy, v ktorých stupne príslušnosti k jednotlivým triedam vyhovujú intervalom vyberie do množiny vyhovujúcich listov. Následne prechádza cestu od každého listu do vrcholu stromu. Pokiaľ existuje otázka, ktorá sa v teste ešte nenachádza, tak túto otázku do testu zaradí.
- Report najčastejších odpovedí – užívateľ zadá stupne vierohodnosti, podľa ktorých systém nájde najpodobnejší list fuzzy rozhodovacieho stromu. Systém z nájdeného

listu prejde cestu až do vrcholu stromu, pričom v každom vrchole sa nachádza asociačný atribút predstavujúci otázku a z každého vrcholu vychádza vetva, ktorá predstavuje odpoveď na danú otázku. Vznikne postupnosť odpovedí, ktorá udáva najčastejšie odpovede na otázky pre danú triedu. Uvediem príklad:

Užívateľ ako vstup zadá stupne vierohodnosti 0.3 pre prvú triedu a 0.7 pre triedu druhú. Systém nájde najpodobnejší list z ktorého vytvorí cestu do vrcholu. Pre túto cestu zodpovedá nasledujúce produkčné pravidlo:

```
IF("Sklon" = "Naklonený") & IF("Hrúbka" = "tenká") then B = [0.332, 0.668]
```

Toto produkčné pravidlo predstavuje najpravdepodobnejšie zadané odpovede na otázky, ktoré radia respondenta do výslednej triedy.

4 ZÁVER

Diplomová práca mala stanovených niekoľko cieľov. Prvým cieľom práce bolo oboznámenie sa so základmi hĺbkovej analýzy údajov. Ja som sa v práci venoval rozhodovacím stromom, preto som sa zameral predovšetkým na túto tému. V diplomovej práci som pracoval prevažne s fuzzy dátami, preto som sa v niekoľkých kapitolách venoval aj prepojeniu fuzzy dát a hĺbkovej analýzy.

Nosná časť diplomovej práce bol návrh a implementácia aplikácie pre vyhodnocovanie dotazníkov definovaných pomocou fuzzy logiky. Táto časť diplomovej práce je v praktickej časti práce, ktorá začína od kapitoly 3. V tejto časti som opísal už existujúcu aplikáciu, ktorú som vytvoril pre potrebu inžinierskeho projektu. Túto aplikáciu som postupne rozšíril tak, aby bola schopná spracovať zadané dotazníky. Pre potreby aplikácie sa mi podarilo úspešne implementovať algoritmy pre vyhodnocovanie neurčitých dát a to usporiadaný a neusporiadaný fuzzy rozhodovací strom. Správnosť algoritmov som overoval na známych dátových množinách. Pomocou týchto algoritmov som vytvoril fuzzy systém pre spracovanie dotazníkov, ktorého funkcionalita je rozanalyzovaná v kapitole 0. Algoritmy som ešte pred začiatkom vývoja aplikácie porovnal, tak že som vyjadril chybu a presnosť prijatého rozhodnutia pre jednotlivé dátové množiny, ktoré sú priložené v prílohe k práci.

Ďalší stanovený cieľ diplomovej práce bol zameraný na overenie funkcionality a použiteľnosti aplikácie. Pre tento cieľ som mal využiť dáta nazbierané počas inžinierskeho projektu, ale toto nebolo možné z dvoch dôvodov. Počas inžinierskeho projektu sa nám nepodarilo nazbierať dostatočné množstvo dát a aj tie dáta, ktoré boli nazbierané, tak nebolo ich možné vyhodnotiť. To znamená, že nebola známa hodnota výstupného atribútu ani pre jednu inštanciu v dátovej množine. Vyhodnotiť dáta bez známej hodnoty výstupného atribútu pomocou systému, ktorý som vytvoril nie je možné. Preto som dátové množiny generoval, čo mohlo výrazne ovplyvniť štruktúru vzniknutých fuzzy rozhodovacích stromov. Aj keď dáta boli menej kvalitné, napriek tomu boli schopné preukázať funkčnosť jednotlivých funkcií fuzzy systému a v konečnom dôsledku použiteľnosť celej aplikácie.

Vo vývoji aplikácie je stále možné pokračovať. Priestor na doplnenie funkcionality vidím v možnosti do fuzzy systému aplikácie doplniť ďalšie algoritmy. Hlavný prínos

práce je v rýchlom a automatizovanom spôsobe vyhodnocovania dotazníkov a pohodlnom získaní požadovaných reportov.

BIBLIOGRAFIA

1. DVORSKÁ, D. et al. **Elektronická učebnica pedagogického výskumu**. [S.l.]: KEGA.
2. CLIFTON, C. **Encyclopædia Britannica:** Definition of Data Mining. [S.l.]: [s.n.], 2010.
3. LEVASHENKO, V.; ZAITSEVA, E.; KOVALÍK, Š. **Projektovanie systémov pre podporu rozhodovania na základe neurčitých dát**. Žilina: EDIS ZU, 2013.
4. LEVASHENKO, V.; KOVALÍK, Š.; MATIAŠKO, K. **CLASSIFICATION BASED ON STABLE FUZZY DECISION TREE**. Žilina: [s.n.], 2005.
5. QUINLAN. **Induction of Decision Trees**. [S.l.]: Mach. Learn., 1986.
6. BAHETY, A. **Extension and Evaluation of ID3 – Decision Tree Algorithm**. Maryland: University of Maryland, College Park.
7. QUINLAN. **C4.5:** Programs for Machine Learning. [S.l.]: Morgan Kaufmann Publishers, 1993.
8. XINDONG, K. Q. G. Y. M. M. A. N. O. **Top 10 algorithms in data mining**. London: Springer-Verlag, 2007.
9. MAIMON, O.; ROKACH, L. **Data Mining and Knowledge Discovery Handbook**. New York: Springer Science+Business Media, 2010.
10. YUAN, S. **Fuzzy Sets and systems**. [S.l.]: Elsevier, 1995.
11. WANG, X. et al. **On the optimization of fuzzy decision trees**. Harbin: Hebei University, 2000.
12. LEE, H.-M. et al. **An efficient fuzzy classifier with feature selection based on fuzzy entropy**. [S.l.]: IEEE, 2001.
13. JANG, R. J.-S. **Structure determination in fuzzy modeling:** a fuzzy CART approach. Orlando: IEEE, 1994.
14. GUOXIU, L. **A comparative study of three Decision Tree algorithms:** ID3, Fuzzy ID3 and Probabilistic Fuzzy ID3. Rotterdam : Erasmus University Rotterdam, 2005.
15. LEVASHENKO, V.; ZAITSEVA, E. **Fuzzy Decision Trees in Medical Decision Making Support System**. Zilina: University of Zilina, 2012.
16. LEVASHENKO, V.; MARTINCOVÁ, P. **FUZZY DECISION TREE FOR PARALLEL PROCESSING**. Žilina: University of Žilina, 2005.
17. PAWLAN, M.; CASTILLO, C. **JavaFX Overview**. Redwood City: Oracle, 2013.

18. SPN. **Lexikón Informatiky**. Bratislava: SPN, 1991.

19. KEREKESOVÁ, K.; MALAťÁKOVÁ, M. **ID3**. Košice: TECHNICKÁ UNIVERZITA V KOŠICIACH, 2011.

20. AUJESKý, M. **Analýza citlivosti SVM so zameraním sa na medicínske prostredie**. Žilina: ŽILINSKÁ UNIVERZITA, 2015.

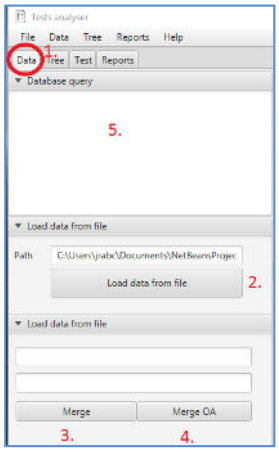
21. PIATETSKY-SHAPIRO; FRAWLEY. **Discovery, analysis, and presentation of strong rules**. Cambridge: [s.n.], 1991.

Príloha A

Užívateľská príručka pre vyhodnocovanie testov zadaných pomocou neurčitých dát

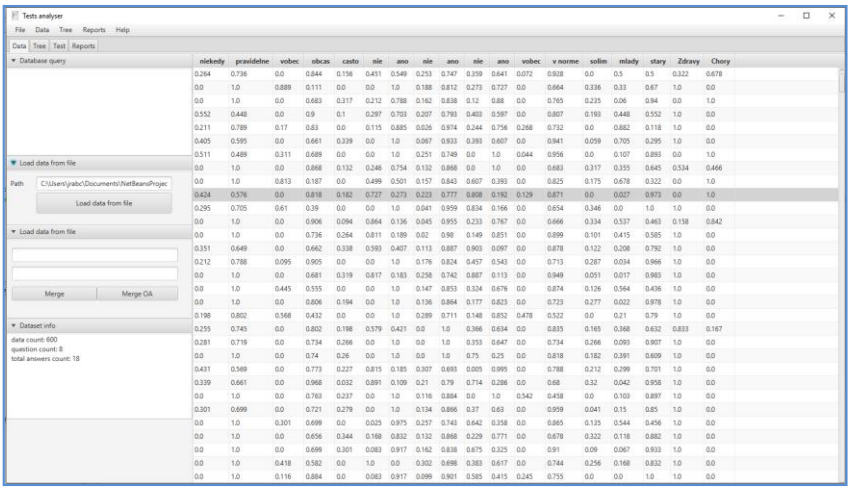
Užívateľ musí ako prvé načítať dáta. Všetka logika, ktorá zabezpečuje načítanie dát je umiestnená v záložke *data*. Dáta je možné načítať zo súboru, alebo z databázy.

1. Záložka Data – dátová logika aplikácie. V tejto záložke sa načítavajú vyhodnotené testy.
2. Tlačidlo *Load data from file* – po kliknutí na toto tlačidlo sa zobrazí okno, v ktorom treba otvoriť *csv* súbor s dátami. Tento súbor sa automatizovane načíta a vytvorí sa dátová množina
3. Tlačidlo *Merge* – po kliknutí na toto tlačidlo sa zobrazia dva formuláre na otvorenie súboru s dátovou množinou. Obidva tieto súbory sa následne načítajú a po načítaní sa dátové množiny spoja. Výstupný atribút bude posledný atribút množiny, ktorá bola načítaná z druhého súboru.
4. Tlačidlo *MergeOA* – Plní podobnú funkciu ako tlačidlo v bode 3. Rozdiel je v tom, že sa nespoja obidve dátové množiny, ale k prvej dátovej množine sa pridá výstupný atribút druhej dátovej množiny a tak sa vytvorí nová dátová množina. Výstupný atribút vzniknutej dátovej množiny je pôvodný výstupný atribút druhej dátovej množiny.



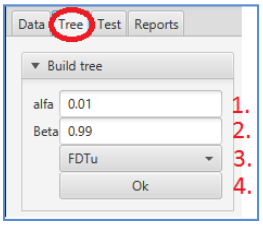
Obrázok 18. Ovládací panel načítavania dát

V záložke *data,* sa zobrazuje tabuľka celej dátovej množiny. Stĺpce tabuľky obsahujú *tooltip text*, ktorý vypíše priemerné stupne príslušnosti v konkrétnom stĺpci. V tejto záložke sa zobrazujú aj základné informácie o dátovej množine, ako je počet inštancií (vyplnených testov), počet všetkých otázok a celkový počet možností na jednotlivé otázky.



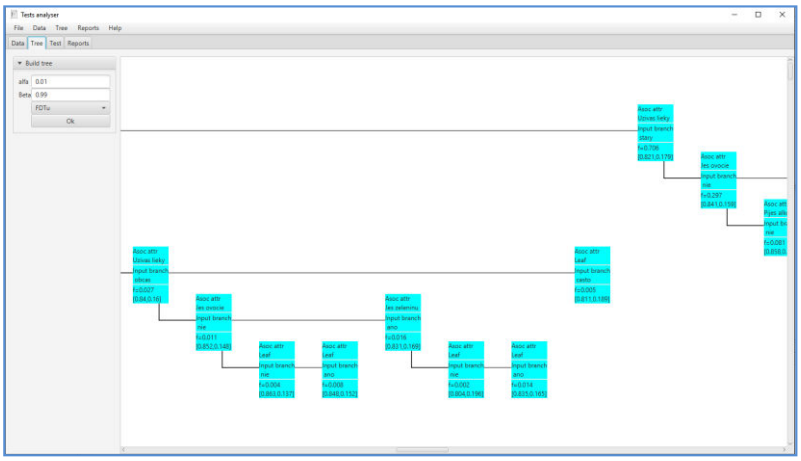
Obrázok 19. Snímka obrazovky záložky *Data*.

Po načítaní dát je potrebné skonštruovať fuzzy rozhodovací strom. Tento krok je nevyhnutý, pretože na základe fuzzy rozhodovacie stromu sú načítané dáta analyzované. Strom sa vytvára v záložke *Tree*.



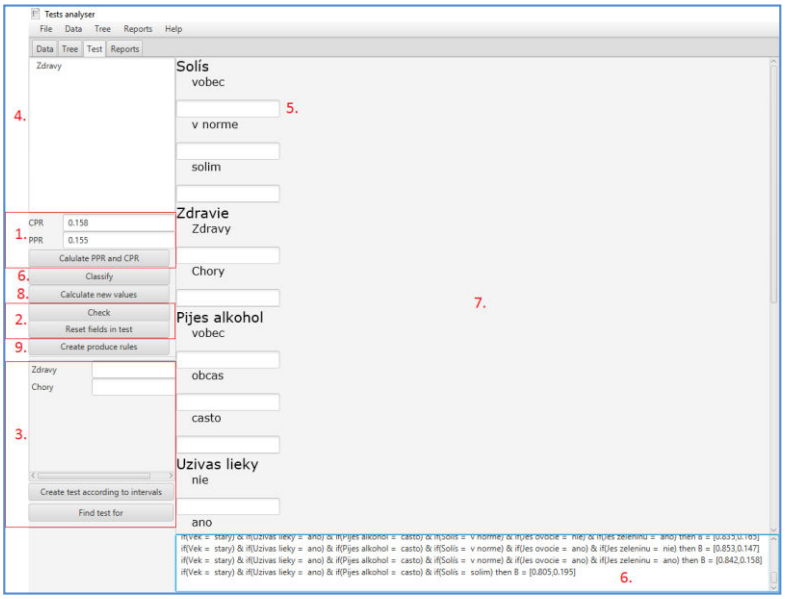
1. Prahová hodnota *alfa* – udáva minimálnu frekvenciu výskytov prípadov. Ak v niektorom vrchole frekvencia výskytu prípadov bude menšia ako zadaná hodnota alfa, tak tento vrchol sa stane listom.
2. Prahová hodnota *beta* – táto hodnota udáva maximálny stupeň vierohodnosti. Pokiaľ v niektorom vrchole vierohodnosť presiahne túto hranicu, tak vrchol sa stane listom.
3. Užívateľ si môže vybrať algoritmus, ktorý chce pre tvorbu fuzzy rozhodovacieho stromu použiť. Na výber má dva a to usporiadaný a neusporiadaný fuzzy rozhodovací strom.
4. Tlačidlo *OK* – po kliknutí na toto tlačidlo sa vytvorí fuzzy rozhodovací strom. Po dokončení konštrukcie fuzzy rozhodovacie stromu sa strom vykreslí.

Nastavenie hodnôt ovplyvní výslednú veľkosť stromu, tým sa ovplyvňuje aj množstvo odporúčaných otázok, preto tieto parametre nie sú nastavené automaticky, ale užívateľom.



Obrázok 20. Snímka obrazovky záložky *Tree*.

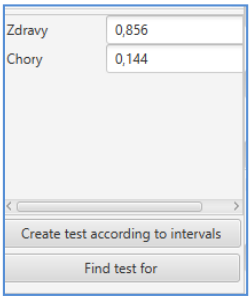
Na vyhodnocovanie testov slúži záložka *Test.* V tejto záložke je potrebné mať vytvorený fuzzy rozhodovací strom.



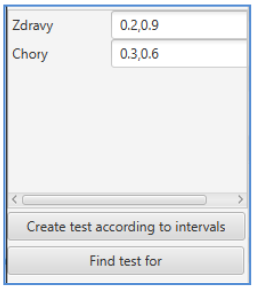
Obrázok 21. Snímka obrazovky záložky *Test*

V časti okna označenej číslom 7 sa okamžite po vytvorení dátovej množiny zobrazí test. Za otázkou nasledujú textové políčka označené na obrázku číslom 5. Do týchto políčok sa vpisujú stupne príslušnosti pre neklasifikované testy. Na obrázku sa nachádzajú tlačidlá *Classify* a *Calculate new values*, označené číslami 6 a 8. Funkcionalita týchto tlačidiel využíva spomínané textové polia. Podľa hodnôt v poliach sa vykonáva jednoduchá klasifikácia, prípadne je možné vyjadriť hodnoty príslušnosti ku každej triede výstupného atribútu. Výsledky klasifikácie sa zobrazia v (4). Textové polia je potrebné korektne vyplniť. Musia obsahovať reálne čísla, ktorých súčet pre každú otázku sa rovná hodnote jedna. Overiť, či vyplnené polia spĺňajú túto podmienku je možné po kliknutí na tlačidlo *check (2)*. Ak niektoré textové pole nespĺňa podmienku, tak sa vyfarbí na červeno, ak spĺňa, tak bude vyfarbené zelenou farbou. Po kliknutí na tlačidlo *Create produce rules* (9) systém vytvorí produkčné pravidlá, ktoré sa vypíšu do (6). Posledná funkcionalita (3), ktorá je umožnená v tejto záložke je vytvárať testy na základe vetiev stromu. V aplikácii sú pre tento účel implementované dva spôsoby:

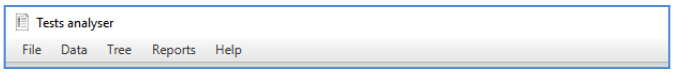
1. Pomocou tlačidla *Find test for* – V časti (3) sa nachádzajú textové polia pre každú hodnotu výstupného atribútu. Na obrázku výstupný atribút nadobúda hodnoty *Zdravý* a *Chorý*. Do políčok vedľa názvov hodnôt je potrebné zadať stupne vierohodnosti, ktoré chceme hľadať. Systém potom nájde najpodobnejší list v strome, z ktorého prejde cestu až do koreňa. Otázky na tejto ceste predstavujú vygenerovaný odporúčaný test.



2. Tlačidlo *Create test according to intervals* – Do textových polí je potrebné zadať interval pre každú hodnotu výstupného atribútu. Podľa zadaných intervalov systém vyberie všetky listy fuzzy rozhodovacieho stromu, ktorých stupne vierohodnosti spadajú do príslušných zadaných intervalov. Takto sa vytvorí množina otázok, ktoré predstavujú odporúčaný test.



Ovládanie aplikácie je možné aj pomocou lišty menu. Lišta menu je nasledovná.

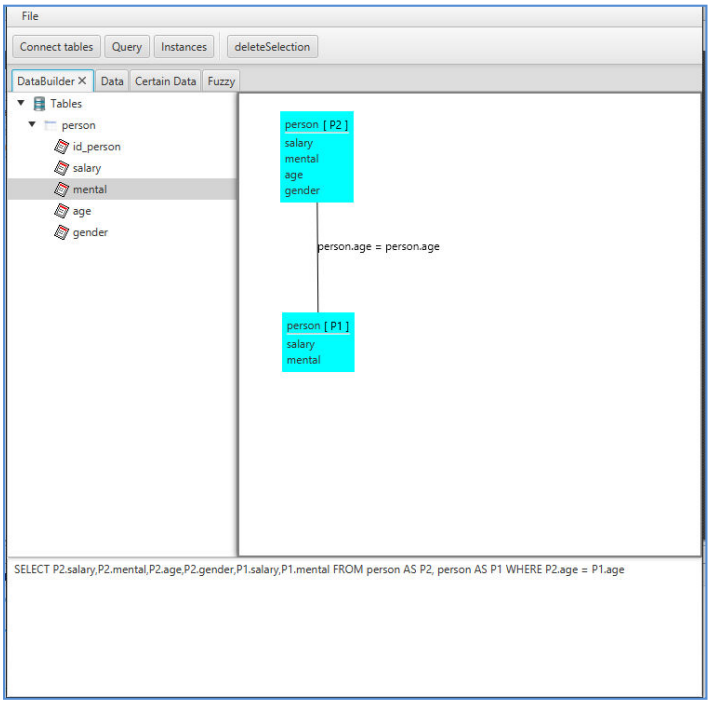


V menu *File* je možné uložiť a načítať stav aplikácie. Taktiež je možné aplikáciu v tomto menu ukončiť. Menu *Data* a záložka *Data* umožňujú rovnakú funkcionalitu. V menu *Tree* je možné načítať a uložiť strom do súboru. Týmto sa môže ušetriť čas pri ďalšom vyhodnocovaní rovnakého testu, nakoľko by nebolo potrebné budovať strom odznova. Pomocou menu, ale aj záložky *Rreports* sa vytvárajú reporty, ktoré sú popísané v kapitole Tvorba výstupných reportov.

Príloha B

Užívateľská príručka pre vyhodnocovanie testov zadaných pomocou určitých dát

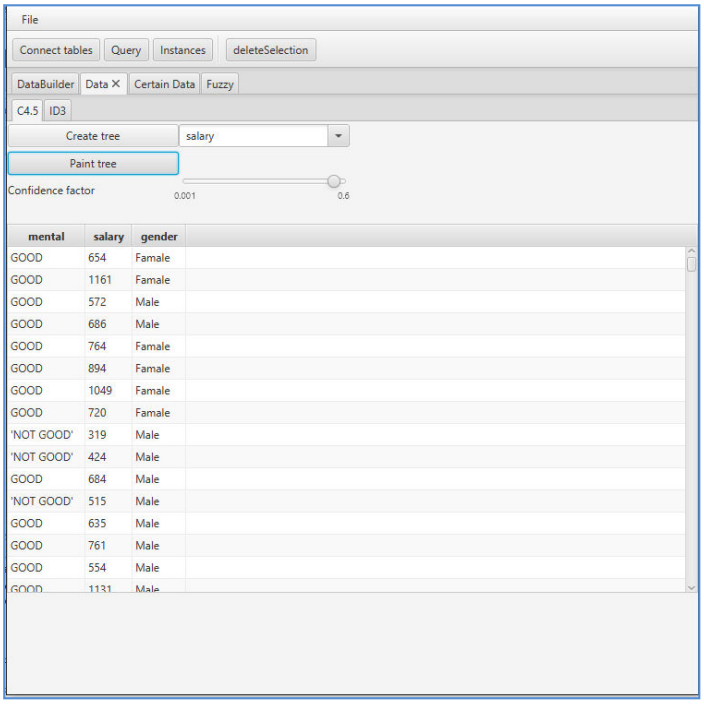
Aj v tejto aplikácií je potrebné najskôr načítať dáta z ktorých sa vytvorí dátová množina. Robí sa to pomocou nasledujúceho okna.



Obrázok 22. Snímka obrazovky záložky *DataBuilder*

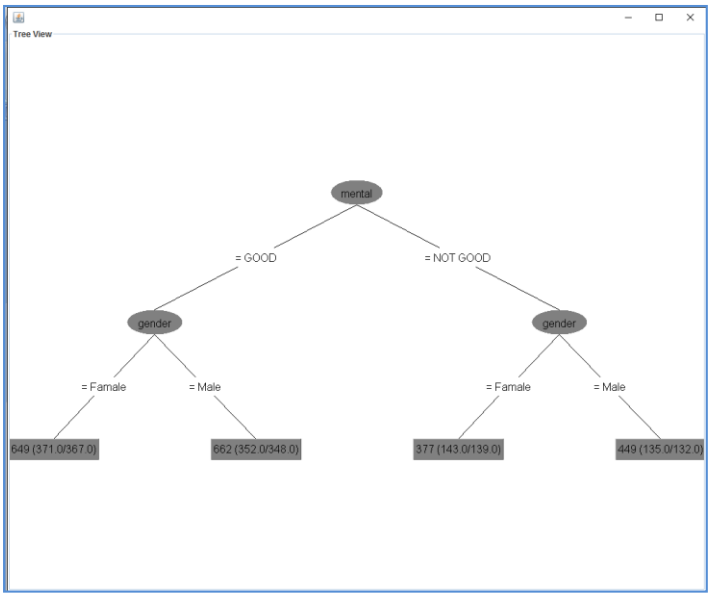
V ľavej časti sa nachádzajú v stromovej štruktúre (1) vypísané tabuľky databázy spolu s ich atribútmi. Tieto položky je možné pomocou *drag and drop* funkcionality pretiahnuť na plátno (2). Následne je možné vytvárať spojenia medzi tabuľkami (1), čím sa medzi tabuľkami vytvorí čiara. Po dvojkliknutí na vytvorenú čiaru sa zobrazí okno, kde sa zadajú prepojovacie atribúty. Keď sú dáta namodelované je potrebné kliknúť na tlačidlo *Query* (3), ktoré vygeneruje databázový dotaz do textového poľa (4). Dotaz je možné dodatočne upraviť. Po úprave dotazu je potrebné stačiť tlačidlo

Instances, ktoré vygeneruje dátovú množinu podľa zadaného databázového dopytu v textovom poli (4).



Obrázok 23. Snímka obrazovky záložky *Data*

V záložke *Data* si užívateľ môže vytvoriť rozhodovací strom, tak isto si môže prezrieť celú dátovú množinu, ktorá sa zobrazí do tabuľky. Pred vytvorením stromu je vhodné nastaviť *confidence factor,* ktorý sa využíva pre zastavenie rastu stromu. Po kliknutí na tlačidlo *Paint tree* sa strom vykreslí.



Obrázok 24. Snímka obrazovky vykresleného C4.5 stromu

Príloha C

CD nosič, ktorý obsahuje:

- Text diplomovej práce vo formáte PDF a docx
- Zdrojové kódy aplikácie a priložené dátové množiny
- Spustiteľný súbor
- Dokumentáciu