2016

Technická univerzita
v Košiciach

Kristína Machová

Nové trendy v strojovom učení

Štatistický prístup

doc. Ing. Kristína Machová, PhD.
Katedra kybernetiky a umelej inteligencie
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Vydané s podporou projektu KEGA grantu MŠVVaŠ SR č. 034TUKE-4/2014 „Integrácia študijných programov v odboroch Kybernetika a Umelá inteligencia“.

Edícia vedeckých spisov Fakulty elektrotechniky a informatiky TU Košice.
Lektorovali: doc. *Ing. Marián Mach, PhD.*
Ing. Peter Bednár, PhD.

ISBN: **978-80-553-2602-3**

PREDSLOV

V posledných rokoch nepribúdajú nové metódy strojového učenia, ktoré by boli založené na celkom nových princípoch. Skôr je tu snaha využiť existujúce techniky na riešenie stále nových úloh v rozličných oblastiach. Takouto modernou oblasťou, ktorou sa v posledných rokoch zaoberá mnoho odborníkov z oblasti strojového učenia, je analýza subjektívneho obsahu textu tak z hľadiska polarity a z hľadiska emócií v nich obsiahnutých ako aj analýza, ktorá by viedla k odhadu, resp. klasifikácii autorít. Väčšinou sa analyzujú krátke texty v rámci konverzačného obsahu, ktorý vzniká prevažne na sociálnych sieťach tvoriacich fenomén tejto doby. Tieto dáta v rámci konverzačného obsahu sa hromadia v sociálnych sieťach vo veľkom objeme, čo vedie k často pertraktovanému pojmu „Big Data“.

Možno práve pre veľký objem dát, ktoré je potrebné spracovať, sa aj v strojovom učení presadzujú metódy, ktoré sú charakteristické štatistickým prístupom. Preto aj predkladaná vysokoškolská učebnica bude orientovaná na tento nový trend v používaní existujúcich metód strojového učenia, ktorý je možné charakterizovať ako *štatistické učenie*. V rámci štatistického učenia sa vysokoškolská učebnica bude venovať regresnej analýze v predikcii so zameraním na lineárnu, viacnásobnú (multiple) a nelineárnu regresiu ako aj regresnej analýze v klasifikácii so zameraním na logistickú regresiu. Taktiež bude samostatná kapitola venovaná Metóde podporných vektorov (Support Vector Machines - SVM), ktorá sa v poslednej dobe veľmi úspešne používa na riešenie množstva aktuálnych problémov. Keď už je reč o často používaných prístupoch, nemôžeme opomenúť ani zloženú klasifikáciu pomocou techniky *náhodné stromy* (Random Forest), čo je modifikácia metódy Bagging (Bootstrap AGGregation).

Metódy zloženého učenia Bagging a Boosting boli popísané v našej predchádzajúcej publikácii monografického charakteru „Strojové učenie v systémoch

spracovania informácií“ a väčšina klasických metód strojového učenia, okrem regresných metód a SVM, bola popísaná v našej publikácii - vysokoškolskej učebnici „Strojové učenie. Princípy a algoritmy.“ Teda predkladaný text dopĺňa už existujúcu učebnú literatúru pre študentov predmetu - kurzu „Strojové učenie“, ktorý sa ponúka na Katedre kybernetiky a umelej inteligencie, FEI Technickej univerzity v Košiciach študentom druhého stupňa v rámci dvoch študijných programov a to „Inteligentné systémy“ a „Hospodárska informatika.

Predkladaná vysokoškolská učebnica sa nesnaží byť vyčerpávajúcim textom z predkladanej oblasti a nerobí si ani nárok na úplnosť. Je vhodná pre tých, ktorí chcú získať základný prehľad oblasti, ale aj pre tých, ktorí majú hlbší záujem o prezentovanú problematiku a detaily nami uskutočnených experimentov v rámci metód, ktoré sú obsahom tejto publikácie. Na Katedre kybernetiky a umelej inteligencie sa venujeme tejto oblasti tak vo výskume, ako aj v pedagogickej činnosti, preto by vzniknutá publikácia mohla vyplniť medzeru, ktorá v tejto oblasti informatickej literatúry v našich podmienkach neustále trvá.

Naše poďakovanie na tomto mieste patrí aj obom recenzentom za starostlivé prečítanie rukopisu, a za cenné pripomienky a námety. Poďakovanie si taktiež zaslúži diplomant Ing. Jaroslav Štefaník, ktorý spolupracoval na riešení problému odhadu autorít konverzačného obsahu, a ktorého výsledky táto publikácia tiež uvádza, ako príklad použitia regresných metód.

Košice, september 2016

autor

Obsah

1 ÚVOD

1.1 Štatistické učenie

Pokúsime sa vysvetliť pojem štatistické učenie na príklade. Predstavte si, že máte ako štatistický konzultant nájsť riešenie problému „ako zvýšiť predajnosť nejakého produktu.“ Predpokladajme že máte k dispozícii dáta o predajnosti tohto konkrétneho produktu v predajnom reťazci, ktoré boli zozbierané v rámci stoviek predajní z celej krajiny. Spolu s týmito dátami máte k dispozícii aj dáta týkajúce sa reklamy toho istého produktu v rámci viacerých médií (sociálne siete, TV, časopisy a pod.). Ako štatistický poradca nemáte skúsenosti priamo s úspešnými technikami predaja, ale môžete predajcovi pomôcť nepriamo, keďže vidíte v dátach súvislosť medzi reklamou a predajnosťou. Môžete mu pomôcť tak, že pre neho z dát naučíte model, ktorý bude schopný predikovať predajnosť produktu na základe rozpočtu na reklamu v konkrétnych médiách. Tento model bude reprezentovaný funkciou, ktorej závislá premenná Y bude predstavovať predajnosť produktu a nezávislá premenná X bude predstavovať rozpočet na reklamu. Keďže môžete reklamovať vo viacerých médiách, musíte uvažovať viac nezávislých premenných napríklad $X1$ pre sociálne siete, $X2$ pre TV a $X3$ pre časopisy. Teda model hľadáte v tvare $Y = f(X) + \varepsilon$, resp. $Y = f(X1, X2, \ldots Xn) + \varepsilon$ (kde ε je chybová konštanta). Dá sa predpokladať, že táto závislosť je lineárna. Ale keby sme hľadali závislosť príjmu na počte rokov strávených vzdelávaním, tam by závislosť bola skôr nelineárna, keďže dlhé základné vzdelávanie ako vzdelávanie vo vyššom veku majú oveľa menší vplyv na príjem ako vzdelávanie v strednom veku, hlavne vysokoškolské, vrátane PhD. štúdia.

V princípe môžeme mať dva dôvody, prečo odhadovať funkciu f a to je kvôli predikcii a inferencii.

1. *Predikcia.* Chceme predikovať (odhadnúť) akú hodnotu bude mať závislá premenná Y v budúcnosti, keď hodnota nezávislej premennej X bude známa. Napríklad sledujeme dlhšiu dobu ako

sa mení spotreba elektrickej energie v sieti v priebehu dňa a chceme odhadnúť aká bude jej spotreba o dva dni o 15:00 hodine. Resp. na základe týchto pozorovaní vieme predikovať nižšiu spotrebu energie po 22:00 hodine večer a teda po tejto hodine znížime cenu energie, aby sme predišli preťaženiu siete cez deň v špičkových hodinách. Inými slovami motivujeme ľudí odoberať elektrickú energiu v nočných hodinách.

2. *Inferencia.* Chceme pochopiť, ako zmena nezávislej premennej X ovplyvňuje zmenu závislej premennej. V tomto prípade nemôže byť funkcia *f* čiernou skrinkou (ako pri predikcii), lebo práve jej charakter - presná forma nás zaujíma.

V rámci inferencie nás zaujímajú odpovede na nasledujúce otázky. Ktoré prediktory (nezávislé premenné *X1, ..., Xn*) majú reálne vplyv na zmenu odpovede (závislej premennej Y)? Vieme identifikovať tieto najdôležitejšie prediktory? Aká je konkrétna závislosť každého prediktora jednotlivo a odpovede? Závislosť odpovede na niektorých prediktoroch môže byť opačná, ako závislosť na ostatných prediktoroch. Môžu byť závislosti medzi *Y* a každým prediktorom adekvátne sumarizované použitím lineárnej rovnice, alebo je táto závislosť komplikovanejšia? Ak sa odvoláme na príklad na začiatku, môžeme túto otázku rozmeniť na drobné nasledujúcim spôsobom. Ktoré konkrétne média reálne zvyšujú predajnosť? Aký konkrétny nárast v predajnosti prináša tento konkrétny spôsob reklamy (napríklad v sociálnych sieťach)? Doteraz bolo rozoberaný vplyv reklamy na predajnosť. To je zjednodušený pohľad, lebo sú tu ďalšie premenné tohto problému, ktoré môžu vplývať na predajnosť (môžu inferovať vyššiu mieru predajnosti). To sú napríklad ceny, zľavy, kapacity skladov a podobne.

Existujú aj také problémy, ktoré sú zaujímavé tak z hľadiska inferencie ako aj z hľadiska predikcie. Napríklad, môžeme skúmať charakter vplyvu kriminality, zonácie, kvality vzduchu, blízkosti rieky, blízkosť škôl a obchodov, veľkosť domu alebo bytu či záhrady na cenu nehnuteľnosti,

čo spadá do paradigmy inferencie. Ale takisto nás môže zaujímať aj odhad ceny nehnuteľnosti a to, či je táto konkrétna cena nadhodnotená alebo podhodnotená. A to je predikčný problém.

Metódy, pri ktorých sa sústreďujeme na hľadanie parametrov známej funkcie sa nazývajú parametrické metódy. Existuje aj iný prístup, ktorý sa sústreďuje na nájdenie vhodných funkcií, teda ide o neparametrické metódy. Viac o tom je možné nájsť v [1].

1.2 Správnosť a interpretovateľnosť predikcie

V rámci štatistického učenia máme k dispozícii množstvo metód, o ktorých budeme ešte hovoriť v ďalších kapitolách tejto vysokoškolskej učebnice. Niektoré z nich sú menej flexibilné a viac reštrikčné, ako napríklad lineárna regresia, ktorá ponúka iba jeden tvar funkcie. Iné metódy sú zase flexibilnejšie a často presnejšie, ako napríklad metódy zloženej klasifikácie ako je bagging, boosting alebo náhodné lesy „Random forests“ (budú im venované niektoré z nasledujúcich podkapitol).

Ak je to tak, zdá sa logické rozhodnúť iba pre použitie tých druhých flexibilnejších a presnejších. Avšak nie je to také jednoduché, lebo metódy zloženej klasifikácie produkujú príliš komplikovaný výstup a sú ťažko interpretovateľné, aj keď na úrovni čiernej skrinky môžu byť presnejšie.

Teda vstupuje nám do toho aspekt interpretovateľnosti modelu, ktorý je často v protiklade so správnosťou modelu. Interpretovateľnosť je dôležitá hlavne pri úlohách, kde je cieľom *inferencia*. Napríklad, ak chceme porozumieť vzťahu medzi závislou a nezávislou premennou, potom nám je viac užitočný reštrikčný ale jednoduchý lineárny model.

Ak je cieľom predikcia, potom nie je interpretovateľnosť modelu tak dôležitá a bude lepšie použiť flexibilnejší model. Ale ani to nie je stále pravda, lebo niekedy práve menej flexibilná a jednoduchšia metóda bude presnejšia. To súvisí s preučením vysoko flexibilných metód.

1.3 Regresia verzus klasifikácia

Predpokladajme, že máme nové pozorovanie v oblasti, pre ktorú máme naučený model použitím niektorej metódy strojového učenia. Niekedy potrebujeme pre toto nové pozorovanie predikovať resp. odhadnúť (vypočítať) presnú hodnotu závislej premennej. V tom prípade ide o regresiu. Ak by sme to isté nové pozorovanie mali zaradiť do jednej z definovaných kategórií, resp. tried, potom by šlo o klasifikáciu.

Môžeme sa na to pozrieť aj s iného uhlu pohľadu. Premenné, resp. atribúty môžu byť vo všeobecnosti kvantitatívne alebo kvalitatívne. Kvantitatívne premenné sú spravidla numerické hodnoty ako napríklad vek, príjem či cena. Na druhej strane kvalitatívne (kategorické) premenné (atribúty) nadobúdajú jednu hodnotu z viacerých preddefinovaných ako napríklad (muž, žena), (malý, stredný, veľký) alebo (áno, nie). Problémy s kvantitatívnou odpoveďou zvykneme označovať ako regresné problémy, zatiaľ čo tie s kvalitatívnou (kategorickou) odpoveďou označujeme ako klasifikačné problémy.

Ale nie je to vždy také jednoznačné. Napríklad lineárna regresia sa typicky používa pri kvantitatívnej odpovedi, ale iný druh regresie – logistická regresia sa používa pri kvalitatívnej odpovedi, konkrétne pri dvojtriednej binárnej odpovedi, teda sa používa na riešenie klasifikačných problémov a je považovaná za klasifikačnú metódu. No v prípade, že by neurčovala triedu ale počítala pravdepodobnosť triedy, potom by mohla byť považovaná za regresnú metódu. Aj ďalšie metódy strojového učenia ako KNN - K najbližších susedov alebo boosting môžu byť s úspechom použité pri hľadaní tak kvantitatívnej ako aj kvalitatívnej odpovede, teda pri riešení tak regresných ako aj klasifikačných problémov. Spomenuté metódy budú rozoberané v ďalších kapitolách.

2 REGRESNÁ ANALÝZA

V prvom rade je potrebné si ujasniť, že regresnú analýzu môžeme v rámci strojového učenia s úspechom použiť tak na predikciu ako aj na klasifikáciu. Regresná analýza sa v prvom rade sústreďuje na nájdenie závislosti javov v tvare nejakej funkcie na základe sledovania týchto javov v určitom časovom okne. Táto závislosť sa potom môže v budúcnosti použiť na predikciu hodnoty závislej premennej, ktorá reprezentuje závislý jav. To platí pre lineárnu a nelineárnu regresiu, ktoré sa s úspechom môžu použiť na odhad, estimáciu konkrétnej hodnoty premennej sledovaného javu. Na druhej strane logistická regresia môže poslúžiť na klasifikáciu nejakého objektu do jednej z dvoch tried. Obidvom týmto prípadom bude venovaná osobitná podkapitola z nasledujúcich.

V prírode ako aj v matematike, ktorá ju modeluje, nevznikne žiaden jav bez toho aby nebol vo vzájomnom vzťahu s inými javmi. Preto pri štúdiu viacrozmerných súborov pozorovaní je dôležité poznať závislosť dvoch alebo viacerých veličín. Na hľadanie takej závislosti sa s úspechom používajú metódy regresnej a korelačnej analýzy [2]. V rámci otvorenej kapitoly sa budeme zaoberať regresnou analýzou a to konkrétne nasledujúcimi typmi závislostí medzi premennými: funkčnou závislosťou, voľnou závislosťou a štatistickou závislosťou.

Pod *funkčnou závislosťou* rozumieme prípad, keď každej hodnote jednej veličiny odpovedá práve jedna hodnota pre každú z ostatných veličín, na ktorých je pôvodná veličina závislá. Rovnako to platí aj naopak. V prípade, ak hodnotám jednej veličiny zodpovedajú rôzne hodnoty nejakej inej veličiny, hovoríme o *voľnej závislosti*. *Štatistická závislosť* predstavuje voľnú závislosť medzi kvantitatívnymi štatistickými znakmi. Práve závislosťou týchto kvantitatívnych znakov sa zaoberá regresná analýza, pričom to môže byť závislosť kvantitatívneho znaku na jednom kvantitatívnom znaku alebo viacerých znakov [2]. Na základe spomenutých závislostí kvantitatívnych znakov delíme regresiu na jednorozmernú regresiu (jednoduchá regresia) a viacrozmernú regresiu (viacnásobná regresia).

2.1 Jednorozmerná a viacrozmerná regresia

O jednorozmernej regresii hovoríme v prípade, keď sledujeme závislosť dvoch kvantitatívnych znakov *X* a *Y* (viď Obr.1).

	Prax [mes]	Vek
2	14	21
3	4	22
4	8	23
5	12	23
6	12	24
7	2	25
8	10	27
9	4	29
10	8	32
11	8	34

Obr.1 Príklad závislosti jedného kvantitatívneho znaku na jednom kvantitatívnom znaku, ktorá vedie k jednorozmernej regresii

Ak sledujeme závislosť znaku (napríklad veličiny či atribútu) *Y* na 2,3 ...,n kvantitatívnych znakoch (veličinách, premenných), vtedy ide o viacrozmernú regresiu (viď. Obr.2).

Veličinu *X* (na Obr.2 je ich viac – sú štyri) nazývame nezávislou premennou, ktorá sa v niektorých zdrojoch nazýva aj vysvetľujúcou premennou. Závislou premennou, alebo tiež vysvetľovanou premennou, je veličina *Y*. Dôležité je uvedomiť si, že regresná úloha spočíva v zistení formy závislosti *Y* na *X* a jej následnom matematickom vyjadrení funkciou. Túto funkciu nazývame tiež regresná funkcia. Určenie stupňa sily, akým sa daná závislosť prejavuje, je už úlohou korelačnej analýzy [2].

Predpokladajme, že *X* je pevná a *Y* je náhodná premenná, a nech podmienená stredná hodnota náhodnej premennej *Y* je funkciou, teda závisí od premennej *x* a parametrov a_0, a_1, ..., a_K. Potom platí (1):

$$E(Y|x) = \varphi = \varphi(\mathrm{x}, a_0, a_1, \dots a_k) \qquad (1)$$

Pričom *E* je symbol strednej hodnoty, | znamená splnenie

podmienok za týmto znakom. Funkciu φ nazývame regresnou funkciou a predpokladáme, že jej tvar je nám známy. Parametre $a_0, a_1, \dots a_k$ voláme tiež regresné parametre, respektíve regresné koeficienty [3]. Základný jednoduchý regresný model tak môžeme zapísať v tvare (2):

$$y_i = f(x_i) + E_i, \tag{2}$$

kde x_i, y_i sú hodnoty premennej (znaku) *X,* respektíve *Y* a E_i je náhodná chyba, ktorú môžeme tiež nazvať ako reziduálna zložka. f predstavuje funkciu, ktorú nazývame regresnou, a ktorá predstavuje deterministickú zložku. Predpokladá sa, že náhodné chyby sú náhodné veličiny, ktoré majú normálne rozdelenie, pričom ich smerodajná odchýlka je $\sigma > 0$ a majú nulovú strednú hodnotu. Teda platí nasledujúci vzťah (3):

$$\mathcal{E}_i \sim norm(0, \sigma),\ i = 1,2,3, \dots, n. \tag{3}$$

Definujme pojem *rezíduum* ako rozdiel medzi pozorovanou, respektíve empirickou hodnotou označovanou y_i a teoretickou hodnotou, teda hodnotou vypočítanou, ktorú označujeme $\hat{y}_i$. Rezíduum predstavuje bodový odhad náhodnej chyby $\mathcal{E}_i$. Označujeme ho e_i, a vypočíta sa podľa vzťahu (4):

$$e_i = y_i - \hat{y}_i. \tag{4}$$

S týmto pojmom súvisí aj pojem reziduálny súčet štvorcov, ktorý sa využíva pri metóde najmenších štvorcov (bude jej venovaná podkapitola) a je to najlepší spôsob, ako optimalizovať hľadanie regresnej funkcie. Reziduálny súčet štvorcov sa označuje SSE (z anglického Sum of Square Errors) a vypočítame ho nasledovne:

$$\sum_{i=1}^{n} e_i^2. \tag{5}$$

Ako bolo spomenuté viacrozmerná regresia skúma závislosť medzi dvoma a viacerými premennými a môže byť podľa [4] reprezentovaná nasledovným modelom:

$$y_i = a_0 + a_1 x_{i1} + a_2 x_{i2} + \cdots + a_n x_{in} + \mathcal{E}_i,$$

(6)

kde:

y_i ... je hodnota závislej premennej,

x_{ij} ... je hodnota j-tej nezávislej premennej,

a_j ... je neznámy regresný koeficient a

ξ_i ... je náhodná chyba.

Všetky doterajšie vzťahy boli prebraté z [2].

domác-nosť	čistý mesačný príjem na 1 člena	počet členov domácnosti	mesačné výdavky na priem. tovar	mesačné výdavky na služby	mesačné výdavky na potraviny
Mnohonásobná regresia					
	y	x1	x2	x3	x4
1	6761	3	1965	1431	4519
2	6099	1	1164	684	3124
3	11029	6	5592	1988	5623
4	11785	4	6486	1426	3135
5	8198	5	2757	1622	3195
6	6641	1	1264	1381	3154
7	8618	4	2771	2300	2510
8	9349	6	1787	4379	6354
9	10836	3	3991	2172	3467
10	5891	1	1184	852	2956
11	10245	4	2312	3444	3348
12	13990	5	6076	4661	4648
13	11779	6	4135	3171	5556
14	11945	4	3734	4760	4456
15	12771	2	6308	2242	3293

Obr.2 Príklad závislosti jedného atribútu na viacerých premenných, ktorá vedie ku viacrozmernej regresii

Okrem delenia na jednorozmernú a viacrozmernú rozdeľujeme regresiu podľa toho, či je modelovaná lineárnou alebo nelineárnou funkciou premenných. Tak jednorozmerná ako aj viacrozmerná regresia môže byť lineárna ale aj nelineárna. Máme teda štyri možnosti.

2.2 Lineárna a nelineárna regresia

Regresnú funkciu nazývame lineárnou, ak je lineárnou funkciou neznámych premenných. Príkladom lineárnych regresných funkcií sú nasledujúce funkcie [5]:

- $y = a + bx$ (lineárna regresia),
- $y = a + bx + cx^2$ (kvadratická regresia),
- $y = a + bx + \ldots + b_r x^r$ (polynomiálna regresia),
- $y = a + \frac{b}{x}$ (hyperbolická regresia).

Vo všeobecnosti najjednoduchšou, teda lineárnou regresnou funkciou je funkcia:

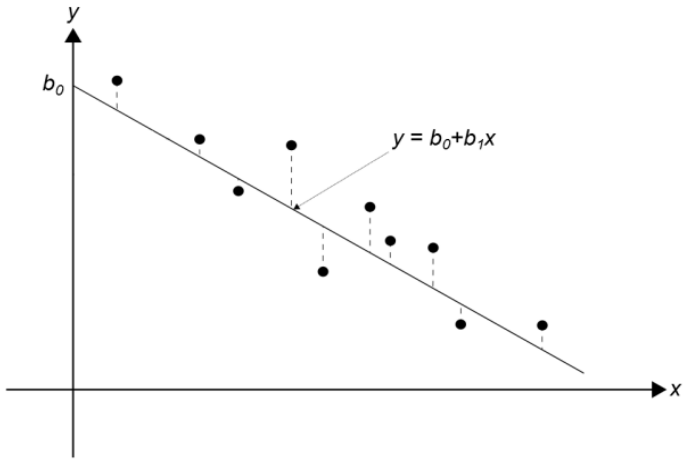
$$f(x) = a_0 + a_1.x \tag{7}$$

Hodnoty a_0 a a_1 voláme parametre regresnej priamky, ktoré v niektorých zdrojoch zvyknú byť označované aj ako regresné koeficienty. Regresný koeficient a_0 predstavuje priesečník regresnej priamky s osou o_y. Tento koeficient sa tiež nazýva úrovňová konštanta. Koeficient a_1 vyjadruje sklon priamky k osi o_x, - smernicu regresnej priamky. Pokiaľ s rastom nezávislej premennej *X* dochádza v priemere tiež k rastu hodnôt závislej premennej *Y*, označujeme tento sklon priamky ako kladný. Spomínanú závislosť tak označujeme ako pozitívnu, respektíve priamu závislosť. Ak je regresný koeficient a_1 záporný, tak pri raste hodnôt nezávislej premennej dochádza v priemere k poklesu hodnôt závislej premennej, a v tomto prípade hovoríme o negatívnej alebo nepriamej závislosti [2]. Presne takúto nepriamu závislosť ilustruje v Obr.3 konkrétne vo forme jednorozmernej lineárnej regresie.

Vo všeobecnosti sú hodnoty a_0 a a_1 označované ako nezávislé parametre základného súboru. Z tohto tvrdenia vyplýva, že je nutné ich odhadnúť, a to pomocou *n* nezávislých pozorovaní dvoch veličín X a Y. Výsledkom týchto pozorovaní sú usporiadané dvojice $(x_1, y_1), \ldots, (x_n, y_n)$, kde y_i predstavujú hodnoty závislej premennej Y a x_i sú hodnoty nezávislej premennej X. Tieto dvojice (pozorovania) je možné znázorniť bodmi

v dvojrozmernom priestore (viď Obr.3).

Geometrickým znázornením súboru spomínaných dvojíc hodnôt je bodový graf. Ide o geometrické zobrazenie v rovine, kde na zvislú os o_y nanášame príslušné hodnoty závislej premennej a na vodorovnú os o_x sú nanášané hodnoty nezávislej premennej. Výsledkom tohto postupu je geometrické znázornenie n bodov v rovine, pričom z ich vzájomnej polohy je možné posúdiť regresnú závislosť znakov *X* a *Y.*



Obr.3 Príklad jednorozmernej lineárnej regresie s nepriamou (negatívnou závislosťou)

Hlavným cieľom jednoduchej lineárnej regresie je danými bodmi preložiť priamku, teda nájsť takú lineárnu regresnú funkciu, ktorá najlepšie vystihuje a charakterizuje polohu daných *n* bodov. Jednoduchý lineárny regresný model v rámci dvojrozmerného priestoru môžeme teda zapísať v nasledujúcom tvare:

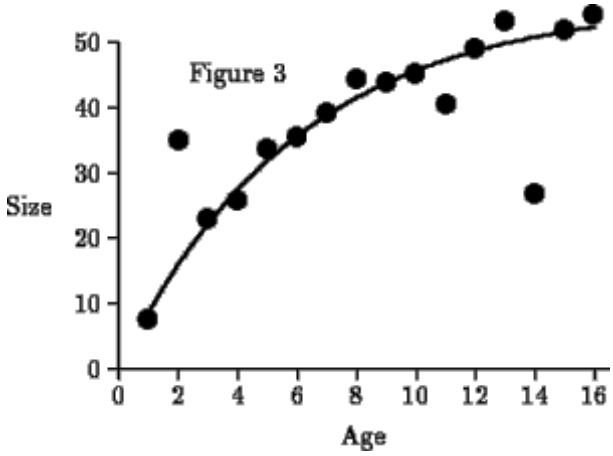
$$y_i = a_0 + a_1.x_i + \mathcal{E}_I, i = 1,2,3 \dots n, \quad (8)$$

pričom jeho vyrovnávajúcu regresnú priamku zapíšeme v tvare:

$$\hat{y}_i = \hat{a}_0 + \hat{a}_1.x_i, \quad (9)$$

kde $\hat{y}_i$ je teoretická, teda vypočítaná hodnota premennej Y a $\hat{a}_0$, $\hat{a}_1$ sú bodové odhady parametrov a_0 a a_1. Z pozorovaných dát sa tieto bodové odhady získavajú najčastejšie Metódou najmenších štvorcov [2], o ktorej sa ešte v tejto práci bude pojednávať.

Niekedy máme k dispozícii dáta, ktoré nie je možné modelovať lineárnou priamkou a musíme sa uchýliť ku nelineárnej regresii. Na Obr.4 je znázornený model nelineárnej regresie. O tomto type regresie hovoríme v prípade, ak bodovým grafom prekladáme krivku, ktorej rovnica nie je lineárnou vzhľadom na neznáme parametre. Toto je kľúčové si uvedomiť. V rovnici môžu byť aj druhé mocniny, ale ak sa netýkajú hľadaných parametrov, potom pôjde o lineárnu regresiu. V takomto prípade môžu niekedy vzniknúť problémy pri hľadaní minima metódou najmenších štvorcov. Preto sa snažíme takýto typ regresie transformovať na lineárnu funkciu. Pri niektorých funkciách stačí použiť jednoduchú transformáciu premenných [5].



Obr.4 Model nelineárnej regresie

Vo všeobecnosti môžeme povedať, že lineárna regresia má vždy jednoznačné riešenie, čo nie je prípad nelineárnej regresie. Ak sa stretneme so závislosťou nelineárneho charakteru, je potrebné zvoliť vhodnú nelineárnu funkciu. Odhad parametrov pri nelineárnych modeloch je časovo náročný. Možné príklady nelineárnej funkcií podľa [2]:

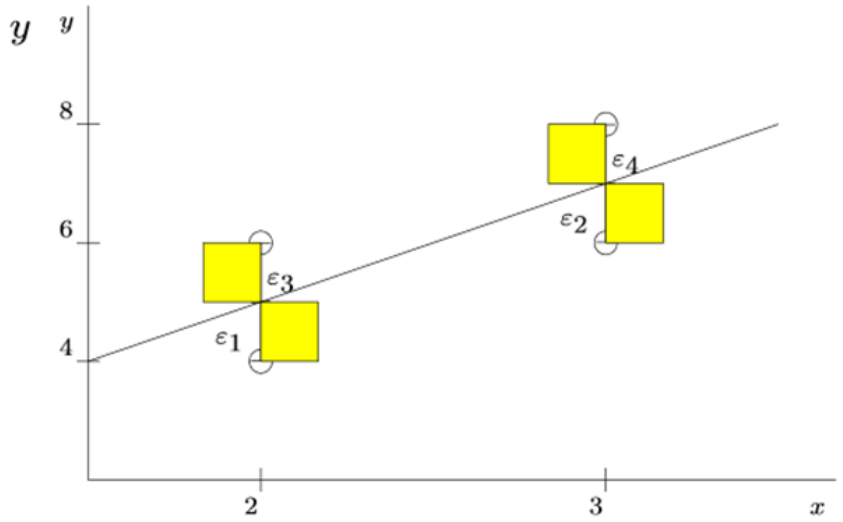
- regresná parabola 2. stupňa
  $f(x) = a_{0+}a_1.x + a_2.x^2$,
- regresná mocninová funkcia
  $f(x) = a_0.x^{a_1}$,
- regresná exponenciálna funkcia
  $f(x) = a_0.a_1^x$,
- regresná hyperbola
  $f(x) = a_0 + \frac{a_1}{X}$,
- regresná logaritmická funkcia
  $f(x) = a_{0+}a_1.lnx$.

Výber najlepšieho typu regresnej funkcie nie je vždy jasný od začiatku, preto sa za najlepší považuje podľa [6] ten, ktorý je najlogickejší, pri ktorom sa rozdelenie reziduálnych odchýlok čo najviac približuje k normálnemu rozdeleniu, ktoré zabezpečí to, že rozdelenie reziduálnych odchýlok bude mať približne rovnakú variabilitu, a pri ktorom budú reziduálne odchýlky čo najmenšie a napokon, ktorý je najjednoduchšou krivkou a vykazuje najväčšiu tesnosť závislostí.

2.3 Metóda najmenších štvorcov

Aproximáciu funkcie prostredníctvom interpolačných polynómov používame v prípade vysokej presnosti nameraných hodnôt. Väčšinou v praxi sú tieto hodnoty získané experimentálne teda môžu byť zaťažené oveľa väčšími chybami, ako sú chyby spôsobené zaokrúhľovaním. Preto je potrebné použiť optimalizačnú metódu a ako najvhodnejšia je matematicko-štatistická metóda najmenších štvorcov. Táto metóda sa môže použiť pri oboch typoch regresií, tak lineárnej ako aj nelineárnej. Pomáha pri aproximácii nameraných dát priamkou. Táto metóda minimalizuje chyby, ktoré predstavujú rozdiely medzi vypočítanými hodnotami závislej premennej (pomocou danej regresnej funkcie) a empirickými nameranými hodnotami tejto premennej. Tieto rozdiely, nazvané *reziduálne odchýlky*, môžu mať kladné aj záporné hodnoty a práve z toho dôvodu sa tieto hodnoty najprv umocňujú na druhú a potom sumujú, čo ilustruje Obr.5.

Využitie a aplikácia tejto metódy je rozšírená v rôznych vedných oboroch, ako štatistika, ekonómia, geodézia a pod. Viac o metóde SSE je možné nájsť v [2].



Obr.5 Ilustrácia metódy najmenších štvorcov [2]

Na grafickom zobrazení metódy najmenších štvorcov (viď. Obr.5 predstavuje os *x* hodnoty nezávislej premennej a os *y* hodnoty závislej premennej. Samotná priamka predstavuje regresnú funkciu. Prázdne krúžky reprezentujú jednotlivé pozorovania (trénovacie príklady), pričom žlté štvorce reprezentujú štvorce odchýlok pozorovanej a vypočítanej hodnoty závislej premennej.

Uvažujme nasledovný lineárny regresný model:

$$y_i = a_0 + a_1 . x_i + \mathcal{E}_I, i = 1,2,3 \dots n, \qquad (10)$$

a jemu odpovedajúcu vyrovnávajúcu regresnú priamku v tvare:

$$\hat{y}_i = \hat{a}_0 + \hat{a}_1 . x_i, \qquad (11)$$

kde $\hat{y}_i$ je teoretická, teda vypočítaná hodnota premennej *Y* a $\hat{a}_0$ a $\hat{a}_1$ sú bodové odhady parametrov a_0, a_1. Tieto

bodové odhady sa z pozorovaných dát získavajú Metódou najmenších štvorcov minimalizáciou súčtov štvorcov (SSE z anglického Sum of Square Errors) reziduálnych odchýlok empirických hodnôt y_i od teoretických hodnôt závislej premennej $\hat{y}_i$ nadobúdal svoju minimálnu hodnotu. SSE sa počíta podľa vzťahu:

$$SSE(\hat{a}_0, \hat{a}_1) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{a}_0 - \hat{a}_1 x_i)^2 \quad (12)$$

Reziduálny súčet štvorcov je dôležitou veličinou, nakoľko sa používa na štatistický odhad rozptylu bodov $[x_i, y_i]$ okolo teoretickej regresnej krivky. Z matematickej analýzy vyplýva, že minimum uvedenej funkcie dvoch premenných zistíme tak, že položíme rovné nule jej prvé parciálne derivácie podľa $\hat{a}_0$ a $\hat{a}_1$:

$$\frac{\partial SSE}{\partial \hat{a}_0} = 0 \; \frac{\partial SSE}{\partial \hat{a}_1} = 0 \quad (13)$$

Po dosadení do predchádzajúceho vzorca dostaneme nasledujúcu sústavu rovníc (14) a (15):

$$\frac{\partial}{\partial \hat{a}_0}\left(\sum_{i=1}^{n} (y_i - \hat{a}_0 - \hat{a}_1 x_i)^2\right) = \sum_{i=1}^{n} 2(y_i - \hat{a}_0 - \hat{a}_1 x_i).(-1) = 0$$

$$\frac{\partial}{\partial \hat{a}_1}\left(\sum_{i=1}^{n} (y_i - \hat{a}_0 - \hat{a}_1 x_i)^2\right) = \sum_{i=1}^{n} 2(y_i - \hat{a}_0 - \hat{a}_1 x_i).(-x_i) = 0$$

(14) a (15)

Po úprave predchádzajúcej sústavy rovníc dostaneme nasledovnú sústavu dvoch lineárnych rovníc (16) a (17) s dvoma neznámymi. Takúto sústavu nazývame aj sústavou normálnych rovníc:

$$n \,.\, \hat{a}_0 + \hat{a}_1 . \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i, \quad (16)$$

$$\hat{a}_0 . \sum_{i=1}^{n} x_i + \hat{a}_1 . \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i \, . \, y_{i.} \tag{17}$$

Vyriešením tejto sústavy lineárnych rovníc dostaneme výsledné vzorce (18) a (19) pre výpočet bodových odhadov $\hat{a}_0$ a $\hat{a}_1$, čím vlastne dostaneme presnú polohu samotnej regresnej priamky v dvojrozmernom priestore:

$$\hat{a}_1 = \frac{n . \sum_{i=1}^{n} x_i \, . \, y_i - \sum_{i=1}^{n} x_i \, . \sum_{i=1}^{n} y_i}{n . \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}, \tag{18}$$

$$\hat{a}_0 = \frac{\sum_{i=1}^{n} y_i - \hat{a}_1 . \sum_{i=1}^{n} x_i}{n} = \bar{y} - \hat{a}_1 . \bar{x}. \tag{19}$$

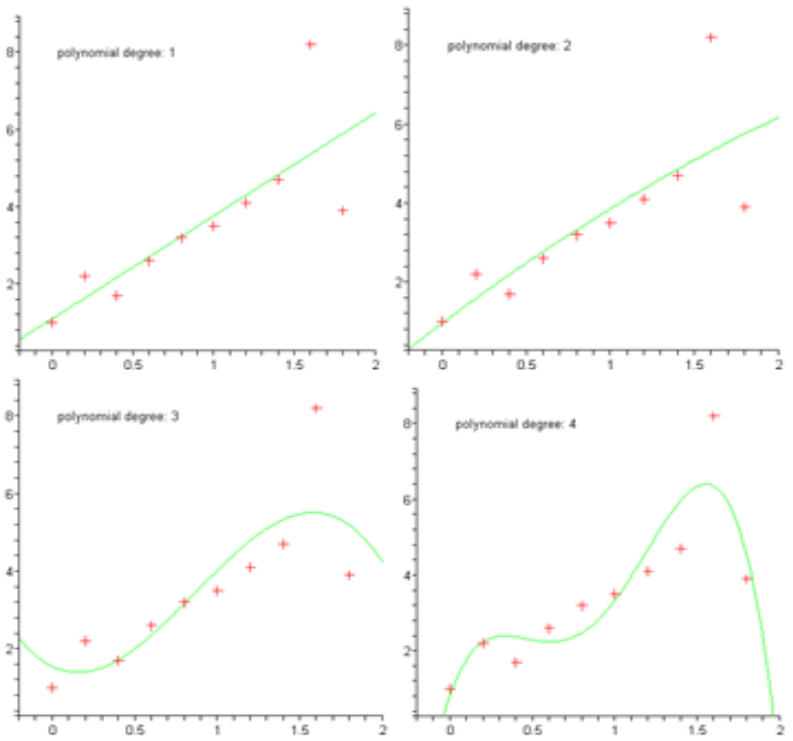
Doteraz sme sa zaoberali aproximáciou bodového grafu priamkou, resp. polynómom prvého stupňa. Iná možnosť je aproximácia parabolou, kedy sa body preložia kvadratickou funkciou, resp. polynómom druhého stupňa (viď (20)):

$$y_i = a_0 + a_1 . x_i + a_2 x_i^2 \tag{20}$$

V takom prípade parametre a_0, a_1, a_3 získame riešením nasledovnej sústavy rovníc:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i \, . \, x_i \\ \sum y_i . \, x_i^2 \end{bmatrix} \tag{21}$$

Obr.6 ilustruje aproximáciu zadaných hodnôt polynómami stupňa s=1,2,3 a 4.



Obr. 6 Ukážka aproximácie zadaných bodov polynómami stupňa s = 1,..,4

2.4 Implementácie (ne)lineárnej regresie

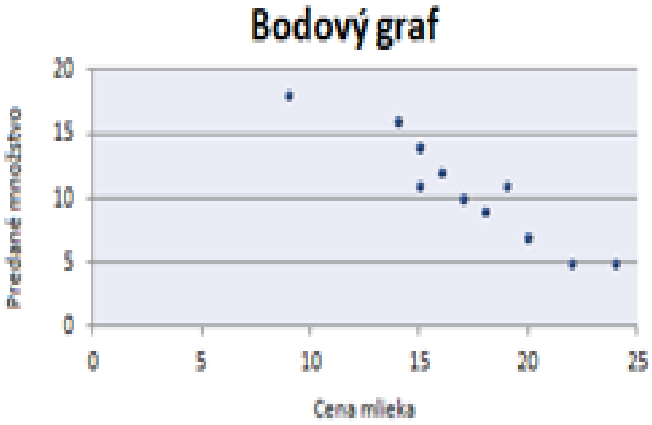
2.4.1 Predaj mlieka

Na tomto veľmi jednoduchom príklade je ilustrovaná náročnosť realizácie výpočtov hoci jednoduchej lineárnej regresie [7]. V Tab.1 sú uvedené pozorované údaje o množstve predaného mlieka a jeho cene, ktorým odpovedá bodový graf na Obr.7. Tento obrázok naznačuje, že vzťah medzi závislou premennou *Y* - množstvom predaného mlieka, a nezávislou premennou *X* - cenou mlieka je lineárny. Je zrejmé, že s narastajúcou cenou mlieka klesá jeho predaj, teda ide o nepriamu (negatívnu) lineárnu závislosť. Nech je našou úlohou zistiť, o koľko klesne predaj mlieka s nárastom jednotkovej ceny. To vieme zistiť, ak bodmi na Obr.7 preložíme priamku

a vypočítame jej smernicu. Keďže počítame jednoduchú lineárnu regresiu, zapíšeme ju v tvare $y_i=b_0+b_1 \cdot x_i$, kde x je vysvetľujúca premenná, y je vysvetľovaná premenná, b_0 je absolútny člen a b_1 je smernica priamky. Je potrebné vykonať podporné výpočty, ktoré sú zobrazené v Tab.2 a následne metódou najmenších štvorcov odhadnúť oba parametre b_0 a b_1.

Tab.1 Pozorované množstvo predaného mlieka a jeho cena

Týždeň	Predaj za týždeň	Cena v predajni
1	9	18
2	5	24
3	18	9
4	14	15
5	10	17
6	12	16
7	7	20
8	11	15
9	5	22
10	16	14
11	14	15
12	11	19



Obr.7 Bodový graf podľa údajov z Tab.1

Pre odhad parametrov b_0 a b_1 použijeme nasledujúce vzorce (22) a (23):

$$\boldsymbol{b_1} = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n\sum_{i=1}^{n} x^2 - (\sum_{i=1}^{n} x_i)^2} = \frac{12 \text{ x } 2074 - 204 \text{ x } 32}{12 \text{ x } 3642 - 204^2} = -0{,}977$$

$$\boldsymbol{b_0} = \bar{y} - b_1\bar{x} = 11 - (-0{,}977)\text{x } 17 = 27{,}609$$

(22) a (23)

Výsledná rovnica regresnej priamky je $\hat{y}_i = 27{,}609\text{-}0{,}977x_i$.

2.4.2 Odhad autorít sociálnych sietí

Tento problém je omnoho zložitejší. V prvom kroku bolo potrebné vyextrahovať údaje z portálu „sme.sk“, uložiť do pamäte programu a následne uložiť do databázy, ktorá by umožňovala vykonávanie ďalších potrebných operácií. Vyextrahované údaje obsahovali hodnoty nasledovných premenných pre každého účastníka webovej diskusie: priemerné hodnotenie prispievateľa, „karma“, priemerný počet znakov príspevku, priemerná poloha v strome,

priemer reakcií na príspevok diskutéra a samotný počet príspevkov. Ukážka týchto údajov sa nachádza v Tab.3. Tieto údaje boli rozšírené o závislú premennú *Y*, ktorá je nazvaná hodnotenie a reprezentuje názor experta alebo názor ostatných účastníkov diskusie na to nakoľko je hodnotený účastník diskusie autoritou (viď Tab.3). Hodnotenie ostatných účastníkov diskusie bola nazvaná „múdrosť davu“.

Tab.2 Podporné výpočty regresnej priamky podľa (22) a (23).

	yi	xi	xiyi	Yi^2	xi^2
n=12	9	18	162	81	324
$\bar{y}$ =11	5	24	120	25	576
X prie.= 17	18	9	162	324	81
	14	15	210	196	225
	10	17	170	100	289
	12	16	192	144	256
	7	20	140	49	400
	11	15	165	121	225
	5	22	110	25	484
	16	14	224	256	196
	14	15	210	196	225
	11	19	209	121	361
SPOLU	**132**	**204**	**2074**	**1638**	**3642**

V prípade ohodnotenia expertom bol každý jeden príspevok prečítaný a zvolený expert mu pridelil výsledné hodnotenie autoritatívnosti v intervale od 1 do 100. V prípade, ak mal jeden diskutér viac príspevkov, jeho výsledná autorita sa vypočítala ako podiel súčtu všetkých jeho hodnotení autoritatívnosti a počtu príspevkov v diskusii.

Čo sa týka ohodnotenia podľa múdrosti davu, dôraz bol

kladený na názory veľkého počtu ľudí, ktorí sa navzájom neovplyvňovali. Základná myšlienka tohto prístupu je, že keď sú správne agregované aj nedokonalé hodnotenia jednotlivcov, vieme získať výsledok, ktorý je lepší ako najlepší získaný odhad. Toto je podobný princíp, na akom pracuje aj zložená klasifikácia, resp. učenie súborom metód, ktorým je venovaná 5. kapitola.

Tab.3 Ilustrácia dát (hodnôt nezávislých premenných pre jednotlivých účastníkov webovej diskusie, pričom každému účastníkovi je venovaný jeden riadok) rozšírených o závislú premennú Hodnotenie

Nickname	Hodnotenie	Karma	Počet znakov	Poloha v strome	Priemer reakcií	Počet príspevkov	Hodnotenie
Peter Z	60	108	26	0	1	1	60
V12	80	182	220	2	0,5	2	100
fer	80	171	548,5	3	2,5	2	90
sandokan555	80	162	57,5	4	0,5	2	90
Peter_S	50	99	112,5	6	0	2	50
Darkman	80	167	117	3	0	1	90
Jesse Pinkman	40	74	210,5	1,5	1,5	2	40
rmm	60	108	22	1	1	1	60
ViViD	80	180	306	3,333333333	1	3	90
visnait	70	130	206	3	0	1	80
Axel Wers	80	190	146	3,5	1,5	2	100
gott	80	145	171	4	1	1	80
M@jo	80	169	402	3,666666667	0,333333333	3	90
piajjnmo	80	172	132,5	2	1	2	90
kekonomov_duch	80	170	34	1	2,5	2	90

Odhad autority bol realizovaný prostredníctvom lineárnej, polynomiálnej a tiež nelineárnej regresie, pričom všetky tri funkcie boli učené na základe hodnotenia experta a aj múdrosti davu. S polynomiálnou regresiou je možné prostredníctvom transformácie pracovať ako s lineárnou regresiou. Teda celkovo bolo naučených 6 funkcií viacnásobnou regresiou, keďže sme mali k dispozícii nie jednu ale šesť nezávislých premenných. Funkcie boli naučené v programovacom prostredí Matlab, ktoré disponuje okrem iného aj metódami regresnej analýzy. Pred samotným výpočtom bolo potrebné zadefinovať učený tvar funkcie pre každú z troch typov regresie. Tieto tri navrhnuté tvary funkcií, ktorých parametre boli počítané, sú uvedené v rovniciach (24) až (26).

Lineárna regresia:

$$y = c1 * H + c2 * K + c3 * PR + c4 * PZ + c5 * PS + c6 * PP \quad (24)$$

Polynomiálna regresia:

$$y = c1 * H^3 + c2 * K^2 + c3 * PR + c4 * PZ + c5 * PS + c6 * PP \quad (25)$$

Nelineárna regresia:

$$y = c1 * H^{c7} + c2 * K^{c8} + c3 * PR^{c9} + c4 * PZ^{c10} + c5 * PS^{c11} + c6 * PP^{c12}, \quad (26)$$

kde: *H* je hodnotenie, *K* je karma, *PR* je počet reakcií, *PZ* je počet znakov, *PS* je poloha v strome, *PP* je počet príspevkov. Výsledné naučené funkcie sú uvedené v Tab.4.

Výsledkom učenia je šesť funkcií, z ktorých je potrebné vybrať tú najrelevantnejšiu, ktorá by dokázala s čo najlepšou presnosťou určiť autoritu diskutéra. Preto na testovacej vzorke bola testovaná úspešnosť všetkých modelov. Testovanie bolo založené na výpočte priemeru odchýlok. Tieto odchýlky boli vypočítané z rozdielov medzi stanovenými hodnotami závislej premennej (či už expertom alebo davom) a hodnotami vypočítanými pomocou naučených závislostí. Do úvahy boli braté absolútne hodnoty odchýlok. Nakoniec boli všetky získané hodnoty spriemerované, aby sme dostali jednu hodnotu pre každý druh regresie. Priemerné hodnoty odchýlok sú prezentované v Tab.5, z ktorej vyplýva, že najpresnejšia je lineárna regresia učená z múdrosti davu.

Tento naučený model priniesol lepšie výsledky ako metóda ohodnotenia podľa názoru experta, ktorý hodnotí príspevok diskutéra podľa svojho subjektívneho názoru, zatiaľ čo pri modeli naučenom z múdrosti davu sa berie do úvahy hodnotenie početnejšej skupiny používateľov. Viac v [8].

Tab.4 Nájdené závislosti hodnoty autority

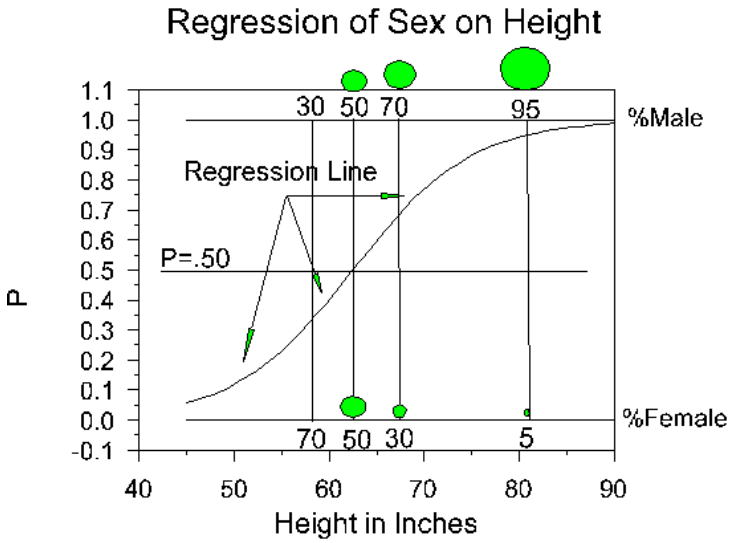
Lineárna regresia EXPERT
0,4383 * H + 0,0746 * K + (-3,4386) * PR + 0,0281 * PZ + (-2,1932) * PS + 8,0102 * PP
Lineárna regresia DAV
0,43854 * H + 0.325 * K + (-0.0853) * PR + 0,002 * PZ + (-0,2928) * PS + 1,0728 * PP
Polynomiálna regresia EXPERT
0,0001 * (H)^3 + 0,0004 * (K)^2 + (-2,0557) * PR + 0,0303 * PZ + (-1,5539) * PS + 12,1589 * PP
Polynomiálna regresia DAV
0,0001 * (H)^3 + 0,0009 * (K)^2 + 1,9875 * PR + 0,0043 * PZ + 0,7473 * PS + 6,7507 * PP
Nelineárna regresia EXPERT
0.0382*(H)^1.7192+ (-0.3295)* (K)^0.9590+ (-0.6269)*(PR)^ 3.2394+ 0.4470*(PZ)^0.6810+ 0.1825*(PS)^0.0001+ 20.2509*(PP)^ 0.2977
Nelineárna regresia DAV
0.0185*(H)^1.8135+ 141.5704*(K)^(-78.3900)+ (-0.5562)*(PR)^ 0.0001+ 0.0018 *(PZ)^1.0457+ (-0.0011) *(PS)^3.7717+ 37.6642*(PP)^0.0038

Tab.5 Priemerné hodnoty odchýlok pre každý typ regresie

Trénovacia množina		
Typ regresie	**Hodnotenie podľa**	**Priemer odchýlok**
Lineárna regresia	**EXPERT**	**17,34888586**
	MÚDROSŤ DAVU	**3,29984747**
Polynomiálna regresia	**EXPERT**	**24,01229093**
	MÚDROSŤ DAVU	**8,791176793**
Nelineárna regresia	**EXPERT**	**18,11310462**
	MÚDROSŤ DAVU	**6,56185723**

3 LOGISTICKÁ REGRESIA

Predpokladajme, že máme k dispozícii dátovú množinu, v ktorej odpoveď padne do jednej z dvoch kategórií Muž alebo Žena v závislosti na výške osoby. Logistická regresia nemodeluje presne túto odpoveď ale skôr pravdepodobnosť, že odpoveď padne do jednej z týchto dvoch kategórií čo ilustruje Obr.8. Teda modelujeme nasledovné pravdepodobnosti: *Pr(pohlavie = Žena | výška), Pr(pohlavie = Muž | výška).*



Obr.8 Model závislosti pravdepodobnosti pohlavia na výške osoby založený na logistickej regresii [9]

Z Obr.8 je jasné, že keď je osoba veľmi maličká, takmer isto je to žena. Na druhej strane, keď výška osoby presiahne určitú hodnotu, takmer isto je to muž.

Trochu si to zjednodušíme a budeme pracovať s inými hodnotami závislej premennej a to Áno a Nie namiesto Muž a Žena, alebo inak povedané Áno - je to muž alebo Nie – nie je to muž, je to žena. Teda nás bude zaujímať pravdepodobnosť, že ide o muža *Pr(muž = Áno | výška)* v závislosti na výške osoby. Vieme, že táto

pravdepodobnosť je číslo medzi 0 a 1. Ak chceme zostaviť predikčný model, musíme stanoviť hranicu tejto pravdepodobnosti, za ktorou budeme osobu považovať za muža, napr. *Pr(muž = Áno | výška) > 0.5*. Ak chceme byť striktnejší, resp. presnejší, môžeme zvoliť vyššiu hodnotu hranice - prahu, napr. *Pr(muž = Áno | výška) > 0.7*. Vo všeobecnosti by sme mohli povedať, že potrebujeme modelovať pravdepodobnosť závislej premennej *Y* na nezávislej premennej X, teda *Pr(Y = 1|X),* kde bolo Áno nahradené hodnotou 1. V literatúre [1], bol zavedený skrátený zápis tejto pravdepodobnosti vo forme Pr(X). Pri tomto skrátenom zápise potom môžeme povedať, že hľadáme závislosť Pr(X) na X. Teda ak budeme vychádzať z toho, čo sme sa naučili z lineárnej regresie, povieme, že hľadáme nasledovný model:

$$P(X) = \beta_0 + \beta_1 \cdot X \tag{27}$$

To je najjednoduchší model, ktorý ale sa nepodobá na ten model, ktorý potrebujeme dosiahnuť. Tento model je reprezentovaný lineárnou priamkou, teda pre veľké hodnoty výšky budeme mať oveľa väčšie hodnoty pravdepodobnosti ako *1*. Teda takáto predikcia nie je dosť citlivá ani vhodná na odhad pravdepodobnosti, ktorá musí byť medzi hodnotami *0* a *1*. Aby sme sa vyhli tomuto problému, musíme nájsť iný model použitím funkcie, ktorá bude dávať výstup ako hodnotu z intervalu $\langle 0,1 \rangle$ pre všetky hodnoty *X*. Bolo by možné nájsť mnoho funkcií, ktoré spĺňajú túto podmienku. Jednou z nich je aj logistická funkcia (28):

$$p(X) = \frac{e^{\beta_0+\beta_1 \cdot X}}{1 + e^{\beta_0+\beta_1 \cdot X}} \tag{28}$$

Táto funkcia je ilustrovaná aj na Obr.8, z ktorého je jasné, že závislá premenná nemôže nadobúdať hodnotu menšiu ako 0 ani hodnotu väčšiu ako 1 pri akýchkoľvek hodnotách nezávislej premennej. Logistická funkcia produkuje krivku tvaru S

Ako sa k tomuto vzťahu môžeme dopracovať? Uvažujme osobu s určitou výškou. Aká je šanca že tento človek je muž? Šanca vo všeobecnosti je pomer pravdepodobnosti, že niečo platí ku pravdepodobnosti, že to neplatí.

Teda Šanca = p(pravda)/p(nepravda), pričom p(nepravda) = 1-p(pravda). Z toho vyplýva nasledovný vzťah (29) pre šancu nejakého javu *X*. Táto šanca sa zvykne označovať *„odds“* a môže nadobúdať hodnoty medzi 0 a ∞.

$$odds(X) = \frac{p(X)}{1-p(X)}. \tag{29}$$

Ak niečo platí pre 1 zo 4 ľudí (jeden zo štyroch má pozorovanú vlastnosť), šanca výskytu tejto vlastnosti je ¼ lebo jej pravdepodobnosť sa rovná 0.2 a teda Odds=0.2/(1-0.2)=0.25. Teda šanca aj pravdepodobnosť sú veľmi malé. Ak vezmeme do úvahy optimistickejší prípad, keď nejakú vlastnosť má 9 z 10 ľudí, potom p(X) = 0.9 implikuje odds=0.9/(1-0.9)=9.

Platí, že logaritmus šance má lineárny vzťah k prediktorom a v dôsledku toho platí aj nasledovný vzťah:

$$\log \frac{p(X)}{1-p(X)} = \beta_0 + \beta_1 \cdot X. \tag{30}$$

Ľavá strana tejto rovnice sa zvykne nazývať „log-odds“ alebo „logit“. Platí, že logistický regeresný model má logit, ktorý je lineárne závislý na *X*. Ak β_1 má pozitívnu hodnotu, potom nárast *X* implikuje nárast *p(X)*. Ak β_1 má negatívnu hodnotu, potom nárast *X* je spojený s poklesom *p(X)*.

Keď obidve strany rovnice (30) vložíme ako argumenty do exponenciálnej funkcie, dostaneme nasledovný vzťah:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 \cdot X} \tag{31}$$

Ak túto rovnosť upravíme takým spôsobom, že použijeme rovnicu (27) ako substitúciu dostaneme vzťah (28) teda našu logistickú regresnú funkciu.

3.1 Odhad regresných koeficientov

Koeficienty β_0 a β_1 sú neznáme a môžu byť odhadnuté na základe dostupných trénovacích údajov. V kapitole 2. týkajúcej sa lineárnej regresie bolo uvedené, že na odhad

regresných koeficientov sa používa metóda najmenších štvorcov. Aj v prípade logistickej regresie môžeme použiť nelineárnu metódu najmenších štvorcov. Však lepšie bude použiť všeobecnejšiu *metódu maximálnej podobnosti*, ktorá má lepšie štatistické vlastnosti. Teda snažíme sa nájsť také hodnoty koeficientov β_0 a β_1, ktoré zabezpečia, aby sme zo vzťahu (28) dostali hodnotu $p(X)$ blízku jedna pre všetky X, pre ktoré platí daná hypotéza a $p(X)$ blízke 0 pre všetky X, pre ktoré táto hypotéza neplatí. Táto interpretácia môže byť matematicky formalizovaná rovnicou (32), ktorá sa nazýva *funkcia podobnosti*:

$$l(\beta_0, \beta_1) = \sum_{i:y_i=1} p(x_i) \sum_{i':y_i'=0} \big(1 - p(x_i')\big). \tag{32}$$

Aby sme získali odhad týchto parametrov $\hat{\beta}_0$ a $\hat{\beta}_1$ musíme nájsť maximum funkcie podobnosti. Ako bolo spomenuté funkcia podobnosti predstavuje veľmi všeobecný prístup, používaný s úspechom v mnohých nelineárnych modeloch. Metóda najmenších štvorcov je špeciálnym prípadom tejto funkcie podobnosti.

Vytváranie predikcií. Predpokladajme, že máme k dispozícii dáta o platobnej neschopnosti v závislosti na výške balansu na kreditnej karte. Potrebujeme z nich naučiť model založený na logistickej regresii, ktorý bude predikovať pravdepodobnosť bankrotu. Najprv musíme odhadnúť (vypočítať) hodnoty koeficientov tohto modelu na základe dát, ktoré máme k dispozícii. Povedzme, že dostaneme výsledok uvedený v Tab.6.

Tab.6 Hodnoty koeficientov modelu založeného na logistickej regresii

Koeficient	Hodnota koeficientu	Std. error
$\widehat{\boldsymbol{\beta}}_{\mathbf{0}}$	-10.6513	0.3612
$\hat{\beta}_1$	0.0055	0.0002

Keď máme odhadnuté koeficienty, môžeme pristúpiť k samotnej predikcii. Ak je výška balansu 1000 Euro,

potom odhad pravdepodobnosti bankrotu podľa vzťahu (28) vyzerá nasledovne:

$$\hat{p}(X) = \frac{e^{-10.6513+0.0055\cdot 1000}}{1+e^{-10.6513+0.0055\cdot 1000}} = 0.00576 \tag{33}$$

Táto pravdepodobnosť bankrotu vyšla menšia ako 1%. Ak sa však hodnota balansu zvýši na 2000 Euro, potom aj pravdepodobnosť bankrotu sa podstatne zvýši na hodnotu 0.586 čo predstavuje 58,6%. Tento príklad bol prevzatý z [1], kde je možné nájsť viac informácií o logistickej regresii ako aj o odhade regresných koeficientov.

3.2 Viacnásobná logistická regresia

Pri viacnásobnej logistickej regresii riešime problém predikcie binárnej odpovede použitím viacerých prediktorov (atribútov). Teda modelujeme závislú premennú, ktorá môže nadobúdať dve hodnoty a je závislá na viacerých nezávislých premenných. Zoberieme rovnicu (30), ktorá bola odvodená pre jednoduchú logistickú regresiu a zovšeobecníme ju tak že namiesto jednej nezávislej premennej použijeme *p* nezávislých premenných – *p* prediktorov $(X_1, \ldots, X_p)$, čím dostaneme vzťah (34):

$$\log\frac{p(X)}{1-p(X)} = \beta_0 + \beta_1 \cdot X_1 + \cdots + \beta_p \cdot X_p \tag{34}$$

Z tohto vzťahu môžeme odvodiť nasledovnú rovnicu pre odhad pravdepodobnosti predikovaného javu – pravdepodobnosti hodnoty závislej premennej:

$$p(X) = \frac{e^{\beta_0+\beta_1\cdot X_1+\cdots+\beta_p\cdot X_p}}{1+e^{\beta_0+\beta_1\cdot X_1+\cdots+\beta_p\cdot X_p}} \tag{35}$$

Podobne ako pri jednoduchej logistickej regresii použijeme metódu maximálnej podobnosti na odhad koeficientov $\beta_0, \beta_1, \ldots, \beta_p$.

3.3 Logistická regresia pre viac ako dve triedy

Zatiaľ čo pri viacnásobnej regresii sa uvažuje viac prediktorov, teda nezávislých premenných, pri viactriednej regresii sa uvažuje viac ako dve závislé premenné – viac ako dve triedy. Ako príklad môže poslúžiť skupina troch najčastejších diagnóz traumatologickej ambulancie na pohotovosti a to: *infarkt, predávkovanie liekmi a epileptický záchvat*. Teda potrebujeme modelovať dve pravdepodobnosti:

Pr(Y = infarkt | X) a *Pr(Y = predávkovanie liekmi | X)*, pričom pre poslednú diagnózu platí :

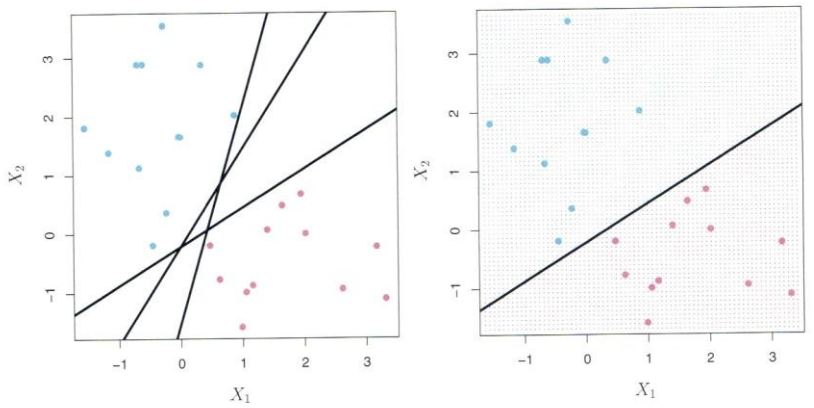
Pr(Y = epileptický záchvat | X) = 1 - Pr(Y = infarkt | X) - Pr(Y = predávkovanie liekmi | X).

Logistická regresia pre dve triedy z predchádzajúcej podkapitoly má svoje rozšírenie vo forme logistickej regresie pre viac ako dve triedy, no v praxi nie je tendencia ho veľmi používať. Pre viac triednu klasifikáciu sa najčastejšie používa diskriminačná analýza, o ktorej sa viac možno dozvedieť zo zdrojov [1] a [10].

4 METÓDA PODPORNÝCH VEKTOROV

Metóda podporných vektorov je aj v slovenských vedeckých kruhoch známa pod skratkou SVM (ang. *Support Vector Machines*) [1]. Táto metóda sa objavila v komunite vedcov „počítačiarov“ v roku 1990 a do dnešných čias je veľmi populárna, pravdepodobne preto, že pri riešení mnohých problémov dáva najlepšie výsledky. SVM vzniklo ako zovšeobecnenie jednoduchého a veľmi intuitívneho klasifikátora nazvaného *Klasifikátor maximálneho rozpätia* (ang. Maximal Margin Classifier) [1]. Avšak tento klasifikátor nie je možné aplikovať na väčšinu dát, ktoré nie sú lineárne separabilné. To znamená, že dokáže spracovať iba dáta, ktorých triedy je možné separovať lineárnou priamkou (hranicou). Preto bolo navrhnuté rozšírenie tohto klasifikátora nazvané *Klasifikátor podporných vektorov* (ang. Support Vector Classifier) [10], ktoré je možné aplikovať na väčšinu dát. Ďalším rozšírením je *Stroj podporných vektorov* (ang. Support Vector Machine), ktorý je schopný pracovať aj s nelineárnymi hranicami tried.

Aby sme sa dostali ďalej, musíme definovať pojem *hyperrovina* (ang. Hyperplane). Hyperrovina v *p* dimenzionálnom priestore je podobný útvar ale v *p-1* dimenzionálnom priestore. Napríklad je hyperrovina 2_dimenzionálneho priestoru musí byť 1_dimenzionálna a teda je to priamka. Podobne hyperrovina 3_dimenzionálneho priestoru musí byť 2_dimenzionálny útvar a teda rovina. Takáto hyperrovina sa môže použiť na separovanie príkladov dvoch tried od seba, teda na klasifikáciu. Príkladom takého typu klasifikácie je aj takzvaná *Lineárna prahová jednotka* (ang. Linear Threshold Unit) [11]. Problémom je, že lineárne separovateľné dáta je možné väčšinou separovať viacerými hyperrovinami, ako to ilustruje Obr.9. V jeho ľavej časti sú nakreslené tri rôzne riešenia – tri hyperroviny, teda priamky, keďže ide o dvojrozmerný priestor. Musíme použiť nejaké hodnotiace kritérium kvality týchto troch riešení, aby sme vybrali jedno, ktoré je ilustrované v pravej časti obrázku.



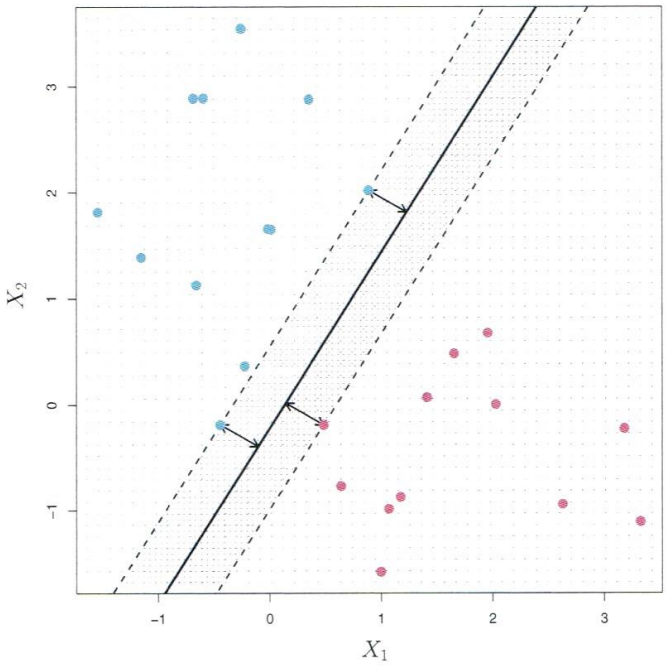
Obr.9 Ilustrácia lineárnej prahovej jednotky použitej na separovanie príkladov dvoch tried [1]

Vo všeobecnosti existuje nekonečné množstvo takýchto hyperrovín v prípade, že máme k dispozícii dáta, ktoré sú perfektne separovateľné. A práve *Klasifikátor maximálneho rozpätia* (ang. Maximal Margin Classifier) je tá metóda, ktorá nám pomôže vyhnúť sa problému výberu jedného najlepšieho riešenia.

4.1 Klasifikátor maximálneho rozpätia

Úlohou tohto klasifikátora je nájsť takú separujúcu hyperplochu, ktorá je najďalej od trénovacích príkladov. Aby sme ju našli, musíme vypočítať všetky vzdialenosti trénovacích príkladov od separujúcej hyperplochy. Najmenšia z týchto vzdialeností sa nazýva okraj. Na Obr.10 sa nachádzajú dva okraje. Ešte musíme vedieť zistiť, na ktorej strane hyperplochy vrátane okrajov sa nachádzajú testovacie príklady. Potom ich vieme klasifikovať. Tak sa dostávame ku klasifikátoru maximálneho rozpätia, ktorý je ilustrovaný na Obr.10. Hyperplocha maximálneho rozpätia je nakreslená plnou čiarou. Vzdialenosti od plnej ku čiarkovaným čiaram – okrajom, predstavujú rozpätie. Táto separačná hyperplocha je reprezentovaná priamkou umiestnenou

v strede širšieho pásu, ktorý bol vložený medzi dve triedy.



Obr.10 Ilustrácia kalsifikácie do dvoch tried pomocou klasifikátora maximálneho rozpätia [1]

Klasifikátor, ktorý má veľké rozpätie na trénovacích príkladoch, bude mal mať veľké rozpätie aj na testovacích príkladoch a teda bude presnejší. Klasifikátor maximálneho rozpätia je často úspešný, ale taktiež môže viesť k preučeniu pri vysoko dimenzionálnom priestore príkladov.

Na Obr.10 vidíme, že tri trénovacie príklady majú rovnakú vzdialenosť od separujúcej hyperplochy a ležia na prerušovaných čiarach, ktoré indikujú rozpätie, alebo šírku okrajov. Tieto trénovacie príklady sú reprezentované vektormi, ktoré vychádzajú zo separujúcej hyperplochy a smerujú k okrajom a nazývame ich *podporné vektory*. Ak sa tieto trénovacie príklady posunú hoci máličko, posunie sa aj poloha hyperplochy maximálneho rozpätia. Ak sa posunú iné trénovacie príklady, ktoré nie sú podpornými

vektormi, poloha separujúcej hyperplochy ostane rovnaká. Teda poloha hyperplochy maximálneho rozpätia závisí de facto na malom počte dôležitých pozorovaní – na malom počte podporných vektorov.

Ako teda môžeme skonštruovať hyperplochu maximálneho rozpätia? Predpokladajme, *n* trénovacích príkladov: $x_1, \ldots, x_n$ a klasifikačné triedy označkované: $y_1, \ldots, y_n \in \{-1, 1\}$. Hľadanie hyperplochy maximálneho rozpätia je potom optimalizačný problém, pri ktorom sa hľadá maximálne *M* definované rovnicou (36):

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) \geq M, i = 1, \ldots, n \tag{36}$$

Pričom platí formula (37)

$$\sum_{j=1}^{p} \beta_j^2 = 1. \tag{37}$$

Obmedzenie definované v rovnici (36) garantuje, že každý trénovací príklad bude na správnej strane hyperplochy, keďže ona je definovaná nasledovne pomocou (38).

$$\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) = 0. \tag{38}$$

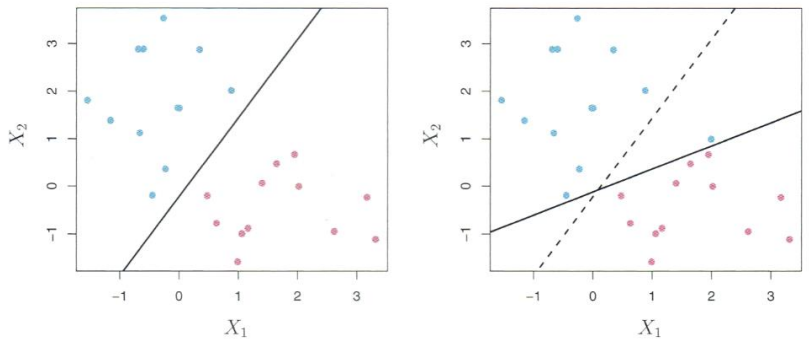
Dá sa dokázať, že kolmá vzdialenosť od trénovacieho príkladu k hyperploche sa rovná:

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right). \tag{39}$$

Je potrebné si uvedomiť, že *M* reprezentuje rozpätie, resp. okraje hyperplochy, preto je potrebné *M* maximalizovať. Ak by *M=0*, to znamená že deliaca hyperplocha by nemala žiadne rozpätie ani okraje. Ak sú trénovacie príklady oboch tried v priestore premiešané a teda priestor nie je separabilný, potom sa klasifikátor maximálneho rozpätia nemôže použiť. Ale je možné rozšíriť tento koncept použitím takzvaných mäkkých okrajov „soft margin“, na ktorých je vybudovaný klasifikátor podporných vektorov.

4.2 Klasifikátor podporných vektorov

Klasifikátor maximálneho rozpätia z predchádzajúcej podkapitoly nevyhnutne perfektne a striktne klasifikuje všetky trénovacie príklady ale na druhej strane je veľmi senzitívny na individuálne pozorovania. Ak zmeníme polohu hoci len jedného príkladu následkom bude dramatická zmena polohy hyperplochy maximálneho rozpätia, čo ilustruje Obr.11. V ľavej časti obrázku je nájdená optimálna hyperplocha pre zadané dáta – trénovacie príklady. Stačí pridať jeden jediný nový trénovací príklad (viď pravú časť obrázku) a poloha hyperplochy sa pootočí o takmer 40 stupňov. Na viac táto nová hyperplocha je menej uspokojujúca ako predchádzajúca, pretože má tenšie rozpätie.



Obr.11 Ilustrácia citlivosti klasifikátora maximálneho rozpätia na zmenu polohy individuálnych pozorovaní [1]

Riešenie s príliš tenkým rozpätím je problém, lebo dostatočná vzdialenosť klasifikovaného príkladu od deliacej hyperplochy je zárukou toho, že príklad bol správne klasifikovaný.

Preto bolo navrhnuté rozšírenie na taký klasifikátor, ktorý nebude musieť perfektne klasifikovať všetky trénovacie príklady do jednej z dvoch tried, ale v niekoľkých málo prípadoch mu je dovolené urobiť chybu. Na druhej strane bude robustnejší vo vzťahu k individuálnym pozorovaniam

(trénovacím príkladom) a bude lepšie klasifikovať väčšinu trénovacích príkladov. Obetujeme chybnú klasifikáciu zopár trénovacích príkladov za lepšiu klasifikáciu zvyšnej väčšiny, teda použijeme niečo ako mäkké okraje (soft margins). Budeme mäkkí, lebo prižmúrime oči nad nesprávnou klasifikáciou v niektorých prípadoch. Takýto klasifikátor sa niekedy nazýva *klasifikátor mäkkého okraja* (Soft Margin Classifier), ale oveľa známejší názov tohto rozšíreného klasifikátora je *Klasifikátor podporných vektorov*. Tento klasifikátor pripúšťa, že niektoré trénovacie príklady sú vo vnútri rozpätia a to buď na správnej strane deliacej hyperplochy, ale niekedy môžu padnúť aj na opačnú stranu hyperplochy. Aj to sa zriedkavo pripúšťa aj keď je to veľmi nevítaný úkaz, keďže odpovedá chybnej klasifikácii. Riešením tohto problému je ďalšia optimalizácia, ktorá je popísaná v zdroji [1].

4.3 Stroje podporných vektorov

Stroje podporných vektorov je preklad z anglického názvu Support Vector Machines (SVMs), ktorý je aj v slovenských vedeckých kruhoch bežne rozšírený. Preto v nasledujúcom texte budeme sa na tento klasifikátor odvolávať skratkou SVM alebo SVMs. Aby to nebolo také jednoduché, v literatúre sa uvádza táto skratka v dvoch obdobách v jednotnom aj množnom čísle. Existujú SVMs - Support Vector Machine**s**, ktoré predstavujú skupinu dvoch klasifikátorov a to po,prvé *Klasifikátor s nelineárnou rozhodovacou hranicou* (Classification with Non-linear Decision Boundaries) a po druhé *Stroj podporných vektorov* - SVM (Support Vector Machine). Zatiaľ čo klasifikátor s nelineárnou rozhodovacou hranicou používa nelineárne funkcie na simuláciu deliacej hyperroviny, stroj podporných vektorov (! jeden) pracuje s kernelom [12].

Klasifikátor podporných vektorov, o ktorom sme hovorili v predchádzajúcej podkapitole, je veľmi dobrou možnosťou pre klasifikáciu separabilných dát do dvoch tried, ak hranica medzi tými dvoma triedami môže byť lineárna. Avšak v praxi máme často k dispozícii dáta, ktoré vyžadujú skôr hranicu simulovanú nelineárnou funkciou, napríklad

kvadratickou funkciou, kubickou funkciou alebo polynomiálnou funkciou vyššieho rádu. Vtedy musíme použiť iný klasifikátor a najjednoduchšie je siahnuť po Klasifikátore s nelineárnou rozhodovacou hranicou. Aby sme si priblížili tento klasifikátor uvažujme namiesto *p* atribútov $X_1, X_2, \ldots, X_p$ v *p* dimenzionálnom priestore rozšírenú množinu *2p* atribútov $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$.

Potom musíme maximalizovať M prostredníctvom nájdenia takých koeficientov $\beta_0, \beta_{11}, \beta_{12}, \ldots, \beta_{p1}, \beta_{p2}, \epsilon_1, \ldots, \epsilon_n$ aby platilo (40)

$$y_i\left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2\right) \geq M(1 - \epsilon_i), \tag{40}$$

pričom platí:

$$\sum_{i=1}^{n} \epsilon_i \leq C, \epsilon_i \geq 0, \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1 \tag{41}$$

Mohli by sme uvažovať namiesto kvadratického polynómu polynómy vyšších radov, alebo použiť iné zložitejšie nelineárne funkcie namiesto polynómov a skončiť pri obrovskom množstve atribútov až by bola výpočtová zložitosť nezvládnuteľná. Preto je v takom prípade lepšie použiť stroj podporných vektorov (jeden) SVM.

4.4 Stroj podporných vektorov

Stroj podporných vektorov (SVM) je ako bolo naznačené rozšírenie klasifikátora podporných vektorov, ktoré je možné použiť aj na priestore príkladov s vysokou dimenzionalitou pomocou kernelovej metódy [12]. Predpokladajme, že skalárny súčin dvoch *r-vektorov a, b* je $\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$. Potom skalárny súčin dvoch pozorovaní (trénovacích príkladov) je daný nasledovne:

$$\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{42}$$

Dá sa dokázať, že pre lineárny klasifikátor podporných vektorov platí

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \tag{43}$$

Pričom sa predpokladá *n* parametrov $\alpha_i, i = 1, \ldots, n$ pre každý trénovací príklad. Na odhad parametrov $\beta_0, \alpha_1, \ldots, \alpha_n$ potrebujeme $\binom{n}{2} = n(n-1)/2$ skalárnych súčinov $\langle x_i, x_i' \rangle$ medzi všetkými pármi trénovacích príkladov. Pretože α_i je nenulové iba pre podporné vektory (ak trénovací príklad nie je podporným vektorom potom jeho $\alpha_i = 0$) môžeme vzťah (43) upraviť nasledovne:

$$f(x) = \beta_0 + \sum_{i \epsilon S} \alpha_i \langle x, x_i \rangle, \tag{44}$$

kde S je množina podporných vektorov. Teda lineárny klasifikátor je založený hlavne na výpočte skalárnych súčinov. Predpokladajme, že môžeme tieto skalárne súčiny nahradiť zovšeobecnením skalárneho súčinu vo forme kernelu $K(x_i, x_i')$ čo je funkcia, ktorá kvantifikuje podobnosť dvoch pozorovaní. V najjednoduchšom prípade môže mať tvar (45)

$$K(x_i, x_i') = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{45}$$

Tento vzťah je známy ako lineárny kernel. Avšak mohli by sme použiť aj iný tvar kernelovej funkcie, napríklad (46).

$$K(x_i, x_i') = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{46}$$

Vzťah (46) je známy ako polynomiálny kernel stupňa *d*, kde *d* je pozitívne celé číslo. Použitím polynomiálneho kernela v klasifikátore podporných vektorov je možné dosiahnuť oveľa flexibilnejšie tvary deliacich hraníc na separovanie príkladov dvoch tried od seba.

Ak je klasifikátor podporných vektorov kombinovaný s niektorým nelineárnym kernelom, ako napríklad vo vzťahu (46), potom výsledný klasifikátor je známy ako SVM (support vector machine) a má nasledovnú formu:

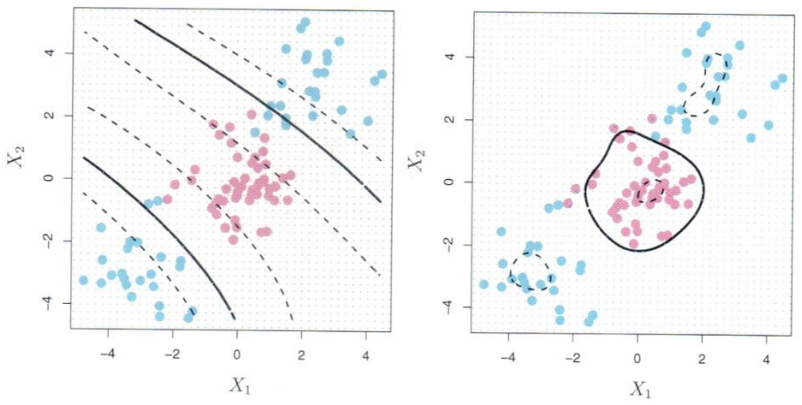
$$f(x) = \beta_0 + \sum_{i \epsilon S}^{n} \alpha_i K(x, x_i). \tag{47}$$

Ľavá časť Obr.12 ilustruje príklad SVM s polynomiálnym kernelom. Keby sme aplikovali na dáta v ľavej časti Obr.12 lineárny klasifikátor podporných vektorov, bola by chyba klasifikácie vysoká. Ak položíme *d=1* bude SVM redukované na klasifikátor podporných vektorov.

Ďalšia alternatíva nelineárneho kernela je populárny radiálny kernel, ktorý má nasledovnú formu:

$$K(x_i, x_i') = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{48}$$

V rovnici (48) je γ kladná konštanta. V pravej časti Obr.12 je ilustrované použitie SVM v kombinácii s radiálnym kernelom a ako vidieť výsledok separovania dvoch tried od seba je veľmi dobrý.



Obr.12 SVM s polynomiálnym kernelom stupňa 3 na nelineárnych dátach v ľavej časti obrázku a SVM s radiálnym kernelom v pravej časti obrázku [1]

Použitie kernelových funkcií má výhodu aj v tom, že znižuje výpočtovú náročnosť, lebo stačí vypočítať $K(x_i, x_i')$ namiesto $\binom{n}{2}$ skalárnych súčinov. To je dôležité, lebo v mnohých reálnych aplikáciách SVMs sa pracuje s tak veľkým priestorom atribútov, až sa výpočet stáva nezvládnuteľný.

5 UČENIE SÚBOROM METÓD

Učenie súborom metód (Ensemble Learning) [10] sa niekedy nazýva aj učením pomocou zloženého klasifikátora [13] alebo učenie pomocou združených metód [14]. V rámci učenia súborom metód je vytváraný jeden zložený model, ktorý sa skladá z viacerých partikulárnych – čiastkových modelov, ktoré môžu mať rovnakú ale aj odlišnú štruktúru [15]. Pri rovnakej štruktúre jednoducho použijeme tú istú metódu strojového učenia opakovane. Pri odlišnej štruktúre môže byť každý čiastkový model natrénovaný inou metódou strojového učenia. Učenie súborom metód je motivované reálnym životom, keď o viacerých možnostiach ako realizovať nejaké významné rozhodnutie necháme rozhodovať viacero autorít v danej oblasti. Jednoducho budú hlasovať o finálnom rozhodnutí o výsledku. Viaceré možnosti ako rozhodnúť je možné v oblasti strojového učenia interpretovať ako viaceré triedy a autority je možné nahradiť čiastkovými klasifikátormi teda modelmi, ktoré boli natrénované jednou alebo viacerými metódami strojového učenia pri použití tých istých alebo aj rozdielnych parametrov. Prečo je dobré si takto komplikovať život? Preto, lebo učenie súborom metód prináša spravidla zvýšenie presnosti a stability modelov. Nevýhodou je väčšia výpočtová náročnosť súvisiaca s väčšou zložitosťou modelu. Avšak, niekedy je možné čiastkové modely trénovať paralelne a tak znížiť vysokú výpočtovú náročnosť.

Zloženou klasifikáciou má zmysel sa zaoberať, pretože zvyšuje efektívnosť klasifikácie slabých klasifikátorov pomocou hlasovania viacerých slabých, alebo tzv. partikulárnych klasifikátorov o konečnej klasifikácii do konkrétnej kategórie. Slabý klasifikátor je taký, ktorého presnosť je len o málo vyššia ako presnosť náhodnej klasifikácie. Preto práve pre tieto klasifikátory je vhodné hľadať techniky zvyšovania efektívnosti, napríklad zloženou klasifikáciou. Efektívnosť klasifikácie vo všeobecnosti závisí od viacerých skutočností. Hlavne od vhodnosti výberu klasifikačného algoritmu pre danú úlohu a od kvality trénovacej množiny. Kvalita trénovacej množiny závisí od

toho, či sú sledované hodnoty všetkých relevantných atribútov, či nie sú medzi atribútmi aj irelevantné atribúty, nakoľko presné sú pozorovania (trénovacie príklady) - či nie sú zašumené a pod. Za predpokladu, že trénovacia množina nie je dostatočne kvalitná, aby zabezpečila vysokú presnosť klasifikácie, je možné formovať rôzne výbery z trénovacej množiny.

Teda môžeme povedať, že hlavný cieľ učenia súborom metód je vytvoriť predikčný model (zložený klasifikátor) kombinovaním sily kolekcie viacerých jednoduchých základných modelov [10]. Poznáme viacero druhov týchto zložených klasifikátorov. Najznámejšie sú Bagging, Random Forest (Náhodný les), Boosting a Stacking (Stohovanie). Každému z týchto modelov bude venovaná jedna z nasledujúcich podkapitol. Bagging a náhodný les sú modely učené súborom metód vhodné pre úlohu klasifikácie, kde „komisia“ stromov hlasuje za predikovanú triedu. Boosting je tiež metóda založená na hlasovaní komisie, ale je komplikovaniejšia v tom, že slabé klasifikátory sa vyvíjajú v čase a členovia komisie majú vážené hlasy. Stacking je nový prístup kombinujúci silu veľkého počtu modelov vychádzajúcich z rozdielnych trénovacích algoritmov strojového učenia.

Učenie súborom metód pozostáva z dvoch úloh. Po prvé je potrebné natrénovať základné klasifikátory z trénovacích dát. Po druhé je potrebné kombinovať tieto základné klasifikátory do zloženého klasifikátora, resp. zloženého prediktora. To platí napríklad pri metódach bagging a náhodný les. Metóda boosting ide ďalej a buduje skupinový klasifikátor tým, že uskutočňuje reguláciu a kontrolované hľadanie vo vysoko dimenziounálnom priestore slabých klasifikátorov.

Pri samotnom učení zloženého klasifikátora, sa najprv musia naučiť partikulárne klasifikátory, pretože na základe ich hlasovania sa rozhodne o výslednej klasifikácii nového príkladu zloženým klasifikátorom. Ako čiastkové, partikulárne modely sú veľmi dobre použiteľné stromové modely. Budeme uvádzať príklady použitia metód bagging a boosting, kde v úlohe partikulárnych, resp. základných klasifikátorov bol použitý algoritmus generujúci

rozhodovacie stromy C4.5 [16]. Experimenty s uvedenými metódami optimalizácie výsledkov základných klasifikačných algoritmov boli aplikované na riešenie problému kategorizácie textových dokumentov [17].

5.1 Bagging

Bagging je metóda na zlepšovanie výsledkov klasifikačných algoritmov, ktorá bola navrhnutá Breimanom [18]. Jej názov je odvodený zo slovného spojenia "**b**ootstrap **agg**regat**ing**". Táto technika formuje rozličné výbery z pôvodnej trénovacej množiny a nad každou podmnožinou – výberom natrénuje partikulárny slabý klasifikátor niektorým zvoleným algoritmom strojového učenia. O výsledku klasifikácie nového prípadu sa rozhodne takzvaným „hlasovaním" všetkých partikulárnych klasifikátorov.

Pre partikulárne klasifikátory platí, že sú naučené tým istým algoritmom strojového učenia, ale každý z nich bol natrénovaný na inej podmnožine trénovacích príkladov. Teda všetky natrénované partikulárne modely majú rovnakú štruktúru a vo výsledku rovnakú váhu. Preto metóda bagging dokáže nielen dobre odhadnúť najpravdepodobnejšiu triedu ale aj pravdepodobnosti jednotlivých tried. Keďže trénovanie partikulárnych modelov je nezávislé (čo neplatí v prípade boostingu, kde každý partikulárny model je postavený na predchádzajúcom) je možné toto trénovanie realizovať v paralelnej forme.

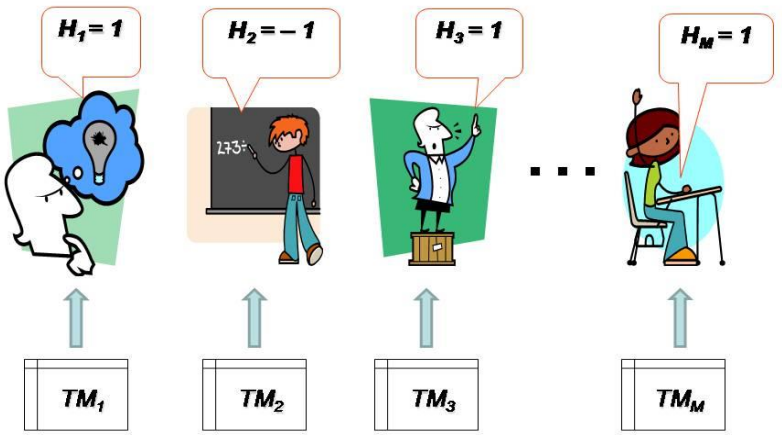
Samotná metóda baggingu pracuje nasledovne. V prípade dvojtriednej klasifikácie, klasifikačný algoritmus generuje klasifikátor $H: D \rightarrow \{-1,1\}$ na základe trénovacej množiny, teda kolekcie textových dokumentov *D.* Každý dokument je buď klasifikovaný do triedy (1), alebo nie je klasifikovaný do triedy(-1) daným klasifikátorom. Definičný obor klasifikátora *H* je množina dokumentov *D* a obor funkčných hodnôt klasifikátora *H* je množina kategórií *C*. Metóda bagging vytvára postupnosť *M* partikulárnych klasifikátorov H_m, $m=1,\ldots,M$, kde *M* je počet všetkých výberov (vzoriek)

trénovacej množiny. Tieto partikulárne klasifikátory sú kombinované do zloženého klasifikátora podľa vzorca (49):

$$H(d_i, c_j) = sign\left(\sum_{m=1}^{M} \alpha_m H_m(d_i, c_j)\right) \quad (49)$$

V tomto vzorci okrem hlasov partikulárnych klasifikátorov H_m vystupujú aj konštanty α_m, ktoré sú vypočítané algoritmom baggingu alebo aj boostingu a hodnotia príspevok každého partikulárneho klasifikátora ku presnosti výslednej klasifikácie. Úlohou týchto konštánt je ponechať väčší vplyv presnejším partikulárnym klasifikátorom.

Vzťah (49) je možné interpretovať ako hlasovanie partikulárnych klasifikátorov, kde trénovací príklad - dokument d_i je klasifikovaný do triedy c_j, pre ktorú hlasuje väčšina partikulárnych klasifikátorov. Parameter α_m, $m=1,\ldots,M$ posilňuje vplyv presnejších klasifikátorov na konečnú predikciu triedy. Hlasovanie partikulárnych klasifikátorov ilustruje Obr.13. Za predpokladu, že počet partikulárnych klasifikátorov je $M=4$, potom podľa Obr.13 tri partikulárne klasifikátory hlasujú za zaradenie dokumentu d_i triedy c_j a jeden hlasuje proti ($H_1=H_3=H_4=1$, $H_2=-1$). Teda suma H_m je kladné číslo a výsledok funkcie signum je 1, teda výsledné rozhodnutie zloženého klasifikátora je zaradiť dokument d_i do triedy c_j . Pričom sa predpokladá, že $\alpha_1= \alpha_2= \alpha_3= \alpha_4=1$.



Obr.13 Hlasovanie partikulárnych klasifikátorov [13]

Vyššie popísaný postup reprezentuje metódu, ktorá sa nazýva „*základná verzia baggingu*“ [18]. Avšak, sú známe aj ďalšie podobné stratégie – „*bagging like strategies*“, ktoré rozdeľujú pôvodnú trénovaciu množinu na *M* rovnako veľkých podmnožín. Nad týmito podmnožinami sa natrénuje *M* partikulárnych klasifikátorov, ktoré sa združia do jedného zloženého klasifikátora. Tieto baggingové stratégie sú ilustrované Tab.7, kde originálna trénovacia množina pozostáva zo šestnástich trénovacích príkladov označených veľkými písmenami abecedy.

Najznámejšie z týchto stratégií sú disjunktné partície (disjoint partitions), malé vrecia (small bags), malé vrecia bez opakovania (no replication small bags) a disjunktné vrecia (disjoint bags).

Stratégia *disjunktných partícií* používa náhodné rozdelenie príkladov trénovacej množiny do podmnožín, pričom každý príklad sa v podmnožinách nachádza iba raz. Vo všeobecnosti platí, že ak tvoríme *M* podmnožín, každá z nich bude obsahovať *1/M* časť pôvodnej množiny. Zjednotenie podmnožín vytvorí opäť pôvodnú množinu. Klasifikátor *H* získaný zložením partikulárnych

klasifikátorov H_m naučených na disjunktných partíciách, dosahuje spravidla najlepšie výsledky zo všetkých baggingových stratégií.

Tab.7 Rozličné stratégie delenia trénovacej množiny na partikulárne podmnožiny

Pôvodná množina	A B C D E F G H I J K L M N O P			
Disjunktné partície	A B C D	E F G H	I J K L	M N O P
Malé vrecia (MV)	A C H L	B P L P	D I O H	K C F K
MV bez opakovania	A C H L	O P L N	D I O H	K C F P
Disjunktné vrecia	A B C D C	E F G H E	I J K L J	M N O P O

V stratégii *malých vriec* je každá podmnožina príkladov generovaná nezávisle od ostatných náhodným výberom trénovacích príkladov s možnosťou opakovania. Jeden príklad sa môže nachádzať vo viacerých podmnožinách a aj v tej istej podmnožine sa môže vyskytovať opakovane. Zjednotenie partícií nemusí dávať pôvodnú trénovaciu množinu. Táto stratégia dáva najhoršie výsledky.

Stratégia *malých vriec bez opakovania* sa od predchádzajúcej líši tým, že žiadny trénovací príklad sa nenachádza viackrát v tej istej podmnožine.

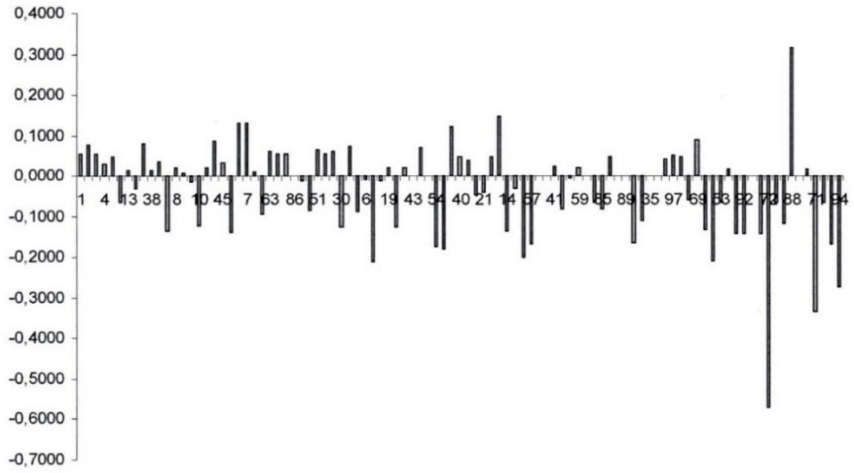
V poslednej stratégii nazvanej *disjunktné vrecia* sa početnosť podmnožín najprv rovná *1/M*-tej časti pôvodnej množiny. Potom sa ich početnosť zväčší, pretože v každej sa zopakuje náhodne vybratý prvok. Toto zväčšenie nemusí byť iba o jednotku, pretože replikovať sa môže viac prvkov. Počet replikácií musí byť rovnaký v každej podmnožine. Efektívnosť tejto stratégie je podobná

efektívnosti disjunktných partícií a za určitých podmienok môže byť aj lepšia.

Bola vykonaná séria testov metódy bagging nad dvoma kolekciami dokumentov: Reuter's-21578 a internetového portálu televízie Markíza. Kolekcia Reuter's-21578 obsahovala 674 kategórií a 24242 termov - slov v anglickom jazyku. Po predspracovaní pomocou lematizácie (steming) a po odstránení neplnovýznamových (stop) slov bol počet termov redukovaný na 19864. Druhá kolekcia – Markíza obsahovala dokumenty klasifikované do 96 kategórií a obsahovala 26785 dokumentov v slovenskom jazyku. Dokumenty oboch kolekcií boli rozdelené na trénovaciu a testovaciu časť v pomere 2:1.

Vykonané testy sa týkali efektívnosti baggingových stratégií a určenia minimálneho počtu základných klasifikátorov potrebných na zlepšenie výsledkov slabej klasifikácie ako aj určenia počtu klasifikátorov, zvyšovaním ktorého sa už efektívnosť klasifikácie výrazne nezvýši.

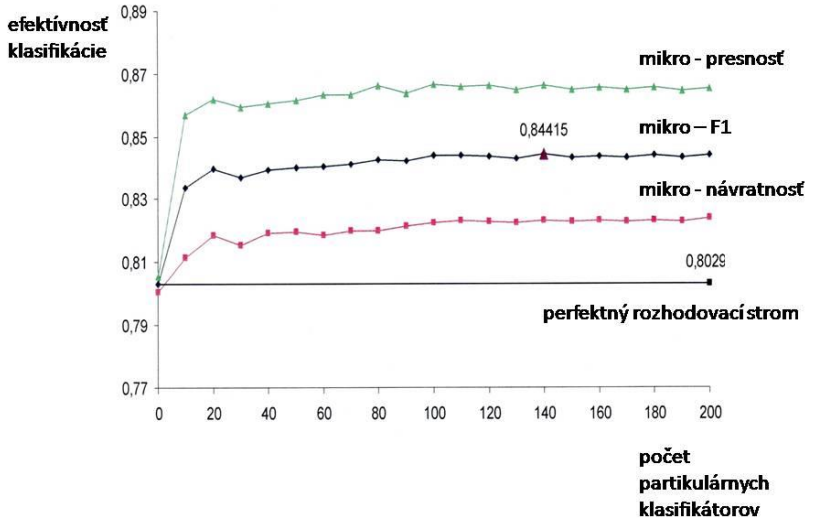
Efektívnosť baggingových stratégií bola overovaná porovnávaním s výsledkami klasifikácie perfektným rozhodovacím stromom. Perfektný rozhodovací strom plnil v týchto experimentoch úlohu silného klasifikátora (neorezaný). Efektívnosti tohto silného klasifikátora sa snažili vyrovnať slabé klasifikátory (orezané) skladaním hlasov väčšieho počtu slabých klasifikátorov v zloženej klasifikácii. Výsledok tohto porovnania je graficky znázornený na Obr.14 pre každú triedu zvlášť. Triedy boli usporiadané klesajúco podľa frekvencie ich výskytu zľava do prava. Interpretácia je nasledovná: klasifikácia pomocou baggingových stratégií dáva lepšie výsledky ako perfektný rozhodovací strom pri frekventovanejších kategóriách (ľavá polovica obrázku, kde je viac špičiek smerom nahor). Pri kategóriách s nízkou frekvenciou výskytu dáva lepšie výsledky perfektný rozhodovací strom.



Obr.14 Rozdiely v presnosti medzi zloženou klasifikáciou založenou na baggingu a perfektným rozhodovacím stromom na kolekcii dokumentov Reuter's

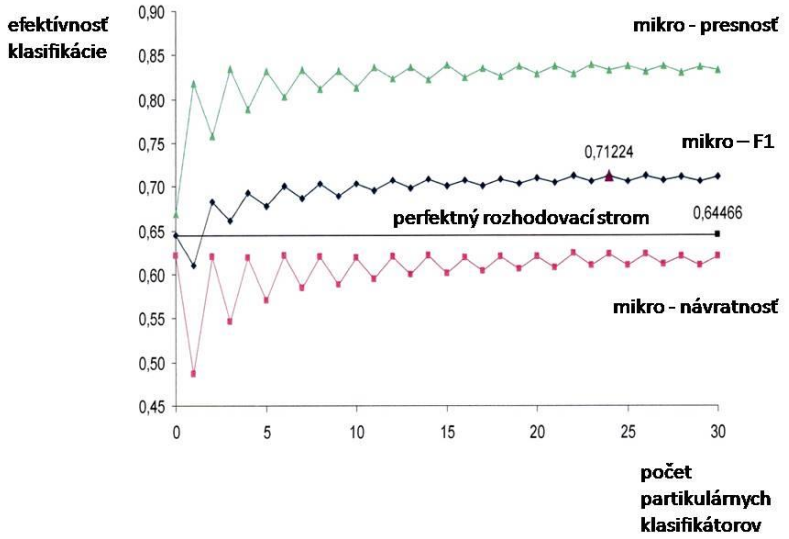
V rámci *určenia minimálneho počtu partikulárnych klasifikátorov* bola skúmaná aj závislosť efektívnosti metódy bagging od počtu klasifikátorov v zloženom klasifikátore. Boli vykonané experimenty pri ktorých bolo najprv uvažovaných 200 klasifikátorov a tento počet bol postupne redukovaný, pričom sa zaznamenávala zmena efektívnosti klasifikácie. Takto získané výsledky (viď Obr.15) boli porovnávané s efektívnosťou perfektného rozhodovacieho stromu, ktorá je znázornená rovnou čiarou.

Z Obr.15 je možné vyčítať, že metóda bagging dosahuje stabilné výsledky už pri počte 40 partikulárnych klasifikátorov pre kolekciu Reuter's. To znamená, že zvyšovanie počtu klasifikátorov už nezlepší výrazne efektívnosť klasifikácie. Uspokojivé výsledky by sa dosiahli už pri počte 20 partikulárnych klasifikátorov. Podobné experimenty boli vykonané nad dokumentmi internetového portálu televíznej spoločnosti Markíza. Výsledky sú ilustrované obrázkom Obr.16. Z tohto obrázka je možné vyčítať, že 10 až 15 partikulárnych klasifikácií postačí na dosiahnutie stabilných výsledkov zloženej klasifikácie.



Obr.15 Efektívnosť klasifikácie baggingom v závislosti od počtu partikulárnych klasifikátorov v zloženom klasifikátore na kolekcii dokumentov Reuter's

Podrobnejší opis experimentov s baggingom je možné nájsť v [19]. Ako bolo spomenuté, každý partikulárny klasifikátor je trénovaný na podmnožine dostupnej dátovej množiny a môže sa stať, že sa niektoré záznamy zvýšia. Zvyšné, nepoužité záznamy je možné použiť na Hold-Out validáciu partikulárneho modelu. Vyčíslením presnosti všetkých partikulárnych modelov a ich spriemernením je možné sa rýchlo dostať k odhadu presnosti zloženého klasifikátora – modelu. V takom prípade nie je nutná krížová validácia.



Obr.16 Efektívnosť klasifikácie baggingom v závislosti od počtu partikulárnych klasifikátorov v zloženom klasifikátore na kolekcii dokumentov Markíza

Technika Baggingu sa väčšinou používa na skladanie partikulárnych klasifikátorov typu rozhodovací strom. Avšak, ako partikulárny klasifikátor môže byť použitá aj niektorá z metód regresnej analýzy. Pri riešení úlohy regresie, generujeme regresný strom toľkokrát, koľko máme výberov trénovacej množiny. Zo všetkých odhadov danej premennej za všetky regresné stromy vypočítame priemernú hodnotu a to bude vlastne výsledok hlasovania komisie slabých klasifikátorov. Typickým príkladom baggingu je aj zložený model náhodného lesa (random forest), ktorému je venovaná nasledujúca podkapitola.

5.2 Náhodné lesy

Metóda *náhodného lesa* (Random Forests) [15], [20] generuje zložený klasifikátor podobne ako metóda baggingu. Je to modifikácia baggingu, ktorá buduje veľkú kolekciu de-korelovaných stromov a spriemerňuje ich. Tento zložený klasifikátor je tvorený viacerými

rozhodovacími stromami bez orezania. Vyšší počet partikulárnych rozhodovacích stromov umožňuje vyšší počet predikcií a teda aj vyššiu presnosť zloženej predikcie pomocou hlasovania v porovnaní s individuálnym stromovým modelom. Jednotlivé predikcie by mali byť na sebe nezávislé, teda aj jednotlivé stromové modely by mali byť nezávisle a práve preto sa používa náhodný výber atribútov pre každý stromový model. Kvôli náhodnému výberu atribútov sa tejto technike hovorí *náhodné lesy*. Výsledná trieda podľa zloženého klasifikátora sa určuje hlasovaním všetkých partikulárnych rozhodovacích stromov, podobne ako v baggingu. Keďže sú jednotlivé partikulárne stromy na sebe nezávislé, je vhodné a jednoduché ich spracovať paralelne. Ďalej, aj náhodný výber trénovacej množiny každého partikulárneho stromu umožňuje ho validovať na dátach, ktoré neboli vybraté na jeho trénovanie, čo uľahčuje validáciu náhodného lesa. Tento prístup je teda rýchly a dosť presný, preto niet divu, že sa v posledných rokoch používa veľmi často. Na druhej strane potrebuje zadať niektoré parametre a to počet rozhodovacích stromov v modeli a počet náhodne zvolených atribútov v každom strome.

Teda náhodné lesy je možné považovať za vylepšenie techniky Baggingu spôsobom, ktorý „de-koreluje“ stromy, inak povedané robí partikulárne klasifikátory (stromy) na sebe nezávislé. Špecifikom náhodných lesov je, že ako partikulárny klasifikátor sa používa výhradne rozhodovací strom. Na generovanie každého partikulárneho stromu sa nepoužíva celá množina atribútov, ale iba náhodne zvolená podmnožina týchto atribútov. Konkrétne, pri výbere testovacieho atribútu v uvažovanom uzle stromu je bratých do úvahy povedzme m atribútov z celkového počtu p atribútov. Ale zvolený je len jeden z m atribútov. Pri ďalšom uzle (poduzle) je znova uvažovaných iba m atribútov, ale to iná nová podmnožina m atribútov (iný výber podmnožiny) ako v predchádzajúcom prípade a znova je zvolený jeden atribút, podľa hodnôt ktorého sú generované ďalšie vetvy stromu. Teda zakaždým, keď potrebujeme uskutočniť delenie uzlu a vytvorenie nových vetiev stromu, zakaždým uvažujeme celkom novú, náhodne zvolenú podmnožinu

atribútov.

Musíme si uvedomiť aj to, že pri každom delení stromu v rámci náhodných lesov, algoritmu nie je dovolené uvažovať väčšinu atribútov – prediktorov [1]. Aj keď sa to zdá byť na prvý pohľad neracionálne, má to svoj dôvod. Predpokladajme, že v dátovej množine existuje jeden veľmi silný prediktor v rámci skupiny mierne silných prediktorov. Potom zákonite väčšina partikulárnych stromov použije práve tento silný prediktor ako testovací atribút vo svojom koreni. Potom by sa mohli jednotlivé partikulárne stromy dosť podobať a teda bude medzi nimi silná korelácia, čo je nežiadúce. Spriemerňovanie vysoko korelovaných klasifikátorov (stromov) neprinesie dosť veľkú redukciu odchýlky a teda ani veľkú výhodu v porovnaní s použitím jedného stromu namiesto lesa. A práve preto sa pri delení stromu používa iba podmnožina prediktorov a to vždy iná. To vedie k tomu, že pravdepodobnosť uvažovania silného prediktora pri delení stromu bude iba $(p-m)/p$ a teda aj iné prediktory dostanú viac šance. Tento princíp sa nazýva procesom de-korelácie stromov a vďaka nej budú vygenerované stromy lepšie spolupracovať a dávať spoľahlivejšie výsledky.

Teda hlavným rozdielom medzi náhodnými lesmi a baggingom založeným na generovaní stromov ako partikulárnych klasifikátorov je voľba veľkosti m podmnožiny prediktorov. Napríklad, ak $m=p$, potom náhodný les sa stáva baggingom. Čím menšia hodnota m bude použitá pri budovaní náhodného lesa, tým spoľahlivejší náhodný les vznikne, hlavne v prípade, ak medzi prediktormi bolo veľké množstvo korelovaných prediktorov.

Pri riešení väčšiny problémov konkrétny výkon a presnosť náhodných lesov sú veľmi podobné technike boostingu, ktorá bude popísaná v nasledujúcej podkapitole. Avšak náhodné lesy je možné jednoduchšie natrénovať a ladiť ako boosting. Možno aj preto sú náhodné lesy populárne a sú v súčasnosti aj implementované v množstve softvérových balíčkov zameraných na strojové učenie. Keďže stromy, či už regresné alebo klasifikačné, sú notoricky zašumené, veľmi získavajú na presnosti práve

v procese spriemerňovania, ktoré je základom učenia súborom metód. Algoritmus náhodného lesa určeného pre regresiu alebo klasifikáciu je podľa [10] nasledovný:

Začiatok algoritmu

1. Pre b = 1 až B platí:
   a. Z trénovacích dát vyber vzorku Z^*, ktorá bude veľkosti N
   b. Na týchto dátach nechaj vyrásť strom náhodného lesa T_b tak, že rekurzívne budú opakované nasledovné kroky pri každom terminálnom uzle stromu, až kým sa nedosiahne stanovená hodnota minimálnej veľkosti uzlu.
      i. Náhodne vyber m premenných z celkového počtu p premenných.
      ii. Vyber najlepšiu premennú pre delený uzol stromu z týchto m premenných.
      iii. Rozdeľ uzol stromu do dvoch dcérskych uzlov.
2. Výstupom je súbor stromov $\{T_b\}_1^B$.

Aby mohla byť vykonaná predikcia pre nové pozorovanie x, musí platiť:

Pre regresiu:

$$\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{50}$$

Pre klasifikáciu: Nech $\hat{C}_b(x)$ je predikcia triedy podľa b-tého stromu náhodného lesa. Potom platí (51)

$$\hat{C}_{rf}^B(\mathrm{x}) = \text{majoritný hlas}\left\{\hat{C}_b(x)\right\}_1^B. \tag{51}$$

Koniec algoritmu.

Samotný autor tohto algoritmu Leo Breiman aj iní považujú

náhodné stromy za najsprávnejšie a najlepšie interpretovateľné zo všetkých techník učenia súborom metód. Naviac vyžadujú veľmi málo ladenia. Dosahujú iba 4,88% klasifikačnej chyby, pričom bagging dosahuje 5,4% chybu klasifikácie. Ďalšou výhodou náhodných lesov je, že nemôžu byť preučené.

Dôležitou vlastnosťou náhodných lesov je používanie takzvaných „Out-Of-Bag“ (OOB) výberov, ktoré sú definované nasledovne. Pre každé pozorovanie (TP) $z_i = (x_i, y_i)$ je zvolený jeho prediktor náhodného lesa spriemernením iba tých stromov, ktoré korešpondujú s tými príkladmi, v ktorých sa z_i nevyskytuje. Odhad OOB chyby je takmer identický tomu odhadu, ktorý bol získaný N násobnou krížovou validáciou.

5.3 Boosting

Ďalší z prístupov k učeniu súborom metód je založený na myšlienke použitia váh trénovacích príkladov pri generovaní konkrétneho výberu trénovacej množiny. Tento prístup sa nazýva metóda „boosting“ [21] a [22]. Boosting je jednou z najsilnejších nápadov ako uskutočňovať strojové učenie v posledných dvaciatich rokoch. Podobne ako bagging aj boosting je všeobecným prístupom, ktorý môže byť aplikovaný na mnohé štatistické učiace metódy tak pre regresiu ako aj klasifikáciu. My sa ale budeme obmedzovať na aplikáciu tohto prístupu na rozhodovacie stromy. Motiváciou pre boosting je kombinovať výstupy mnohých slabých klasifikátorov za účelom vytvorenia výkonnej klasifikačnej „komisie“. Z tohto hľadiska sa zdá že jednoducho podobný baggingu a iným prístupom k učeniu založeným na hlasovaní. V zdroji [10] sa dokazuje, že boosting je od základu iným prístupom. Vychádza sa z faktu, že slabý klasifikátor je iba trochu lepší ako náhodný generátor. Účelom boostingu je opakovane aplikovať slabý klasifikátor v sekvencii nad dátami, ktoré sú opakovane modifikované. Výsledkom tohto procesu je sekvencia slabých klasifikátorov. Predikcie všetkých týchto klasifikátorov sú kombinované do väčšinového hlasu podľa vzorca (49).

Čím sa teda hlavne líši boosting od baggingu? Partikulárne stromy sa trénujú sekvenčne a každý nový partikulárny strom by mal byť lepší, lebo sa učí z chýb predošlého. Každý nový strom je generovaný z modifikovanej verzie originálnej dátovej množiny. Teda všetky výbery trénovacej množiny v rámci boostingu obsahujú celú trénovaciu množinu, ale v každom výbere majú príklady iné váhy. Táto myšlienka sa dá jednoducho ilustrovať na príklade: ak niektorý trénovací príklad má váhu 2, akoby sa v trénovacej množine nachádzal dvakrát. Keď má váhu 0, akoby sa v trénovacej množine nenachádzal. Samozrejme vo všeobecnosti váha nemusí byť celočíselná. Väčšinou to býva číslo od 0 do 1.

Táto metóda učenia súborom metód podľa [23] využíva jeden učiaci algoritmus a spoločnú trénovaciu množinu na trénovanie súboru klasifikačných modelov, ktoré sa líšia iba tým, že jednotlivé trénovacie príklady mali v procese učenia iné váhy. Na začiatku majú všetky trénovacie príklady rovnakú váhu. Po natrénovaní prvého modelu sú váhy trénovacích príkladov prvýkrát zmenené v závislosti od správnosti predpovede modelu. To sa potom opakuje toľkokrát, koľko partikulárnych modelov je potrebné naučiť. Váhy príkladov, ktoré boli správne klasifikované sa znižujú, pretože ich dôležitosť pre ďalšie trénovanie klesla, keďže v ich prípade bola dosiahnutá úspešná predpoveď triedy. Naopak, príkladom, ktoré boli v predchádzajúcej iterácii nesprávne klasifikované sa priradí väčšia váha. Teda väčší dôraz sa kladie na príklady nesprávne klasifikované a na následnú korekciu ďalším partikulárnym klasifikátorom naučeným z ďalšieho výberu trénovacej množiny s novými váhami. Ako koeficienty váhovania sa používajú presnosti jednotlivých partikulárnych modelov. Po zmenení váh trénovacích príkladov sa opakuje trénovací proces, ktorého výsledkom je skupina modelov alebo inak povedané zložený klasifikátor. Finálna klasifikácia sa tvorí ako hlasovanie partikulárnych klasifikácií, podobne ako pri metóde „bagging“.

Teda metódu boosting je možné považovať za zdokonalenie metódy bagging. Podobne ako pri metóde bagging aj metóda boosting generuje postupnosť

partikulárnych klasifikátorov H_m, $m=1,\ldots,M$ pre M rôznych výberov trénovacej množiny. Tieto klasifikátory sa kombinujú do výsledného klasifikátora presne tým istým spôsobom ako pri baggingu. Trénovacia množina je modifikovaná prostredníctvom distribúcie váh trénovacích príkladov - dokumentov $d_i \in D$, kde D je kolekcia dokumentov. Množina váh je pred naučením prvého klasifikátora nastavená uniformne. Pre každú iteráciu sa zvýšia váhy tých trénovacích príkladov, ktoré boli nesprávne klasifikované predchádzajúcim klasifikátorom H_{m-1}. Naopak váhy tých príkladov, ktoré boli klasifikované správne sa znížia. Proces učenia sa takto sústredí na nesprávne klasifikované príklady. Najznámejší z boostingových algoritmov AdaBoost.MH2 [22] reprezentuje algoritmus klasifikácie do viac ako dvoch tried. Tento algoritmus generuje klasifikátory $H_m: DxC \rightarrow R$, ktoré predikujú triedy $c_j \in C$ na základe rozhodovacej funkcie $sign[H(d_i, c_j)]$. V experimentoch bol použitý nasledovný boostingový algoritmus:

Začiatok algoritmu

1.Inicializuj distribúciu váh $w_{1(i,j)} = 1 / (|D||C|)$,
$i = 1, \ldots, |D|$, $j = 1, \ldots, |C|$

2.Pre $m = 1, \ldots, M$

2.1.Generuj klasifikátor $H_m: D \times C \rightarrow R$

2.2.Urči parameter $\alpha_m \in R$.

2.3.Modifikuj distribúciu váh podľa pravidla (52)

$$w_{m+1(i,j)} = \frac{w_{m(i,j)} \exp(-\alpha_m y_{i,j} H_m(d_i, c_j))}{Z_m} \qquad (52)$$

kde Z_m je normovacia konštanta, ktorá garantuje, že platí (53)

$$\sum_{i=1}^{|D|} \sum_{j=1}^{|C|} w_{m+1(i,j)} = 1 \tag{53}$$

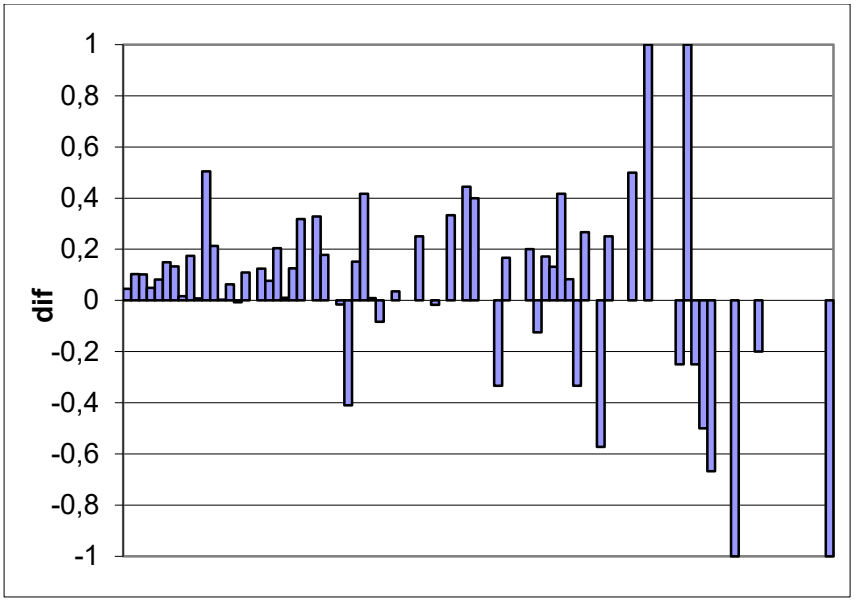
3. Výstupom je zložený klasifikátor, ktorého matematický model je uvedený vo vzťahu (49).

Koniec algoritmu

Premenná $y_{i,j}$ je určená ako $y_{i,j} = +1$ ak $d_i \in c_j$ a ako $y_{i,j} = -1$ ak $d_i \notin c_j$. Opísaný algoritmus bol modifikovaný tak, aby výpočet váh nebol ovplyvňovaný nepresnosťou spôsobenou zaokrúhľovaním váh.

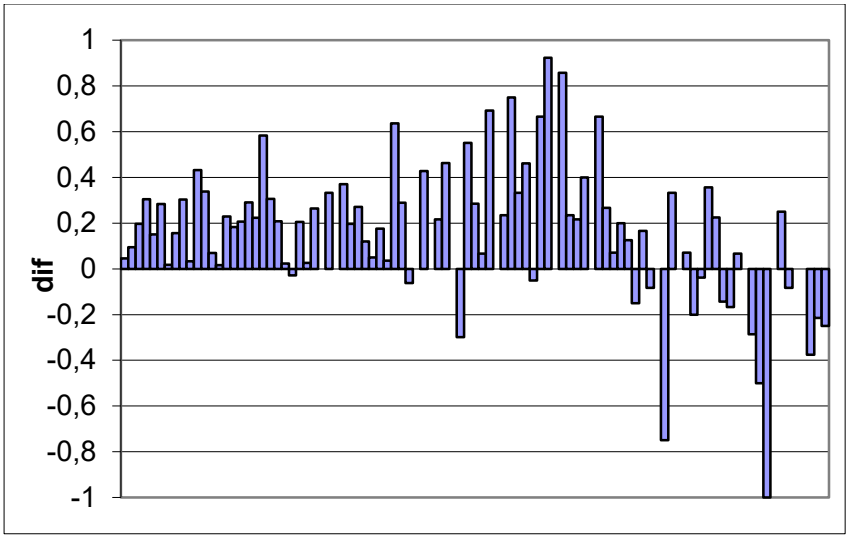
Bola vykonaná séria testov metódy boosting nad dvoma kolekciami dokumentov Reuter's-21578 a internetového portálu televízie Markíza. Vykonané testy ukázali zmysel použitia metódy boosting. Pomocou experimentov sa podarilo zistiť minimálny počet partikulárnych klasifikátorov potrebných na zlepšenie výsledkov slabej klasifikácie ako aj počet klasifikátorov, zvyšovaním ktorého sa už presnosť klasifikácie výrazne nezvýši.

V rámci testovania *efektívnosti metódy boosting* bolo zistené, že metóda boosting (kladné hodnoty rozdielu „dif") dáva vo všeobecnosti lepšie výsledky ako perfektný rozhodovací strom (záporné hodnoty rozdielu „dif"), ako ilustruje Obr.17 pre kolekciu Reuter's a Obr.18 pre kolekciu Markíza.



Obr.17 Rozdiely v presnosti medzi zloženým klasifikátorom založenými na boostingu a perfektným rozhodovacím stromom na kolekcii Reuter‘s

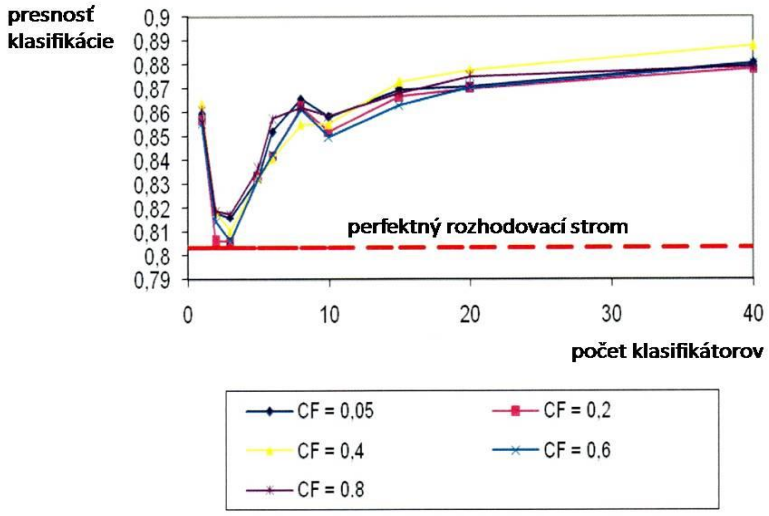
Presnosť klasifikácie bola vyhodnotená pre každú triedu – kategóriu zvlášť. Kategórie boli usporiadané klesajúco podľa frekvencie ich výskytu. Z výsledkov ilustrovaných v Obr.17 a Obr.18 vyplýva, že zložený klasifikátor (boosting) dáva lepšie výsledky ako perfektný rozhodovací strom pre vyššie frekvencie výskytu. Tento záver je odôvodniteľný, keďže nízky počet výskytov niektorej kategórie v trénovacej množine neposkytuje dostatok informácie pre mechanizmus zloženej klasifikácie.



Obr.18 Rozdiely v presnosti medzi zloženým klasifikátorom založeným na boostingu a perfektným rozhodovacím stromom na kolekcii „Markíza“

Bolo potrebné taktiež určiť *minimálny počet partikulárnych klasifikátorov.* Počet partikulárnych klasifikátorov bol limitovaný počtom 100 klasifikátorov. Následne bol počet klasifikátorov znižovaný a bol sledovaný dopad na efektívnosť klasifikácie.

Obr.21 ilustruje skutočnosť, že už pri približne 5 základných klasifikátoroch dával boosting výsledky porovnateľné s perfektným rozhodovacím stromom (v Obr.19, 20 a 21 je perfektný rozhodovací strom reprezentovaný prerušovanou čiarou) ak bola efektívnosť zloženej klasifikácie meraná parametrom F_1. Nad počtom približne dvadsať základných klasifikátorov bola efektívnosť zloženej klasifikácie prakticky konštantná a o 5% lepšia ako pri perfektnom rozhodovacom strome. V ďalších parametroch efektívnosti boli dosiahnuté podobné výsledky. Konkrétne, na dosiahnutie uspokojivej presnosti klasifikácie (Obr.19) bol potrebný minimálny počet cca 15 klasifikátorov a na dosiahnutie uspokojivej návratnosti klasifikácie (Obr.20) bol potrebný minimálny počet cca 10 klasifikátorov.



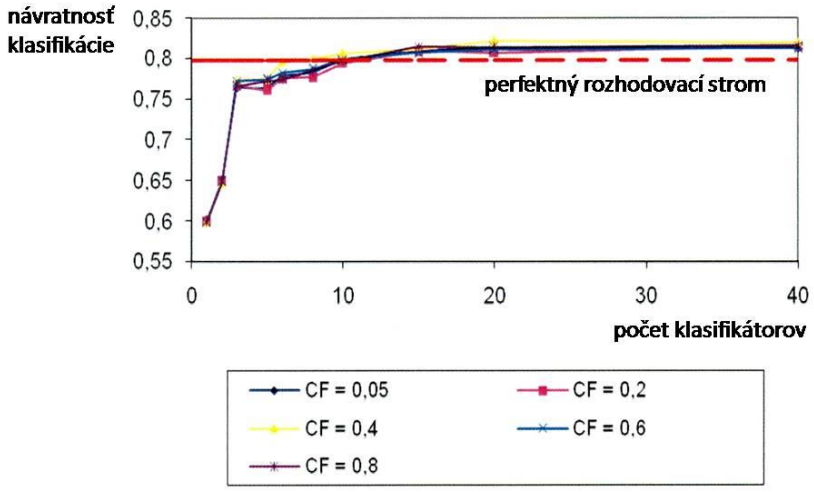
Obr.19 Závislosť presnosti zloženej klasifikácie od počtu základných klasifikátorov v boostingu na kolekcii Reuter's-21578

Zaujímavé je, že na Obr.19 spočiatku klesá presnosť klasifikácie s rastom počtu klasifikátorov v metóde boosting. Dôvodom je preučenie partikulárneho klasifikátora. Toto preučenie spôsobuje prechodný nárast zložitosti klasifikátora na úroveň perfektného rozhodovacieho stromu. Postupne sa presnosť boostingu vyrovná perfektnému rozhodovaciemu stromu a presiahne ho. Podrobný opis výsledkov uvedených experimentov s boostingom je možné nájsť v [24].

Je potrebné spomenúť, že použitie tak baggingových ako aj boostingových stratégií má zmysel iba vtedy, keď sa pracuje so slabými klasifikátormi. Silný klasifikátor sa nepotrebuje „radiť" s množstvom ďalších klasifikátorov.

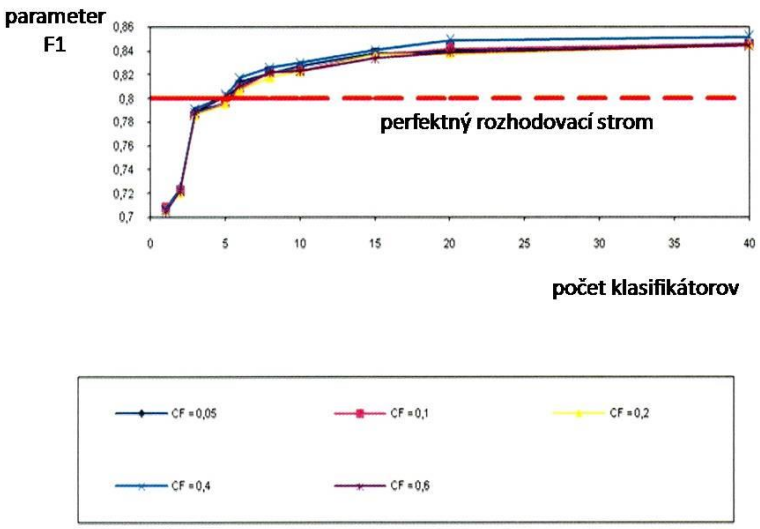
Ukázalo sa, že metódy zloženej klasifikácie (bagging, náhodné stromy a boosting) naozaj môžu zlepšiť výsledky klasifikácie slabými klasifikátormi. Avšak, použitie baggingu, náhodných stromov a boostingu má aj svoje nevýhody, a to je strata jednoduchosti, prehľadnosti

a ilustratívnosti singulárneho rozhodovacieho stromu. Taktiež sa zvyšuje výpočtová zložitosť.



Obr.20 Závislosť návratnosti zloženej klasifikácie od počtu základných klasifikátorov v boostingu na kolekcii Reuter's-21578

Porovnanie týchto metód preferuje boosting pred baggingom. Bagging je jednoduchšou metódou, avšak boosting dáva lepšie výsledky. Na kolekcii dokumentov Reuter's-21578 sa ukázalo, že uspokojivá efektívnosť klasifikácie sa dosahuje pri baggingu medzi 20 až 40 základnými klasifikátormi, zatiaľ čo pri boostingu sa uspokojivá efektívnosť klasifikácie dosahuje medzi 5 až 15 základnými klasifikátormi. Lepšie výsledky boostingu je možné odôvodniť faktom, že táto metóda používa váhovanie trénovacích príkladov. Keďže sa zvyšujú váhy príkladov nesprávne klasifikovaných v predchádzajúcej iterácii, proces učenia sa sústreďuje na nesprávne klasifikované príklady. Tým sa učenie zrýchľuje a spresňuje. Porovnanie oboch metód je uvedené aj v [25].



Obr.21 Závislosť parametra efektívnosti zloženej klasifikácie F1 od počtu základných klasifikátorov v boostingu na kolekcii Reuter's-21578

5.4 Stacking

Metóda Stackingu (Stacked Generalization) [14], [15] je staršou metódou ako Bagging a Boosting a teda v určitom slova zmysle ich predchodcom. V súčasnosti sa menej používa. Možno preto, že keďže kombinuje modely natrénované viacerými rozdielnymi metódami strojového učenia, je zložitejší a náročnejší na trénovanie ale aj na formovanie výsledného rozhodnutia. Stacking nevyužíva metódu hlasovania ani váhovania, ale takzvaný koncept meta-učenia. Proces učenia pri Stackingu pozostáva z dvoch častí. Najprv sa natrénujú partikulárne klasifikátory na tej istej trénovacej množine. Pri trénovaní každého partikulárneho klasifikátora sa použije iná metóda strojového učenia a teda bude mať špecifickú štruktúru ako aj silné a slabé stránky. V druhej časti sa natrénuje meta-klasifikátor, ktorý zohľadní výsledky čiastkových klasifikátorov ako aj ich presnosť. Výsledkom tohto

trénovania bude znalosť meta-klasifikátora o tom, ktoré partikulárne klasifikátory dávajú relevantné odpovede v akých špecifických prípadoch v rámci definovaných úloh buď predikcie alebo klasifikácie. To je možné reprezentovať pomocou klasifikačného stromu, ktorého 0-tú úroveň tvoria partikulárne klasifikátory a meta-klasifikátor reprezentuje úroveň 1.

Ak by však bola použitá tá istá trénovacia množina na trénovanie partikulárnych klasifikátorov ako aj meta-klasifikátora, potom by tu bolo reálne riziko preučenia. Riešením by mohlo byť rozdelenie dátovej množiny na dve časti. Prvá časť by sa potom použila na trénovanie partikulárnych klasifikátorov a testovanie meta-klasifikátora, zatiaľ čo druhá časť by sa použila na trénovanie meta-klasifikátora a testovanie partikulárnych klasifikátorov. Všetky modely by potom boli trénované a testované na nezávislých dátach.

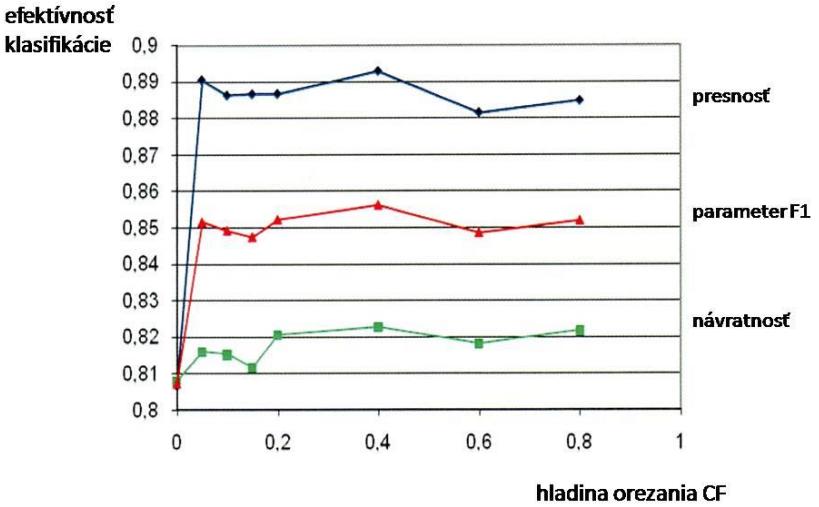
5.5 Diskusia k učeniu súborom metód

V predchádzajúcich podkapitolách boli popísané experimenty s metódami bagging a boosting, teda metódami zloženej klasifikácie, ktoré v úlohe partikulárnych klasifikátorov používali binárne rozhodovacie stromy vygenerované algoritmom C4.5. Avšak tento algoritmus generuje perfektné rozhodovacie stromy. Pre overenie efektívnosti, resp. výkonnosti zloženej klasifikácie bolo potrebné použiť slabé klasifikátory, ako už bolo spomenuté. Slabé klasifikátory boli v experimentoch získané orezaním perfektných binárnych rozhodovacích stromov. Efektívnosť týchto slabých klasifikátorov bola v experimentoch porovnávaná s efektívnosťou perfektného rozhodovacieho stromu, ktorý nebol orezaný. Taktiež bolo sledované či a ako sa zvýši efektívnosť slabých klasifikátorov so zvyšovaním ich počtu v zloženej klasifikácii (bagging a boosting) stále v porovnaní s perfektným rozhodovacím stromom.

V súvislosti s generovaním slabých klasifikátorov vyvstala otázka, ako veľmi ich orezať, teda na akej hladine významnosti *CF* a akým algoritmom. Hladina významnosti

v orezávacej technike sa niekedy stručne označuje ako hladina orezania. V našich experimentoch bola použitá konkrétne orezávacia technika „*algoritmus pesimistického odhadu chyby*“, ktorý tento odhad určuje vypočítaním štandardnej odchýlky odhadnutej presnosti za predpokladu binominálneho rozdelenia. Pre danú hladinu významnosti je spodná hranica odhadu vzatá ako veľkosť sily pravidla.

Pre veľké množiny dát je pesimistický odhad veľmi blízko k presnosti na trénovacích dátach (štandardná odchýlka je veľmi malá). Napriek tomu, že táto metóda nie je štatisticky úplne korektná, našla si využitie v praxi. Obr.22 ilustruje ako sa mení výkonnosť klasifikátora, konkrétne rozhodovacieho stromu, v závislosti od hladiny orezania *CF*. Najlepšie výsledky boli dosiahnuté pre *CF=0.4*. Preto v rámci experimentov s baggingom a boostingom bola použitá práve táto hladina orezania.



Obr.22 Parametre efektívnosti klasifikácie rozhodovacími stromami v závislosti od hladiny orezania *CF*

Vyššie prezentované techniky učenia súborom metód generujú takzvané zložené klasifikátory, teda modely primárne určené na riešenie úlohy klasifikácie do tried. Ale

jednotlivé spomínané techniky je možné veľmi efektívne aplikovať s istými obmenami aj na riešenie úlohy regresie. Napríklad namiesto mechanizmu hlasovania môže byť použitý aritmetický priemer alebo namiesto váhovaného hlasovania sa môže použiť váhovaný priemer.
Konkrétne techniky Bagging, Boosting a Stacking je možné použiť pre úlohy regresie pod podmienkou určitých zmien, ktoré vyplývajú z charakteru cieľovej veličiny. Napríklad v prípade Boostingu [26] je potrebné nahradiť určovanie správne či nesprávne klasifikovaných trénovacích príkladov porovnávaním absolútnej alebo kvadratickej chyby aktuálneho trénovacieho príkladu voči priemernej chybe trénovacej množiny. Ak je chyba nižšia (vyššia) ako priemerná, potom sa bude váha daného trénovacieho príkladu znižovať (zvyšovať).
Jednou z techník zloženého učenia je aj takzvaná *Aditívna regresia* (Additive Regression), ktorá bola špeciálne navrhnutá pre úlohu regresie [14]. Táto technika používa rozdielne typy regresných modelov a vytvára z nich zložený model. Spojenie rozdielnych typov modelov prináša zisk v podobe vzájomného dopĺňania a kompenzácie slabých stránok. Táto technika svojou podstatou pripomína techniku Stackingu. Aditívna regresia štartuje s prázdnou množinou modelov a postupne sa pridávajú partikulárne modely, ktoré sa trénujú sekvenčne, aby bola splnená podmienka, že presnosť celkového zloženého modelu sa nebude znižovať. Po natrénovaní nového partikulárneho modelu sa zmenia váhy trénovacích príkladov na základe dosiahnutej presnosti. Toto pripomína techniku Boosting s tým rozdielom, že sa používajú viaceré typy algoritmov strojového učenia zameraných na regresiu, pričom pri Boostingu sa to nerobí. Tam sa používa jedna metóda strojového učenia pre všetky partikulárne klasifikátory.

6 METÓDY ZALOŽENÉ NA STROMOCH

V rámci strojového učenia sa reprezentácia pomocou stromu, resp. acyklického grafu, využíva tak pri klasifikačných ako aj regresných metódach. Stromový graf reprezentuje segmentáciu priestoru príznakov alebo priestoru prediktorov do určitého počtu jednoduchých oblastí. Taký strom predstavuje sumár pravidiel na segmentáciu priestoru príznakov (prediktorov). Takéto prístupy, založené na segmentácii priestoru príznakov, alebo inak povedané na delení priestoru príkladov na podpriestory, sa zvyknú nazývať „metódy založené na stromoch“. Typickým znakom týchto metód založených na stromoch je jednoduchá interpretácia, ktorá sa ale stráca v prístupoch ako bagging, náhodné stromy a boosting, ktoré sú založené na generovaní veľkého počtu stromov, ktoré sú kombinované do jednoduchej predikcie vychádzajúcej z konsenzu.

Stromové grafy v oblasti strojového učenia sa typicky nazývajú rozhodovacie stromy a je možné ich aplikovať tak na regresné ako aj klasifikačné metódy.

6.1 Regresné stromy

Proces budovania regresného stromu pozostáva v jeho najjednoduchšej podobe z dvoch krokov:

1. Rozdelenie priestoru príznakov (prediktorov) použitím možných hodnôt $X_1, X_2, ..., X_p$ na J neprelínajúcich sa oblastí $R_1, R_2, ..., R_j$.
2. Pre každé pozorovanie, ktoré padne do oblasti Rj, potom pre tento príklad sa vykoná predikcia vo forme odpovede spojenej s oblasťou R_j.

Napríklad, predstavme si, že v rámci prvého kroku dostaneme dve oblasti R_1 s odpoveďou 10 a R_2 s odpoveďou 20. Potom pre dané pozorovanie $X = x$, ak $x \in R_1$ (resp. R_2), tak bude predikovaná hodnota 10 (resp. 20).

Ako teda vieme skonštruovať – vytvoriť jednotlivé oblasti R_1 až R_j ? Musíme minimalizovať RSS (Residual Sum of

Squares), inak povedané štvorcovú chybu danú (54):

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i - \hat{y}_{R_j})^2 \tag{54}$$

Kde ^y_{R_j} je očakávaná odpoveď pre trénovacie pozorovanie j-tého boxu. Nanešťastie, je výpočtovo nemožné uvažovať a preveriť každé možné delenie do *J* boxov. Preto sa používa lakomý alebo zištný „greedy“ prístup zhora-dole známy ako „rekurzívne binárne delenie“. Pri každom delení sa vytvoria dve nové vetvy v strome smerom nadol. Tento prístup bol nazvaný lakomý, lebo v každom kroku v procese budovania stromu sa berie jedno najlepšie delenie a nepozerá sa na to aké iné delenie by mohlo priniesť lepší strom v nejakom budúcom kroku.

V procese rekurzívneho binárneho delenia stromu musíme najprv zvoliť prediktor – atribút X_j a pre neho takzvaný „cutpoint“ bod rezu *s* tak, aby rozdelenie priestoru daného atribútu na dve oblasti $\{X|X_j < s\}$ a $\{X|X_j >= s\}$ viedlo k minimálnej chybe RSS. Ak budeme uvažovať všetky atribúty X_1 až X_p a všetky možné hodnoty bodov rezov pre každý atribút je potrebné pri každom delení zvoliť testovací atribút a jeho bod rezu tak aby výsledný strom mal najmenšiu chybu RSS. Konkrétnejšie, pre každé *j* a *s* definujeme páry oblastí R_1 a R_2 nasledovne:

$$R_1(j,s) = \{X|X_j < s\}\ a\ R_2(j,s) = \{X|X_j \geq s\} \tag{55}$$

To znamená, že musíme nájsť *j* a *s* také, aby minimalizovali rovnicu (56).

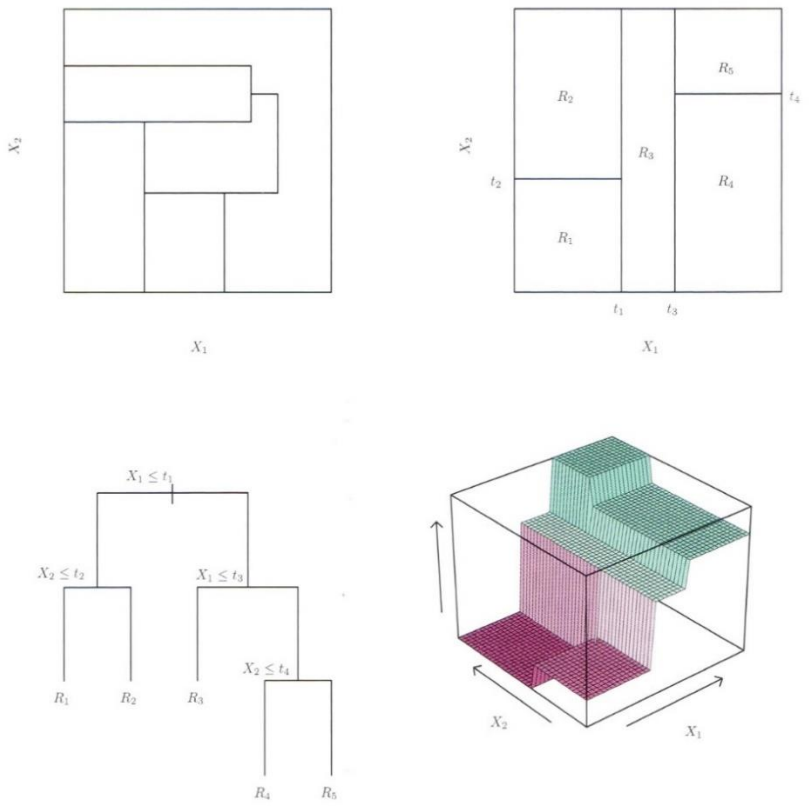
$$\sum_{i:x_i\in R_1(j,s)}(y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i\in R_2(j,s)}(y_i - \hat{y}_{R_2})^2 \tag{56}$$

kde $\hat{y}_{R_1}$ je odpoveď pre trénovací príklad z oblasti $R_1(j,s)$ a

$\hat{y}_{R_2}$ je odpoveď pre trénovací príklad z oblasti $R_2(j, s)$.

Tento proces opakujeme a hľadáme najlepší prediktor – atribút a najlepší bod rezu aby sme vytvorili delenie v uvažovanej oblasti také, aby RSS bolo minimálne. To znamená, že uvažovanú oblasť ďalej delíme na podoblasti. Tento proces hľadania sa opakuje dovtedy, kým nie je splnená ukončovacia podmienka, ktorá napríklad musí zabezpečiť, aby počet príkladov v každej podoblasti neklesol pod 5 príkladov. Ilustrácia výsledného delenia na päť podoblastí je na Obr.23.

Takto vygenerované stromy môžu byť príliš komplexné (zložité). Menšie stromy sú lepšie z hľadiska interpretovateľnosti a menšieho biasu. Menšie stromy by sa dali dosiahnuť aj tak, že by generovanie stromu skončilo, ak by RSS prekročilo nejakú vopred stanovenú hranicu. Avšak lepšia stratégia je dovoliť, aby bol vygenerovaný veľký strom a potom ho orezať. Toto orezanie prestavuje nájdenie podstromu, ktorý bude mať nižšiu testovaciu chybu ako pôvodný preučený strom. Avšak testovanie každého z možných podstromov by bolo príliš ťažkopádne, keďže ich môže byť veľmi veľa. Preto je vhodné vybrať z tohto množstva malú množinu podstromov a tú uvažovať. Na to nám môže poslúžiť metóda *Orezávanie cena – komplexnosť* (Cost Complexity Pruning) taktiež známa ako *Orezávanie najslabšieho spojenia* (Weakest Link Pruning) [1]. Táto metóda sa sústreďuje iba na sekvenciu stromov s pozitívnym (v zmysle pozitívneho čísla) ladiacim parametrom α. Takéto budovanie regresného stromu je možné uskutočniť pomocou nasledovného algoritmu.



Obr.23 Rozdelenie dvoj dimenzionálneho priestoru príznakov (atribútov) na päť oblastí - partícií. Ľavý Horný: delenie nie je výsledkom rekurzívneho binárneho delenia. Pravý Horný: výstup rekurzívneho binárneho delenia. Ľavý Dolný: binárny strom, ktorý odpovedá deleniu v pravom hornom obrázku. Pravý Dolný: mapa predikčných plôch (oblastí) v perspektíve, ktorá odpovedá stromu v ľavom dolnom obrázku [1].

Začiatok algoritmu budovania regresného stromu:

1. Generujeme veľký strom aplikovaním rekurzívneho binárneho delenia nad trénovacími dátami. Ukončíme toto generovanie vtedy, keď každý terminálny (listový) uzol bude obsahovať menej trénovacích príkladov, ako je stanovený minimálny počet.

2. Aplikujeme „orezávanie cena komplexnosť“ na v prvom kroku vygenerovaný veľký strom a získame postupnosť najlepších podstromov, ako funkcie parametra *α*.
3. Vykonajme K-násobnú krížovú validáciu na získanie parametra *α*, pričom sa trénovacia množina rozdelí do K ohrád (podmnožín).
    a. Opakujme kroky 1. a 2. nad všetkými podmnožinami okrem k-tej.
    b. Testujme štvorcovú chybu predikcie na dátach z k-tej podmnožiny ako funkciu *α.*

    Vypočítajme priemernú hodnotu chyby pre každú hodnotu *α* a vyberme *α* s minimálnou priemernou chybou.
4. Vráťme podstrom z kroku 2., ktorý odpovedá zvolenej hodnote *α*.

Koniec algoritmu.

Pre každú hodnotu α existuje odpovedajúci podstrom $T \subset T_0$ taký, že hodnota výrazu vo formule (57) je tak malá ako je to len možné.

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{57}$$

Absolútna hodnota $|T|$ reprezentuje počet terminálnych uzlov stromu T a R_m odpovedá m-tému terminálnemu (listovému) uzlu. Ak sa parameter *α* rovná 0, potom sa $T = T_0$.

6.2 Klasifikačné stromy

Klasifikačný strom sa od regresného stromu líši hlavne v tom, že sa používa na predikciu skôr kvalitatívnej ako kvantitatívnej odpovede. Pri regresnom strome je predikovaná odpoveď pre daný príklad daná priemernou

hodnotou všetkých odpovedí (v tvare numerických hodnôt) všetkých trénovacích príkladov, ktoré patria do toho istého terminálneho (listového) uzla, teda do tej istej oblasti. Na druhej strane pri klasifikačnom strome je predikovaná najfrekventovanejšie sa vyskytujúca trieda medzi trénovacími príkladmi v oblasti, do ktorej patrí aj príklad (pozorovanie), pre ktorý sa vykonáva predikcia. V tomto prípade nie je možné použiť RSS ako kritérium na vetvenie stromu. Alternatívou k tomu môže byť odhad klasifikačnej chyby (Classification Error Rate), ktorý reprezentuje chybne klasifikované trénovacie príklady uvažovanej oblasti, ktoré nepatria do predikovanej triedy. Častejšie sa v oblasti strojového učenia klasifikačné stromy nazývajú rozhodovacími stromami.

6.2.1 Reprezentácia naučenej znalosti rozhodovacím stromom

Rozhodovací strom je reprezentácia pojmu, ktorá veľmi názorne ilustruje proces učenia. Preto je táto reprezentácia vhodná na riešenie úloh, ktoré vyžadujú spoluprácu s odborníkmi iných vedných odborov a teda laikmi v strojovom učení. Rozhodovacie stromy sa môžu používať v rôznych oblastiach, napríklad v znalostných systémoch na automatické generovanie báz znalostí, v objavovaní znalostí naprieč rozličnými oblasťami, či v rámci vyhľadávania a klasifikácie dokumentov alebo ekonomických, medicínskych, hutníckych, elektrotechnických a mnohých ďalších doménach.

Rozhodovací strom predstavuje reprezentáciu rozhodovacej procedúry [27] pre klasifikáciu príkladov do príslušných tried. Táto reprezentácia je acyklickou grafovou štruktúrou – stromom, ktorý obsahuje koreňové, medziľahlé a listové uzly. Uzly reprezentujú triedu alebo testovací atribút. Hrany rozhodovacieho stromu reprezentujú hodnoty testovacieho atribútu. Z každého uzla vychádza toľko hrán, koľko hodnôt má testovací atribút v danom uzle.

Na začiatku generovania rozhodovacieho stromu sa nachádzajú všetky trénovacie príklady, (celá trénovacia množina) v koreňovom uzle. Príklad trénovacej množiny z medicínskej oblasti je uvedený na Obr.24.

Pre každý nelistový uzol (koreňový a medziľahlé uzly) sa vyberie najvhodnejší atribút. Hovoríme, že každý nelistový uzol reprezentuje špecifický test. V rámci medziľahlých uzlov sa trénovacia množina člení na podmnožiny resp. strom sa vetví na podstromy pre každý výsledok špecifického testu, t.j. pre každú hodnotu zvoleného atribútu. Vo všeobecnosti test môže byť aj niečo zložitejšie (napríklad priamka pre lineárnu regresiu alebo krivka pre nelineárnu regresiu). Inak povedané každý nelistový uzol obsahuje testovaciu podmienku, ktorá rozdeľuje priestor príkladov na podpriestory podľa výsledku testu, teda podľa hodnôt zvoleného atribútu. Každý listový uzol rozhodovacieho stromu reprezentuje jednu klasifikačnú triedu.

Por.č.	A1	A2	A3	T
1.	v norme	áno	vyzretý	+
2.	malý	áno	vyzretý	+
3.	v norme	nie	vyzretý	-
4.	v norme	áno	nezrelý	-
5.	veľký	nie	vyzretý	-
6.	v norme	áno	čiast. vyzretý	+
7.	malý	áno	čiast. vyzretý	+
8.	veľký	nie	čiast. vyzretý	-

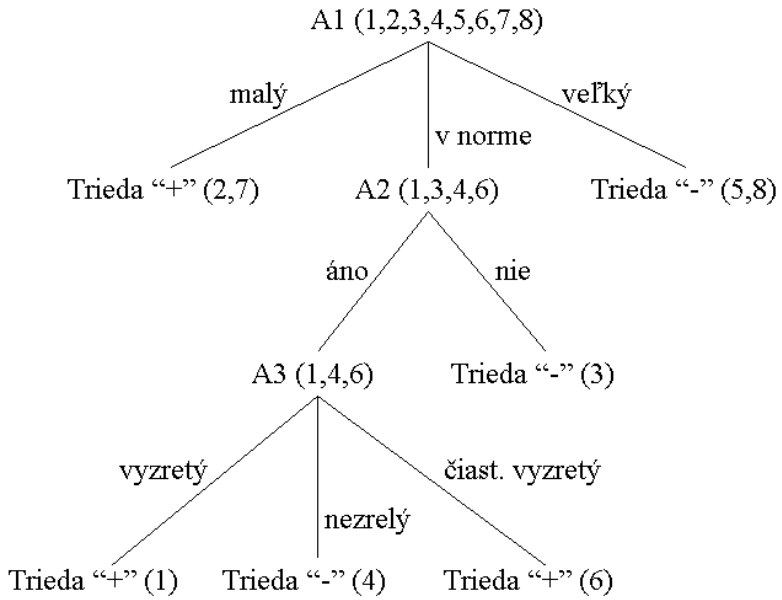
Obr.24 Množina trénovacích príkladov z oblasti prenatálnej medicíny s tromi atribútmi (A1 **– prírastok hmotnosti ženy,** A2 **– zistený patologický prietok u dieťaťa,** A3 **– typ placenty)**

Z vyššie uvedeného vyplýva, že pre definíciu rozhodovacieho stromu je potrebné určiť nasledovné:

- pre listový uzol názov triedy, do ktorej budú klasifikované všetky trénovacie príklady, asociované s daným listovým uzlom

- pre nelistový uzol testovací atribút, podľa ktorého sa rozhodovanie vetví na jednoduchšie podstromy pre každú hodnotu zvoleného testovacieho atribútu.

Majme príklad medicínskej bázy znalostí zameranej na určovanie patologického plodu. Trénovacia množina pozostávajúca z ôsmych príkladov je znázornená na Obr.24. Z tejto trénovacej množiny je možné vygenerovať rozhodovací strom, ktorý je zobrazený na Obr.25.



Obr.25 Rozhodovací strom vygenerovaný algoritmom ID3 (čísla v uzloch sú čísla trénovacích príkladov)

Každý trénovací príklad je asociovaný s jedným listovým uzlom. Preto je možné rozhodovací strom prepísať do sady produkčných pravidiel. Generovaním produkčných pravidiel z rozhodovacieho stromu sa zaoberá aj [28]. Klasifikačné, resp. produkčné pravidlá je možné získať nasledovne:

- Každá cesta v rozhodovacom strome od jeho koreňa po niektorý listový uzol predstavuje jedno produkčné pravidlo

- Ľavá strana, t.j. podmienková časť pravidla bude obsahovať všetky testovacie podmienky v nelistových uzloch na danej ceste v tvare: $A_i=a_i$, pričom každý testovací atribút A_i bude nadobúdať hodnotu $\boldsymbol{a_i}$ prislúchajúcu zvolenej vetve.
- Pravá strana pravidla bude obsahovať názov triedy zodpovedajúcej listovému uzlu, v ktorom cesta končí. Do tejto triedy bude zaradený každý trénovací príklad, spĺňajúci podmienky ľavej strany pravidla.

Napríklad rozhodovací strom na Obr24 je možné prepísať do nasledujúcej množiny pravidiel pre triedu (+), teda pre patologický plod:

1. *AK A1=malý*

   POTOM trieda (+)

2. *AK A1=v norme & A2=áno & A3=vyzretý*

   POTOM trieda (+)

3. *AK A1=v norme & A2=áno & A3=čiast vyzretý*

   POTOM trieda (+)

Podobne je možné generovať z daného rozhodovacieho stromu aj pravidlá pre triedu (-), keď nejde o patologický plod.

Klasifikácia nového príkladu sa potom uskutočňuje nasledovne: *Ak nový príklad spĺňa všetky testy na ceste v strome od koreňa po listový uzol (podmienky podmienkovej časti pravidla), potom je daný príklad zaradený do triedy v listovom uzle (v záverovej časti pravidla).*

6.2.2 Generovanie rozhodovacích stromov

Algoritmy tejto skupiny sa zvyknú označovať aj ako algoritmy induktívneho učenia [29], [30] a [31]. Zameriavajú sa na generovanie reprezentácie naučenej znalosti v tvare rozhodovacieho stromu. Indukovanie rozhodovacieho stromu je ukážkou prístupu „rozdeľuj a panuj“, pri ktorom sa úlohy delia na podúlohy, teda trénovacia množina sa

delí na podmnožiny, alebo priestor príkladov sa delí na podpriestory, lebo sa rekurzívne generujú - indukujú podstromy.

Algoritmy pre generovanie rozhodovacích stromov sú typicky založené na princípe budovania stromov zhora nadol. Pri tomto postupe je pôvodný priestor príkladov (trénovacia množina) delený na jednotlivé podpriestory, charakterizované hodnotami testovacích atribútov. Toto delenie sa uskutočňuje rekurzívne, až kým nie je splnená ukončovacia podmienka. Väčšina algoritmov na generovanie rozhodovacieho stromu končí indukciu, keď sa v každom listovom uzle nachádzajú iba príklady jednej triedy. To sú algoritmy uskutočňujúce perfektnú klasifikáciu. Môže byť definované aj iné ukončovacie kritérium, napríklad: *keď podpriestor obsahuje aspoň. 90% príkladov tej istej triedy, stáva sa listovým uzlom a daný uzol sa ďalej nevetví.* V prípade, že už boli vyčerpané všetky atribúty a v niektorom podpriestore sa stále nepodarilo splniť ukončovacie kritérium, je množina atribútov nedostatočná, alebo pôvodná množina príkladov protirečivá.

Všeobecný postup generovania rozhodovacieho stromu zhora nadol môže byť nasledovný.

Začiatok algoritmu:

1) Ak je pre každý podpriestor splnené ukončovacie kritérium, generovanie sa ukončí.
2) Inak:
    a. Zvolí sa podpriestor obsahujúci príklady klasifikované do viacerých tried.
    b. Pre zvolený podpriestor sa vyberie jeden testovací atribút, ešte nepoužitý pre daný podpriestor príkladov.
    c. Zvolený podpriestor príkladov sa rozdelí na ďalšie podpriestory podľa hodnôt zvoleného testovacieho atribútu.

Koniec algoritmu.

Bod 1) zodpovedá prípadu, keď generovanie rozhodovacieho stromu je už ukončené. Bod 2) pokračuje v generovaní, keď expanduje medziľahlý uzol o ďalšiu úroveň.

Ešte je potrebné zodpovedať otázku, ako voliť testovacie atribúty v kroku 2b). Pri vhodnej voľbe bude rozhodovací strom minimálny. Na druhej strane, nevhodná voľba testovacích atribútov môže značne zväčšiť rozmery generovaného stromu. Konkrétne algoritmy môžu používať rôzne spôsoby výberu testovacieho atribútu, väčšinou informačné miery. Odpoveď na nastolenú otázku sa nachádza v popisoch konkrétnych algoritmov.

Rozhodovacie stromy sa dnes používajú v mnohých systémoch. Jedným z prvých bol systém EPAM (Elementary Perceiver and Memorizer) publikovaný v [32]. Tento systém bol považovaný za kognitívno-simulačný model učenia pojmov človekom. Ďalší model CLS [33] používa pri generovaní rozhodovacieho stromu heuristický prístup (Lookahead Method). V Huntovom učení pojmov (Hunt's Concept Learning) má pôvod aj známy algoritmus C4.5, o ktorom pojednáva jedna z nasledujúcich podkapitol.

Algoritmus ID3. Základným algoritmom generujúcim rozhodovací strom metódou zhora nadol je algoritmus ID3 (Iterative Dichotomizer 3) [31]. Voľne by sa dal názov algoritmu preložiť ako Iteratívny dvojtriedny klasifikátor. Bol vyvinutý Rossom Quinlainom v doméne znalostí šachistu o hraní šachových koncoviek.

Ukončovacie kritérium tohto algoritmu je, že každý podpriestor obsahuje iba príklady jednej triedy. Ak je množina atribútov dostatočná, možno uvedeným postupom zostrojiť rozhodovacie stromy korektne klasifikujúce všetky trénovacie príklady. Hovoríme, že rozhodovací strom generovaný algoritmom je:

- perfektný keď správne klasifikuje všetky trénovacie príklady

- rozmerovo optimálny resp. minimálny keď je rozhodovacia procedúra čo najjednoduchšia, s čo najmenším počtom testov hodnôt atribútov.

Algoritmus ID3 pre výber testovacieho atribútu využíva Shannonovu teóriu informácie [34], ktorá na meranie množstva informácie používa entrópiu. Claude Shannon objavil informačnú teóriu, ktorá spôsobila revolúciu v oblasti komunikácie a taktiež prispela k úspechu učenia pomocou rozhodovacích stromov. Základom tejto teórie je nasledovné:

Ak správy $x_1, x_2, \ldots, x_n$ sú možné s pravdepodobnosťami $p(x_1), p(x_2), \ldots, p(x_N)$ a tieto vytvárajú úplný súbor pravdepodobností, teda platí:

$$\sum_{j=1}^{N} p(x_j) = 1 \tag{58}$$

potom entrópiu, resp. neurčitosť daného súboru správ možno vyjadriť formulou (59):

$$H = -\sum_{j=1}^{N} p(x_j) \log_2 p(x_j) \tag{59}$$

Tento vzťah je známy ako Shannonova entrópia. Čím je entrópia súboru správ vyššia, tým menej určitý je obsah budúcej správy a tým menšie množstvo informácie získame, keď správu prijmeme. Naopak, čím je entrópia súboru správ nižšia, tým väčšie množstvo informácie získame.

Rozhodovací strom môže byť považovaný za zdroj informácie, ktorá pre nejaký konkrétny príklad formuje správu o klasifikácii daného príkladu do nejakej triedy. Keď niektorý uzol stromu obsahuje iba príklady tej istej triedy, entrópia v tomto uzle je nulová, pretože klasifikačné rozhodnutie pre príklady prislúchajúce tomuto uzlu je pevne dané, keďže pravdepodobnosť jednej triedy je 1 a pravdepodobnosť ostatných tried je nulová.

Z toho vyplýva, že entrópia v listových uzloch je nulová,

zatiaľ čo v koreňovom uzle a medziľahlých uzloch má nenulovú hodnotu. Pre vygenerovanie minimálneho rozhodovacieho stromu by entrópia mala čo najrýchlejšie klesnúť na nulovú hodnotu. Algoritmus ID3 používa heuristiku, ktorá usiluje o čo najvyšší pokles entrópie lokálne v každom kroku.

Nech uzol *S* obsahuje n_1 príkladov klasifikovaných do triedy *T1* a n_2 príkladov zaradených do triedy *T2.* Potom pravdepodobnosť, že nejaký príklad prislúchajúci uzlu *S* rozhodovacieho stromu bude klasifikovaný do triedy *T1* resp. *T2* je

$$\frac{n_1}{n_1+n_2} \quad resp. \quad \frac{n_2}{n_1+n_2} \tag{60}$$

Potom entrópia v danom uzle *S,* presnejšie povedané entrópia súboru trénovacích príkladov prislúchajúcich uzlu *S,* bude určená nasledovne pomocou vzťahu (61):

$$H(S) = -\sum_{j=1}^{2} \frac{n_j}{n_1+n_2} \log_2 \frac{n_j}{n_1+n_2} \tag{61}$$

Predpokladajme, že je možné v uzle *S* použiť atribút *A* s hodnotami a_1 a a_2 pre rozdelenie príkladov z uzla *S* do dvoch disjunktných podmnožín *S1* a S2. Ak v uzle *S* má m_1 príkladov hodnotu atribútu *A* rovnú a_1 a m_2 príkladov má hodnotu a_2, potom celková entrópia v uzle *S* s použitím atribútu *A* sa určí ako:

$$H(S,A) = \sum_{j=1}^{2} \frac{m_j}{m_1+m_2} H(S_j) \tag{62}$$

Informácia získaná použitím atribútu *A* v uzle *S* je označovaná ako informačný zisk a počíta sa podľa vzťahu (63):

$$I(A) = H(S) - H(S, A) \tag{63}$$

V každej iterácii sa v kroku 2b) algoritmu pre generovanie rozhodovacieho stromu metódou zhora nadol vyšetrujú všetky ešte nepoužité atribúty a ako testovací atribút sa vyberá ten, ktorý maximalizuje informačný zisk resp. minimalizuje entrópiu t.j. neurčitosť. Stratégiou algoritmu ID3 je čo najrýchlejší pokles entrópie na nulu a tak generovanie minimálneho rozhodovacieho stromu.

Algoritmus ID3 je možné jednoducho rozšíriť aj na prípad, keď príklady budú klasifikované do viacerých tried a keď atribúty môžu nadobúdať viac ako dve hodnoty. Ak jednotlivé triedy budú *T1, T2, ..., TN*, potom vzťah pre výpočet entrópie uzla *S* bude mať nasledovný tvar:

$$H(S) = -\sum_{j=1}^{N} p(T_j) \log_2 p(T_j) \tag{64}$$

kde $p(T_j)$ je pravdepodobnosť toho, že nejaký príklad, patriaci uzlu *S*, bude klasifikovaný do triedy T_j. Ak atribút *A* môže nadobúdať hodnoty $a_1, a_2, ..., a_m$, potom vzťah pre výpočet neurčitosti uzla *S* s testovacím atribútom *A* bude mať tvar:

$$H(S, A) = \sum_{j=1}^{m} p(a_j) H(S_j) \tag{65}$$

kde $p(a_j)$ je pravdepodobnosť toho, že nejaký príklad, patriaci uzlu *S* má hodnotu atribútu $A=a_j$.

Pre použitie algoritmu musí platiť podmienka neprotirečivosti (nekontradikčnosti) trénovacích príkladov. Nutnosť platnosti tejto podmienky vyplýva z toho, že ID3 nie je odolný voči zašumeným údajom na vstupe. Zo spôsobu určovania pravdepodobností vyplýva, že iba dodržanie dvoch dodatočných podmienok, a to podmienky neredundantnosti príkladov (nepovolenie viacnásobného výskytu trénovacieho príkladu v trénovacej množine) a podmienky vzájomnej nezávislosti atribútov, garantuje

nájdenie minimálneho rozhodovacieho stromu.

Rozhodovacie stromy sa spočiatku aplikovali hlavne na problémy diagnostiky chorôb. Jednou z týchto aplikácií algoritmu ID3 je aj program uvedený v [35], ktorý je využívaný v lekárstve na spracovanie a vyhodnotenie EEG, konkrétnejšie kmeňových sluchových evokovaných potenciálov. V tomto prípade modifikácia spočíva v zavedení diskretizácie hodnôt atribútov, optimálnej hĺbky stromu a v spolupráci s jednotkou zhlukovania.

Algoritmus ID5R. Pomerne známy je aj algoritmus ID5R (Inductive Dichotomizer 5 Recursive), ktorý predstavuje inkrementálnu modifikáciu algoritmu ID3. Využíva sa v prípadoch, keď nie všetky trénovacie príklady sú k dispozícii naraz, ale stávajú sa známymi postupne (odoberanie z reálneho procesu v určitých časových okamihoch). Ide o riešenie učiacej úlohy typu on-line. Niekedy nie je možné čakať, až budú známe všetky trénovacie príklady na začiatku generovania rozhodovacieho stromu a teda je potrebné generovať rozhodovací strom na základe tých príkladov, ktoré už sú známe a byť schopný po príchode každého ďalšieho trénovacieho príkladu aktualizovať už vygenerovaný rozhodovací strom.

Pre túto úlohu je samozrejme možné použiť aj niektorý neinkrementálny algoritmus, napr. ID3. Toto riešenie má však niekoľko nevýhod:

- pri príchode každého nového príkladu je potrebné realizovať indukciu opäť od začiatku
- pri novej indukcii sa nevyužívajú informácie získané v predchádzajúcich indukciách
- počet príkladov neustále narastá, čím sa predlžuje doba potrebná pre opätovné vygenerovanie rozhodovacieho stromu.

Preto je vhodné využívať inkrementálnu indukciu, ktorá po príchode nového príkladu inicializuje iba modifikáciu už existujúceho stromu. Takúto inkrementálnu indukciu vykonáva aj algoritmus ID5R. Aj tento algoritmus, podobne ako ID3, využíva maximalizáciu informačného zisku pre

výber testovacieho atribútu. Na rozdiel od ID3 si uchováva v každom uzle rozhodovacieho stromu dostatočnú informáciu na to, aby mohol zistiť, či aj po príchode nového príkladu bude predtým zvolený testovací atribút ešte ten najvhodnejší, teda bude stále maximalizovať informačný zisk, alebo či je potrebné ho nahradiť iným atribútom. Pri nutnosti zmeny testovacieho atribútu mu uchovávaná informácia umožňuje nájsť nový atribút a všetky podstromy reštrukturalizovať podľa nového stavu, zmeneného príchodom nového trénovacieho príkladu. Podrobnejšie o inkrementálnej indukcii rozhodovacích stromov pojednáva [36].

6.2.3 Algoritmus C4.5

Ďalším zaujímavým algoritmom tejto skupiny je algoritmus C4.5 [37], ktorý navrhol Ross Quinlan. Ide vlastne o vylepšený a doplnený algoritmus ID3. Algoritmus C4.5 rozlišuje dva typy atribútov: nominálne a reálne. Používa pomenovanie diskrétne (nominálne) a spojité (reálne) atribúty. Ak je potrebné používať iné typy atribútov, je potrebné ich vhodne pretransformovať na jeden z týchto atribútov. Je to neinkrementálny algoritmus, budujúci strom zhora nadol. Tento algoritmus je založený na princípe "Rozdeľuj a panuj" nad množinou *M* trénovacích príkladov. Predpokladajme, že trénovacie príklady majú byť klasifikované do tried $T_1, T_2, ...T_K$. Potom môžu nastať tri prípady:

- Množina *M* obsahuje jeden alebo viac príkladov patriacich do tej istej triedy T_i. Rozhodovací strom pre množinu *M* je listový uzol s triedou T_i.
- Množina *M* neobsahuje žiadne príklady. Rozhodovací strom je opäť listový uzol, ale triedu nie je možné určiť z množiny *M*. Môže byť určená napr. ako majoritná trieda zo všetkých trénovacích príkladov, alebo ako majoritná trieda v rodičovskom uzle.
- Množina *M* obsahuje príklady patriace do viacerých tried. Vyberie sa test vedúci k rozdeleniu množiny *M* do podmnožín, obsahujúcich príklady s rovnakou hodnotou testovacieho atribútu. Testovacia

podmienka predstavuje atribút, podľa hodnôt $\{a_1, ...a_n\}$ ktorého sa M rozdelí do podmnožín $M_1, M_2, ...M_n$, kde M_i obsahuje všetky príklady z M, vyhovujúce podmienke s hodnotou atribútu a_i. Rozhodovací strom sa v tomto prípade skladá z rozhodovacieho uzla s testovacím atribútom a jednou podmienkou pre každú hodnotu atribútu. Tento postup je opakovaný rekurzívne pre každú vetvu rozhodovacieho uzla.

Algoritmus C4.5 sa od ID3 líši okrem iného aj druhým prípadom, keď množina M neobsahuje žiadne príklady. Tento algoritmus vytvára perfektný rozhodovací strom za predpokladu neprotirečivosti trénovacích príkladov, podobne ako algoritmus ID3. Existencia protirečivých príkladov môže signalizovať nedostatočné množstvo atribútov, a teda nie dosť precízny popis príkladov v údajovej časti. V prípade protirečivých príkladov je možné ukončovacie kritérium, predpokladajúce klasifikáciu všetkých príkladov v listovom uzle do jednej triedy upraviť tak, aby majoritná trieda v listovom uzle tvorila aspoň napríklad 90%.

Pomerové kritérium zisku. Algoritmus C4.5 používa na výber testovacej podmienky (testovacieho atribútu) kritérium zisku, ktoré je založené na informačnej teórii, podobne ako algoritmus ID3. V takom tvare, ako ho používa ID3, však toto kritérium zisku uprednostňuje výber testovacích atribútov s väčším počtom hodnôt. Tým vytvára určitý druh prehľadávacej preferencie. Uvažujme extrémny prípad, že v trénovacej množine sa nachádza diskrétny atribút, ktorý má pri každom trénovacom príklade inú hodnotu. Teda má toľko hodnôt, koľko je v trénovacej množine príkladov. Paradoxne práve tento atribút by mal najmenšiu entrópiu a teda aj najväčší informačný zisk. V dôsledku toho by mal taktiež najväčšiu šancu pre výber do testovacej podmienky. Lenže to by sa rozhodovacom strome prejavilo ako vetvenie, ktoré by viedlo k podmnožinám a im odpovedajúcim uzlom, obsahujúcim iba jeden príklad. Takéto vetvenie je neefektívne. Uvedený nedostatok je možné odstrániť normalizovaním informačného zisku, t.j. zavedením pomerového kritéria

zisku.

Nech sa uzol *S* použitím atribútu *A* rozvetví na *m* vetiev. Potom pomerová entrópia bude daná vzťahom (66):

$$H_p(S,A) = -\sum_{j=1}^{m} p(a_j)\log_2(p(a_j)) \tag{66}$$

kde $p(a_j)$ je pravdepodobnosť toho, že nejaký príklad patriaci uzlu *S* prešiel vetvou a_j do poduzla S_j, keďže nadobúda pre atribút *A* hodnotu a_i. Pomerová entrópia má tendencie narastať s rastúcim počtom vetiev. Potom pomerovým informačným ziskom, daným vzťahom (67)

$$I_p(S,A) = \frac{I(S,A)}{H_p(S,A)} \tag{67}$$

sa obmedzí výber testovacej podmienky s mnohými výstupmi, čím sa vytvorí nový druh prehľadávacej preferencie. I(S,A) sa určí podľa vzťahu, ktorý na určenie informačného zisku používa algoritmus ID3. Pomerový informačný zisk vedie ku generovaniu rozmerovo menších rozhodovacích stromov.

Spojité atribúty. Testovacia podmienka pre spojité atribúty pozostáva z prahovej hodnoty, ktorá rozdeľuje usporiadanú množinu čísel na dve podmnožiny. Trénovacie príklady sú najprv usporiadané vzostupne podľa hodnôt daného spojitého atribútu. Vznikne usporiadaná množina hodnôt spojitého atribútu $\{v_1, v_2, \ldots, v_m\}$.

Prahová hodnota ležiaca medzi dvoma hodnotami v_i a v_{i+1}, rozdelí množinu príkladov na množiny: $\{v_1, v_2, \ldots, v_i\}$, $\{v_{i+1}, v_{i+2}, \ldots, v_m\}$.Existuje $m-1$ takýchto rozdelení pri celkovom počte trénovacích príkladov *m*. Pre všetky rozdelenia sa vypočíta ohodnocovacia funkcia a vyberie sa rozdelenie s maximálnym ohodnotením. Ohodnocovacou funkciou môže byť informačný zisk alebo pomerový informačný zisk. Prahová hodnota medzi dvoma hodnotami spojitého atribútu v_i a v_{i+1} sa určí ako ich priemerná hodnota.

Neznáme hodnoty atribútov. V reálnych údajoch sa často stáva, že niektoré hodnoty atribútov nie sú známe. Môžu chýbať z rôznych príčin. Z dôvodu zašumenia, nedôležitosti atribútu, nedbanlivosti vkladajúceho, alebo sú skutočne neznáme. Algoritmus generovania rozhodovacieho stromu je potrebné upraviť, aby bol schopný vysporiadať sa s takýmto neúplným vstupom.

Príklady trénovacej množiny s neurčenou triedou sú zbytočné (informačne nezaujímavé), preto sú pri spracovávaní odfiltrované. Odfiltrovať aj príklady s chýbajúcimi hodnotami atribútov by bolo najjednoduchšie, ale nie vždy to je vhodné. Do úvahy pripadá napríklad doplnenie chýbajúcich hodnôt najčastejšie sa vyskytujúcou hodnotou.

Algoritmus C4.5 nevykonáva predspracovanie údajov pred začatím generovania rozhodovacieho stromu podľa [38] je schopný sa s tým vyrovnať počas samotného generovania.

K tomu je potrebné upraviť kritérium pre výber testovacej podmienky, a taktiež spôsob zaradzovania neúplných príkladov do jednotlivých uzlov. Entrópia v danom uzle *H(S)* sa vypočíta tým istým spôsobom ako predtým. K vypočítaniu *H(S,A)* sú uvažované iba príklady so známymi hodnotami uvažovaného atribútu. Modifikovaný informačný zisk sa vypočíta nasledovne pomocou formuly (68):

$$I(S,A) = p(A_k)(H(S) - H(S,A)) - p(A_{unk})$$

(68)

kde: $p(A_k)$ je pravdepodobnosť výskytu známych hodnôt atribútu a $p(A_{unk})$ je pravdepodobnosť výskytu neznámych hodnôt atribútu. Podobne sa upraví pomerová entrópia:

$$H_p(S,A) = -\sum p(a_j)\log_2 p(a_j) - p(a_{unk})\log_2 p(a_{unk})$$

(69)

kde: $p(a_j)$ je pravdepodobnosť toho, že príklad patriaci

uzlu S prešiel vetvou a_j do poduzla S_j. Pomerový informačný zisk ostáva v pôvodnej podobe.

Orezávanie klasifikačných stromov. Keď sa pri generovaní rozhodovacieho stromu kladie dôraz na presnosť klasifikácie, môže sa stať, že rozhodovací strom bude preučený teda „overfitting“. Pri takomto strome sa dá zaručiť presná klasifikácia iba u trénovacích príkladov. Pri klasifikácii testovacích príkladov potom často sú výsledky testov nelichotivé. Rozhodovací strom totiž odráža aj náhodné súvislosti, spôsobené nedokonalým výberom pozorovaní – trénovacích príkladov. Tieto problémy sa prejavia vznikom veľmi tenkých vetiev, ktoré nezriedka obsahujú iba jeden trénovací príklad. Taký strom je vhodné orezať teda skrátiť niektoré tenké vetvy. Je možné uviesť horný odhad dĺžky vetiev stromu, ktorý je zárukou požadovanej presnosti [29]. Vhodná dĺžka vetiev je menšia ako podiel logaritmov rozsahu trénovacej množiny a počtu uvažovaných atribútov. Toto číslo sa nazýva *optimálna hĺbka stromu* [35].

Orezávanie rozhodovacieho stromu je jednou z možností, ako znížiť chybu klasifikácie a zároveň zmenšiť rozhodovací strom. Presnosť klasifikácie na učiacich údajoch sa síce zmenší, ale dá sa predpokladať, že sa zvýši presnosť na neznámych príkladoch.

Pri klasických metódach orezávania sa strom najprv vygeneruje a potom sa postupne zjednodušuje. Existujú metódy, ktoré zjednodušujú rozhodovací strom už počas jeho generovania. Budovanie a následné orezávanie rozhodovacieho stromu je síce pomalšie, ale hodnovernejšie a v neposlednom rade aj jednoduchšie, pretože je založené na náhrade podstromu listovým uzlom.

Techniky orezávania, podrobnejšie popísané aj v [39], sa delia podľa toho, či pracujú aj s testovacou, alebo iba s trénovacou množinou. V prvom prípade sa rozhodovací strom vygeneruje na základe trénovacej množiny a následnému orezaniu poslúži iná, testovacia množina. Za predpokladu dostatočného množstva dôveryhodných údajov vznikne týmto postupom kvalitnejšia rozhodovacia procedúra.

Príkladom takéhoto spôsobu orezávania rozhodovacieho stromu sú nasledovné techniky:

- Orezávanie cena-komplexnosť
- Redukcia chyby

Metóda *redukcie chyby* rozdistribuuje testovacie príklady do listových uzlov originálneho rozhodovacieho stromu. Potom pre každý nelistový uzol *S* stromu *T* sa preskúma ako by sa zmenil počet chybne klasifikovaných príkladov, ak by sa *S* nahradil jeho najlepším listovým uzlom. Ak nový strom klasifikuje s rovnakým, alebo menším počtom chýb a *S* neobsahuje žiadny podstrom s takouto vlastnosťou, potom je *S* nahradený daným listom. Tento proces sa ukončí vtedy, keď by už každá ďalšia náhrada zvýšila počet chybne klasifikovaných príkladov z testovacej množiny.

Reprezentantom druhého typu techník orezávania, ktoré pracujú iba s trénovacou množinou, je *pesimistické orezávanie.* Táto technika odhaduje chybu klasifikácie. Používa pritom heuristickú metódu, ktorá vyžaduje počet všetkých trénovacích príkladov *N* a počet chybne klasifikovaných príkladov v danom uzle *E*. Algoritmus C4.5 rozširuje túto heuristiku o voľbu úrovne pesimistického orezávania CF. V ďalšom sa navrhuje prírastok predikovanej chyby klasifikácie *U(E,N).* Celková predikovaná chyba je daná súčtom prírastku chyby a počtu chybne klasifikovaných príkladov:

$$E' = E + U(E, N) \tag{70}$$

Predikovaná chyba nelistového uzla sa určí súčtom predikovaných chýb jeho poduzlov. Orezávanie postupuje rozhodovacím stromom zdola nahor. Je možné nahradiť uzol:

- listovým uzlom s triedou, ktorá sa najčastejšie vyskytuje v podstrome
- alebo jedným z poduzlov v jeho vetvách.

Pre každú možnosť sa vypočíta predikovaná chyba uzla a porovnáva sa s predikovanou chybou uzla pred orezaním. Aplikuje sa tá možnosť, ktorou sa dosiahne menšia predikovaná chyba. Ak sa nahradí uzol poduzlom, poduzol môže byť taktiež orezaný v ďalšej iterácii. Keďže predikovaná chyba uzla sa môže len znížiť, výsledkom je menšia predikovaná chyba celého stromu. Výhodou tejto metódy je, že jej stačí trénovacia množina. Táto výhoda je veľmi cennou pri veľmi často sa vyskytujúcom nedostatku príkladov.

Technika malého okna. Pri spracovaní veľkých databáz sa môže ukázať operačná pamäť nedostatočnou. Tento problém môže vyriešiť nepriama metóda spracovania údajov, nazývaná technika malého okna (windowing). Táto metóda náhodne vyberie podmnožinu príkladov z množiny trénovacích príkladov, nazývanú „okno“. Z okna je nasledovne vygenerovaný rozhodovací strom. Vygenerovaným stromom sa klasifikujú príklady, ktoré nie sú zahrnuté v okne. Chybne klasifikované príklady sa zaradia do okna, z ktorého sa opäť vygeneruje nový rozhodovací strom. To sa opakuje dovtedy, kým posledný vygenerovaný strom nebude správne klasifikovať všetky príklady mimo okna. Často sa stáva, že konečné okno obsahuje iba malú časť z celkovej množiny trénovacích príkladov.

V algoritme C4.5 sa nevyberajú príklady do okna náhodne, ale preferovaním výberu, ktorý vytvorí distribúciu tried v okne zhodnú s distribúciou tried v trénovacej množine. Algoritmus sa môže zastaviť skôr, než rozhodovací strom správne klasifikuje všetky príklady mimo okna. Napríklad, keď sa už neznižuje celková chyba klasifikácie, alebo ak narastie veľkosť okna nad obmedzujúcu hranicu. Včasné ukončenie môže v prípade zašumených údajov zabrániť neúprosnému rastu okna, ktoré smeruje k pohlteniu všetkých trénovacích príkladov.

Okrem možnosti obísť obmedzenie nedostatku pamäti, je použitie techniky malého okna výhodné pri redundancii príkladov. Na druhej strane, generovaním rozhodovacieho stromu v každom cykle sa predĺži celkový čas spracovania. Neočakávaným prínosom je vyššia presnosť ním

vytvorených stromov. Je možné vygenerovať viacero rozličných stromov a následne vybrať najlepší vzhľadom na predikovanú chybu. Takto vygenerovaný strom je často presnejší, ako strom generovaný z celej trénovacej množiny. Ide to však na úkor času, ktorý s počtom generovaných stromov narastá.

Zoskupovanie hodnôt diskrétnych atribútov. Pri generovaní stromu klasickým spôsobom sa rekurzívne generujú podstromy pre každú hodnotu atribútu. Ak má niektorý atribút veľmi veľa hodnôt (rádovo desiatky) výsledný rozhodovací strom sa stane neprehľadným. Preto je potrebné atribúty s veľkým počtom hodnôt diskretizovať, teda nahradiť pôvodnú rozsiahlu množinu atribútov novou menšou množinou diskrétnych hodnôt atribútov. Počet hodnôt môžu ovplyvniť aj požiadavky na tvar výsledného rozhodovacieho stromu. Môže byť požadovaná rôzna hĺbka a šírka stromu. Aby bol požadovaný strom zrozumiteľný pre používateľa, potom je vhodné zaviesť počet maximálne sedem rôznych diskrétnych hodnôt pre jeden atribút. Psychologické testy často definujú tento počet ako maximálny počet alternatív, ktoré je človek schopný alebo ochotný uvažovať.

Diskrétne hodnoty atribútov je možné zoskupovať viacerými spôsobmi [40]. Jednou z metód je binárne rozdelenie s dvoma výslednými podmnožinami. Ďalším spôsobom je určenie všetkých rozdelení množiny hodnôt daného atribútu a výber najvhodnejšieho, podľa nejakého ohodnocovacieho kritéria. Tento prístup však vedie ku kombinatorickej explózii výpočtového procesu.

Ďalšou možnosťou je využitie heuristiky (ako to robí algoritmus C4.5), ktorá je menej výpočtovo náročná, a ktorá pracuje nasledovným spôsobom:

- Najprv sa v rámci inicializácie vytvorí inicializačné rozdelenie, pozostávajúce z *N* jednoprvkových množín (práve toľko, koľko je hodnôt atribútu A). Pre toto rozdelenie sa vypočíta informačný zisk alebo pomerový informačný zisk.
- Vytvoria sa všetky možné rozdelenia zlúčením dvoch množín (všetky možné dvojice). Rozdelenia

sa opäť ohodnotia skórovacou funkciou a vyberie sa najlepšie rozdelenie.

- Zlučovanie množín rekurzívne postupuje dovtedy, kým nie je dosiahnuté hraničné rozdelenie s dvoma množinami.
- Zo všetkých rozdelení sa vyberie výsledné rozdelenie s maximálnym ohodnotením.

Pri výbere testovacieho atribútu v konkrétnom uzle sa nielen vyberá ešte nepoužitý atribút, ale sa aj spúšťa vyššie uvedený algoritmus, ktorý určí najvhodnejšie zoskupenie hodnôt daného atribútu.

Zoskupovanie hodnôt atribútov výrazne zvýši celkový čas generovania rozhodovacieho stromu. Z hľadiska dĺžky výpočtu by bolo výhodnejšie vykonať zoskupovanie hodnôt atribútov už pred začiatkom generovania rozhodovacieho stromu. Tento prístup má však tiež svoju nevýhodu. Nedokáže podchytiť rozličné znalosti v rôznych častiach rozhodovacieho stromu.

6.3 Strom verzus lineárny model

Regresný a klasifikačný strom predstavujú dosť odlišné prístupy. Taktiež lineárna regresia a regresný strom nie je to isté. Napríklad lineárna regresia predpokladá model vo forme (71):

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j, \tag{71}$$

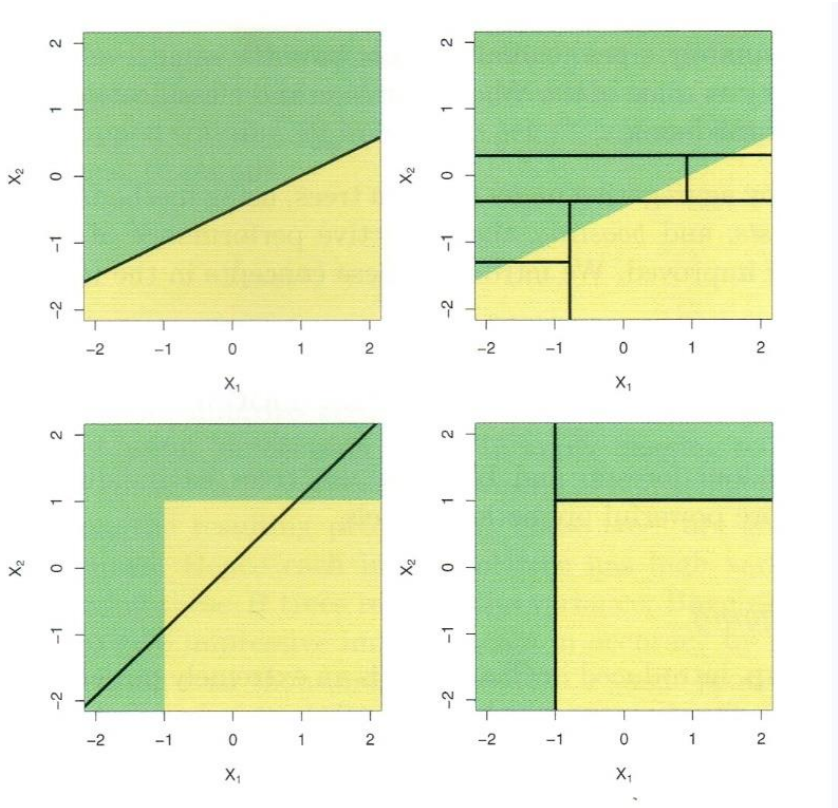
zatiaľ čo regresný strom je možné reprezentovať nasledovným modelom:

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1(x \in R_m) \tag{72}$$

kde $R_1, \ldots, R_m$ predstavujú partície priestoru príznakov (atribútov) z Obr.23. Ťažko povedať, ktorý z týchto modelov je lepší. Ak je priestor lineárne separovateľný je vhodné použiť lineárnu regresiu, ktorá je jednoduchá a v tomto prípade bude dávať dobré výsledky asi lepšie ako regresný

strom, čo vyplýva aj z Obr.26 z porovnania dvoch horných podobrázkov. Avšak, ak je vzťah medzi atribútmi (prediktormi) a odpoveďou (predikciou), teda vzťah medzi vstupom a výstupom, nelineárny a komplexný, potom je vhodnejší model regresného stromu. V tomto prípade regresný strom prekoná klasické prístupy, čo najlepšie ilustruje Pravá Horná časť Obr.26. Prekoná znamená, že po odhade chyby v rámci krížovej validácie, bude vykazovať menšiu chybovosť. Avšak niekedy zvažujeme nielen presnosť prístupu ale aj interpretovateľnosť prístupu a tak si môžeme zvoliť model regresného stromu aj v prípade, že by nebol presnejší.

V horných podobrázkoch na Obr.26 máme dáta, ktoré sú lineárne separabilné. Preto lineárna regresia na týchto dátach vykazuje takmer nulovú chybu (Ľavý Horný). To sa nedá povedať o regresnom strome (Pravý Horný). Pri rozdelení na podoblasti prístupom vlastným budovaniu regresného stromu, nie je možné presne oddeliť tieto dáta – iba približne. Teda je tam nenulová chyba. V dolný podobrázkoch máme dáta iného charakteru. Sú separabilné ale nie lineárne. Preto lineárna regresia vykazuje dosť veľkú chybu (Ľavý Dolný), zatiaľ čo generovanie regresného stromu dáva takmer bezchybný výstup (Pravý Dolný).



Obr.26 Ilustrácia vzťahu medzi rozhodovacím stromom a lineárnym modelom [1].

6.4 Výhody a nevýhody stromových modelov

Tak regresné ako aj klasifikačné stromy majú radu výhod aj nevýhod:

- Naučená znalosť v tvare stromu sa dá ľahko vysvetliť ľuďom a to aj laikom v oblasti informatiky. Niekedy sú stromy ľahšie vysvetliteľné ako lineárna regresia.
- Existuje predpoklad, že rozhodovacie stromy najlepšie zrkadlia spôsob, ako sa rozhoduje človek.

- Stromy ako grafy môžu byť vizualizované a teda ľahko interpretovateľné dokonca aj nie expertom, ak sú malé.
- Stromy dokážu ľahko zvládnuť aj kvalitatívne prediktory (atribúty) bez potreby vytvárať fiktívne premenné.
- Na druhej strane stromy často nemajú rovnakú úroveň správnosti predikcie ako iné regresné alebo klasifikačné prístupy.

ZÁVER

Predložená vysokoškolská učebnica je zameraná na oblasť metód strojového učenia, ktoré majú štatistickú podstatu. Zvyknú sa označovať ako metódy štatistického učenia. Učebnica prináša popis najúspešnejších a v súčasnosti najčastejšie používaných metód štatistického učenia. Táto vysokoškolská učebnica je určená pre predmet – kurz s názvom Strojové učenie, ktorý sa ponúka v rámci dvoch študijných programov na Katedre kybernetiky a umelej inteligencie Fakulty elektrotechniky a informatiky na Technickej univerzite v Košiciach. Tento učebný text dopĺňa starší učebný text autora tejto publikácie s názvom „Strojové učenie – princípy a algoritmy“ a taktiež monografiu „Strojové učenie v systémoch spracovanie informácií“.

Väčšina metód uvedenej vysokoškolskej učebnice sa používa aj dnes na riešenie aktuálnych problémov. Autorka publikácie má skúsenosti s výskumom, kde sa tieto metódy aplikujú na spracovanie textov, napríklad krátkych textov vznikajúcich v rámci konverzačného obsahu hromadeného na sociálnych sieťach. Preto sa často ako príklady použitia týchto metód používajú príklady klasifikácie dokumentov, dolovanie konverzačného obsahu a štruktúry konverzácie za účelom analýzy sentimentu a odhadu autoritatívnosti prispievateľov do webových diskusií.

POUŽITÁ LITERATÚRA

[1] James, G., Witten, D., Hastie, T., Tibshirani, R.: *An Introduction to Statistical Learning – with Applications in R.* Springer Texts in Statistics, DOI: 10.1007/978-1-4614-7138-7_2, Springer Science+Business Media New York, 2013, 426 ps

[2] Ostertágová, E.: *Aplikovaná Štatistika*. 2. dopln. vyd. Košice : EQUILIBRIA, 2013. s. 123-127. ISBN 978-80-8143-067-1.

[3] Buša, J., Pirč, V., Schrotter, Š.: *Numerické metódy, pravdepodobnosť a matematická štatistika*. 1. vyd. Košice 2006. s.137. ISBN 80-8073-632-4

[4] Analýza závislej intervalovej premennej Y na nezávislých intervalových premenných X: Viacnásobná regresia [online] . [cit. 2015-02-18]. Dostupné z: http://rimarcik.com/navigator/mr.html

[5] Rektorys, Karel. *Přehled užité matematiky*. 5., nezměn. vyd. Praha: Státní nakladatelství technické literatury, 1988, 2 sv. Česká matice technická (SNTL)

[6] Analýza závislosti dvoch veličín [online]. [cit. 2016-08-26]. Dostupné z: http://www.fberg.tuke.sk/upam/SSD4%20novy.pdf

[7] Lineárny regresný model [online]. [cit. 2016-08-26]. Dostupné z: http://fsi.uniza.sk/kkm/old/publikacie/ek/ek_kap_3.pdf

[8] Machová, K., Štefaník, J.: *Regression Methods in the Authority Identification within Web Discussions*. In: Computational Collective Intelligence, Lecture Notes in Artificial Intelligence, Vol. LNAI 9329, no. 1 (2015), © Springer-Verlag Berlin Heidelberg, 2015, 203-212, ISSN 0302-9743

[9] Kováč, J.: Strojové učenie [online]. [cit. 2016-09-07]. Dostupné z: http://people.tuke.sk/kristina.machova/prezentacieSU/10strojove%20ucenie%202012%20JozoKovac.ppt

[10] Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Second Edition.* DOI:10.1007/b94608_1, Springer Science+Business Media New York, LLC 2009, 2016, 745 ps, ISBN 978-0-387-84857-0

[11] Machová, K.: *Machine Learning. Principles and Algorithm*. Faculty of Electrical Engineering and Informatics, Technical University of Košice, Elfa s.r.o, 2002, Košice, 117s., ISBN 80 89066 51 8

[12] Jackson, R., Weisstein, E.W.: *Kernel - MathWorld--A Wolfram* [online]. [cit. 2016-09-07]. Dostupné z: http://mathworld.wolfram.com/Kernel.html

[13] Machová, K.: *Strojové učenie v systémoch spracovania informácií*. Faculty of Electrical Engineering and Informatics, Technical University of Košice, Elfa s.r.o, 2009, Košice, 85 s., ISBN 978-80-8086-130-8

[14] Krammer, P.: *Dolovanie v údajoch so zameraním na interpretovateľnosť modelov*. Dizertačná práca, FIIT STU, Bratislava, 2015, 105 s.

[15] Witten, I.H., Frank, E., Hall, M.A.: *Data Mining – Practical Machine Learning Tools and Techniques*. Third Edition, Elsiever, 2011

[16] Quinlan, J.R.: *Bagging, boosting and C4.5*. In Proc. of the Fourteenth National Conference on Artificial Intelligence, 1996

[17] Schapire, R.E., Singer, Y.: *BoostTexter: A Boosting – based System for Text Categorization*. Machine Learning, 39(2/3), 2000, 135-168

[18] Breiman, L.: *Bagging predictors*. Technical Report 421, Department of Statistics, University of California at Berkeley, 1994

[19] Machová, K., Barčák, F., Bednár, P.: *A Bagging Method Using Decision Trees in the Role of Base Classifiers*. Acta Polytechnica Hungarica, Vol.3, No.2, 2006, Budapest Tech., Hungary, 121-132, ISSN 1785-8860

[20] Breiman, L.: *Random Forests*. Machine Learning 45 (2001), 5-32

[21] Schapire, R., Freund, Y.: Boosting the margin: A new explanation for the efectiveness of voting methods. The annals of statistics, 26(5):1651-1686, 1998.

[22] Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning, 37(3), 1999, 297-336

[23] Janecek, A., Gansterer, W.: *Efficient Feature Reduction and Classification Methods*: Application in Drug Discovery and Email Categorizatio, 2009

[24] Machová, K., Puszta, M., Bednár, P.: An Improvement of the Classification Algorithm Results. Teaching Mathematics and Computer Science, Debrecen, Hungary, Vol.4, No.6, 2006, 131-142, ISSN 1589-7389

[25] Machová, K., Puszta, M., Barčák, F., Bednár, P.: A Comparison of the Bagging and the Boosting Method Using the Decision Trees Classifiers. ComSIS - Computer Science and Information Systems, Vol. 3, Number 2, 2006, Novi Sad, Serbia and Montenegro, 57-72, ISSN 1820-0214

[26] Schonlau, M.: Boosted Regression (Boosting): An introductory tutorial and a Stata plugin. The Stata Journal 2005, Vol.5, No.3, pp. 330-354

[27] Quinlan, J.R.: Decision trees and decision making. IEEE Transactions Systems, Man and Cybernetics 20, 2, 1990, pp. 339-346

[28] Quinlan, J.R.: Generating production rules from decision trees. Proc. of the Tenth International Joint Conference on Artificial Intelligence, San Mateo, CA: Morgan Kaufmann, 1987, pp. 304-307

[29] Russell S.J., Norvig P.: Artificial Intelligence. A Modern Approach. Prentice-Hall International, Inc., USA, 1995, ISBN 0-13-360124-2

[30] Michalski, R.S., Stepp, R.E.: A Theory and Methodology of Inductive Learning. In: Michalski R.S., Carbonnell J.G., Mitchell T.M. (eds.): Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann, 1983

[31] Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 1, 1986, pp. 81-106

[32] Feigenbaum, E.A.: The simulation of verbal learning behavior. Proc. of the Western Joint Computers Conference, 19, 1961, pp. 121-131

[33] Hunt, E.B., Marin, J., Stone, P.T.: Experiments in Induction. Academic Press, New York, 1966.

[34] Shannon, C.E., Weaver, W.: The Mathematical Theory of the Communication. University of Illinois Press, Urbana, 1949

[35] Štepánková, O., Hetmerová, A., Kraus, J.: Některé možnosti použití strojového učení v lékařství. Lékař a Technika, 29, 1998, pp. 56-62

[36] Utgoff, P.E., Bradly, C.F.: Incremenral induction of decision trees. Machine Learning 4, 2, 1991, pp. 161-186

[37] Quinlan, J.R.: C4.5 Programes for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California, 1993, ISBN 1-55860-238-0

[38] Quinlan, J.R.: Unknown attribute values in induction. Proc. of the Sixth International Machine Learning Workshop, San Mateo, CA: Morgan Kaufmann, 1989, pp. 164-168

[39] Mach, M.: Získavanie znalostí pre znalostné systémy. Vydavateľstvo ELFA s.r.o., Košice, 1997, 104 ps

[40] Quinlan, J.R.: Decision trees and multi valued attributes. In J.E. Hayes, D. Michie, and J. Richards (eds.), Machine Intelligence 11, Oxford, UK: Oxford University Press, 1988, pp. 305-318

Táto publikácia bola vytvorená za podpory projektu KEGA č. 034TUKE-4/2014 “Integrácia študijných programov v odboroch Kybernetika a Umelá inteligencia”.

NÁZOV: **Nové trendy v strojovom učení – štatistický prístup**
AUTOR: Machová Kristína
RECENZENTI: doc. Ing. Marián Mach, PhD., Ing. Peter Bednár, PhD.
VYDAVATEĽ: Technická univerzita v Košiciach
ROK: 2016
ROZSAH: 96 strán
NÁKLAD: 50 ks
VYDANIE: prvé
FORMÁT: b5
TLAČ: Fakulta elektrotechniky a informatiky, TU
VYTLAČENÉ: september 2016

ISBN: **978-80-553-2602-3**

Rukopis neprešiel jazykovou úpravou.
Za odbornú a obsahovú stránku zodpovedá autor.