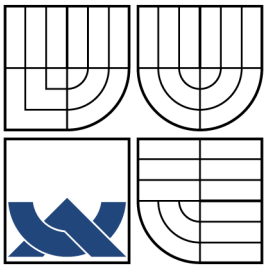
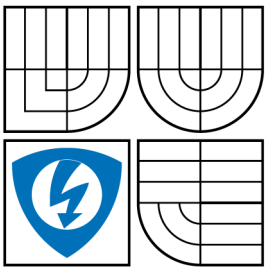


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

POČÍTAČOVÁ ANALÝZA MEDICÍNSKÝCH OBRAZOVÝCH DAT

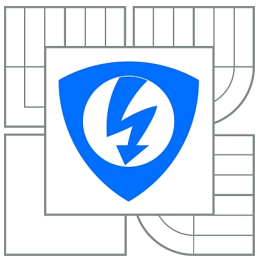
COMPUTER ANALYSIS OF MEDICAL IMAGE DATA

DIPLOMOVÁ PRÁCA
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. RÓBERT KRAJČÍR

VEDÚCI PRÁCE
SUPERVISOR
Ing. VÁCLAV UHER

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Róbert Krajčír ***ID:*** 125499
Ročník: 2 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Počítačová analýza medicínských obrazových dat

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s metodami výpočtu lokálních příznaků ve 3D. Navrhněte a implementujte metody, s pomocí kterých lze počítat lokální 3D příznaky. Funkčnost metod ověřte na reálných datech z MRI.

DOPORUČENÁ LITERATURA:

[1] Kovalev, V. A., Kruggel, F., Gertz, H. J., & von Cramon, D. Y. (2001). Three-dimensional texture analysis of MRI brain datasets. Medical Imaging, IEEE Transactions on, 20(5), 424-433.
[2] Mahmoud-Ghoneim, D., Toussaint, G., Constans, J. M., & de Certaines, J. D. (2003). Three dimensional texture analysis in MRI: a preliminary evaluation in gliomas. Magnetic resonance imaging, 21(9), 983-987.
[3] Li, X., Xia, H., Zhou, Z., & Tong, L. (2010, October). 3D texture analysis of hippocampus based on MR images in patients with alzheimer disease and mild cognitive impairment. In Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on (Vol. 1, pp. 1-4). IEEE.

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 28.5.2014

Vedoucí práce: Ing. Václav Uher
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Táto práca sa zaoberá analýzou medicínskych obrazových dát pomocou rôznych štatistických a numerických metód implementovaných v prostrediach Eclipse a Rapidminer využitím jazyku Java. Použité sú sady snímok (rezov), ktoré sú výsledkom vyšetrenia mozgu rôznych pacientov magnetickou rezonanciou. Jednotlivé segmenty v tomto 3D obraze sú podrobené výpočtu niekoľkých lokálnych príznakov, na základe ktorých sú vygenerované dátové sady pre použitie v trénovacích algoritmoch. Schopnosť týchto alogritmov úspešne identifikovať zdravé alebo choré tkanivo je následne otestovaná prakticky na dostupných dátach.

KĽÚČOVÉ SLOVÁ

lokálne príznaky, skleróza multiplex, 3D, diagnostika, vyhľadávanie, strojové učenie, Rapidminer

ABSTRACT

This work deals with medical image analysis, using variety of statisic and numeric methods implemented in Eclipse and Rapidminer environments in Java programming language. Sets of images (slices), which are used here, are the results of magnetic resonance brain examination of several subejcts. Segments in this 3D image are analyzed and some local features are computed, based on which data sets for use in training algorythms are generated. The ability of successful identification of healthy or unhealthy tissues is then practically tested using available data.

KEYWORDS

local features, Multiple sclerosis, 3D, diagnosis, search, machine learning, Rapidminer

KRAJČÍR, Róbert *Počítačová analýza medicínských obrazových dat*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 53 s. Vedúci práce bol Ing. Václav Uher

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Počítačová analýza medicínských obrazových dat“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Václavovi Uhrovi za jeho ochotu, s akou mi pomáhal k jej vytvoreniu. Moje uznanie mu patrí za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy a námety k práci. Tiež by som chcel poďakovať tímu SPL za poskytnutie podkladov a možnosti zúčastniť sa na ich výskume.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výskum popísaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených z projektu SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Vzhľadom na postupné starnutie svetovej populácie je v modernej medicíne stále častejšou témou problematika nervových a mozgových chorôb, resp. rôznych degenerácii, ktoré sa hlavne medzi staršou generáciou ľudí dajú s trochou nadsázky považovať za civilizačné. Diagnózy ako Alzheimerova či Parkinsonova choroba, skleróza multiplex alebo tumory (či už zhubné alebo nie) dokážu výrazne obmedziť, prípadne aj ohroziť život človeka alebo jeho spoločenské uplatnenie.

Keďže vo svete má počet týchto prípadov narastajúci trend, a to nielen kvôli starnutiu populácie, ale aj kvôli modernému spôsobu života, je na lekárov vyvíjaná stále vyššia záťaž v rámci stanovovania diagnóz, prevencie a hľadania účinného spôsobu liečby. Táto práca (hlavne diagnostická časť) býva často časovo náročná, pretože aj skúsený lekár musí podrobne analyzovať snímky z CT vyšetrenia a hľadať v obraze fragmenty, ktoré sem nepatria. Pravdepodobnosť chyby pri tomto hľadaní je pritom stále relatívne vysoká, keďže ľudský zrak na niektoré malé objekty nemusí byť dostatočne citlivý a ľudský chybový faktor tiež nemožno úplne vylúčiť.

Možnosťou pomoci lekárom a zdravotníckym pracovníkom je využitie moderných počítačových technológii, ktoré by zabezpečili čiastočnú, prípadne aj úplnú automatizáciu tohoto procesu. Implementáciou správneho algoritmu je možné počítač naučiť prehľadávať obrazy, a to nielen v 2D, ale aj v 3D súradniciach. Takto vytvorený program by teda mohol porovnávať jednotlivé pixely obrazu v rôznych smeroch, vyhľadávať rôzne asociácie a pracovať omnoho citlivejšie ako ľudský zrak, keďže by pracoval priamo s číselnou hodnotou pixelov.

Cieľom tejto práce je teda na základe dostupných štúdií a materiálov vytvoriť algoritmus, ktorý bude schopný vypočítať rôzne vlastnosti jednotlivých segmentov 3D obrazu a následne ich porovnávať a klasifikovať na základe modelu vytvoreného niektorým z algoritmov strojového učenia.

1 ANALÝZA DÁT V MEDICÍNE

1.1 Príznaky mozgových chorôb

1.1.1 Všeobecný popis

Ľudský mozog môže podliehať vplyvu rôznych chorôb a patogénnych javov, ktoré sú v konečnom dôsledku schopné obmedziť jeho funkciu a človeku privodiť závažné zdravotné a sociálne komplikácie, v najhoršom prípade až smrť. Moderná medicínska veda je schopná väčšinu týchto mechanizmov (aj vďaka rozvoju techniky) popísať a vysvetliť, avšak vo väčšine prípadov stále nie sú známe presné a zaručené postupy liečby, ktorá by bola naozaj účinná a dospela až k úplnému vyliečeniu. Sústreďuje sa teda na včasnú diagnózu (včasnou liečbou je možné nástup choroby výrazne oddialiť) a spomaľovanie priebehov ochorení a úľavu od symptómov.

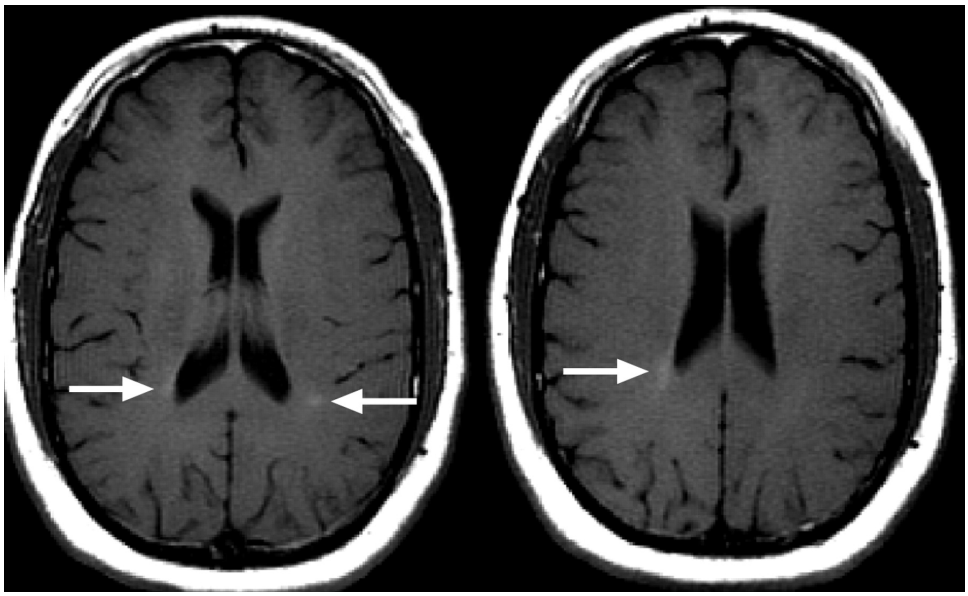
Medzi najbežnejšie choroby a degenerácie, ktoré postihujú populáciu hlavne staršej, ale aj mladšej generácie, sa radia skleróza multiplex (SM) (tiež známa ako roztrúsená mozgovomiešna skleróza) a Alzheimerova choroba (ACh). Hlavne počet chorých s ACh sa odhaduje až na 50% všetkých pacientov s demenciou [2] a pri neobjavení účinnej liečby sa vzhľadom na starnutie populácie očakáva výrazný nárast.

Pôvod týchto chorôb je odlišný - kým SM je autoimunitné ochorenie, ACh má viac degeneratívnu príčinu. Napriek tomu sú si ich dôsledky a príznaky v mnohých oblastiach podobné. Obe choroby charakterizujú poruchy vo vedení nervových vzruchov v centrálnej nervovej sústave - mozgu a mieche. Postupom času spôsobujú rozpad alebo blokáciu nervových spojení a zakončení a tým čiastočnú alebo aj úplnú invaliditu či demenciu (hlavne ACh). Pri sklerózach je toto spôsobené napádaním nervových zakončení vlastným imunitným systémom, vznikom zápalov a následnou obmedzenou schopnosťou neurónu vysielať alebo prijímať akékoľvek podnety. Alzheimerova choroba sa naopak vyznačuje postupným samovoľným rozpadom zakončení, vo výsledku s väčším dôsledkom na pamäť a osobnosť človeka.

1.1.2 Zmeny v štruktúre mozgu

Proces popísaný vyššie spôsobuje tiež zmeny v štruktúre a hustote mozgového tkaniva, ktoré je možné zachytiť pri vyšetrení MRI. Pri SM napríklad imunitný systém napáda izolačnú látku neurónových výbežkov - myelin. V lekárskej praxi je tento proces nazývaný atak a na pacientovi sa na rozdiel od ACh (ktorá napreduje pomaly a bez očividných príznakov) prejavuje aktívne obmedzením pohybu a/alebo prudkými bolesťami. V miestach, kde takýto zápal odoznel, možno pozorovať zjazvenie v bielej mozgovej hmote (tzv. lézie). Táto deformácia je detekovateľná pri

vyšetrení, ako je možno vidieť na obr 1.1 [5].



Obr. 1.1: Lézie na mozgu pacienta trpiaceho SM

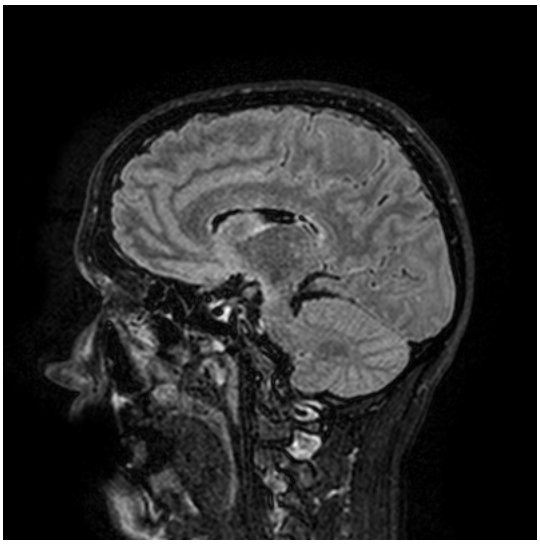
Pri skúmaní Alzheimerovej choroby post mortem sa zistilo, že z dosiaľ neznámych príčin sa nervové bunky a vlákna rozpadávajú. V mozgovej kôre možno pozorovať chumáče takýchto rozpadnutých spojení. Toto sa prejaví na zmene hustoty tkaniva, takže pri vyšetrení magnetickou rezonanciou (MRI) to možno vidieť ako zmenu odtieňa snímky. Podľa niektorých štúdií [13] je možno na základe zmeny hustoty tkaniva časti mozgu nazývanej hippocampus (zodpovedná za ukladanie spomienok do dlhodobej pamäte) konštatovať, že pacient má ranné štádium ACh. Nástup choroby je ale veľmi pozvoľný, takže vo včasných fázach sú tieto zmeny voľným zrakom takmer nepozorovateľné.

Tu sa objavujú prvé podnety na vypracovanie tejto práce - výhody implementácie vyhľadávacieho algoritmu počítačom, ktoré je schopné proces vyhľadávania artefaktov v snímkach pacientov s ACh alebo SM značne spresniť a urýchliť, sú v prípade úspešnosti nesporné.

1.2 Diagnostika magnetickou rezonanciou

Vyšetrenie magnetickou rezonanciou je v dnešnej dobe jedným z najpoužívanejších diagnostických nástrojov v liečbe nervových a ďalších ochorení. Keďže je to neinvazívne vyšetrenie s dobrou citilivosťou a jasne zaberá aj mäkké tkanivá, značne sa rozšírilo aj napriek finančnej náročnosti prístrojového vybavenia. Jej princíp spočíva vo vystavení pacienta do silného (až 3T) homogénneho magnetického poľa, pričom

sa sníma odozva na krátky rádiofrekvenčný impulz. Výstupné obrazy sú v odtieňoch šedej, spravidla vo forme množstva 2D rezov (príklad nám dostupných snímok je na obr. 1.2), sú ale známe aj techniky, ktoré vykresľujú objemový 3D obraz.



Obr. 1.2: Snímka jedného rezu pri MRI vyšetrení mozgu

Proces vyhodnotenia takýchto snímok prebieha na mnohých klinikách stále manuálne, kedy rádiológ prezerá snímku po snímke a vizuálne hľadá fragmenty, ktoré do zdravého tkaniva nepatria. V prípade podozrenia na sklerózu sa zameriava na lézie, čiže zjazvenie po zápale, prípadne pozoruje hustotu tkaniva. Keďže počet snímok predstavuje často viac ako 200, aj skúsenému špecialistovi táto práca zaberie veľa času, pričom nie je vylúčené, že niektoré fragmenty prehliadne. Taktiež je v mnohých prípadoch problém nájsť presnú hranicu medzi abnormalitou (tumorom, léziou, degenerovanou oblasťou apod.) a zdravým tkanivom. Asistencia vo forme programu, ktorý je schopný na jednotlivé fragmenty rádiológa upozorniť, prípadne presnejšie vyznačiť oblasť, ktorú zaberajú, je teda vítaná a aktívne skúmaná po celom svete.

1.2.1 Možnosti automatizácie prehľadávania

Využitie výpočtovej techniky ponúka potenciál spracovania 2D rezov do 3D obrazu, s následnou možnosťou vyhľadávať asociácie medzi pixelmi v smere všetkých troch osí. Doterajšie možnosti spracovania 2D obrazu sú na profesionálnej úrovni, hlavne jazyk JAVA, používaný v multimédiách, je vhodným a často používaným prostriedkom pre spracovanie podobných dát (analýza textúr, predikcia pohybu apod). Dôvodov na jeho úspešnosť je viac - vďaka použitiu virtuálneho stroja prispôsobeného

hardwaru (Java Virtual Machine) je použitie jazyka nezávislé na platforme, čo prinieslo masové rozšírenie. Vďaka tomu, a tiež vďaka svojej objektovo orientovanej architektúre, sa JAVA stala obľúbeným nástrojom aj v oblasti spracovania obrazu.

Avšak spracovanie iba v 2D smere môže vynechať, resp. obísť dôležité informácie obsiahnuté v snímkach. Možnosť 3D spracovania a hľadania súvislostí medzi jednotlivými rezmi má stále veľký potenciál skúmania a množstvo rôznych prístupov ako takéto obrazy vyhodnotiť, napríklad koexistenčné matice skúmané v [7] a [8].

Rovnako je v tejto oblasti atraktívna, a z výskumného hľadiska zaujímavá možnosť implementácie učiacich sa algoritmov, akým je aj proces navrhnutý v [14]. Najjednoduchšie formy analýzy porovnávajú snímky s vopred natrénovanou definovanou množinou snímok, ktoré boli analyzované skúseným rádiológom a presne označené ako snímky zdravého alebo chorého jedinca. Zavedenie učiaceho sa programu by znamenalo priebežné rozširovanie trénovacej množiny s minimálnymi zásahmi zvonku a spresňovanie budúcich výsledkov.

Snímky tiež nemusia byť, a spravidla ani nie sú analyzované a porovnávané po pixeloch. Takýto prístup nie je veľmi presný, pretože neumožňuje širší záber a rozpoznávanie väčších objektov v obraze. Často sa používa práve metóda porovnávania jednotlivých oblastí snímku, a to na základe numerických metód, napríklad štatistického vyhodnotenia hodnôt pixelov a následného vypočítania rôznych lokálnych príznakov, na základe ktorých je možno označiť snímku (resp. pacienta) ako chorého alebo zdravého [8]. Lokálne príznaky používame aj v tejto práci, preto sú na ďalších stranách popísané podrobnejšie.

1.3 Aplikácia numerických metód

1.3.1 Využitie štatistiky

Pre porovnanie dvoch alebo viacerých skupín objektov (v našom prípade oblasti s pixelmi) môže byť práve niektorá štatistická hodnota jedným z možných porovnávacích kritérii. Výpočtom týchto hodnôt (napríklad minimum, maximum, rôzne druhy odchýlok apod.) dostaneme malé množstvo údajov, ktoré je ale schopné reprezentovať celú skupinu, resp. oblasť navonok. Môžeme teda použiť aj túto disciplínu, čím v konečnom dôsledku dostaneme viacej informácii o skladbe konkrétnej oblasti. Tieto môžu byť využité na ďalšiu klasifikáciu snímku, prípadne môžu odhaliť na ktoré oblasti sa zamerať viac (napríklad príliš nízka alebo vysoká priemerná hodnota pixelov v oblasti môže indikovať zmenu hustoty tkaniva).

Z výpočtového hľadiska bývajú štatistické metódy pomerne rýchle, pretože neobsahujú moc veľa operácii. V tejto práci bola využitá voľne stiahnuteľná knižnica Apache commons - math, konkrétne jej časť `commons.math.stat.descriptive.*`

ktorá sa venuje popisnej štatistike. Ako prehľadne uvádza manuál [1], princíp fungovania knižnice je vo vytvorení samostatnej inštancie niektorej z podtried. Pole vytvorené konštruktorom tejto triedy je potom naplnené hodnotami, ktorých skupinové štatistické vlastnosti budeme následne skúmať. Ďalším krokom je už iba využitie predpripravených metód `get`, ktoré vracajú výsledky výpočtov. Knižnica teda pracuje rýchlo, čo bolo odskúšané aj v rámci tejto práce vytvorením triedy `VolumeStatistics`, ktorá je využívaná v triede `Statistics3Dfeatures`, čo je trieda prispôsobená pre inštaláciu do prostredia Rapidminer, ktorému je v tejto práci venovaná samostatná sekcia 3.1.

Vytvorená trieda je teda jednoducho (troma vnorenými cyklami, každý pre jeden smer x,y,z) naplnená hodnotami pixelov a vracia základné štatistické hodnoty oblasti ako minimum, maximum, rozptyl, smerodajná odchýlka, geometrický a aritmetický priemer a rôzne iné. Príklad kódu je uvedený nižšie (výstup je krátený o ošetrenie pretečení oblasti, nastavenie premenných apod.). Odpovedajúci operátor v Rapidminer, ktorý bol za účelom implementácie tejto triedy v novom prostredí vytvorený, má názov *local_3d_statistics*.

```
public class VolumeStatistics {
 int x,y,z,area;
 SummaryStatistics stat = new SummaryStatistics();

public int setStatValues(ThreeDimIOObject image) {

 for (int k = z-area; k < z+area; k++) {
  for (int i = x-area; i < x+area; i++) {
   for (int j = y-area; j < y+area; j++) {
    stat.addValue(image.getVoxel(i, j, k));
 }
  }
   }
 return 1;
 }
```

Volanie štatistických výsledkov v triede `Statistics3Dfeatures` priamo na výstup:

```
features.setFeature(getAttributeName(image) + "_" +
"GeoMean", volStat.getGeoMean());
features.setFeature(getAttributeName(image) + "_" +
"Deviation", volStat.getDeviation());
```

1.3.2 Lokálne príznaky

Vo všeobecnosti je ťažké presne definovať čo je lokálny príznak, keďže definície sa v rôznej literatúre líšia. V rámci spracovania obrazu sa dá povedať, že príznakom je istá časť snímku, ktorá je niečím zaujímavá a môže byť použiteľná pre ďalšie spracovanie, vyhľadávanie alebo porovnávanie. Príznakom môže byť bod, hrana, alebo aj skupina bodov, ktoré sa spravidla odlišuje od zvyšného snímku svojou intenzitou, farbou alebo textúrou, ale ako kritérium môžu byť použité aj rôzne iné parametre podľa potreby vyvíjanej aplikácie.

Vyhľadávanie takýchto príznakov býva medzi prvými krokmi pri spracovaní obrazu a vyhľadávaní rôznych fragmentov. V momente, ako je objavený bod alebo oblasť záujmu, môže byť táto rozšírená a podrobená ďalšiemu podrobnejšiemu skúmaniu - napríklad rôzne merania alebo výpočty. Tieto sú následne spracované na deskriptory (popisné prvky) a využívané v ďalších aplikáciach. Vzhľadom na to, že tieto procesy bývajú často počiatočnými fázami zložitejších algoritmov, vo veľkej miere na nich závisí úspešnosť a efektivita celého systému.

Pre rozlíšenie typov príznakov tiež neexistuje žiadne všeobecne známe pravidlo alebo definícia, ktorá by ich striktne klasifikovala. Vo voľne dostupných zdrojoch však možno spravidla objaviť tieto najrozšírenejšie typy príznakov:

- Okraje (edges) - hranica medzi dvoma oblasťami snímku
- Rohy (corners) - označenie prudkých zmien okrajov alebo iných bodov záujmu
- Oblasti záujmu (regions of interest) - opis obrazu v zmysle skupín/oblastí bodov (narozdiel od rohov presnejšie v nevýrazných a jemných oblastiach)
- Vyvýšeniny (ridges) - používané na rozlišovanie podlhovastých objektov ako cievy alebo cesty

2 STROJOVÉ UČENIE

Samotné procesy strojového učenia, prípadne dolovania znalostí, pozostávajú z viacerých procedúr, z ktorých každá je dôležitá a má veľký podiel na kvalite výsledku. Všetky použité algoritmy sú spravidla pre každú aplikáciu takéhoto učenia nastavené, resp. vybrané tak, aby v danom kontexte podávali čo najlepšie výsledky. Preto proces, ktorý v prípade nejakého typu dát podáva dobré výsledky, môže v inom úplne zlyhať (napr. proces naučený identifikovať v obrázku automobil už nemusí byť tak úspešný v prípade kamiónu). Najdôležitejšie fázy a postupy strojového učenia sú bližšie opísané práve v tejto kapitole.

Učenie ako také vieme klasifikovať nasledovne [4]:

- Nekontrolované - vstupné vzorky dátovej sady nemajú vopred označenú triedu, do ktorej by mali prináležať. Takéto učenie iba rozdelí dáta do kategórii podľa vzájomnej podobnosti, nie je však schopné vyhodnotiť správnosť takéhoto rozdelenia, ani definovať význam jednotlivých kategórii (napr. na základe podobnosti rozdelí snímky zdravého a chorého človeka, nie je však schopné povedať ktorý je ktorý).
- Čiastočne kontrolované - pri budovaní modelu pracuje s označenými aj neoznačenými vzorkami. Označené vzorky definujú konečné triedy a ostatné sú použité na spresnenie hraníc medzi nimi.
- Aktívne - užívateľ sa môže aktívne zapojiť do procesu prostredníctvom dotazu algoritmu a poskytnúť označenie nejakej vzorke pomocou svojej znalosti.

2.1 Vytvorenie množiny dát

Výber vhodných dát, ich spracovanie a celá fáza tzv. pre-procesingu je jednou z kľúčových častí dolovania znalostí. Platí totiž všeobecne známe pravidlo, že každý učiaci sa algoritmus môže byť iba tak dobrý, aká vysoká je kvalita a kvantita trénovacích dát. Bežne sa totiž stýkame s problémami zarušených, duplikátnych alebo úplne bezvýznamných dát, ktoré istým spôsobom kontaminujú vstupnú dátovú sadu. Rovnako platí, že vstupný dátam by sme mali rozumieť a chápať ich. Nemá žiadny zmysel porovnávať trénovacie dáta, ktoré pre nás nemajú výpovednú hodnotu a nemáme ani približnú predstavu čo môžeme očakávať.

Od vstupných dát spravidla vyžadujeme najmä nasledujúce vlastnosti:

- Validita - atribúty majú správny dátový typ, nevybočujú z povoleného rozsahu, nie sú prázdne, ak treba sú unikátne (napr. rodné číslo) apod.
- Presnosť - ak atribút odkazuje na nejakú skutočnosť, táto by mala platiť (napr. neexistujúce PSČ).

- Kompletnosť - pokiaľ možno všetky atribúty by mali obsahovať nejaké hodnoty, pokiaľ možno platné.
- Konzistencia - dva záznamy v dátovej sade by si nemali navzájom odporovať.

V priebehu pre-procesingu sme schopní množstvo nedostatkov v horeuvedených vlastnostiach odstrániť, hoci je to často krát na úkor inej vlastnosti. Chýbajúce hodnoty je možné nahradiť hodnotou východziou (default), náhodnou, prípadne ich odstrániť z dátovej sady, čo platí aj pre duplikátne alebo kolidujúce záznamy za podmienky, že sa trénovacie dáta nestanú príliš riedkymi.

Samotné atribúty pritom môžeme deliť podľa operácii, ktoré s nimi môžeme podnikať. Nie všetky učiace sa algoritmy sú totiž vhodné na všetky typy atribútov. Rozdelenie aj s uvedenými operáciami je nasledovné:

- nominálne ($=\neq$)
- binárne (pravda, nepravda)
- ordinálne ($=\neq<>$)
- numerické (kvantitatívne)
  - interval ($=\neq<>+-$)
  - percento ($=\neq<>+-\times\div$)

Z pohľadu konečnosti množiny hodnôt, ktoré môžu atribúty nadobúdať, ich môžeme ešte ďalej označiť ako diskrétne alebo spojité (spravidla s pohyblivou desatinnou čiarkou) [4].

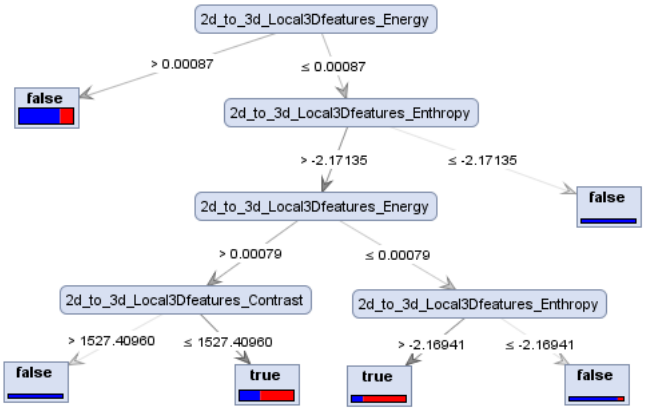
2.2 Trénovacie algoritmy

Tieto algoritmy tiež bežne nazývame klasifikátory, keďže nám sprostredkovávajú funkciu radenia (klasifikácie) vstupných záznamov do jednotlivých tried. Proces klasifikácie spravidla pozostáva z dvoch hlavných fáz - trénovací krok a klasifikačný krok. V prvej fáze je zostavené mapovanie, respektíve funkcia, ktorá na základe hodnôt vstupných parametrov priraďuje hodnotu konečnej triedy. V druhej fáze nastáva samotná klasifikácia pomocou testovacích dát. Poznáme rôzne druhy klasifikačných algoritmov, práca a základné princípy niektorých z nich sú vysvetlené v tejto sekcii.

2.2.1 Rozhodovacie stromy

Rozhodovacie stromy sú jedným s najjednoduchších a užívateľsky najprívetivejších algoritmov používaných na strojové učenie. Používajú sa (tak ako aj ostatné obdobné algoritmy) na klasifikáciu objektov alebo dátových záznamov (inštancií) do preddefinovaných tried na základe ich atribútov (príznakov). Často sa vyskytujú v rôznych vedných disciplínach, ako napríklad finančníctvo, marketing, medicína, počítačové vedy a rôzne ďalšie. Ich hlavnou prednosťou je, že človek nemusí byť expertom

aby ich vedel interpretovať, keďže sa vyobrazujú graficky s hierarchickou štruktúrou tvorenou na základe jednoduchých podmienok. Spomínaná štruktúra má tvar prevráteného stromu, ktorý začína od uzlu bez vstupov (koreň, ang. root). Tento sa ďalej rozvetvuje do interných uzlov (charakterizované jedným vstupom a jedným a viac výstupmi) a koncových listov (konečné, rozhodujúce uzly, ang. leaves), ktoré reprezentujú triedy do ktorých zaraďujeme vstupné objekty, ako je uvedené na obrázku 2.1. Uzol sa môže rozvetviť na základe rôznych kritérii (spravidla vopred nastavených - zameraných na presnosť, informačnú hodnotu apod.). Samotný proces rozvetvovania a konečný vzhľad stromu vie užívateľ ovplyvniť zadaním parametrov ako napríklad maximálna hĺbka (počet úrovní) stromu, počet listov a hodnota informačného zisku (Gain), pri ktorom sa uzol ešte rozvetví. Algoritmus rozvetvovania, ako aj spôsob výpočtu kritérii je presne popísaný v [12].



Obr. 2.1: Rozhodovací strom

Štruktúra klasifikácie je teda zrejmá - vezmeme vstupný objekt a smerom od koreňa dolu porovnávame hodnoty jeho atribútov (príznakov), čím postupujeme celým stromom až ku konečnému klasifikačnému listu. Tento postup (a zároveň celý strom) by nemal byť príliš dlhý, pretože pokiaľ je strom príliš rozsiahly, s najväčšou pravdepodobnsťou nebude veľmi presný [12].

2.2.2 Neurónové siete

Vzorom a impulzom pre vznik algoritmov tohoto typu bolo skúmanie centrálneho nervového systému živých tvorov. Je známe, že nervové bunky tvoria obrovskú sieť, ktorá je schopná práce na komplikovaných procesoch, výpočtoch a jej pamäťové kapacity sú obrovské. Na základe toho sa teda začal vyvíjať algoritmus, ktorý by mal byť adaptibilný a schopný paralelných a kolektívnych výpočtov v spolupráci

viacerých uzlov, ktoré svojou štruktúrou a usporiadaním pripomínajú práve nervový systém živých organizmov.

Základná myšlienka, na základe ktorej sú tieto algoritmy spravidla stavané, je že neurónové siete by mali byť schopné spočítať akúkoľvek vypočítateľnú (resp. vyčísliteľnú) funkciu a každý uzol by mal byť ľahko implementovateľný na samostatnej jednotke (procesore), čím sa dosiahne určitá forma nezávislosti a zvýšenia výpočtového výkonu [6], čo je výhodné hlavne v dnešnej dobe počítačov s viacerými procesormi alebo vláknami.

Neurónová sieť sa dá definovať ako orientovaný graf. Model biologického neurónu sa v tejto oblasti nazýva preceptrón. Tento prijíma signály (vstupné objekty, vzory) $\bar{x}$ vo forme binárnych alebo reálnych hodnôt použitím váhového vektoru $\bar{w}$. Jeho výstup je teda váhovaná suma vstupov [9]:

$$o = f(\bar{w} \cdot \bar{x}) = f\left(\sum_{j=1}^{n+1} w_j x_j\right) \tag{2.1}$$

kde f je tzv. aktivačná funkcia preceptrónu.

Pokiaľ je táto aktivačná funkcia binárna (bipolárna) a používa tzv. diskrétny preceptrón, dostávame napr. pre 2-rozmerný priestor dva podpriestory a preceptrón dokáže klasifikovať vzory do dvoch tried, čiže rieši lineárne separovateľné problémy (AND, OR) [9].

Učenie (t.j. hľadanie funkcie medzi šturkúrou jej objektov a ich vlastnosťami) s použitím neurónových sietí prebieha nasledovne: máme učiaci signál s, ktorý je funkciou $\bar{x}$, $\bar{w}$ a prípadne aj spätnej väzby od užívateľa (ktorý v tomto prípade vystupuje v roli akéhosi učiteľa) podľa toho, či ide o nekontrolované alebo čiastočne kontrolované strojové učenie. V diskrétnych krokoch sa následne mení váha nasledovne [9]:

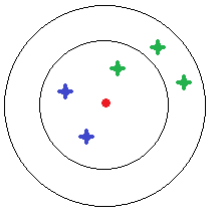
$$w_j(t+1) = w_j(t) + \Delta w_j(t) = w_j(t) + \alpha s(t) x_j(t), \tag{2.2}$$

kde konštanta α je tzv. rýchlosť učenia ($0 < \alpha \leq 1$).

Pre riešenie problémov komplikovanejšieho charakteru ako je trieda lineárne separovateľných problémov sa nepoužívajú uvedené jednoduché neurónové siete, ale komplikovanejšie algoritmy, ako je napríklad gradientový algoritmus (navrhnutý p. Rumelhartom) využívajúci viacvrstvé neurónové siete, tiež známy aj ako metóda spätného šírenia. Táto metóda otvorila možnosti implementácie neurónových sití na zložitejšie nelineárne problémy, čím sa viacvrstvé neurónové siete ešte viac spopularizovali.

2.2.3 Algoritmus *k*-NN

Skratka označujúca tento algoritmus pochádza z anglického k-Nearest-Neighbors (k najbližších susedov), kde už názov samotný napovedá, aký je princíp fungovania tejto metódy. Jedná sa o neparametrickú metódu, ktorá sa radí k najjednoduchším algoritmom strojového učenia a ktorá klasifikuje vstupné dáta na základe blízkosti hodnôt ich atribútov k atribútom iných vstupných objektov (z pohľadu priestoru alebo roviny v ktorej sa tieto hodnoty nachádzajú). Pre lepšiu predstavivosť je dobrou analógiou vykreslenie hodnôt atribútov na obrázku 2.2, kde k-NN algoritmus vyberá do jednej triedy body ležiace najbližšie k sebe. Inak povedané - objekt je predikciou zaradený do tej triedy, v ktorej sa nachádza väčšina zo zaradeného (vopred klasifikovaného a označeného) počtu k jeho susedov. V prípade, že $k = 3$, je červený bod zaradený do modrej triedy. Ak je $k = 5$, zaradí algoritmus tento bod do zelenej triedy. Hodnoty k sa najčastejšie volia ako nepárne čísla, aby sa zamedzilo



Obr. 2.2: Klasifikácia algoritmom k-NN

výskytu nerozhodných výsledkov (špeciálny prípad je $k = 1$ - klasifikuje sa podľa jedného najbližšieho suseda).

Pre určenie vzdialenosti medzi dvoma bodmi v n-rozmernom priestore sa najčastejšie využíva štandardná euklidovská vzdialenosť [4]:

$$d(x, y) = \sum_{i=1}^{n} \sqrt{(y_i - x_i)^2} \tag{2.3}$$

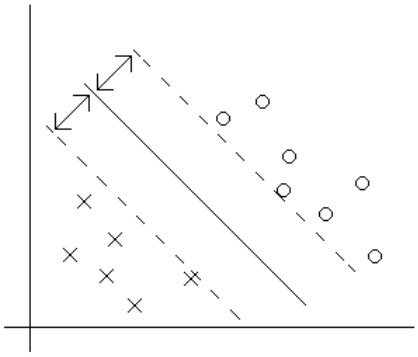
kde x a y sú body, medzi ktorými určujeme vzdialenosť pre n-rozmerný priestor udávaný počtom atribútov. Pre nominálne (nečíselné) hodnoty atribútov sa ich hodnote buď priradí číslo a postupuje sa štandardne, alebo sa využije iný výpočet vzdialenosti (napr. Hammingova vzdialenosť pre dva reťazce) [4].

Metóda dáva najlepšie výsledky pre nízku dimenzionalitu vstupných dát (počet atribútov / príznakov jedného objektu). V prípade vyšších hodnôt (napr. 10 a viac) je euklidovská vzdialenosť najbližších a najvzdialenejších objektov takmer ekvidištantná, takže metóda stráca svoj zmysel. Rovnako dáva algoritmus veľmi slabé výsledky pri vyššom pomere zašumených alebo chýbajúcich dát.

2.2.4 Metóda podporných vektorov

Jedna z najnovších metód nesie názov pochádzajúci z anglického Support Vector Machine (SVM). Dá sa využiť na klasifikáciu lineárnych aj nelineárnych dát a používa mapovanie, ktorým transformuje vstupné dáta (trénovacie) do viacrozmerného priestoru. V tomto priestore následne hľadá tzv. nadrovinu, ktorá bude od seba rozdeľovať dáta prináležiace rôznym výstupným triedam. Nadrovinu v tomto kontexte chápeme podľa jej štandardnej geometrickej definície - je to podpriestor (dimenzie n-1) daného priestoru (dimenzie n), ktorý túto dimenziu delí na dva podpriestory (polopriestory). Za týmto účelom sú využité podporné vektory a nimi definované veľkosti „okrajov“ okolo nadrovín. V tejto časti sa budeme venovať iba prípadu, keď sú vstupné dáta lineárne oddeliteľné, keďže druhý prípad lineárne neoddeliteľných dát z veľkej časti vychádza z tohoto základu a obsahuje iba niekoľko rozdielov (napríklad viacero nelineárnych mapovaní), ktoré sú prehľadne vysvetlené v [4].

Pri lineárne separovateľných dátach teoreticky existuje nekonečné množstvo nadrovín (priamok pre 2D, rovín pre 3D priestor apod.), ktoré sú schopné správne oddeliť dáta prináležiace jednej triede od tých patriacich inej. Uvádzané vysvetlenie je koncipované na príklade dvoch výstupných tried. Snažíme sa nájsť takú nadrovinu, ktorá má smerom k najbližším dátovým objektom jednej aj druhej triedy čo najširší prázdny okraj, ako je zobrazené na obrázku 2.3 (body prináležiace jednotlivým výsledným triedam označené tvarom, deliaca nadrovina, t.j. v 2D priestore priamka zobrazená plnou čiarou a okraje čiarkovanou). Toto je z toho dôvodu, že do budúc-



Obr. 2.3: Okraje deliacej nadroviny

nosti predpokladáme vyššiu variabilitu vstupných dát než sú tie trénovacie, a preto je vhodná akási „rezerva“, vďaka ktorej bude klasifikácia presnejšia. Body deliace priamo na okraji sú samotnými podpornými vektormi. Matematicky môžeme našu

deliacu nadrovinu pre účely tohoto algoritmu zapísať ako [4]:

$$\mathbf{W} \cdot \mathbf{X} + b = 0 \tag{2.4}$$

kde $\mathbf{W} = \{w_1, w_2, ..., w_n\}$ je váhovací vektor pre počet n atribútov a b je výchylka.

V prípade, že máme pre každý dátový objekt dva vstupné atribúty, t.j. z nich vytvorený priestor bude dvojrozmerný (2D), môžeme vzťah 2.4 prepísať ako rovnicu roviny nasledovne:

$$w_0 + w_1 x_1 + w_2 x_2 = 0 \tag{2.5}$$

kde parameter b sme nahradili výrazom w_0 s rovnakou funkciou.

Na základe horeuvedeného výrazu teda vieme pomocou jednoduchých nerovností vyjadriť, či nejaký bod leží na jednej alebo druhej strane takto definovanej roviny, a teda či prináleží jednej alebo druhej triede.

2.3 Vyhodnocovanie presnosti algoritmov

Dôležitým krokom po vytvorení (resp. natrénovaní) nového modelu je overenie presnosti ním poskytovaných výsledkov. Toto je samozrejme nemožné pri nekontrolovanom učení (bez učiteľa), kde vstupné dáta nie sú nijak označené a teda nevieme rozlíšiť, ku ktorej triede by mali skutočne patriť. Naopak je táto časť procesu veľmi žiadúca (priam nevyhnutná) pri ostatných druhoch strojového učenia, keďže presnosť nám neposkytne len obraz o úspešnosti, ale umožní nám aj porovnávať jednotlivé algoritmy medzi sebou.

Testovať výsledky na tých istých dátach, na ktorých bol model natrénovaný, je ale nezmyselné, pretože by sme dostali skreslené údaje. Z tohoto dôvodu je teda potrebné vstupné dáta rozdeliť na trénovacie a testovacie, prípadne aj na dáta validačné. Využitím týchto dát môžeme modelu poskytnúť nové dáta, ktoré môže klasifikovať na základe vytvoreného modelu, no sú pritom aj označené skutočnou triedou do ktorej by mali patriť, čo nám umožňuje ľahko zistiť správanie sa modelu a umožniť jeho úpravu, doladenie, alebo voľbu iného algoritmu.

Jeden zo základných nástrojov na posudzovanie presnosti strojového učenia je tzv. matica zámien (z ang. confusion matrix). Pokiaľ uvažujeme s m výstupnými triedami, je táto matica tabuľkou o rozmeroch $m \times m$, pričom záznam $A_{i,j}$, kde $i, j \in \{1, 2, ..., m\}$, predstavuje množstvo vzoriek skutočnej triedy i modelom zaradených (označených) do triedy j. Je teda zrejmé, že od dobrého klasifikátoru budeme chcieť, aby prevažná väčšina záznamov ležala na diagonále tejto matice a ostatné záznamy v matici boli pokiaľ možno nulové [4]. Príklad matice zámien pre dve štandardné triedy true a false z prostredia Rapidminer je uvedený na obr. 2.4.

accuracy: 71.19%			
	true false	true true	class precision
pred. false	1135	278	80.33%
pred. true	591	1012	63.13%
class recall	65.76%	78.45%	

Obr. 2.4: Matica zámien (confusion matrix)

V prípade, že nás viac zaujíma presnosť v rozličných triedach, môžeme z tejto matice vypočítať aj ďalšie parametre, akými sú senzitivita (vyjadruje koľko pozitívnych vzoriek bolo správne identifikovaných), špecifickosť (správne identifikované negatívne vzorky) a precíznosť (koľko vzoriek označených ako pravdivých má v skutočnosti hodnotu pravda). Z týchto hodnôt vieme následne odvodiť aj celkovú presnosť podľa nasledujúcich vzťahov [4]:

$$senzitivita = \frac{TruePos}{Pos} \tag{2.6}$$

$$specifickost = \frac{TrueNeg}{Neg} \tag{2.7}$$

$$preciznost = \frac{TruePos}{TruePos + TrueNeg} \tag{2.8}$$

$$presnost = senzitivita\frac{Pos}{Pos + Neg} + specifickost\frac{Neg}{Pos + Neg} \tag{2.9}$$

2.3.1 Testovanie

Ako už bolo spomenuté, je dôležité testovať naučený model na nezávislých dátach. Toto býva ale často krát problémom, keďže vstupné dáta nebývajú nekonečné a prípadov, kedy je dát „nazvyš“ je skôr menej. Preto boli zavedené metódy testovania, ktoré sa s týmto faktom vedia vysporiadať a vedia pracovať s trénovacími dátami aj na testovanie. V každom prípade ale platí že validačné dáta by v žiadnej z týchto množín nemali byť zahrnuté.

Základnými metódami využívanými v prípade nedostatku dát sú tzv. preťahovacia metóda (z ang. holdout method) a jej rozšírenie - náhodné podvzorkovanie (z ang. random subsampling). V tejto metóde sú vstupné trénovacie dáta náhodne rozdelené medzi trénovacie a testovacie (najčastejšie v pomere $^2/_3$ - $^1/_3$, ale nemusí to byť pravidlo). Pri náhodnom podvzorkovaní je uvedená metóda zopakovaná viacnásobne po sebe a výslednú presnosť dostávame spriemerovaním takto získaných čiastkových presností [4].

Cross-validation

Krížová validácia, známejšia pod jej anglickým názvom, je sofistikovanejšou testovacou metódou, ktorá rozdeľuje vstupné dáta do k vzájomne sa vylučujúcich (na sebe nezávislých) skupín (resp. podmnožín) rovnakej veľkosti. S nimi vykoná precedúru učenia sa a testovania dokopy k-krát, pričom pri každej iterácii je jedna skupina (zakaždým iná) použitá na testovanie a zvyšných $k - 1$ skupín je použitých na trénovanie. Každá skupina je teda raz použitá na trénovanie a v ostatných pokusoch na učenie. Špeciálnym prípadom je, keď k je rovné množstvu vstupných vzoriek, takže v jednej iterácii sa vždy pre trénovanie vynechá iba jedna vzorka. Z toho vyplýva aj označenie leave-one-out (vynechaj jedno).

Rozdelenie dát do podmnožín môže prebiehať viacerými spôsobmi. Možnosti a nápoveda programu Rapidminer (podrobnejšie popísaný v ďalšej kapitole, v sekcii 3.1) ponúkajú napríklad nasledovné:

- lineárne vzorkovanie (linear sampling) - poradie zostáva zachované, dáta sú delené tak ako idú za sebou
- náhodné vzorkovanie (shuffeled sampling) - dáta sú do skupín vyberané náhodne
- prekladané vzorkovanie (stratified sampling) - dáta sú do skupín vyberané náhodne, ale s ohľadom na zachovanie rovnomerného rozloženia vzoriek prináležiacich rôznym triedam vo vytváraných skupinách

Ako je doporučené v [4], je vhodné používať krížovú validáciu s prekladaným vzorkovaním s 10 podmnožinami pre jej nízku odchýlku a rozptyl.

Bootstrap

Pri tejto metóde je hlavným rozdielom oproti ostatným to, že vzorkovanie vstupných dát prebieha nahradzovaním. Znamená to, že aj keď bola vstupná vzorka vybraná na trénovanie, má stále rovnakú pravedepodobnosť že môže byť vybraná znova a pridaná do trénovacej sady ďalší krát. Trénovacia sada teda môže obsahovať duplikáty, ktorých počet je teoreticky neobmedzený. Vzorky, ktoré sa do trénovacej množiny dát nedostali vôbec, vytvoria testovaciu sadu, na ktorej budú výsledky overované.

Existuje niekoľko variácii metódy bootstrap. Ako uvádza [4], najpoužívanejšou je .632 bootstrap. Princíp jej fungovania je v dostupnej literatúre opísaný nasledovne: máme vstupné dáta o veľkosti d vzoriek. Tieto sú navzorkované d krát s použitím nahradzovania, čím vytvoria trénovaciu sadu o veľkosti d vzoriek. Ako bolo spomenuté vyššie, v tejto sade sa môžu vyskytovať duplikáty, čo je vzhľadom na princíp fungovania metódy normálne. Zvyšné dáta vytvoria testovaciu sadu, v ktorej je 36,8% dát, zvyšných 63,2% je práve v bootstrap trénovacej sade (tieto hodnoty sa dajú ľahko odvodiť pomocou teórie pravdepodobnosti).

Uvedený proces zopakujeme k krát, podobne ako pri krížovej validácii, kde pri každej iterácii použijeme testovaciu sadu, ktorá ostala ako zvyšok po vytváraní tej trénovacej, na výpočet presnosti práve zostaveného modelu. Celkovú presnosť teda vieme vyjadriť ako [4]:

$$Acc(M) = \sum_{i=1}^{k} (0,632 Acc(M_i)_{test} + 0,368 Acc(M_i)_{tren}), \tag{2.10}$$

kde $Acc(M_i)_{test}$ je presnosť modelu bootstrap vzorky i aplikovaného na testovaciu sadu i a podobne $Acc(M_i)_{tren}$ je presnosť toho istého modelu aplikovného na originálnu dátovú sadu.

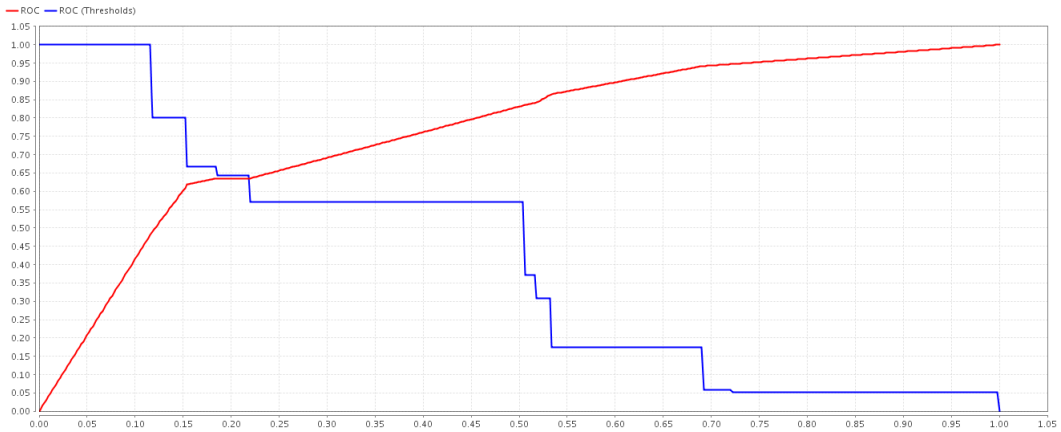
Bootstrap sa hodí na aplikáciu tam, kde máme malú sadu vstupných dát alebo v prípadoch, že štatistické rozloženie vstupných vzoriek je veľmi zložité. Jej menšou nevýhodou je však to, že dáva viac-menej optimistické výsledky.

2.3.2 Grafické zobrazenie kvality

Za účelom porovnávania modelov rôznych algoritmov nepoužívame len numerické atribúty uvedené vyššie v rámci tejto kapitoly, ale osvedčilo sa aj grafické porovnávanie, ktoré poskytuje jednoduchý a jasný obrazový prehľad. Najpoužívanejšie zobrazenia sú pomocou ROC kriviek alebo tzv. Lift grafov.

ROC krivky

Skratka ROC (z ang. Receiver Operating Characteristic, voľne preložiteľné ako „operačná charakteristika prijímača“) je grafické vyjadrenie výkonu klasifikátoru. Vyjadruje závislosť medzi vyššie definovanou hodnotou senzitivity (vzťah 2.6) a po-



Obr. 2.5: ROC krivka (označená červenou farbou)

merom negatívnych vzoriek nesprávne označených ako pozitívne (false-positive). Takéto zobrazenie nám umožňuje vizuálne porovnať závislosť medzi správnym vyhodnocovaním pozitívnych vzoriek voči omylnosti pri pozitívnom označení negatívnej vzorky. Využitie má aj obsah plochy pod krivkou, ktorý je mierou presnosti modelu (1 = ideálny model, 0,5= náhodné hádanie).

Aby sme boli schopní ROC krivku vykresliť, náš model by mal byť schopný určiť pravdepodobnosť úspešnosti predpovede pre každú vzorku. Tieto pravdepodobnosti následne zoradíme zostupne a postupujeme od najnižšej hodnoty k vyšším. Očakávame, že pod danou hodnotou pravdepodobnosti sú všetky hodnoty negatívne a nad ňou pozitívne. Na základe toho určíme hodnoty TruePos, FalsePos, TrueNeg a FalseNeg, vypočítame senzitivitu a pomer nesprávne označených negatívnych vzoriek ku všetkým negatívnym [4]. Tieto hodnoty stačí vyniesť do grafu (senzitivita spravidla na os y) a ROC krivku máme hotovú.

Lift grafy

Lift grafy sú mierou efektívnosti učiacich sa algoritmov voči náhodnému výberu. Vychádzajú z ROC kriviek a vedia prehľadne zobraziť, nakoľko je použitie algoritmu v danej fáze výhodnejšie ako hádanie naslepo. Hodia sa hlavne v momente, keď máme v záujme vyberať iba vzorky jednej triedy a o ostatné sa nezaujímame (napríklad pri marketingovom oslovovaní najprv oslovíme tých užívateľov, u ktorých máme väčšiu pravdepodobnosť pozitívnej odpovede). Do grafu vynášame pomer medzi výsledkami predpovedanými strojovým učením a výsledkami bez jeho použitia. Príklady a ďalšie využitie možno nájsť v [3].

3 IMPLEMENTÁCIA LOKÁLNYCH PRÍZNAKOV

Táto časť práce je zameraná na samotný výber a výpočet príznakov a ich implementáciu do prostredia Rapidminer. Na ďalších stranách je zhrnutý použitý matematický aparát a tiež kód a prostredie, v ktorom je využívaný.

3.1 Prostredie Rapidminer

Rapidminer je prostredie využívajúce jazyk JAVA, ktoré poskytuje procedúry, triedy a metódy pre využitie na data mininig (zber údajov), strojové učenie a rôzne druhy analýz. Pre tieto účely obsahuje efektívne vstupné procesy, prispôsobené grafické užívateľské rozhranie (GUI) a je tiež ľahko rozšíriteľný o nové funkcie, užívateľom definovateľné rozšírenia a operátory [10], alebo aj moduly (plugin).

Štruktúra procesov v prostredí Rapidminer je definovaná pomocou kódu XML a je reprezentovaná prehľadným grafickým rozhraním, ktoré umožňuje plynulú náväznosť a vetvenie procesov podľa potreby (tzv. process graph). Dôvodom pre jeho použitie v tejto práci je hlavne ľahké prepojenie s kódom a triedami definovanými v prostredí Eclipse a dobré vlastnosti pri výpočte množstva lokálnych príznakov (export do tabuľky, preskakovanie neznámych hodnôt, možnosť grafického spracovania, úprava vstupných parametrov a i.). Prostredie tiež ponúka dobré možnosti škálovatelnosti - pri ďalšej analýze následne iba stačí dodefinovať ďalší operátor podľa potreby a prepojiť výstupy medzi sebou v reťazci procesov, prípadne ho vložiť ako podproces do už vytvoreného bloku operátorov, ktorý je možné ľahko reprodukovať ďalej a vytvoriť tak modulárny systém podobný stavebnici. Dostávame teda vhodnú kombináciu efektivity a grafickej jednoduchosti, ktorú (v konečnej podobe) zvládne obsluhovať aj menej programovo zdatný užívateľ. Podrobnosti a ďalšie funkcie sú prehľadne uvedené v [11].

Princíp na ktorom Rapidminer pracuje je v základe relatívne jednoduchý. Každý vstupný objekt je definovaný nejakým počtom atribútov (textové, číselné, logické a iné), čím dostávame vstupnú dátovú sadu. Príprava takýchto dátových setov (úprava vstupných dát do formátu spracovateľného počítačom) je často krát najnáročnejšou fázou celého procesu data miningu. V našom prípade našťastie ako dátová sada slúžia samotné pixely, ktoré sú popísané svojou polohou a hodnotou. Tieto údaje následne využijeme pre modelovanie, resp. predikciu výsledných hodnôt - v tomto prípade zdravotného stavu pacienta na základe nálezu nežiadúcich fragmentov pomocou výpočtu hodnôt lokálnych príznakov.

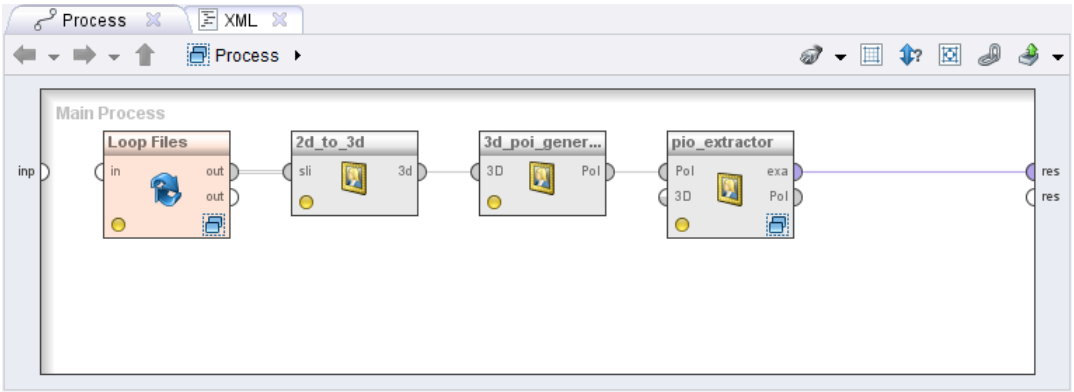
Ďalšími užitočnými vlastnosťami Rapidmineru (z programátorského hľadiska) sú podobne ako v Eclipse kontrola syntaxe a ponuka rýchlych opráv (quick fix), alebo možnosť vkladania tzv. breakpoint - označenie miesta na zastavenie procesu a následná možnosť krokovania za účelom hľadania chýb v kóde.

Samotné prostredie sa integruje do programu Eclipse importovaním ako nový projekt z online repozitára. Po úspešnom nakopírovaní je pre správnu funkciu integrácie jednotlivých tried vytvorených v Eclipse do Rapidminer nevyhnutné zabezpečiť, aby projekt využíval inštalované knižnice Java JDK (a nie JRE), pretože obsahujú potrebné vývojárske nástroje. Vytvorenie nadstavby prebieha pomocou súborov build.xml (súbory budujúce rozšírenia pre Rapidminer) obsiahnutých v každej triede, ktorú chceme v novom prostredí využívať. Tieto súbory sú následne nástrojom Ant Build / install prevedené na odpovedajúcu súčasť programu Rapidminer, ktorý nám ostáva iba spustiť prostredníctvom jeho triedy `RapidminerGUI`.

3.2 Voľba a matematický výpočet príznakov

Ako už bolo spomenuté v sekcii 1.2.1, vhodným spôsobom na identifikáciu objektov v 2D alebo 3D obrazoch je využitie lokálnych príznakov. V tejto práci bol vzhľadom na doterajšiu prácu skupiny SPL zvolený postup výpočtu viacerých príznakov pre oblasť premennej veľkosti. Postup, na ktorý nadväzujem a doteraz vytvorený sled operátorov sa dá popísať nasledovne (viď obr. 3.1):

načítanie obrázkov (rezov) zo súboru → prevod na 3D objekt → vygenerovanie bodov → výpočet



Obr. 3.1: Sled procesov v prostredí Rapidminer

Prehľadávanie súboru a načítanie obrázkov prebieha pomocou operátoru „Loop files“, ktorý sa v Rapidminer-i nachádza už v sade základných procesov. Na svoj výstup odovzdáva jednotlivé rezy, ktoré sa ďalej spracovávajú v rozšírení „2d_to_3d“

(spolu s nasledujúcim operátorom definované už pred začiatkom mojej spolupráce na projekte) na výstupný 3D objekt triedy `ThreeDimIOObject`. Týmto zo sady cca. 255 rezov vytvoríme iba jeden objekt, s ktorým sa v ďalšom reťazci procesov narába oveľa jednoduchšie. Vytvorený objekt je vstupom pre proces, ktorý vytvorí končený počet (stanovený užívateľom) počiatočných bodov vo viacerých snímkach buď náhodne (roztrúsene) alebo v štruktúre mriežky. Tieto body budú následne slúžiť ako stred oblasti, v ktorej sa vykoná výpočet. Jej rozmery budú môcť byť buď 2D - užívateľovi sa poskytne možnosť voľby jednej hodnoty, ktorá sa použije pre súradnice x a y, alebo bude možné nastaviť variantu prehľadávania zadaného (nenulového) počtu susedných rezov (počet bude nezávislý na súradniciach x a y - nie je nevyhnutné aby bol prehľadávaný snímok v smere osi z v rovnakej vzdialenosti ako x a y), čím dosiahneme analýzu oblasti v trojrozmernom priestore.

Nasleduje blok pre výpočet hodnôt vlastných samotným oblastiam. Konkrétne príznaky, ktoré sa budú počítať, boli vybrané na základe inšpirácie z [8]. Zatiaľ sú nimi energia, entropia a kontrast, avšak nie je problém pridať v budúcnosti ďalšie a v už vytvorenom operátore iba doprogramovať vzorec pre ich výpočet. Ako je možno vidieť z výsledkov citovanej práce, práve použitie týchto príznakov je vhodné na identifikáciu a rozlišovanie hraníc medzi rôznymi objektami na snímkach MRI.

Matematický aparát pre výpočet parametrov danej oblasti je nasledovný [8]:

$$Energia = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P(i,j)^2 \tag{3.1}$$

$$Kontrast = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i-j)^2 P(i,j) \tag{3.2}$$

$$Entropia = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P(i,j) \log P(i,j) \tag{3.3}$$

kde $P(i,j)$ je hodnota pravdepodobnostnej funkcie pre zhodu farebnej hodnoty i prvého pixelu s hodnotou porovnávaného pixelu j. N_g pritom udáva použitý počet farieb, teda aj maximálnu možnú hodnotu pixelu.

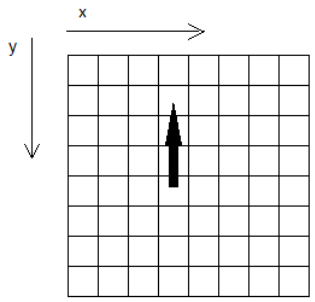
3.3 Vytvorenie nového operátoru

V našom operátore budeme hľadať príbuznosť dvoch pixelov v danom smere a vzdialenosti. Ich vzájomná poloha je teda jednoznačne určená vektorom, ktorého maximálna dĺžka je obmedzená aktuálnou veľkosťou prehľadávanej oblasti a uhol jeho natočenia je závislý na voľbe vyhľadávania asociácii iba v jednom alebo viacerých rezoch. Pre 2D teda môže tento uhol nadobúdať hodnoty 0°, 45°, 90° alebo 135°,

pričom pri 3D zobrazeniach je táto množina rozšírená o 180°, 225°, 270° a 315°. Toto je z dôvodu, že v 2D zobrazení by sme pri rozšírenej množine testovali aj tak tie isté dvojice.

Zo vzorcov uvedených v minulej sekcii vyplýva, že je potrebné vypočítať hodnotu pravdepodobnostnej funkcie pre každú farebnú (presnejšie odtieňovú) kombináciu a následne použiť funkciu suma. Vzhľadom na podobnosť výpočtu vybraných príznakov (použitie sumácie všetkých kombinácii) je z pohľadu rýchlosti výpočtu vhodné nepočítať hodnoty pravedpodobnostnej funkcie 256^2-krát (pre každú kombináciu odtieňov) pri výpočte každého príznaku zvlášť. Lepšie riešenie je vytvorenie matice 256x256 hodnôt, ktoré je síce zaberie malé množstvo pamäte navyše, ale zbavíme sa nutnosti opakovať množstvo operácii a zrýchlime tak celý proces. Takéto zrýchlenie sa prejaví hlavne pri vyššom počte počítaných príznakov, keďže v každej ďalšej oblasti začína výpočet odznova a pri tomto riešení sa napríklad 3 príznaky vypočítajú iba v jednom cykle.

Pre výpočet hodnôt pravdepodobnostnej funkcie bol použitý vzorec obsiahnutý v [8] a vytvorená samostatná funkcia `ComputeProbability` so štruktúrou popísanou nižšie. Základným princípom funkcie je prehľadanie všetkých pixelov v oblasti a ich porovnanie s pixelom, ktorý je vzdialený o definovaný vektor prepočítaný na súradnicový rozdiel *dx*, *dy* a *dz*. Parameter *dz* dostávame priamo ako vstup od užívateľa (kde definujeme ako ďaleko od súčasného snímku chceme prehľadávať), *dx* a *dy* dostaneme jednoduchým prepočtom podľa zadanej veľkosti vektora a jeho uhlu (viď obr. 3.2).



Obr. 3.2: Ukážka vektoru o veľkosti 3 a uhle $\Theta = 90°$

Keďže v tomto prípade sa súradnica *x* vôbec nemení, *dx* ostáva nulové a *dy* sa nastaví podľa prepínača switch (výstup je krátený):

```
switch (angle) {
 case 90:
  deltaY=-(getParameterAsInt(DISTANCE));
  break;
```

Samotná trieda `ComputeProbability` dostáva na svoj vstup vytvorenú 3D sadu rezov a počiatočný bod `point` vygenerované v predchádzajúcich procesoch (ako je podrobnejšie popísané v predchádzajúcej sekcii 3.2) a užívateľom zadané hodnoty veľkosti oblasti a smerového vektoru pre porovnávanie. Parametre i a j sú hodnotami pixelov, ktoré porovnávame, t.j. snažíme sa zistiť aká je pravdepodobnosť toho, že ak jeden voxel má hodnotu i, potom voxel vzdialený o definovaný vektor má hodnotu j.

Celý proces výpočtu hodnoty pravepodobnostnej funkcie musí byť ošetrený proti pretečeniu definovanej oblasti, aby sa zamedzilo porovnávaniu dvojíc voxelov, v ktorých sa jeden nachádza za hranicami. Je treba správne definovať cykly, hlavne ich začiatok a koniec, čo je docielené dvojicou podmienok if/else. Na základe týchto podmienok sa vykoná cyklus so správnou syntaxou. Ako príklad je uvedený cyklus pre prípad, že obe hodnoty dx a dy sú kladné.

```
public double ComputeProbability
(ThreeDimIOObject image, Voxel point, int area, int dx,
int dy, int dz, int i, int j)
{
        double P=0;
        double R=(2*area+1-Math.abs(dx))*
         (2*area+1-Math.abs(dy))*(2*area+1-Math.abs(dz));

if (dx>0) {
 if (dy>0) {
  for (int Z = point.getZ()-area;
   Z < point.getZ()+area-dz; Z++) {
    for (int X = point.getX()-area;
     X < point.getX()+area-dx; X++) {
     for (int Y = point.getY()-area;
      Y < point.getY()+area-dy; Y++) {
       if (image.getVoxel(X, Y, Z)==i &&
        image.getVoxel(X+dx, Y+dy, Z+dz)==j) {
         P++;
     }
    }
   }
  }
<výstup krátený>

return P*(1/R);
```

Princípom teda je spočítanie všetkých zhodných voxelov a ich podelenie celkovým množstvom voxelov R v oblasti, čím dostaneme požadovaný výsledok.

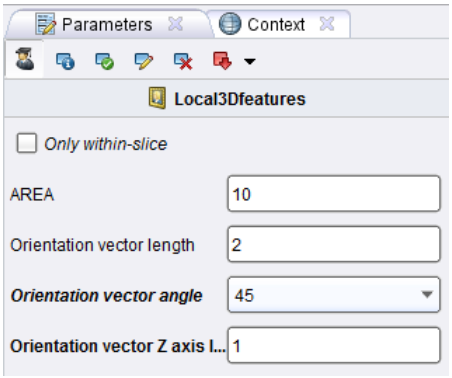
Ako už bolo spomenuté vyššie, všetky zvolené príznaky používajú hodnotu pravepodobnostnej funkcie, preto je možné začať počítať sumy podľa uvedených vzorcov v tom istom cykle, v ktorom počítame samotnú pravdepodobnosť pre každé dve možné farebné kombinácie dvoch voxelov vzdialených od seba o danú vzdialenosť. Hodnoty funkcie sa poukladajú do matice pre jednoduchšie ďalšie použitie. Teoreticky by bolo možné hodnoty uložiť aj do jedinej premennej, ktorú budeme stále prepisovať, ale tým by sme sa pripravili o prípadné ďalšie využitie týchto hodnôt mimo cyklu.

Je teda treba iba previesť vzorce do kódu, ako napríklad nasledovne:

```
for (int i = 0; i < 255; i++) {
 for (int j = 0; j < 255; j++) {
  Dmatrix[i][j]=ComputeProbability(image, point, Area,
   deltaX, deltaY, deltaZ, i, j);
  Energy+=(Dmatrix[i][j]*Dmatrix[i][j]);
   }
  }
 features.setFeature(getAttributeName(image) + "_" +
  "Energy", Energy );
}
```

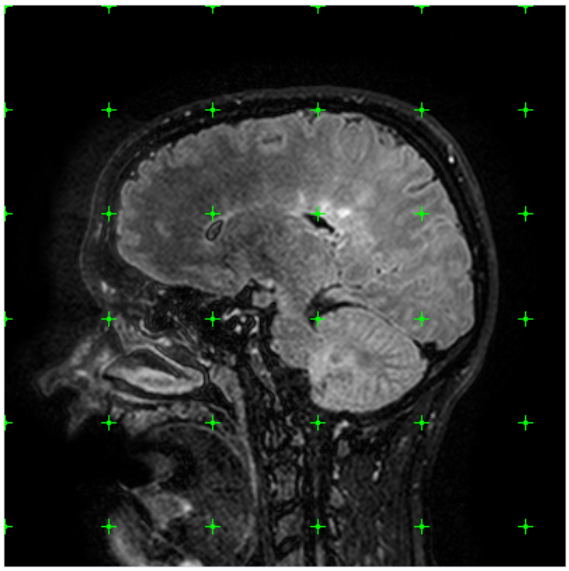
3.4 Výsledky

Ná základe všetkých vyššie uvedených poznatkov a vytvorených procesov bol v prostredí Rapidminer odskúšaný výpočet s parametrami definovanými nasledovne:



Obr. 3.3: Parametre testovacieho procesu

Pre lepšie vyhodnotenie výsledkov bol do procesu vložený breakpoint, konkrétne za operátor *3d_poi_generator*. Budeme tak mať prehľad o rozložení generovaných bodov, čo umožní rozpoznať očakávané chyby (pretečenie v rôznych smeroch) od tých nezamýšľaných. Ako môžeme vidieť na obr. 3.4, štruktúra mriežky v jednom snímku je vytvorená z 36 bodov, pričom prvý riadok a stĺpec sú relatívne blízko okraja snímku, takže môžeme očakávať chybu (program pri tomto pretečení vracia hodnotu príznaku -99999).



Obr. 3.4: Generované body v jednom z rezov

Z tabuľky súradníc, ktorú si v Rapidminer môžeme vygenerovať, je tiež zrejmé, že body sú pravidelne generované v snímkach 1, 75, 149 a 224. Keďže sa teda nachádzajú už v prvej snímke, zaručene tým dostaneme pretečenie v smere osi z vo všetkých prípadoch 3D analýzy, takže všetkých 36 bodov tohoto rezu bude mať vo výsledku nastavenú hodnotu pre tento prípad, ktorá je -88888. V konečnej tabuľke je možné si overiť, že sa tu nachádza presne 36 bodov s hodnotou -88888 a $11 \times 3 = 33$ bodov s hodnotou -99999 (v každom snímku sa nachádza 11 bodov na okraji a generátor vytvára body v dokopy 4 snímkach, avšak prvý snímok už nespĺňa podmienku pre os z).

Samotný výpočet parametrov jednotlivých oblastí je už časovo náročnejší. Kým načítanie rezov zo súboru spolu s ich transformáciou do 3D objektu a generovaním bodov netrvá dlhšie ako 5 sekúnd, vypočet operátoru `pio_extractor`, ktorý obsahuje vytvorený operátor `Local3Dfeatures`, zaberie ďalších takmer 5 minút (celý proces z obr. 3.5 bol vypočítaný presne za 4m 58s).

Toto je spôsobené hlavne množstvom rozsiahlejších cyklov v operátore (napr. 3 vnorené cykly v metóde `ComputeProbabilty` pre každú farebnú kombináciu pre

každú oblasť definovanú generovaným bodom v predchádzajúcom procese). Časová náročnosť samozrejme stúpa s veľkosťou oblasti, keďže sa tým zvyšuje počet dvojíc bodov, ktoré treba porovnávať.

Samotné výsledky tejto práce (hodnoty energie, entropie a kontrastu pre oblasti so stredom v generovaných bodoch) pre vstupné parametre uvedené na obr 3.3 možno nájsť na nasledujúcom obrázku. Tabuľka je usporiadaná podľa hodnôt prvého stĺpca (kontrastu), aby boli vidieť hlavne relevantné hodnoty.

Result Overview | ExampleSet (pio_extractor)

Data View | Meta Data View | Plot View | Advanced Charts | Annotations

ExampleSet (143 examples, 1 special attribute, 3 regular attributes)

Row No.	label	2d_to_3d_Local3Dfeatures_Contr...	2d_to_3d_Local3Dfeatures_Energy	2d_to_3d_Local3Dfeatures_Enthropy
81	0	0.001	0.742	-0.058
51	0	0.008	0.725	-0.085
137	0	0.018	0.699	-0.126
48	0	0.020	0.695	-0.123
93	0	0.068	0.633	-0.205
109	0	0.092	0.651	-0.202
120	0	0.129	0.421	-0.435
12	0	0.210	0.418	-0.498
84	0	0.251	0.338	-0.635
58	0	0.474	0.259	-0.707
27	0	0.511	0.452	-0.515
13	0	0.720	0.120	-1.033
89	0	0.761	0.089	-1.031
49	0	0.909	0.495	-0.485
127	0	1.167	0.280	-0.834
67	0	1.506	0.124	-1.275
76	0	2.196	0.099	-1.268
38	0	2.853	0.041	-1.445
72	0	4.074	0.018	-1.599

Obr. 3.5: Generovaná tabuľka s výslednými hodnotami

4 TESTOVANIE

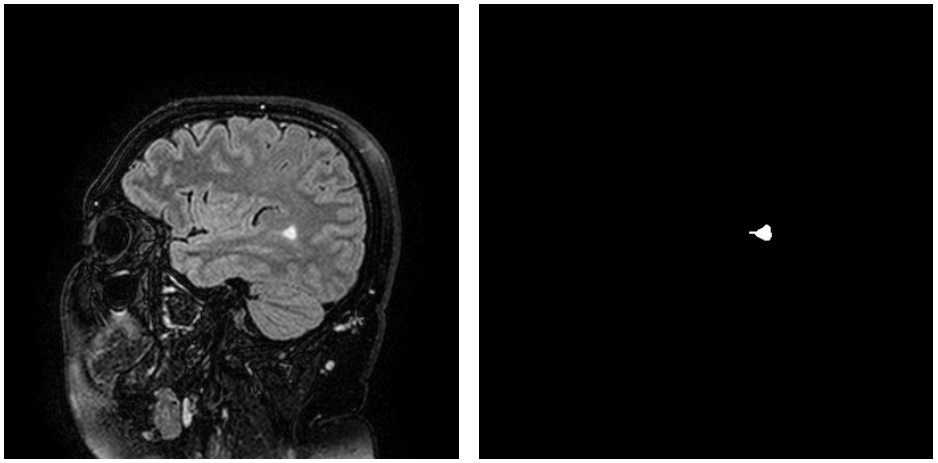
V tejto kapitole je bližšie priblížená praktická časť práce zameraná na testovanie rôznych druhov algoritmov (metódy určovania kvality aj samotné algoritmy boli popísané v kapitole 2) pri spracovávaní dát vytvorených na základe príznakov počítaných spôsobom uvedeným v sekcii 3.3. Generácia trénovacích a testovacích sád a z toho vyplývajúce úpravy procesu a následné použitie týchto sád na určenie presnosti je spolu s výsledkami obsahom nasledujúcich strán.

4.1 Generácia tréningových dát

Ako bolo spomenuté v kapitole 2, pri tomto type strojového učenia je dôležité pracovať s označenými dátami, ktoré by mali mať dostatočné zastúpenie pre správne natrénovanie modelu. V nasledujúcich sekciách je popísaná implementácia týchto podmienok v praktickej časti práce.

4.1.1 Značenie pravej kategórie

Aby sme mohli využiť metódy kontrolovaného učenia a boli schopní posúdiť presnosť klasifikačného algoritmu, potrebujeme poznať skutočnú kategóriu, do ktorej vstupné dáta naozaj patria, a tieto patričným spôsobom označiť. V tejto práci sú všetky vstupné snímky doplnené jednoduchým čierno-bielym snímkom, tzv. maskou, v ktorej lekár - špecialista bielou farbou označil oblasti, ktoré vykazujú patogénne javy a v zdravom tkanive by sa nemali nachádzať. Príklad takejto dvojice snímkov je na obr. 4.1.



(a) Snímka MRI (b) Maska

Obr. 4.1: Snímka MRI so zodpovedajúcou maskou

V prostredí Rapidminer je toto označenie reprezentované príznakom *label*. Tento bude nadobúdať hodnotu 0 v prípade nepravdy (*false*) a 255 v prípade pravdy (*true*). Tieto dve hodnoty svojou logickou hodnotou vyjadrujú, či daný bod leží v oblasti označenej špecialistom. Ak áno, je mu priradené označenie true, v opačnom prípade false. Keďže label má v Rapidminer-i špecifické postavenie a patrí medzi tzv. špeciálne atribúty, je danému bodu priradený hneď pri jeho výbere medzi dátovú sadu v operátore `3d_poi_generator`, čo bolo dosiahnuté pridaním nasledovnej podmienky do kódu operátoru:

```
for (double z = startZ; z <= maxZ; z += step) {
 for (double x =startX; x < maxX; x += step) {
  for (double y = startY; y < maxY; y += step) {
   //255=TRUE (white color in mask)
   if (mask.getVoxel((int)x, (int)y, (int)z)==255) {
    points.addPoint((int)x, (int)y, (int)z, 255);
   } else {
   //0=FALSE
    points.addPoint((int)x, (int)y, (int)z, 0);
   }
  } } }
```

4.1.2 Zabezpečenie rovnomernejšieho rozloženia

Kvôli správnej funkcii klasifikačného algoritmu je tiež dôležité zapezpečiť rovnomerný počet vzoriek jednej aj druhej kategórie. Pokiaľ je vzoriek z jednej ketegórie oveľa menej, môže to mať dôsledky v podobe nepresnosti algoritmu (ktorý napríklad zaradí všetky vzorky do jednej triedy, čo bolo vidieť v pokusoch na tejto práci, alebo vzorky z menšinovej triedy budú mať výrazný podiel zlej kategorizácie), keďže program nemá dostatok údajov na presné natrénovanie. Vzhľadom na to, že pri náhodnom výbere bodov v sade snímok rôznych pacientov je štatisticky prevažná väčšina bodov (až 90%) negatívnych (príznaky nervových alebo nádorových ochorení spravidla zaberajú iba minimum objemu mozgu), bolo pri generácii bodov nutné programu pomôcť zvýšením počtu bodov s pozitívnym výsledkom. Toto bolo dosiahnuté dvoma spôsobmi popísanými nižšie - vytvorením nového operátora a pomocou zvonku zásahom človeka (čiže v podstate istá forma aktívneho strojového učenia).

Nový operátor nesie názov `TruePointGenerator` a ako názov napovedá, jeho úlohou je generácia čo najväčšieho počtu bodov s príznakom *true*. Princíp činnosti je veľmi podobný predchádzajúcemu operátoru `3d_poi_generator`. Na svojom vstupe operátor obdrží sadu bodov vygenerovaných predchádzajúcim operátorom, ktorý ich na základe hodnoty masky označil v príznaku *label* . Množina týchto bodov je jeden

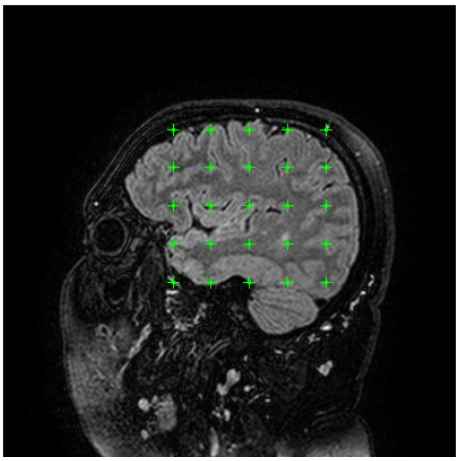
po druhom znovu testovaná, s dôrazom na hodnotu 255, t.j. *true*. Keď aktuálne testovaný bod podmienke vyhovie, spustí sa prehľadávanie vopred zadaného okolia tohoto bodu (veľkosť tejto oblasti je určená parametrom *SIZE*), keďže predpokladáme, že maska nie je tvorená samostatne roztrúsenými bodmi, ale ich zhlukmi. Všetky body z uvedeného okolia sú znovu testované na hodnotu masky (ktorú operátor obdrží druhým vstupným portom) a pridávané do samostatného poľa typu HashSet. Toto pole sa na konci celého prehľadávania zlúči so vstupnou sadou bodov.

Uvedené riešenie muselo byť zvolené z toho dôvodu, že polia typu HashSet (akým je aj pole v objekte `Points3DIOObject`, ktorý prenáša vstupnú sadu bodov) sú citlivé na manipuláciu s ich dátami (hlavne pridávanie alebo odoberanie prvkov poľa), zatiaľ čo je s nimi nejak inak narábané - napríklad sú súčasťou cyklu ktorý ich prehľadáva ako v tomto prípade. Pri programovaní a testovaní praktickej časti práce bolo toto správanie a problémy s indexáciou možné pozorovať na vlastné oči. Akýkoľvek pokus o pridanie prvku teda končí chybovou hláškou a pádom programu. Ďalšou úvahou je, že aj v prípade neprítomnosti tohoto javu by sme sa pridávaním na koniec sady, ktorú práve testujeme, mohli vystaviť nežiadúcemu zacykleniu programu.

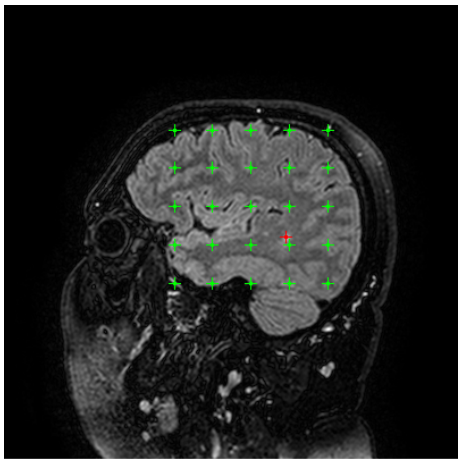
Objekt `Points3DIOObject` bol teda rozšírený o metódu `addCollection`, ktorá zabezpečuje zlúčenie nového a pôvodného poľa. Táto sa zavolá vždy po úplnom dokončení pridávania pravdivých bodov, ako je to zobrazené v nasledujúcom výňatku z kódu:

```
for (Voxel point : points) {
 if (point.getLabel()==255) {
  for (double x = point.getX() - getParameterAsInt(SIZE);
  x <= point.getX() + getParameterAsInt(SIZE); x++) {
   if (x<0 | x>mask3D.getData(ThreeDimIOObject.class).getHeight()) {
   continue;
   }
  for (double y = point.getY() - getParameterAsInt(SIZE);
  y <= point.getY() + getParameterAsInt(SIZE); y++) {
   if (y<0 | y>mask3D.getData(ThreeDimIOObject.class).getWidth()) {
   continue;
   }
   if (mask3D.getData(ThreeDimIOObject.class).getVoxel((int) x,
   (int) y, (int) point.getZ()) == 255) {
    setExtension.add(new Voxel((int)x, (int)y, point.getZ(), 255));
    } else {
    setExtension.add(new Voxel((int)x, (int)y, point.getZ(), 0));
    }
   }  }  }      }
points.addCollection(setExtension);
setExtension.clear();
poiOut.deliver(points);
```

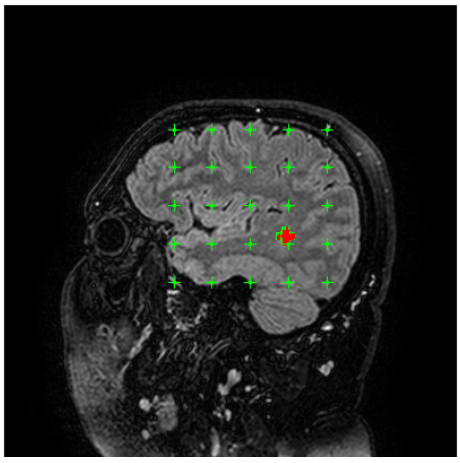
Forma ľudského zásahu je v podobe manuálneho pridania niekoľkých bodov ležiacich v oblasti označenej maskou. K tomuto úkonu bolo nutné pristúpiť z toho dôvodu, že počítač pri náhodnej generácii bodov v rámci celého 3D obrazu takpovediac triafa do neznáma a aj pri bližšom definovaní súradníc v oblasti hľadania by bolo nutné vygenerovať veľké množstvo bodov, aby „trafil“. Štatisticky by pri tom stále trval obrovský nepomer medzi bodmi pravdivými a nepravdivými, a ako bolo už spomenuté vyššie, toto by mohlo spôsobovať nepresnosti v učiacom algoritme. Preto bol za operátor `3d_poi_generator` vložený breakpoint a do snímok boli manuálne vložené ďalšie body „navyše“ označené príznakom *true* do miest, kde je očakávaný výskyt masky. Týchto bodov nemusí byť veľa, keďže zvyšok práce dokončí vyššie popísaný operátor `TruePointGenerator`. Celý postup možno vidieť na obr 4.2.



(a) Body vygenerované pomocou `3d_poi_generator`



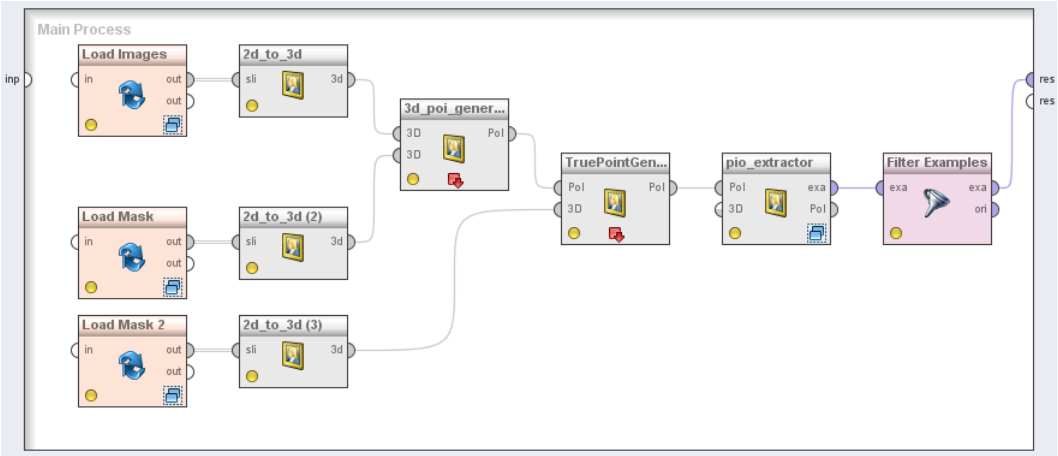
(b) Bod s príznakom true pridaný manuálne



(c) Body generované operátorom

Obr. 4.2: Postup generácie ďalších „true“ bodov

Na konci celého procesu generácie bodov bol ešte vložený operátor `Filter Examples`, ktorý má za úlohu odstrániť dáta, ktoré obsahujú chýbajúce hodnoty. Informačné hodnoty -99999 a -88888 nakonfigurované kvôli možnosti vizuálnej kontroly v sekcii 3.4 boli nahradené hodnotou *NaN* (not a number), ktoré filtračný operátor interpretuje ako chýbajúce dáta a z dátovej sady ich vyradí. Takto na výstupe dostávame sadu úplných dát. Výsledný sled procesov použitý na ich generáciu je zobrazený na obr 4.3.



Obr. 4.3: Generácia dát v Rapidminer

4.2 Testovanie algoritmov

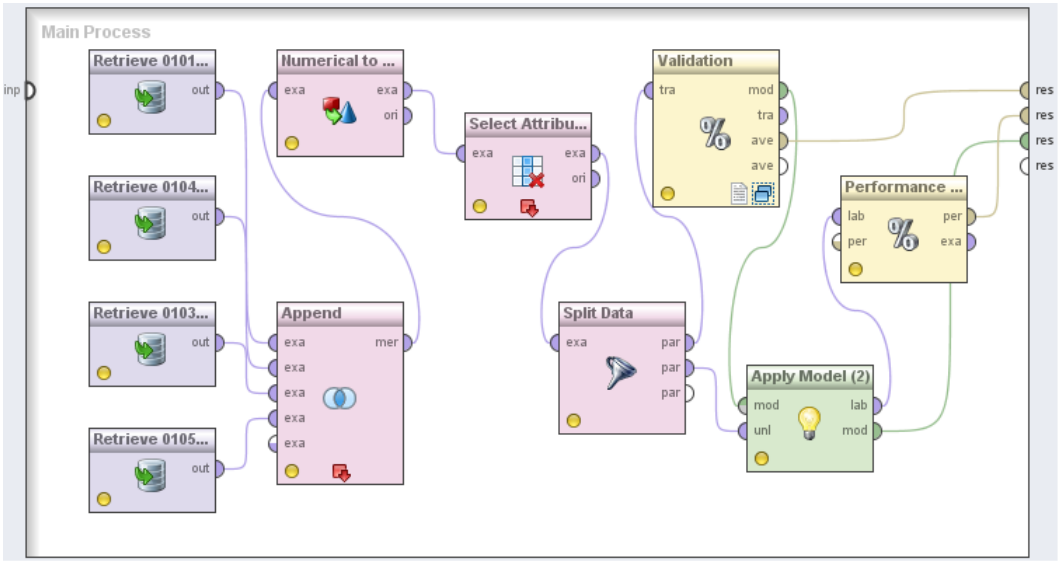
Spôsobom uvedeným v predchádzajúcej sekcii boli vygenerované tréningové dáta zo všetkých desiatich poskytnutých dátových sád - MRI rezov od rôznych pacientov spolu s maskami nakreslenými lekármi-špecialistami. Dáta boli generované s parametrom $AREA = 4$ a pre transparentnosť ukladané zvlášť podľa príslušnosti k sade rezov a zlučované do jednej veľkej sady operátorom `Append`.

Dôležité je tiež konvertovať hodnotu príznaku *label* z numerickej na logickú (binominálnu). Keďže sme logicky tieto hodnoty roztriedili už pri generácii jednotlivých bodov priradením hodnôt 0 pre *false* a 255 pre *true*, nie je to komplikovaná úloha. Prostredie Rapidminer pre takýto úkon ponúka operátor `Numerical to Binominal`, v ktorom stačí nastaviť interval numerickej hodnoty (v našom prípade napríklad 0 - 128), ktorá sa namapuje ako *false*. Všetky ostatné hodnoty budú označené zostávajúcou kategóriou, ktorou je *true*. Správnosť tohoto priradenia môžeme skontrolovať pomocou príznaku *2d_to_3d_Local3Dfeatures_mask_label*, ktorý nie je ničím iným ako manuálne vloženým duplikátom príznaku *label* určeným pre kontrolu tohoto priradenia. Tento duplikát následne odstránime operátorom `Select Attributes`,

v ktorom použijeme možnosť inverzného výberu - na svoj výstup odovzdá všetky hodnoty atribútov okrem *mask_label*.

Takto získané dáta môžeme rozdeliť operátorom `Split Data` podľa vopred zadaného pomeru a metódy (ktoré sú popísané v sekcii 2.3.1) na tréningové a validačné, alebo môžu byť priamym vstupom do operátoru `Validation`, ktorý pozostáva z dvoch častí - tréningovej a testovacej, a rovnako ovláda metódy delenia dát spomenuté vyššie. Konkrétna variácia tohoto operátoru použitá v tejto práci je cross-validation s počtom opakovaní $k = 10$. V rámci tréningovej časti môžeme vo vnútri operátoru meniť a testovať rôzne trénovacie algoritmy popísané v kapitole 2. Tiež máme možnosť meniť parametre týchto operátorov a sledovať vplyv na výsledok (napríklad prejav zmeny hodnoty *minimal gain* na hĺbku a rozvetvenie rozhodovacieho stromu).

V prípade rozdelenia dát na tréningové a validačné použijeme model vytvorený trénovacím algoritmom vo vnútri operátoru `Validation` a rovnakým spôsobom ho aplikujeme na úplne nezávislé validačné dáta (využitím operátoru `Apply Model`), čím dostaneme reálnejší a menej skreslený výsledok (maticu zámien vytvorí operátor `Performance`). Celý sled procesov je prehľadne zobrazený na obr 4.4.



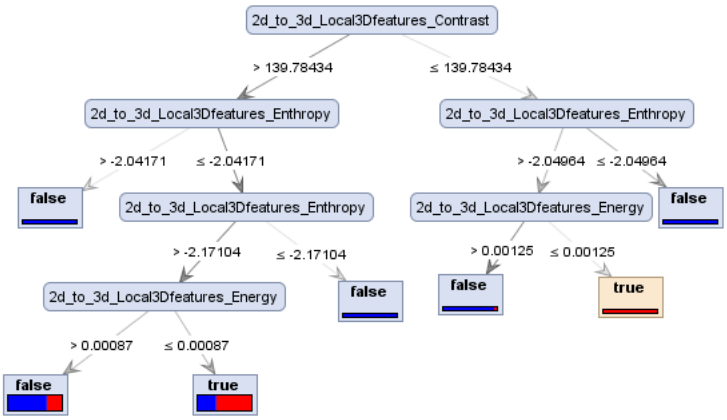
Obr. 4.4: Usporiadanie operátorov pri testovaní

Namiesto použitia operátoru `Split Data` môžeme zvoliť aj iný prístup a dáta si vopred rozdeliť. Keďže predpoklad je, že budeme mať vygenerované veľké množstvo bodov, môžeme to urobiť bez obavy z nedostatočného množstva trénovacích vzoriek. Keďže k dispozícii je 10 vstupných sád, 5 z nich použijeme na trénovanie a testovanie a zvyšné na validáciu modelu. V tomto prípade teda vynecháme spomenutý operátor a nahradíme ho ďalším operátorom `Append` (spolu s nadväzujúcou konverziou na

binominálne hodnoty a odstránením duplikátneho príznaku), ktorý bude dáta z 5 validačných sád združovať do jednej. Tento postup bol uplatnený aj pri testovaní rôznych algoritmov v nasledujúcich častiach práce.

4.2.1 Úspešnosť rozhodovacích stromov

Použitie tohoto typu trénovacieho algoritmu v praxi potvrdilo jeho rýchlosť, keď aj pri zmenách parametrov a veľkej hĺbke stromov bol výsledok vypočítaný v priebehu jednotiek sekúnd. Pri vytváraní modelu s použitím rozhodovacích stromov sa však začali objavovať problémy s nedostatnočným rozvetvením a hĺbkou, kedy sa pri východzích parametroch vytvoril strom iba s jedným listom a všetky vzorky boli označené ako false. Keďže vetvenie stromu závisí hlavne od parametru *minimum gain* (určuje minimálnu hodnotu zisku, ktorú musí uzol dosiahnuť, aby sa rozvetvil), bola táto hodnota upravená (znížená na 0,001) tak, aby vetvenie stromu prebehlo viacnásobne, pričom hĺbka stromu zostala obmedzená na 5 úrovní. Kritérium pre delenie bolo zvolené na *gain_ratio*, keďže dávalo najlepšie výsledky. Všetky ostatné parametre zostali na predvolených hodnotách. Výsledný strom po spustení tohoto procesu možno vidieť na obr. 4.5.



Obr. 4.5: Výsledný rozhodovací strom

Môžeme vidieť, že väčšia časť vzoriek spadá do ľavej časti stromu, kde nakoniec končí klasifikovaná s nemalou chybou. Ako bolo spomenuté vyššie, boli testované rôzne alternatívy rozhodovacieho stromu s úpravou niektorých parametrov, avšak ani v jednom prípade neboli obdržané lepšie výsledky. Bola napríklad odskúšaná variácia s dvoj až trojnásobonou hĺbkou stromu, čo však iba pridalo na neprehľadnosti a výsledok to viacej zhoršilo ako zlepšilo. Modifikáciou minimálneho zisku bolo ovyplyvňované vetvenie stromu. Všetky spomínané možnosti boli tiež overené

s rôznym nastavením rozdeľovacieho kritéria, kde *gain_ratio* dávalo najlepší výsledok a *accuracy* paradoxne najhorší (všetky vzorky v jednej triede bez ohľadu na veľkosť zisku). Najlepšie dosiahnuté výsledky boli práve pre vyššie uvedený strom, ich matica zámien je na obr. 4.6.

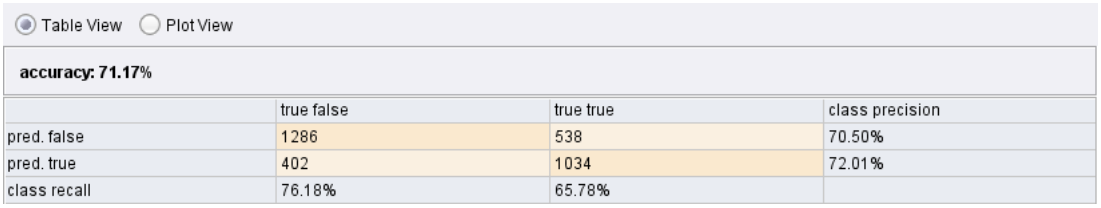


Table View Plot View

accuracy: 71.17%

	true false	true true	class precision
pred. false	1286	538	70.50%
pred. true	402	1034	72.01%
class recall	76.18%	65.78%	

Obr. 4.6: Matica zámien pre rozhodovací strom na obr. 4.5

4.2.2 Úspešnosť metódy podporných vektorov

Metóda podporných vektorov sa ukázala byť o niečo presnejšia ako rozhodovacie stromy, avšak výpočtovo omnoho náročnejšia. Táto náročnosť závisela od zadanej komplexnosti (parameter určujúci akúsi toleranciu zlej klasifikácie vo forme určovania hraníc) a pohybovali sa od desiatok sekúnd v najjednoduchšom prípade až po vyše 5 minút (testované na počítači s dvojjadrovým procesorom Intel i3 @ 2,13GHz; 3GB RAM) pre komplexnosť $C \geq 20$. Najlepšie výsledky boli dosiahnuté pre $C = 0$ (môže byť aj záporné), ostatné parametre boli nastavené tak ako boli predvolené. Algoritmus vygeneroval nasledujúce parametre pre váhy jednotlivých príznakov:

```
Total number of Support Vectors: 5956
Bias (offset): -0.69574
w[2d_to_3d_Local3Dfeatures_Contrast] = 0.13769
w[2d_to_3d_Local3Dfeatures_Energy] = -0.20176
w[2d_to_3d_Local3Dfeatures_Enthropy] = -4.81150
```

Matica zámien pre nezávislé validačné dáta hodnotené modelom s uvedenými parametrami je na obr. 4.7.

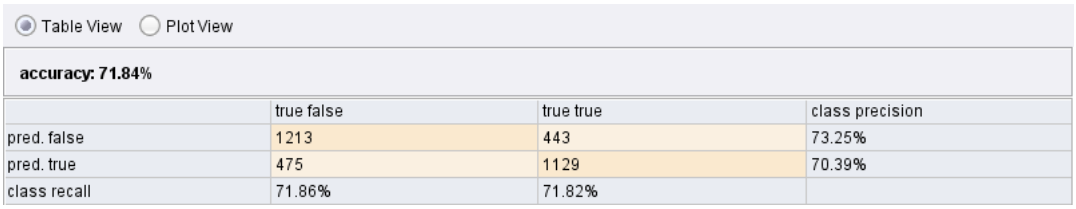


Table View Plot View

accuracy: 71.84%

	true false	true true	class precision
pred. false	1213	443	73.25%
pred. true	475	1129	70.39%
class recall	71.86%	71.82%	

Obr. 4.7: Matica zámien pre SVM

Z uvedenej matice zámien vyplýva, že celková presnosť sa zlepšila iba o desatiny percent (konkrétne o 0,67%), zatiaľ čo doba výpočtu sa niekoľkonásobne predĺžila. Lepšie výsledky dosiahla metóda SVM hlavne v senzitivite - väčší pomer správne identifikovaných pravidivých vzoriek (71,82%), čo je pri takomto type analýzy jeden z najdôležitejších ukazateľov.

Ako bolo spomenuté, boli testované aj parametre s vyššou komplexnosťou, avšak výsledky sa líšili iba minimálne (desatiny percent). V prípade $C = 50$ dokonca úspešnosť dosiahla iba 40% - pravdepodobne nastal jav v angličtine nazývaný „overfitting“ a algoritmus označoval niektoré vzorky opačne.

4.2.3 Úspešnosť algoritmu *k*-NN

Tento algoritmus je jednoznačne na obsluhu a rovnako aj výpočtový výkon a čas najjednoduchším z testovaných možností. Nanešťastie ale vykazuje veľmi nepresné výsledky, ktoré sa pri nízkych hodnotách k (počtu susedov) blížia 50% (jak všeobecná presnosť, ktorá je pri $k = 1$ iba 56,4%, tak senzitivita - 53,2%), čo už je takmer na úrovni náhodného hádania. So zvyšujúcou sa hodnotou počtu susedov síce tieto hodnoty rastú, avšak aj pri vyšších hodnotách dosahujú iba okolo 60% (ako vidieť na obr. 4.8). Testovanie bolo vykonané pre veľkosti $k = \{1, 3, 5, 7, 9, 11, 15, 21, 41\}$, kde sa ale výsledky pre $k = 21$ a $k = 41$ líšili iba o pár málo percent.

```
21-Nearest Neighbour model for classification.
The model contains 5956 examples with 3 dimensions of the
following classes:
  false
  true
```

Table View Plot View

accuracy: 61.07%

	true false	true true	class precision
pred. false	1099	680	61.78%
pred. true	589	892	60.23%
class recall	65.11%	56.74%	

Obr. 4.8: Matica zámien pre k-NN

Boli odskúšané aj ďalšie variácie a možnosti, ktoré operátor k-NN ponúka. Tieto zahŕňajú najmä rôzne metódy výpočtu vzdialenosti medzi bodmi alebo možnosť váhovania - t.j. susedným bodom, ktoré sú k testovanému bližšie, bude pri výpočte priradená väčšia váha ako vzdialenejším. Napríklad v prípade použitia porovnávacieho kritéria *CosineSimilarity* (kosínová podobnosť) bol pozorovaný mierny nárast

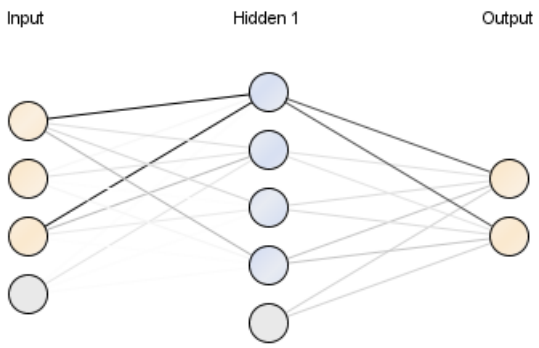
senzitivity na hodnotu 59,48%, avšak na úkor špecifickosti (pomeru správne identifikovaných negatívnych vzoriek). Použitie *KernelEuclideanDistance* (euklidovskej pozitívne definovanej matice vzdialeností) s menením typu matice pri jednotlivých pokusoch tiež neukázalo žiadne výrazné rozdiely. Pri aplikácii váhovania neboli rovnako pozorované žiadne veľké zmeny (iba desatiny percent).

4.2.4 Úspešnosť neurónových sietí

Neurónové siete sa po metóde podporných vektorov ukázali ako druhá časovo najnáročnejšia metóda. Napriek zložitosti a rozsiahlosti problematiky, ktorá sa týmto sieťam venuje, je operátor `Neural Net` v prostredí Rapidminer pojatý vcelku vľúdne aj pre menej skúseného užívateľa, ktorý sa nestratí vo veľkom množstve nastaviteľných parametrov.

Keďže daný operátor používa ako aktivačnú funkciu tzv. sigmoidu, musí byť v prípade, že vstupné atribúty nie sú z intervalu <-1;1>, zaškrtnutý parameter *normalize*, ktorý dáta prevedie na požadovaný formát. Ak sa stane, že hodnoty na vstupe operátoru sú nejakým spôsobom usporiadané, je vhodné ich pred samotnou klasifikáciou premiešať označením poľa *shuffle*. Aj napriek relatívne chaotickému usporiadaniu dát, ktoré na vstup privádzame, môžeme pri tomto premiešaní pozorovať malé zlepšenie výsledkov (niekoľko desatín percenta). Ďalším dôležitým parametrom je *training cycles*, ktorý udáva počet trénovacích cyklov. V každom z nich sa vypočíta chyba oproti skutočnosti a algoritmus spätne upravuje svoje parametre a váhy na jednotlivých uzloch aby v ďalšom cykle túto chybu znížil.

Náčrt vytvorenej neurónovej siete je na obr. 4.9, nasledovaný výstupnými parametrami vypočítanými v 1000 tréningových cykloch (medzivýsledky z vnútornej vrstvy preceptrónov program na výstupe zobrazuje tiež, je ale zbytočné ich sem uvádzať).



Obr. 4.9: Vytvorená neurónová sieť

```
Output
======
Class 'false' (Sigmoid)
-----------------------
Node 1: 5.60691
Node 2: 0.94813
Node 3: 1.13488
Node 4: 1.79735
Threshold: -1.42179

Class 'true' (Sigmoid)
----------------------
Node 1: -5.60951
Node 2: -0.90637
Node 3: -1.11779
Node 4: -1.80533
Threshold: 1.41875
```

Ako je ale pri použití tohoto modelu vidieť na matici zámien (na obr. 4.10), výsledky neurónových sietí sú (aj cez svoj veľký potenciál) v tomto nastavení porovnateľné s rozhodovacími stromami alebo metódou podporných vektorov, oproti ktorej ale majú menší výpočtový čas, čo bolo vcelku prekvapením. Očakávaním totiž bolo, že táto metóda bude najkomplikovanejšia a pre bežný počítač zdĺhavá na výpočet.

Testované boli aj konfigurácie s iným počtom cyklov. S hodnotou 500 sa prevažná väčšina výsledkov klasifikovala ako *true*, takže senzitivita dosahovala takmer 100%, ale celková presnosť klasifikácie bola ceľmi nízka. Pre vyššie počty cyklov sa výsledky zlepšujú, ale už iba vo veľmi pomalom trende (pre 2000 cyklov zlepšenie iba o cca. 0,5%). Pre test zobrazený na týchto stránkach bol tiež vyskúšaný parameter *decay*, ktorý každou iteráciou postupne znižuje rýchlosť učenia (*learning rate* - určuje rýchlosť, akou sa budú meniť váhy medzi jednotlivými iteráciami, v sekcii 2.2.2 označená ako α). Ostatné parametre boli ponechané na predvolenej hodnote.

Table View Plot View

accuracy: 66.35%

	true false	true true	class precision
pred. false	1053	462	69.50%
pred. true	635	1110	63.61%
class recall	62.38%	70.61%	

Obr. 4.10: Matica zámien pre neurónovú sieť

4.3 Vyhodnotenie

Porovnaním výsledkov z matíc na obrázkoch 4.6, 4.7, 4.8 a 4.10 sa dá prísť k záveru, že najlepšie výsledky poskytli rozhodovacie stromy a metóda podporných vektorov, ktorá však bola niekoľkonásobne náročnejšia na čas. Napriek testovaniu rôznych druhov algoritmov spolu s modifikáciou ich parametrov v snahe dosiahnuť lepší výsledok sme na výstupe neobdržali hodnoty presnosti lepšie ako 72%. Táto hodnota je síce lepšia ako náhodné hádanie, avšak pre potenciálne využitie v praxi je rozhodne nedostatočná. Podobné konštatovanie môžeme vysloviť o senzitivite, ktorá je najdôležitejšia vzhľadom na dôraz kladený na správnu identifikáciu pozitívnych vzoriek.

Tieto fakty vedú k predpokladu, že chyba nebola priamo vo výbere alebo spôsobe výpočtu modelu. Výsledky viac naznačujú, že pre použité algoritmy nebolo možné na základe zvolených a vypočítaných príznakov určiť (resp. klasifikovať) výstupnú triedu s vyššou presnosťou. Dáta z oboch tried majú veľmi podobné hodnoty atribútov (príznakov), čím sa niektoré dátové body v myslenej rovine (ako napríklad pri SVM) prekrývajú a sú klasifikované opačne. Toto poukazuje na nevhodný výber príznakov, ich nedostatočný počet alebo nesprávny spôsob ich výpočtu.

Vzhľadom na uvedené domnienky by mohlo byť zaujímavé v budúcnosti otestovať podobný systém (základ je položený v tejto práci) s iným výberom príznakov, čo by vyžadovalo modifikáciu síce iba jedného operátora (`Local3Dfeatures`), avšak bude nevyhnutné odznova vygenerovať body a otestovať výsledky, čo je časovo celkom náročná činnosť. Ako bolo uvedené v predchádzajúcej kapitole, podľa množstva vygenerovaných bodov a hodnoty oblasti okolo nich samotný výpočet príznakov pre jednu vstupnú sadu snímkov (od jedného pacienta) trvá relatívne dlho, k čomu treba pripočítať prácu s generovaním bodov manuálne, kontrolu, čas na zmenu cesty k súborom, ukladanie sád apod. Istý čas tiež zaberie následné trénovanie modelu a testovanie s rôznymi nastaveniami algoritmov, čím sa časovým rozsahom takéhoto a prípade ďalších experimentov s iným výberom príznakov dostávame mimo rozsahu tejto práce.

5 ZÁVER

V práci boli popísané možnosti počítačovej asistencie pri vyšetrovaní a analýze mozgových a nervových chorôb. Predmetom výskumu sú hlavne kvôli stúpajúcemu trendu prípadov vo svete a starnúcej generácii ľudí, ktorá je nimi najohrozenejšia. V práci boli stručne popísané príznaky a dopady dvoch najbežnejších chorôb - roztrúsenej sklerózy a alzheimerovej choroby na ľudské zdravie, myslenie a sociálny život. Taktiež sú spomenuté možnosti analýzy a diagnózy, ktoré sú aktuálne v lekárskej praxi používané. Dnes sa jedná hlavne o vyšetrenie magnetickou rezonanciou. Idea práce je práve v možnosti automatizácie, resp. spresnenia prehľadávania rezov z MRI vyšetrenia a určovanie diagnózy rýchlejšie a presnejšie.

Ďalej sa teda práca venuje opisu algoritmov strojového učenia a metódam, na základe ktorých je možné identifikovať alebo porovnávať jednotlivé fragmenty obrazu. Je možné využitie jak obyčajných štatistických výpočtov, tak tzv. lokálnych príznakov. Na základe naštudovanej literatúry boli teda vybrané niektoré spôsoby výpočtov príznakov, ktoré boli využité v tejto práci.

Samotná implementácia prebehla v prostredí Rapidminer slúžiacom na analýzu dát. Prostredie je v práci opísané, rovnako ako skladba a funkcia procesov a operátorov v ňom (vytvorených priamo v tejto práci a aj v minulosti), s uvedením ďalších rozširujúcich prameňov a návodov. Podľa týchto návodov a s odborným vedením bol v prostredí Eclipse vytvorený nový operátor, resp. trieda *Local3Dfeatures*, ktorá zahŕňa funkciu pre výpočet pravdepodobnostnej funkcie, výpočet príznakov a ďalšie potrebné časti kódu pre inštaláciu do prostredia Rapidminer.

Na základe takto vytvoreného operátoru boli generované sady bodov (s dôrazom na označenie a tiež rovnomerné rozloženie tried pomocou ďalšieho vytvoreného operátoru *TruePointGenerator*), ktoré boli rozdelené na dve veľké sady. Prvá z nich bola trénovacou sadou pre rôzne druhy algoritmov strojového učenia, ktoré vygenerovali klasifikačné modely. Tieto boli testované na druhej (validačnej) polovici dát. Najlepšie výsledky vykazovali tzv. rozhodovacie stromy a tiež metóda podporných vektorov. Namerané hodnoty však nie sú úplne dostatočné na praktické využitie.

Práca ale napriek tomu ponúka ako hlavný prínos novovytvorené operátory a sieť procesov, ktoré slúžia ako základ pre budúce testovanie iných príznakov alebo ich rozdielneho výpočtu bez nutnosti komplikovaného zásahu do vytvorených mechanizmov. Obsahuje tiež vyhodnotenie výkonnosti rôznych trénovacích algoritmov a ich porovnanie medzi sebou, čo podáva lepší obraz nielen o ich všeobecnej presnosti a spôsobe fungovania, ale aj o vhodnosti výberu použitých príznakov.

LITERATÚRA

[1] APACHE SOFTWARE FOUNDATION *Apache commons: commons Math* [online]. 2003, poslední aktualizace 03. 11. 2013 [cit. 03. 12. 2013]. Dostupné z URL: <`http://commons.apache.org/proper/commons-math//userguide/index.html`>.

[2] DIAZ, S.P.R., et al. *The need for a consensus in the use of assessment tools for Alzheimer's disease: the Feasibility Study (assessment tools for dementia in Alzheimer Centres across Europe), a European Alzheimer's Disease Consortium's (EADC) survey.* International Journal Of Geriatric Psychiatry, Volume 20, NO. 8, 2005, Pages 744-748, ISSN 0885-6230.

[3] HAMILTON, H. *Cumulative Gains and Lift Charts* [online]. ©2009, posledná revízia 08.06.2012 [cit. 07. 05. 2014]. Dostupné z URL: <`http://www2.cs.uregina.ca/~dbd/cs831/notes/lift_chart/lift_chart.html`>.

[4] HAN, J., KAMBER, M., PEI, J. *Data Mining: Concepts and Techniques.* 3.vyd. USA: Morgan Kaufmann Publishers, 2011. 708s. ISBN 978-0-12-381479-1.

[5] JANARDHAN, V., SURI, S., BAKSHI, R. *Multiple Sclerosis: Hyperintense Lesions in the Brain on Nonenhanced T1-weighted MR Images Evidenced as Areas of T1 Shortening.* Radiology, Volume 244, NO. 3, 2007, Pages 823-831, ISSN 0033-8419.

[6] KVASNIČKA, V., et al. *Úvod do teórie neuronových sietí.* 1.vyd. Bratislava: Iris, 1997. 262s. ISBN 80-88778-30-1.

[7] KOVALEV, V.A., KRUGGEL, F., GERTZ, H.J., VON CRAMON, D.Y. *Three-Dimensional Texture Analysis of MRI Brain Datasets.* IEEE transactions on medical imaging, Volume 20, NO. 5, MAY 2001, Pages 424-432, ISSN 0278-0062.

[8] MAHMOUD-GHONEIM, D., et al. *Three dimensional texture analysis in MRI: a preliminary evaluation in gliomas.* Magnetic Resonance Imaging, Volume 21, Issue 9, November 2003, Pages 983-987, ISSN 0730-725X.

[9] NÁVRAT, P., BIELIKOVÁ, M., BEŇUŠKOVÁ, Ľ., KAPUSTÍK, I., UNGER, M. Umelé neurónové siete. In *Umelá inteligencia.* 2.vyd. Bratislava: Vydavateľstvo STU, 2006. ISBN 80-227-2354-1. Kapitola 6.

[10] RAPID-I GmbH. *How to extend Rapidminer 5: white paper* [online]. 2012, [cit. 18. 11. 2013]. Dostupné z URL: <http://1xltkxylmzx3z8gd647akcdvov.wpengine.netdna-cdn.com/wp-content/uploads/2013/10/How-to-Extend-RapidMiner-5.pdf>.

[11] RAPID-I GmbH. *Rapidminer 5.0 : Manual* [online]. 2010, [cit. 18. 11. 2013]. Dostupné z URL: <http://1xltkxylmzx3z8gd647akcdvov.wpengine.netdna-cdn.com/wp-content/uploads/2013/10/rapidminer-5.0-manual-english_v1.0.pdf>.

[12] ROKACH, L., MAIMON, O. *Data mining with decision trees: Theory and applications*. 1.vyd. Singapore: World Scientific Pulishing Co. Pte. Ltd., 2007. 244s. ISBN 978-981-277-171-1.

[13] SCHER, A.I., et al. *Hippocampal shape analysis in Alzheimer's disease: A population-based study*. NeuroImage, Volume 36, Issue 1, 15 May 2007, Pages 8-18, ISSN 1053-8119.

[14] TERMENON, M., GRANA, M., BESQA, A., ECHEVESTE, J., GONZALEZ-PINTO, A. *Lattice independent component analysis feature selection on diffusion weighted imaging for Alzheimer's disease classification*. Neurocomputing, Volume 114, 19 August 2013, Pages 132-141, ISSN 0925-2312.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

ACh Alzheimerova choroba

k-NN k najbližších susedov (k Nearest Neighbors)

MRI magnetická rezonancia (Magnetic Resonance Imaging)

SM skleróza multiplex

SVM Support Vector Machine

C komplexnosť

N_{g} použitý počet farieb

$P(i, j)$ pravedpodobnostná funkcia

ZOZNAM PRÍLOH

A OBSAH CD

A.1 DP_Krajcir.pdf

Elektronická verzia tejto diplomovej práce.

A.2 source.zip

Zdrojový kód vytvorený a modifikovaný v rámci praktickej časti tejto diplomovej práce:

- Local3Dfeatures.java
- VolumeStatistics.java
- Statistics3Dfeatures.java
- GenerateMoreTruePoi.java
- PoI3DGenerator.java
- Points3DIOObject.java
- Operators3D.xml

Pre plnú funkcionalitu je nutné triedy implementovať do prostredia Rapidminer spolu s ďalšími triedami vytvorenými tímom SPL, ktoré táto práca využíva. Tieto triedy sú spoločne s celým projektom umiestnené v online repozitári: http://grid.utko.feec.vutbr.cz/usvn/public/svn/ImageProcessingRapidminer5/trunk/IMMI3DExtension/.