Univerzitet u Beogradu

Matematički fakultet

Pejčić Biserka

Primena algoritma stabla odlučivanja u prepoznavanju ponašanja i zdravstvenih rizika kod starijih osoba

MASTER RAD

Beograd, Septembar 2017.

Univerzitet u Beogradu

Matematički fakultet

MASTER RAD

Autor: Biserka Pejčić 1008/2015

Naslov: Primena algoritma stabla odlučivanja u prepoznavanju ponašanja i zdravstvenih rizika kod starijih osoba

Mentor: dr Aleksandar Kartelj, Matematički fakultet

Članovi komisije: dr Mladen Nikolić, Matematički fakultet
dr Filip Marić, Matematički fakultet

Apstrakt

Cilj ovog rada je istraživanje da li i u kojoj meri promene u aktivnostima iz svakodnevnog života starijih osoba, utiču na faktor ili predispozicije osobe – ispitanika prema određenim bolestima starije populacije. Rezultati bi se mogli koristiti za poboljšanje kvaliteta života ili zdravstvenog stanja osoba. Ovaj problem je zanimljiv i sa informatičkog aspekta jer je za njegovo rešavanje potrebno povezivanje više različitih oblasti računarstva, kao što su baze podataka, Veliki skupovi podataka, paralelno programiranje, mašinsko učenje, algoritmi i druge.

U radu se analiziraju rezultati dobijeni primenom algoritama zasnovanih na stablima odlučivanja: ID3, C4.5 i CART. Skup podataka korišćen pri uporednoj analizi pomenutih algoritama dobijen je kao rezultat praćenja određenih aktivnosti u svakodnevnom životu starijih osoba.

Ključne reči u radu: Veliki skupovi podataka, zdravstvo, starija populacija, stabla odlučivanja ID3, C4.5, CART

Sadržaj

1.Uvod

U svetu, a naročito u regionu Evropske unije prosečna starost stanovnika raste. Po statistici iz juna 2017. godine u pojedinim zemljama Evropske Unije živi 19.2% ljudi starijih od 65 godina, što ovu grupaciju stavlja na drugo mesto po veličini odmah iza radno sposobne grupacije koja čini 65.3% ukupno posmatranog uzorka od 510.3 miliona ljudi. Ovo ukazuje na porast grupacije starije od 65 godina za 2.4% u poređenju sa periodom od pre 10 godina. Ovi podaci nam ukazuju na potrebu pronalaženja rešenja i strategije za rastuću populaciju kojoj je potrebno obezbediti adekvatan kvalitet života. Ovo je i glavni motiv za ovo istraživanje.

Prepoznavanje ljudskih aktivnosti postalo je veoma važna tema istraživanja, zbog svojih višestrukih primena u oblastima kao što su mobilno računarstvo (eng. pervasive and mobile computing), video nadzor (eng. surveillance-based security), računarstvo za prepoznavanje sadržaja (eng. context-aware computing), pomoćni sistemi za samostalni život starijih (eng. ambient assistent living) ili socijalna robotika (eng. social robotics). Za pravilan pristup prepoznavanju aktivnosti prikupljeni podaci su ključni izvori. Tehnologija postaje sve više orijentisana ka ljudima kako bi ispunila sve raznovrsnije zahteve u oblasti pružanja ličnih usluga. Drugim rečima, tehnološke usluge moraju se prilagoditi ljudima - korisnicima, a ne obrnuto. Prateći taj trend, prepoznavanje ljudskih aktivnosti postaje prirodna podrška za takve prilagođene tehnologije. Razmotrimo slučaj gde je inteligentan pomoćni sistem u funkciji promovisanja zdravog načina života. Za takav sistem je neophodno da se prepoznaju aktivnosti koje izvršavaju ljudi - korisnici, kako bi se analiziralo koje vrste preporuka bi bile optimalne za poboljšanje zdravlja korisnika.

1.1 Veliki skupovi podataka

Veliki skupovi podataka (eng. Big data) se mogu definisati kao bilo koja vrsta izvora podataka koja ima sledeće tri zajedničke karakteristike:

1. potreba za prihvatanjem velikih skupova novih i skladištenjem postojećih podataka;
2. velika brzina prikupljanja novih podataka, koja je veća od brzine obrade postojećih podataka (brzina kojom pristižu novi podaci veća je od brzine obrade podataka);
3. široka raznovrsnost podataka.

Koncept Velikih skupova podataka je važan upravo zato što omogućava organizacijama da ih prikupljaju i skladište, obrađuju ih i upravljaju njima velikom brzinom. U cilju obrade Velikih skupova podataka, koristi se kombinacija tehnologija razvijenih u poslednjih 50 godina. Kombinacije novih i starih tehnologija pomažu kompanijama i organizacijama da steknu delotvoran uvid u obrađene podatke. Izazov predstavlja mogućnost upravljanja velikim skupovima različitih podataka prilagođenom brzinom i u odgovarajućem vremenskom okviru, koji omogućava analizu prikupljenih podataka u realnom vremenu. Slede neke od bitnijih karakteristika Velikih skupova podataka.

Količina podataka - mnogo faktora doprinosi uvećanju obima podataka (transakcioni podaci skladišteni godinama, tekstualni podaci koji konstantno nadolaze sa društvenih mreža, itd.). U prošlosti je prekomerna količina podataka mogla da prouzrokuje probleme oko skladištenja, ali sa današnjim cenama memorija i uređaja za skladištenje, to više ne predstavlja problem. Sada se javljaju drugi problemi, uključujući određivanje važnosti pojedinih podataka u velikom skupu podataka.

Brzina generisanja i obrade podataka - brza obrada podataka i pravovremeno reagovanje predstavljaju veliki izazov i za najveće svetske kompanije i organizacije.

Raznovrsnost - danas se podaci nalaze u velikom broju različitih formata. Tu se ubrajaju tradicionalne relacione baze podataka, tekstualne datoteke i dokumenti, elektronska pošta, video-zapisi, audio-zapisi, podaci o finansijskim transakcijama, itd. Prema nekim procenama oko 80% podataka nije numeričkog tipa, ali i nenumerički podaci moraju biti uključeni u procedure analize i donošenja odluka u vezi sa njima.

Promenljivost - kao dodatak velikim količinama i brzinama obrade podataka, tok podataka može postati prilično nepravilan sa vremenom.

Složenost - kada se obrađuju Veliki skupovi podataka, oni uobičajeno dolaze iz različitih izvora. U velikom broju slučajeva je složeno uparivati, pročišćavati i transformisati te podatke na bilo koji način. Ipak, neophodno je izvršiti povezivanje - prepoznavanje odnosa među podacima i uspostaviti hijerarhiju podataka, jer u suprotnom količina podataka može da izmakne kontroli i podaci mogu postati neupotrebljivi.

1.2 Tipovi podataka

Podaci se prikupljaju iz različitih izvora i nalaze se u različitim oblicima. Izvori podataka se dele u tri kategorije:

1. *Računarski ili mašinski generisani* - pojam mašinski generisanih podataka se obično odnosi na podatke koje proizvodi mašina bez ljudskog uticaja.
2. *Ljudski generisani* - ovo su podaci koje obezbeđuju ljudi u interakciji sa računarima i/ili svim vrstama "pametnih" uređaja.
3. *Hibridni* - neki stručnjaci tvrde da postoji i treća kategorija koja predstavlja hibrid između prve dve navedene kategorije. Međutim, ovde će se razmatrati samo prve dve.

Sa ekspanzijom razvoja senzora, pametnih uređaja i socijalnih mreža podaci su postali složeni, prvenstveno zato što sada ne uključuju samo tradicionalne strukturirane podatke, već i nestrukturirane ili polustrukturirane podatke.

Strukturirani podaci opisuju podatke koji su grupisani u relacione šeme (tabele, relacije, redovi i kolone u okviru standardnih baza podataka). Organizacija ovih podataka daje mogućnost izvršavanja jednostavnih upita čiji rezultati mogu biti korisne informacije za poslovanje.

Polustrukturirani podaci predstavljaju podatke za koje se ne može reći da su grupisani u neku fiksiranu šemu. Podaci su često nerazdvojivi, ali ipak sadrže oznake koje pomažu pri hijerarhijskom organizovanju ovakvih podataka.

Nestrukturirani podaci su uglavnom podaci koje je teško ubaciti u relacione tabele baza podataka radi analize ili izvršavanja upita nad njima. Podatke ovakvog tipa predstavljaju fotografije, audio i video zapisi, podaci sa društvenih mreža, dokumenti, datoteke i drugo.

1.2.1 Strukturirani podaci

Mašinski generisani strukturirani podaci mogu da uključuju:

- *senzorske podatke* – senzori su uređaji koji mere neku fizičku veličinu električnim putem. Prednosti električnih merenja su: visoka osetljivost, tačnost, brzina i pouzdanost, laka indikacija i prenos signala i mala potrošnja energije. Primeri uključuju radio frekvencijske identifikacione oznake (RFID), "pametne" merače (npr. elektronska brojila za merenje potrošnje električne energije), podatke medicinskih uređaja, GPS podatke, itd. Koriste se minijaturni računarski čipovi da bi se uređaji pratili sa udaljenosti. Kada

prijemnik dobije informacije, one mogu biti prosleđene serveru gde će biti analizirane. Najrasprostranjeniji primer izvora senzorskih podataka su "pametni" telefoni koji imaju senzore kao što je GPS i koji mogu biti korišćeni za razumevanje ponašanja potrošača na novi način (eng. QR code).

- *Internet dnevnik podatke* - kada serveri, aplikacije, mreže i slično rade, oni beleže različite podatke o svojoj aktivnosti. Količina ovih podataka može postati ogromna, a ovi podaci mogu biti iskorišćeni između ostalog za predviđanje narušavanja bezbednosti.
- *podatke u trenutku prodaje* - kada radnik na kasi u prodajnom objektu očita bar-kôd bilo kog kupljenog proizvoda, generišu se svi podaci vezani za proizvod.
- *finansijske podatke* - većina finansijskih sistema su danas programirani, njihov rad se zasniva na predefinisanom skupu pravila, što automatizuje proces. Podaci o trgovanju na berzi su dobar primer za to. Sadrže strukturirane podatke kao što su oznaka kompanije i trenutna vrednost njenih akcija na berzi u dolarima.

Ljudski generisani strukturirani podaci mogu da uključuju:

- *ulazne podatke* - ovo je bilo koji tip podataka koji čovek može uneti u računar, kao što je ime, prezime, godina rođenja, pol, prihod, odgovori na ankete i slično. Ovi podaci mogu biti korišćeni za razumevanje osnovnog ponašanja korisnika ili potrošača.
- *klik podatke* - svaki put kada se klikne na link na nekom sajtu na Internetu, generišu se podaci. Ovi podaci mogu biti analizirani da bi se odredilo ponašanje korisnika - potrošača i utvrdili obrasci kupovine.
- *podatke vezane za igre* - svaki potez koji se napravi u igri može biti zabeležen. Ovi podaci mogu biti korisni za razumevanje kako krajnji korisnici igraju igru i doprineti poboljšanju dizajna, strategija i koncepata samih igara.Neki od navedenih podataka ne moraju biti veliki sami po sebi, kao što su profilni podaci korisnika. Kada se objedine podaci miliona korisnika koji šalju informacije, količina podataka postaje ogromna. Mnogo ovih podataka je vezano za vreme u kom se generišu što može biti korisno za razumevanje obrazaca ili potencijalno predviđanje ishoda. Zajednički zaključak bi bio da ove informacije mogu biti moćne i mogu biti korišćene u različite svrhe.

1.2.2 Nestrukturirani podaci

Nestrukturirani podaci ne prate neki unapred definisani format. Procenjuje se da je količina nestrukturiranih podataka 4 puta veća od količine strukturiranih. Nestrukturirani podaci su zapravo podaci koji se najčešće sreću. Do skoro,

tehnologija nije podržavala druge načine rada sa ovim podacima osim skladištenja i ručne obrade. Nestrukturirani podaci se mogu naći svuda. Većina ljudi i organizacija funkcioniše na osnovu nestrukturiranih podataka. Kao i u slučaju strukturiranih podataka i nestrukturirani podaci mogu biti mašinski ili ljudski generisani. Neki primeri mašinski generisanih nestrukturiranih podataka su:

- *satelitski snimci* - podrazumevaju podatke o vremenskim prilikama ili podatke koje vlade nekih država prikupljaju prilikom satelitskog nadgledanja. Na primer, ***GoogleEarth*** poseduje ogromnu količinu satelitskih snimaka koje obrađuje i spaja na odgovarajući način.
- *naučni podaci* - seizmički snimci, atmosferski podaci, itd.

1.3 Veliki skupovi podataka u zdravstvu

Kod primene Velikih skupova podataka u zdravstvu treba uzeti u obzir da zdravstvo kao delatnost ima specifične vrste podataka. Navešćemo neke:

- podaci o pacijentima;
- podaci o zdravstvenim ustanovama, planovi lečenja;
- podaci o proizvođačima i distribuciji lekova;
- zdravstveno osiguranje, dostupnost zdravstvene zaštite, optimizacija troškova u zdravstvu.

Analiza Velikih skupova podataka je pomogla da se zdravstvo poboljša obezbeđujući personalizovane lekove i preskriptivne analize, intervencije kliničkog rizika i analize predviđanja, umanjeni gubitak i poboljšana raznovrsnost nege, automatizovano spoljašnje i unutrašnje izveštavanje o podacima pacijenta, standardizovanje medicinskih pojmova. Neka poboljšanja su dobro osmišljena, ali realizacija još uvek nije na zadovoljavajućem nivou. Sa dodatno usvojenim konceptima m-Zdravstva (mobilnog) i e-Zdravstva (elektronskog) i primenom mobilnih tehnologija, obim podataka će nastaviti da raste. Ovo uključuje elektronski zdravstveni zapis (karton), podatke u vidu snimaka, podatke dobijene od strane pacijenta, podatke dobijene od strane senzora i druge oblike podataka teške za obradu. Povećava se potreba da se obrati više pažnje na podatke i kvalitet informacija. Veliki skupovi podataka često mogu da znače i postojanje "prljavih" podataka - deo podataka koji je netačan ili nepouzdan, a koji takođe raste sa porastom količine podataka. Ljudske provere u razmerama Velikih skupova podataka su skoro nemoguće i postoji izražena potreba u zdravstvenim službama za "pametnim" (sofisticiranim) alatima za kontrolu tačnosti i verodostojnosti i obradu nepotpunih informacija. Iako su mnoge raspoložive informacije u zdravstvu sada

elektronske i spadaju u oblast Velikih skupova podataka, većinom su nestrukturirane i teške za obradu.

U ovom radu će biti analizirana primena Velikih skupova podataka na praćenje zdravstvenog stanja starijih osoba. Starije osobe imaju specifične zdravstvene probleme koje treba uzeti u obzir. Ponekad im je potrebno dodatno praćenje vitalnih funkcija i/ili dnevnih aktivnosti da bi se procenilo da li se zdravstveno stanje tačno prati (kontroliše). Drugi mogući ciljevi ovih aplikacija su otkrivanje da li se bolest razvija, ili brzo reagovanje kada se desi iznenadna promena zdravstvenih parametara. Obrasci dobijeni od dnevnih rutina mogu biti veoma korisni za pomoć starijim ljudima koji pate od nekih oblika gubitaka u pamćenju. Rezonovanje po ovim obrascima dozvoljava da se predvide akcije i da se obezbede podsetnici na stvari koje treba uraditi (na primer, navođenje kada se osoba ne seća koraka da isprati kompletnu aktivnost). Biće analizirani prikupljeni podaci iz svakodnevnog života starijih osoba i na osnovu promena u njihovoj rutini će se određivati u kojoj meri ispoljavaju faktore rizika vezane za bolesti koje se javljaju kod starijih osoba, kao što su Parkinsonova bolest, kardiovaskularne bolesti, hronične bolesti, demencija, Alchajmerova bolest, astma, respiratorne bolesti, skolonst padovima, slabija pokretljivost…

Ciljevi istraživanja u ovoj oblasti:

- prikupljanje podataka niskog nivoa putem senzora (prikupljanje sirovih podataka);
- integracija - obrada podataka dobijenih pomoću senzora da bi se dobio kontekst informacija i metoda učenja/rezonovanja za prepoznavanje aktivnosti;
- donošenje zaključaka kroz pristupe vođene podacima i vođene znanjima;
- upoznavanje negovatelja i eksperata sa korisnim i bitnim informacijama i zaključcima;

2. Opis problema prepoznavanja ljudskih aktivnosti

Prepoznavanje ljudskih aktivnosti postalo je važna tema istraživanja u oblastima kao što su mobilno računarstvo [Choudhury& Consolvo 2008], pomoćni sistemi za samostalni život [Philipose & Fishkin 2004], socijalna robotika [Fong i sar. 2003], bezbednost zasnovana na nadgledanju [Fernandez-Caballero 2012] i računarstvo za prepoznavanje sadržaja [Laerhoven & Aidoo 2001]. Da bi se omogućilo prepoznavanje aktivnosti u ljudskom okruženju raspoređuju se različite vrste senzora. Oni omogućavaju praćenje ponašanja stanovništva i zapažanje promena u životnoj sredini nastalih ljudskim akcijama. Informacije koje se dobijaju uz pomoć tih senzora obrađuju se putem tehnika za analizu podataka i / ili formalizmima za predstavljanje znanja da bi se stvorili odgovarajući modeli aktivnosti i naknadno se koristili za prepoznavanje aktivnosti. Naučna zajednica je razvila dva glavna pristupa za rešavanje problema prepoznavanja aktivnosti: pristupe zasnovane na podacima i pristupe zasnovane na znanju.

Pristupi zasnovani na podacima koriste veliku bazu podataka dobijenih od senzora za uočavanje modela aktivnosti, koriste tehnike rukovanja podacima i tehnike mašinskog učenja. Sa druge strane, pristupi zasnovani na znanju istražuju bogato prethodno znanje iz domena od interesa za izgradnju modela aktivnosti koristeći inženjering znanja. Većina radova u ovoj oblasti koristi koncept aktivnosti kao osnovu pomoću koje se konstruišu aplikacije kao što su praćenje zdravlja ili pomoćni sistemi za nezavisan život. U sistemima za prepoznavanje aktivnosti zasnovanih na znanju, široko prepoznatljiv nedostatak je da su modeli aktivnosti obično statički, tj. jednom kada su definisani, ne mogu se automatski prilagođavati specifičnostima korisnika [Chen i sar. 2014]. Ovo je vrlo restriktivno ograničenje, jer u opštem slučaju nije moguće definisati kompletni model aktivnosti za svakog korisnika. Eksperti za domen imaju neophodno znanje o aktivnostima, ali ovo znanje ne mora obavezno da generiše kompletne modele u svim slučajevima. Da bi se napravili sistemi za prepoznavanje aktivnosti zasnovani na znanju koji rade u aplikacijama u realnom svetu, modeli aktivnosti moraju se automatski razvijati kako bi se prilagodili različitim ponašanjima korisnika. Zaključak je da su prilagodljivost i evolucija modela aspekti koji bi se mogli pravilno rešiti pristupima zasnovanim na podacima.

2.1 Pravci razvoja

Prepoznavanje ljudskih aktivnosti može se klasifikovati u dve kategorije u odnosu na upotrebe senzora koji se koriste za praćenje aktivnosti:

- *prepoznavanje aktivnosti na osnovu vizije* zasnovano je na upotrebi vizuelno osetljivih objekata kao što su video kamere za praćenje ponašanja korisnika i promena u okolini [Weinland i sar. 2011]. Aktivnost zasnovana na vizuelnom prepoznavanju je dugo bila u fokusu istraživača, ali je van dometa ovog rada.
- *prepoznavanje aktivnosti na osnovu senzora* se zasniva na upotrebi senzorskih mrežnih tehnologija za praćenje aktivnosti [Chen i sar. 2012]. Glavne prednosti senzorski zasnovanog pristupa u odnosu na vizuelno zasnovane su vezane za privatnost i etiku [Yilmaz i sar. 2006], jer se kamere uglavnom posmatraju kao uređaji za snimanje. Generisani podaci sa senzora prilikom senzorskog praćenja su uglavnom vremenska serija promena stanja i/ ili različitih vrednosti parametara koji se obično obrađuju fuzijom (objedinjavanjem) podataka, probabilističkom ili statističkom analizom metoda i tehnika formalnog znanja za prepoznavanje aktivnosti.

2.2 Ranija istraživanja

Postoje dva glavna pristupa za prepoznavanje aktivnosti zasnovana na senzorima u literaturi: pristupi zasnovani na podacima i zasnovani na znanju. Iscrpan pregled oba pristupa može se naći u već spomenutom radu [Chen i sar. 2012a]. Predmet istraživanja u ovom radu je pristup zasnovan na podacima, pa je pažnja posvećena prevashodno ranijim istraživanjima, načinima prikupljanja podataka i načinima obrade prikupljenih podataka.

GatorTech je stariji projekat „pametne kuće“ sproveden na univerzitetu na Floridi [Helal i sar. 2005]. Integrisao je skup senzora i uređaja da bi obezbedio usluge kao što su prepoznavanje glasa i praćenje aktivnosti stanovnika. Izvanredni rani projekti „pametne kuće“ takođe uključuju sledeće: Prilagodljiva svestrana kuća **(MavHome**) sa Univerziteta u Teksaku iz Arlington-a [Cook i sar. 2013b], **PlaceLab** sa MIT-ja [Logan i sar. 2007] i Sistem pametne laboratorije (**Inteligent System Lab** (ISL)) sa Univerziteta u Amsterdamu [van Kasteren i sar. 2010]. **CASAS** projekat „pametne kuće” razvijen je od strane Univerziteta u Vašingtonu 2007 [Chen & Dawadi 2011]. To je projekat istraživanja u više oblasti koji se fokusira na izradu „pametnog kućnog okruženja“ korišćenjem neupadljivih senzora i detektora pokreta. Oblasti istraživanja korišćene u CASAS su tehnologije za pomoć (asistivne tehnologije),

veštačka inteligencija, mašinsko učenje i prepoznavanje aktivnosti. Isti tim je razvio i njihovo skorije istraživanje „ pametna kuća u paketu“ [Cook i sar. 2013c], koja je pojednostavljen dizajn „pametne kuće“, jednostavan je za ugradnju i obezbeđuje funkcionalnosti „pametne kuće” bez potrebe prilagođavanja. Ove sposobnosti uključuju prepoznavanje aktivnosti u realnom vremenu kako događaji pročitani od strane senzora dolaze na ulaz, i prepoznavanje aktivnosti korišćenjem nenadgledanog algoritma učenja. **SWEET-HOME** je nacionalno priznati projekat istraživanja u Francuskoj koji ima za cilj da dizajnira novi sistem „pametne kuće“ zasnovan na glasovnoj (audio, zvučnoj) tehnologiji [Vacher i sar. 2011]. Ovaj projekat ima 3 glavna cilja:

- obezbeđivanje interaktivne tehnologije zasnovane na zvuku koja dozvoljava korisnicima da imaju potpunu kontrolu njihovog kućnog okruženja,
- prepoznavanje stresnih situacija i
- olakšavanje društvenog uključivanja starijih i krhke populacije.

Zanimljiv pravac istraživanja njihovog sistema pametnih kuća je proces odlučivanja pomoću računarstva za prepoznavanje sadržaja (eng. context-aware computing) koji koristi pristup skrivene Markovljeve logičke mreže da poveća sposobnost snalaženja u neizvesnim događajima koji odstupaju od pravih podataka dobijenih od strane senzora [Chahuara i sar. 2014]. Skoriji projekat pametne kuće je „nenametljivo pametno okruženje za nezavistan život“ (**USEFIL**), FP7 projekat koji je započeo 2011 godine. Cilj mu je da obezbedi naprednu i finansijski izvodljivu pomoć u zdravstvenoj nezi u okruženjima „pametnih” kuća [Antoniou i sar. 2014]. Ograničen skup senzora i uređaja (kao što je uređaj koji se nosi na zglobu, kamera, mikrofon i Kinect senzor) korišćen je u postavci „pametne kuće” da bi se prepoznale osnovne fizičke aktivnosti (ležanje, sedenje, hodanje, stajanje, vožnja bicikla, trčanje, penjanje i silaženje po stepenicama) starijih osoba [Billis i sar. 2013]. Cena samog sistema je niska i razvijene su platforme otvorenog koda da olakšaju uopštavanje aplikacije koja smanjuje jaz između napredne tehnologije i populacije koja stari. Autori Qing i Mohan predlažu rešenja za „pametnije i bezbednije kuće” **CSIRO** da povećaju kvalitet života starijih ljudi [Zhang i sar. 2013]. Da bi se ovo postiglo, veliki broj senzora za okruženje je postavljen na razne lokacije unutar „pametne kuće”. Oni služe kao nenametljivi uređaji za praćenje i prepoznavanje ljudskog ponašanja. Platforma za „pametan život uz pomoć” (eng. smart assistive living - **SAL**) omogućava starijima da ostanu u svojim kućama nezavisni koliko je god to moguće [Zhang i sar. 2014]. Od senzora postavljenih u „pametnim kućama” očekuje se da obezbede kontinualno slanje podataka na server. Ekstrakcija, grupisanje i analiza ovih podataka korišćenjem mehanizama mašinskog učenja je

korisna za utvrđivanje dijagnoze i donošenje odluka od strane kliničkih eksperata i zdravstvenih negovatelja.

2.3 Načini prikupljanja podataka

Aktivnosti iz svakodnevnog života koje su merene (praćene) i koje će biti korišćene za analizu mogu se podeliti u nekoliko kategorija, uzimajući u obzir međusobnu povezanost:

- *detekcija određenog stanja organizma* - vreme spavanja, vreme dubokog sna, vreme dremanja, vreme uspavljivanja, broj intervala spavanja/buđenja**;**
- *detekcija opšteg stanja tela* – apetit, broj obroka, zamor, broj padova, otkucaji srca, pamćenje - rad testova CANTAB, nivo bola, telesna težina, test uparivanja reči**;**
- *vreme posvećeno brizi o zdravlju* - vreme provedeno kod lekara, broj poseta lekaru, broj poseta apoteci, vreme provedeno u apoteci;
- *detekcija boravka u određenim prostorijama u kući* - vreme provedeno u kući, promena prostorija, vreme provedeno u kupatilu / toaletu, broj odlazaka u toalet, vreme provedeno u spavaćoj sobi, vreme provedeno u kuhinji, vreme provedeno u dnevnoj sobi, vreme gledanja TV, vreme provedeno u trpezariji;
- *aktivnosti vezane za komunikaciju sa drugima* - broj poziva, broj propuštenih poziva, broj primljenih poziva, broj poseta, broj primljenih poseta;
- *vreme posvećeno fizičkim aktivnostima* - utrošak kalorija usled fizičke aktivnosti, vreme intezivne fizičke aktivnosti, razdaljina hodanja napolju, procenat razdaljine brzog hodanja, procenat razdaljine sporog hodanja, brzina hodanja napolju, hodanje stepenicama, vreme hodanja napolju;
- *vreme provedeno van kuće* - broj izlazaka iz kuće, vreme provedeno napolju, broj poseta bioskopu, vreme provedeno u bioskopu, posete kulturnim manifestacijama, prosečno vreme provedeno u poseti kulturnim manifestacijama, vreme provedeno u javnom parku, broj poseta javnim parkovima, vreme provedeno u prevozu, upotreba javnih sredstava prevoza - pređena razdaljina, broj vožnji javnim prevozom, vreme provedeno u restoranima, broj poseta restoranima, broj poseta ustanovama za smeštaj i boravak starih lica, vreme provedeno u ustanovama za smeštaj i boravak starih lica, vreme provedeno u trgovini, broj poseta trgovini.

2.4 Filtriranje i analiza dobijenih podataka

Pristup zasnovan na podacima je korišćenje tehnike za rukovanje podacima i tehnike mašinskog učenja za uočavanje modela aktivnosti. Što se tiče modela

pristupa, mogu se izdvojiti dve glavne kategorije: generativni pristup i diskriminativni pristup.
Generativni pristup pokušava da izgradi potpuni opis ulaznog skupa (prostora) podataka, obično pomoću probabilističkog modela. Najjednostavniji mogući generativni pristup je „naivni" Bajesov klasifikator, koji se koristio sa obećavajućim rezultatima za prepoznavanje aktivnosti [Bao & Intille 2004], [Brdiczka i sar. 2007], [Cook & Schmitter-Edgecombe 2009], [Tapia i sar. 2004] [van Kasteren & Krose 2007], [Maurer & Rove 2006]. U složenijim pristupima autori se oslanjaju na skriveni Markovljev model [Galata i sar. 1999], [Moeslund i sar. 2006] ili dinamičke Bajesove mreže [Brand i sar. 1997], [Oliver i sar. 2004]. Diskriminativni pristup samo modeluje mapiranje sa ulaza - podatke u izlaze - oznake aktivnosti.
Diskriminativni pristupi uključuju mnoge nadgledane pristupe učenja, kao što su uslovna slučajna polja [Vail i sar. 2007], linearno ili nelinearno diskriminativno učenje, npr. Metoda podržavajućih vektora [Brdiczka 2009] i inkrementalni klasifikatori [Ordonez i sar. 2013]. Postoji nekoliko pristupa koji se ne mogu jasno klasifikovati u diskriminativne ili generativne kategorije, već koriste kombinaciju obe [Lester i sar. 2005], [Pentney i sar. 2008], [Omar i sar. 2010].

Jedan od problema pristupa usmerenih na podatke je potreba za označenim aktivnostima u bazi podataka. Neki autori pokušavaju da prevaziđu ovaj problem u svojim radovima [Rashidi & Cook 2011b], [Rashidi & Cook 2013]. Oni koriste neoznačenu bazu podataka, gde izdvajaju klastere aktivnosti korišćenjem nenadgledanih tehnika učenja. Navedeni klasteri se koriste za obuku i pojačanje skrivenog Markovljevog modela, za koji se pokazalo da može prepoznati nekoliko aktivnosti. Iako predstavljeni pristupi prevazilaze problem zavisno od označenih baza podataka, i dalje pate od nekih drugih tipičnih problema tehnike usmerene na podatke: problem „hladnog" početka (eng. cold start), i problem prevelikog uopštavanja modela aktivnosti (eng. underfitting).

Sa druge strane, prenos učenja se koristi da bi modeli zasnovani na podacima mogli biti višestruko primenljivi. Jedno od istraživanja o prenosu učenja za prepoznavanje aktivnosti sprovedeno je u radu [Cook i sar. 2013a]. Glavna ideja prenosa učenja za prepoznavanje aktivnosti je da pojača proces obuke za novog korisnika i/ili okruženje, prenos stečenog znanja na drugog korisnika i/ili okolinu [Rashidi & Cook 2011a]. Cilj prepoznavanja anomalija je da identifikuje retke šablone u skupu podataka. To su šabloni koji se ne uklapaju u pojam normalnog ponašanja. Da bi se primenile tehnike zasnovane na pristupu vođenom pravilima, mogu biti korišćeni pristup otkriven vremenskim relacijama i pristupi zasnovani na sličnosti. Identifikacija anomalija, posebno u sistemima

za praćenje dnevnih aktivnosti, može biti od velikog značaja za eksperte da donose odluke ili postave dijagnozu u hitnim slučajevima. Takođe može biti korišćena da obezbedi starijim ljudima sa oštećenjima u pamćenju podsetnike ili zvučne signale. Na primer, pacijenti sa depresijom mogu ispoljiti težnju za boravkom u kući i retkim izlascima van kuće, manje želje za razgovorom, neregularno spavanje i smanjen apetit ili unos hrane, dok pacijenti sa dijabetesom mogu da ispolje sklonost ka učestaloj konzumaciji tečnosti (vode) i hrane, pospanost i izlučivanje tečnosti (odlasci u toalet). Prepoznavanje ovih obrazaca može biti od pomoći u dijagnostikovanju ovih bolesti.

3. Stabla odlučivanja

Stabla odlučivanja smatraju se jednim od najpopularnijih pristupa za klasifikaciju podataka. Istraživači iz raznih naučno–tehničkih disciplina kao što su statistika, mašinsko učenje, prepoznavanje obrazaca i istraživanje podataka bavili su se problemom rasta stabla odlučivanja na osnovu raspoloživih podataka. Stablo odlučivanja sadrži koren, odnosno čvor koji nema ulaznih grana. Svi ostali čvorovi imaju tačno jednu ulaznu granu. Čvor sa izlaznim granama zove se unutrašnji čvor. Svi ostali čvorovi zovu se listovi (takođe poznati kao terminalni ili čvorovi odlučivanja). U stablima odlučivanja, svaki unutrašnji čvor deli ulazne podatke (primere) na 2 ili više podprostora prema određenim diskretnim funkcijama ulaznih vrednosti atributa. U najjednostavnijem i najčešćem sljučaju, svaki podatak uzima u obzir jedan atribut, tako da je prostor podataka podeljen prema vrednosti razmatranog atributa. Svaki list stabla je pridružen nekoj klasi i predstavlja najbolju odgovarajuću ciljnu vrednost. Dodeljivanje klase nekom podatku se vrši prolaskom kroz stablo počev od korenog čvora ka listovima. Istraživači se odlučuju za manje složena stabla odlučivanja, pošto mogu biti smatrana razumljivijim. Složenost stabla ima ključan uticaj na njegovu tačnost [Brieman i sar. 1984]. Složenost stabla se eksplicitno kontroliše pomoću kriterijuma zaustavljanja koji se koristi i pomoću primenjene metode potkresivanja. Obično se složenost stabla meri prema jednoj od sledećih metrika:

- ukupan broj čvorova,
- ukupan broj listova,
- dubina stabla i
- korišćen broj atributa.

3.1 Algoritamsko okruženje za stabla odlučivanja

Mehanizmi za formiranje stabla odlučivanja su algoritmi koji automatski grade stablo odlučivanja na osnovu datog skupa podataka. U većini slučajeva cilj je naći stablo odlučivanja takvo da je greška klasifikacije minimalna. Mogu biti definisane i druge funkcije cilja, na primer: najmanji broj čvorova koje stablo mora da ima ili najmanja prosečna dubina koju stablo treba da dostigne. Predstavljanje optimalnog stabla odlučivanja na osnovu podataka smatra se teškim zadatkom. Pokazano je da je građenje minimalnog binarnog stabla sa poštovanjem očekivane greške klasifikacije NP težak algoritam [Hyafil & Rivest 1976]. Čak i traženje minimalnog ekvivalentnog stabla odlučivanja za dato stablo odlučivanja [Zantema & Bodlaender 2000] ili izgradnja optimalnog stabla odlučivanja na osnovu date tablele odlučivanja pokazuje da je u pitanju NP težak

problem [Naumov 1991]. Ovi rezultati ukazuju da je izgradnja optimalnog stabla odlučivanja izvodljiva samo kod manjih problema. Posledično, neophodne su heurističke metode za rešavanje problema izgradnje optimalnog stabla odlučivanja. Ove metode mogu biti podeljene u 2 grupe: s vrha na dole i sa dna na gore. Postoji nekoliko mehanizama za izgradnju stabla odlučivanja s vrha na dole kao što su: **ID3** [Quinlan 1986], **C4.5** [Quinlan 1993], **CART** [Breiman i sar. 1984]. Neki se sastoje od 2 konceptualne faze: rasta i potkresivanja (C4.5 i CART). Drugi mehanizmi za izgradnju stabla odlučivanja primenjuju samo fazu rasta. Uočimo da su ovi algoritmi „pohlepni po prirodi" i grade stabla odlučivanja s vrha na dole rekurzivnim pristupom (takođe poznatom kao „podeli pa vladaj"). U nastavku je prikazan pesudo kod uopštene implementacije formiranja stabla odlučivanja.

Algoritam 3.1 : Uopšteni algoritam formiranja stabla odlučivanja

```
StabloOdlucivanja
Ulaz : primeri, ciljniAtribut , atributi;
if svi primeri imaju istu vrednost ciljnog atributa then
        return novoDrvo(primeri.ciljnaVrednost);
if atributi is empty then
        return najcescaKlasa(primeri);
else
        najboljiAtribut = najboljiAtribut(primeri);
        stablo = novoDrvo(najboljiAtribut);
        for each vrednost v of najboljiAtribut do
                primeriP=primeriSaVrednoscuAtributa (primeri,
                                                najboljiAtribut, v);
                podstablo = stabloOdlucivanja (primeriP,
                                        atributi – najboljiAtribut,
                                        ciljniAtribut);
                dodajPodstabloZaVrednost(stablo, podstablo, v);
return stablo;
```

U svakoj iteraciji, algoritam razmatra particionisanje skupa podataka za formiranje stabla koristeći rezultat funkcije podele na diskretnom skupu podataka, koja se primenjuje na ulazne atribute. Izbor najpogodnije funkcije napravljen je prema nekim merama podele. Nakon izbora odgovarajuće podele, svaki čvor dalje razdvaja skup podataka za formiranje stabla na manje podskupove, dok vrednost mere za svaku od podela ne bude manja od određene vrednosti mere koja mora biti zadovoljena da bi se podela izvršila ili je kriterijum zaustavljanja zadovoljen.

3.2 Kriterijumi za formiranje stabala

U većini slučajeva, funkcije podele na diskretnom skupu podataka dele unutrašnji čvor prema vrednosti jednog atributa. Posledično, mehanizam za izgradnju stabla odlučivanja pronalazi najbolji atribut po kom će izvršiti podelu. Postoji nekoliko kriterijuma podele po vrednosti jednog atributa. Ovi kriterijumi mogi biti formirani na razne načine, kao što su:

- prema poreklu mere: teorija inofrmacija, zavisnost i udaljenost;
- prema strukturi mere: kriterijumi zasnovani na nečistoći, normalizovani kriterijumi zasnovani na nečistoći i binarni kriterijumi.

3.2.1 Kriterijumi zasnovani na nečistoći

Za dati vektor mogućih vrednosti atributa **x** = (x_1, x_2, ..., x_k) ∈ R^k neka je sa **p** = (p_1, p_2, ..., p_k) ∈ R^k označen vektor pridruženih verovatnoća, odnosno učestalost pojavljivanja pri čemu važi $\sum_{i=1}^{k} p_i = 1$. Mera nečistoće je funkcija F : $[0, 1]^{\circ k} \rightarrow$ R takva da zadovoljava sledeće uslove:

1) F (P)>= 0,
2) F (P) je minimum ako ∃i tako da je komponenta $p_i = 1$,
3) F (P) je maksimum ako ∀i, 1<= i<= k, $p_i = 1/k$,
4) F (P) je simetrična u odnosu na komponente skupa P,
5) F (P) je glatka (ima izvod svugde) u svom opsegu.

Primetimo da ako vektor verovatnoće ima komponentu vrednosti 1 (vektor **x** ima samo jednu vrednost), tada je promenljiva definisana kao čista. Sa druge strane, ako su sve komponente jednake, nivo nečistoće dostiže maksimum. Na skupu podataka za formiranje stabla S, p^y je vektor pridruženih verovatnoća za ciljni atribut y. Pri tome i-ta komponenta vektora p^y je označena sa $p_i{}^y = |\{z \,|z \in S, C(z) = c_i\}|/|S|$, gde je c_i vrednost ciljnog atributa y za $i = 1 \ldots |dom(y)|$, a $C(z)$ je vrednost ciljnog atributa za podatak z. Kvalitet podele prema atributu x u skladu sa vrednostima atributa x iz skupa podataka S označen sa $v_j \in dom(x)$ je definisan kao:

$$\Delta F(x, S) = F(p^y) - \sum_{j=1}^{|dom(x)|} \frac{\left|\{z \,|z \in S, C_x(z) = v_j\}\right|}{|S|} \cdot F\left(p^y\left(\{z \,|z \in S, C_x(z) = v_j\}\right)\right).$$

p^y računamo na celom skupu podataka S i na podksupovima skupa S za sve vrednosti $v_j \in dom(x)$, $j = 1 \ldots |dom(x)|$

3.2.2 Informaciona dobit

Informaciona dobit je kriterijum zasnovan na nečistoći koji koristi meru entropije kao meru nečistoće [Quinlan 1987].

$$InformationGain(x,S) = Entropy\,(y,S) - \sum_{v_j \epsilon dom(x)} \frac{\left|\{z\,|z\ \in S\,, C_x\,(z) =\ v_j\}\right|}{|S|} \cdot Entropy(y,\{z\,|z\ \in S\,, C_x\,(z) =\ v_j\})$$

gde je :

$$Entropy\,(y,S) = \sum_{c_j \epsilon dom(y)} -\frac{|\{z\,|z\ \in S\,, C\,(z) =\ c_i\}|}{|S|} \cdot log_2 \frac{|\{z\,|z\ \in S\,, C\,(z) =\ c_i\}|}{|S|}$$

3.2.3 Gini indeks

Gini indeks je kriterijum zasnovan na nečistoći koji meri odstupanja između raspodele verovatnoće ciljne vrednosti. Gini indeks je korišćen u raznim radovima kao što su [Breiman i sar.1984] i [Gelfand i sar. 1991] i definisan je kao:

$$Gini\,(y,S)\ =\ 1\ -\ \sum_{c_j \epsilon dom(y)} \left(\frac{|\{z\,|z\ \in S\,, C\,(z) =\ c_i\}|}{|S|}\right)^2$$

Posledično, kriterijum vrednovanja za izbor atributa a_i je definisan kao:

$$GiniGain(x,S) = Gini(y,S) - \sum_{v_j \epsilon dom(x)} \frac{\left|\{z\,|z\ \in S\,, C_x\,(z) =\ v_j\}\right|}{|S|} \cdot Gini\big(y,\{z\,|z\ \in S\,, C_x\,(z) =\ v_j\}\big)$$

3.2.4 Normalizovani kriterijum zasnovan na nečistoći

Kriterijumi zasnovani na nečistoći opisani iznad orijentisani su prema atributima sa većim vrednostima domena. Naime, preferiraju ulazne atribute sa mnogo vrednosti u odnosu na atribute sa manje vrednosti [Quilian 1986]. Na primer,

ulazni atribut koji predstavlja državni bezbednosni broj (Jedinstveni matični broj građana - JMBG) će verovano dati najviše informacija. Međutim, dodavanje ovog atributa u stablo odlučivanja dovešće do loše generisane tačnosti. Iz tog razloga, korisno je normalizovati mere zasnovane na nečistoći, kao što je opisano u sledećoj sekciji.

3.2.5 Odnos dobiti

Odnos dobiti „normira“ informacionu dobit po sledećoj formuli (Quinlan. 1993):

$$GainRatio(x,S) = \frac{InformationGain(x,S)}{Entropy(x,S)}$$

Uočava se da ovaj odnos nije definisan kada je imenilac 0. Takođe odnos može da teži favorizaciji atributa za koje je imenilac veoma mali. Shodno tome, predlažu se 2 faze. Prvo, informaciona dobit se računa za sve atribute. Zatim, uzimajući u obzir samo atribute koji su se pokazali bar jednako dobro kao prosečno prikupljeni podaci, atribut koji je proizveo najbolji odnos dobiti je izabran. Pokazano je da odnos dobiti teži da nadjača jednostavan kriterijum informacione dobiti, oba u pogledu tačnosti, kao i u pogledu složenosti klasifikatora [Quinlan 1988].

3.2.6 Binarni kriterijumi

Binarni kriterijumi su korišćeni za kreiranje binarnih stabla odlučivanja. Ove mere su zasnovane na podeli domena ulaznih atributa na 2 poddomena. Neka $\beta(x, dom1(x), dom2(x), S)$ označava binarni kriterijum vrednost za atribut x u uzorku S gde su $dom1(x)$ i $dom2(x)$ njegovi odgovarajući poddomeni. Vrednost dobijena za optimalnu podelu atributa domena u 2 uzajamno isključujuća poddomena, koji zajedno čine ceo domen, korišćena je za poređenje atributa.

3.2.7 AUC kriterijum deljenja

Ideja za korišćenje AUC metrike (eng. Area under the ROC curve) kao kriterijuma deljenja je predložena u radu [Ferri i sar. 2002]. Atribut koji sadrži maksimalnu oblast ispod konveksnog omotača ROC krive je odabran. Pokazano je da je deljenje zasnovano na AUC kriterijumu bolje od drugih kriterijuma deljenja zajedno u odnosu na klasifikacionu tačnost i oblast ispod ROC krive. Bitno je zapaziti da za razliku od nečistih kriterijuma, ovaj kriterijum ne radi poređenje između nečistoće roditeljskog čvora sa težinom nečistoće čvorova - dece nakon podele. Ovaj kriterijum je korišćen u radu [Saeb i sar. 2017].

3.2.8 Poređenje kriterijuma deljenja po vrednosti jednog atributa

Studije upoređivanja kriterijuma deljenja opisane u prethodnom delu i druge su sprovedene od strane nekoliko istraživača u poslednjih 30 godina, kao što su [Baker & Jain 1976], [BenBassat 1978], [Mingers 1989], [Fayyad & Irani 1992], [Buntine & Niblett 1993], [Loh & Shih 1997], [Loh & Shih 1999], [Lim i sar. 2000]. Većina ovih upoređivanja su zasnovana na empirijskom rezultatu, iako postoje neki teorijski zaključci. Mnoga od ovih istraživanja ističu da, u većini slučajeva, izbor kriterijuma razdvajanja neće uticati na performanse stabla. Svaki kriterijum je bolji u nekim slučajevima i lošiji u drugim, kao što teorema besplatnog ručka (eng. "no free lunch") pokazuje.

3.3 Kriterijum deljenja po vrednostima više atributa

U kriterijumu deljenja po vrednostima više atributa, nekoliko atributa može učestvovati u jednom čvoru podele. Očigledno, traženje najboljeg kriterijuma podele po vrednostima više atributa je komplikovanije nego traženje najbolje podele po vrednosti jednog atributa. Iako ovaj tip kriterijuma može dramatično da popravi performanse stabla, ovi kriterijumi su mnogo manje popularni nego kriterijumi podele po vrednosti jednog atributa.Većina kriterijuma deljenja po vrednostima više atributa se zasniva na linearnoj kombinaciji ulaznih atributa. Traženje najbolje linearne kombinacije može biti odrađeno korišćenjem "pohlepnih algoritama" [Breiman i sar. 1984], [Murthy 1998], (i/ili) linearnog programiranja [Duda & Hart 1973], [Friedman 1977], [Sklansky & Wassel 1981], [Lin & Fu 1983], [Loh & Vanichetakul 1988], [John 1996] i drugih algoritama [Utgoff 1989a], [Lubinsky 1993], [Sethi & Yoo 1994].

3.4 Kriterijum zaustavljanja

Faza rasta stabla nastavlja se dok se ne ispuni kriterijum zaustavljanja. Sledeći uslovi su uobičajena pravila zaustavljanja:

1. Sve instance skupa podataka za formiranje stabla pripadaju jednoj vrednosti od ciljnog atributa (y);
2. Maksimalna dubina stabla je postignuta;
3. Broj slučajeva u krajnjem čvoru (listu) je manji od minimalnog broja slučajeva koje može imati roditeljski čvor;

4. Ako se čvor podelio, broj slučajeva u jednom ili više čvorova - dece će biti manji od minimalnog broja slučajeva za čvor – dete;
5. Najbolji kriterijum deljenja nije veći od određenog praga.

3.5 Metode potkresivanja (eng. Pruning)

Primenom strogog kriterijuma zaustavljanja teži se da se naprave mala stabla odlučivanja i stabla odlučivanja koja previše uopštavaju. Sa druge strane, korišćenje labavog kriterijuma zaustavljanja teži da generiše velika stabla odlučivanja, prenatrpana u odnosu na skup polaznih podataka za odlučivanje. Za rešavanje ove dileme – labavi ili strogi kriterijum su razvijene metode potkresivanja [Breiman i sar. 1984]. Prema ovoj metodologiji, koristi se labav kriterijum zaustavljanja, dozvoljavajući stablu odlučivanja da prepuni skup polaznih podataka za odlučivanje. Zatim se prepunjeno stablo transformiše u manje stablo uklanjanjem podgrana (podstabala) koje ne doprinose smanjenju tačnosti. Pokazano je u raznim studijama da primena metoda potkresivanja može poboljšati performanse stabla odlučivanja. Drugi glavni motiv za potkresivanje je „zamena tačnosti za pojednostavljenje“ [Bratko & Bohanec 1994]. Kada je cilj napraviti dovoljno precizan kompaktni opis pojmova, potkresivanje je veoma korisno. U ovom procesu, inicijalno stablo odlučivanja je viđeno kao potpuno tačno. Prema tome tačnost potkresanog stabla odlučivanja označava koliko je približno originalnom stablu. Postoji nekoliko tehnika za potkresivanje stabla odlučivanja. Mnoge od njih se primenjuju korišćenjem kretanja kroz čvorove sa vrha na dole ili sa dna na gore. Čvor je potkresan ako se u ovoj operaciji poboljšava određeni kriterijum. Sledeći pododeljak opisuje najpopularnije tehnike potkresivanja.

3.5.1 Potkresivanje složenosti troška

Potkresivanje složenosti troška (takođe poznato kao potkresivanje najslabije karike ili potkresivanje složenosti greške) izvršava se u dve faze [Breiman i sar. 1984]. U prvoj fazi sekvenca stabala T_0, T_1, ..., T_k je kreirana nad podacima za formiranje stabla gde je T_0 originalno stablo pre transformacije i T_k je koreno stablo. U drugoj fazi, jedno od ovih stabala je odabrano kao transformisano stablo prema proceni uopštavanja greške. Stablo T_{i+1} se dobija zamenom jednog ili više podstabala u prethodnom stablu T_i sa odgovarajućim listovima. Podstabla koja su potkresana su ona koja zadržavaju najmanji porast stope greške koja je nastala po potkresanom listu:

$$\alpha = \frac{\varepsilon(pruned(T,t),S) - \varepsilon(T,S)}{|leaves(T)| - |leaves(pruned(T,t))|}$$

Gde $\varepsilon(T,S)$označava stopu greške stabla T po uzorku S i $|leaves(T)|$ označava broj čvorova u T. $pruned(T,t)$ označava stablo dobijeno zamenom čvora t u T sa odgovarajućim listom. U drugoj fazi uopštavanja, greška svakog potkresanog drveta T_0, T_1, ..., T_k je procenjena. Najbolje potkresano stablo je potom odabrano. Ako je dati skup podataka dovoljno veliki, autori sugerišu razdvajanje na skup podataka za odlučivanje i skup potkresivanja. Stabla su kreirana korišćenjem skupa podataka za odlučivanje i izračunata na skupu potkresivanja. Sa druge strane, ako je dati skup podataka nedovoljno velik, predlog je da se koristi metoda unakrsne provere, bez obzira što se time povećava računarska složenost.

3.5.2 Potkresivanje umanjene greške

Jednostavna procedura za potkresivanje stabla odlučivanja, poznata kao potkresivanje, predložena od strane [Quinlan 1987]. U prolasku kroz unutrašnje čvorove sa dna prema vrhu, procedura proverava svaki unutrašnji čvor, da li zamenom sa najčešćom klasom ne smanjuje tačnost stabla. U ovom slučaju čvor je potkresan. Proces se nastavlja dok neko sledeće potkresivanje ne smanji tačnost. Da bi procenili tačnost, [Quinlan 1987] predlaže da se koristi skup potkresivanja. Može se pokazati da se ovaj proces zaustavlja sa najmanjim tačnim podstablom sa poštovanjem datog skupa potkresivanja.

3.5.3 Potkresivanje najmanje greške

Potkresivanje najmanje greške je predloženo u radu [Olaru & Wehenkel 2003]. Prolazi se kroz unutrašnje čvorove sabla sa dna na gore. U svakom čvoru upoređuje procenu 1-verovatnoće sa i bez potkresivanja. Procena 1-stope verovatnoće greške je ispravka jednostavne procene verovatnoće korišćenjem frekvencije. Ako S_t zanemaruje instance koje su stigle do lista t, onda je očekivana stopa greške u ovom čvoru :

$$\varepsilon'(t) = 1 - max_{c_i \epsilon dom(y)} \frac{|\{z \,|z \in S_t \,, C\,(z) = c_i\}| + l \cdot p_{apr}(y = c_i)}{|S_t| + l}$$

gde je $p_{apr}(y = c_i)$ apriori verovatnoća da y uzme vrednost c_i, i l zanemaruje težinu datu apriori verovatnoći. Stopa greške unutrašnjeg čvora je prosečna težina stope greške njegovih grana. Težina je određena prema proporciji instanci kroz svaku granu. Kalkulacija je sprovedena rekurzivno do čvorova. Ako je neki unutrašnji čvor potkresan onda postaje list i njegova stopa greške se računa direktno korišćenjem poslednje jednačine. Kao posledica prethodnog, možemo upoređivati stopu greške pre i posle potkresivanja određenog unutrašnjeg čvora.

Ako potkresivanje ovog čvora ne poveća stopu greške, potkresivanje treba da bude primenjeno.

3.5.4 Pesimistično potkresivanje

Pesimitično potkresivanje izbegava potrebu za potkresanim skupom ili unakrsnom proverom i umesto toga koristi pesimistične statističke uzajamne testove [Quinlan 1993]. Osnovna ideja je da stopa greške procenjene korišćenjem skupa podataka za formiranje stabla nije dovoljno pouzdana. Umesto nje, treba da se koristi realnija mera, poznata kao kontinualna korekcija za binomnu raspodelu:

$$\varepsilon'(T,S) = \varepsilon(T,S) + \frac{|leaves(T)|}{2 \cdot |S|}$$

Ova korekcija i dalje proizvodi optimističnu stopu greške. Posledično, treba da uzmemo u obzir potkresivanje unutrašnjeg čvora t ukoliko njegova stopa greške upada u jednu standardnu grešku stabla na koje referiše, naime [Quinlan 1993]:

$$\varepsilon'(pruned(T,S),S) \leq \varepsilon'(T,S) + \sqrt{\frac{\varepsilon'(T,S) \cdot (1 - \varepsilon'(T,S))}{|S|}}$$

Poslednji uslov je zasnovan na statističkom intervalu poverenja za proporciju. Obično se poslednji uslov koristi tako da T referiše na podstablo čiji je koren unutrašnji čvor t i S predstavlja deo skupa podataka za formiranje stabla koji se odnosi na čvor t. Proces pesimističnog potkresivanja primenjuje prolaz kroz unutrašnje čvorove sa vrha na dole. Ako je neki unutračnji čvor potkresan, onda se svi njegovi potomci uklanjaju iz procesa potkresivanja, rezultujući veoma brzim potkresivanjem stabla.

3.5.5 Potkresivanje zasnovana na grešci

Potkresivanje zasnovano na grešci je varijanta pesimističnog potkresivanja. Implementirano je u poznatom C4.5 algoritmu. Kao i kod pesimističnog potkresivanja, stopa greške se procenjuje korišćenjem gornje granice statističkog intervala pouzdanosti za proporciju.

$$\varepsilon_{UB}(T,S) = \varepsilon(T,S) + Z_\alpha \cdot \sqrt{\frac{\varepsilon(T,S) \cdot (1 - \varepsilon(T,S))}{|S|}}$$

gde $\varepsilon(T,S)$ predstavlja grešku podele stabla T skupa podataka za formiranje stabla S. Z je suprotno od standardne normalne raspodele i α je željeni nivo značaja. Neka $subtree(T,t)$ predstavlja podstablo sa korenom u čvoru t. Neka

je $maxchild(T,t)$ najfrekventnije čvor-dete čvora t (naime većina instanci u S stiže do ovog konkretnog deteta) i neka S_t predstavlja sve instance u S koje dosežu do čvora t. Proces primenjuje prolaz kroz sve čvorove sa dna ka vrhu i upoređuje sledeće vrednosti:

1. $\varepsilon_{UB}(subtree(T,t)\ ,S_t)$.
2. $\varepsilon_{UB}(pruned(subtree(T,t),t)\ ,S_t)$.
3. $\varepsilon_{UB}\big(subtree(T,maxchild(T,t))\ ,S_{maxchild(T,t)}\big)$.

Prema najmanjoj vrednosti proces ili ostavlja stablo u obliku u kojem je sad, potkresuje čvor t ili zamenjuje čvor t podstablom sa korenom u $maxchild(T,t)$.

3.5.6 Upoređivanje metoda potkresivanja

Nekoliko studija pokušavaju da uporede performanse različitih tehnika potkresivanja [Quinlan 1987], [Mingers 1989], [Esposito i sar. 1997]. Rezultati pokazuju da neke metode (kao što su potkresivanje složenosti troška, potkresivanje umanjene greške) teže da preterano potkrešu stablo, na primer prave manje stablo odlučivanja, koje je manje tačno. Druge metode (kao metod potkresivanja zasnovan na grešci, potkresivanje pesimistične greške i potkresivanje najmanje greške) teže da nedovoljno potkrešu stablo. Većina uporednih zaključuje da je teorema besplatnog ručka (eng. “no free lunch”) primenljiva u ovom slučaju: nema metode potkresivanja koja je u bilo kom slučaju bolja od ostalih metoda potkresivanja.

3.6 Drugi problemi

3.6.1 Težina instanci

Neki mehanizmi za formiranje stabla odlučivanja mogu dati različiti tretman različitim instancama. Ovo nastaje prilikom merenja doprinosa svake od instanci u analizi prema datoj težini (između 0 i 1).

3.6.2 Trošak pogrešne podele

Nekim mehanizmima za formiranje stabala odlučivanja moguće je dati numeričke kazne za pogrešno određivanje klase nekoj instanci.

3.6.3 Rukovanje vrednostima koje nedostaju

Vrednosti koje nedostaju su čest slučaj u realnim skupovima podataka. Ova situacija može iskomplikovati formiranje stabla (nepotpun skup podataka za formiranje stabla gde neke od vrednosti nedostaju), kao i klasifikaciju (nova instanca kojoj nedostaju određene vrednosti). Ovaj problem je razrađivan od strane nekoliko istraživača [Friedman 1977], [Breiman i sar. 1984], [Quinlan 1989]. Možemo obraditi vrednosti koje nedostaju u skupu podataka za formiranje stabla na sledeći način: neka $\{z \,|z \in S\,, C_x\,(z) = ?\}$ predstavlja podskup instanci u S čije vrednosti x fale. Kada računamo kriterijum podele koji koristi x, jednostavno zanemarujemo sve instance kod kojih je vrednost u atributu x nepoznata, umesto da se koristi kriterijum deljenja $\Delta\mathrm{F}(x, S)$ koristi se $\Delta\mathrm{F}(x, S\backslash\{z \,|z \in S\,, C_x\,(z) = ?\}\,)$. Sa druge strane, u slučaju vrednosti koje nedostaju, kriterijum deljenja treba da bude smanjen (pojednostavljen) proporcionalno, ako ništa ne može da se nauči iz ovih instanci [Quinlan 1989]. Drugim rečima, umesto da koristimo kriterijum deljenja $\Delta\mathrm{F}(x, S)$ koristimo sledeću ispravku:

$$\frac{|S\backslash\,\{z \,|z \in S\,, C_x\,(z) = ?\}\,|}{|S|}\Delta\Phi(x, S\backslash\,\{z \,|z \in S\,, C_x\,(z) = ?\}\,)$$

Pristup poznat kao "zamenjena podela" implementiran je u CART algoritmu [Breiman i sar. 1984] . Ideja je naći za svaku podelu u stablu podelu koja je može zameniti i koja koristi različite ulazne atribute, a koja najviše liči na originalnu podelu. Ako vrednost ulaznog atributa korišćena u originalnoj podeli nedostaje, onda je moguće koristiti podelu koja je može zameniti. Sličnost između 2 binarne podele u uzorku S je formalno definisana kao:

$$res(x, dom1(x), dom2(x), x', dom1(x'), dom2(x'), S) =$$

$$\frac{|\{z \,|z \in S\,, C_x\,(z) \in dom1(x)\,, C_{x'}\,(z) \in dom1(x')\}|}{|S|}$$
$$+\frac{|\{z \,|z \in S\,, C_x\,(z) \in dom2(x)\,, C_{x'}\,(z) \in dom2(x')\}|}{|S|}$$

Kada se prva podela odnosi na atribut x i njegove podele $dom(x)$ na $dom1(x)$ i $dom2(x)$. Rezervna podela se odnosi na atribut x' i deli njegov domen na $dom1(x')$ i $dom2(x')$.Nedostajuća vrednost može biti procenjena prema drugim instancama [Loh & Shih 1997]. U fazi učenja, ako vrednost nominalnog atributa x u torci q nedostaje, onda se procenjuju prema njegovom obliku sve instance koje imaju istu ciljnu vrednost atributa. Formalno

$$estimate(x, y_q, S) = argmax_{v_j \epsilon dom(x)} \left| \{ z \mid z \in S, C_x(z) = v_j, C(z) = y_q \} \right|$$

gde y_q predstavlja vrednost ciljnog atributa u torki q.

4. Razvojni okvir Apache Spark

Veliki skupovi podataka i nauka o podacima (eng. data science) kao termini i oblasti više ne predstavljaju samo aktuelnost o kojoj svi pričaju – veliki broj korisnika je shvatio prednosti koje donose i uočio potrebu za ovim alatima. Često ti alati mogu biti komplikovani za korišćenje i održavanje, bez neke velike potrebe za tim. U ovom radu korišćen je alat kojem raste popularnost i ušao u široku upotrebu zbog lakoće korišćenja i sjajnih performansi i funkcionalnosti koje donosi za rad sa podacima. Dodatna prednost mu je i zajednica koji svakog dana radi na unapređenjima, kao i odlična dokumentacija. Alat za brzu i efikasnu obradu Velikih skupova podataka je Apache Spark [Apache Spark]. Apache Spark je projekat otvorenog koda koji se razvija pod pokroviteljstvom Apache fondacije i jedan je od popularnijih alata za rad sa Velikim skupovima podataka. Razlog za to leži u filozofiji koja stoji iza projekta:

- Jedinstvena mašina za razvoj klijent-server aplikacija za obradu podataka, bilo da se radi o paketnoj obradi (eng. batch), kontinualnom prenosu (eng. streaming) ili interaktivnoj obradi podataka;
- Razvoj korišćenjem bogatih i jednostavnih aplikativnih programskih interfejsa (API), koji će podržati značajne optimizacije u pogledu performansi;
- Mogućnost integracije sa različitim sistemima gde se čuvaju podaci (imajući u vidu da je premeštanje podataka između sistema skupa operacija) i integracija sa različitim komponentama.

Spark omogućava korišćenje računarskih resursa grupe računara za izvršavanje složenih i zahtevnih poslova pri radu sa podacima. Podatke predstavlja kroz strukture podatak kao što su otporni distribuirani skupovi podataka (eng. resilient distributed datasets (RDD)), okviri podataka (eng. data frames) i skupovi podataka (eng. data sets). Pored memorije, podaci se mogu skladištiti i na disku, i ta odluka je prepuštena programeru. Pri obradi podataka ove strukture se mogu čuvati u radnoj memoriji, što značajno povećava performanse u obradi. Strukture podataka u Sparku su nepromenljive, pa tako svaka operacija nad nekom strukturom generiše novu strukturu. U prethodnom periodu uloženi su veliki napori u razvoj Sparkovih komponenti koje se koriste u obradi podataka. Neke od njih su:

- *Spark Streaming* - omogućava razvoj aplikacija za rad sa podacima koji stižu u toku (eng. streaming). Ima mogućnost automatskog oporavka od otkaza, ako se desi da neka mašina u grupi otkaže, sav posao će se adekvatno rasporediti na druge mašine i neće doći do gubitka podataka.

- *Spark SQL* - omogućava korišćenje SQL upita u istraživanju i obradi podataka, kao i korišćenje u kombinaciji sa drugim komponentama. Korišćenje okvira podataka (eng. data frame) omogućava jednostavan pristup različitim izvorima podataka, kao što su JSON, Parquet, Hive ili relacione baze podataka kroz JDBC driver. Omogućava da se upiti korišćeni u Hive strukturi bez izmene koriste i na Sparku. Kroz JDBC ili ODBC driver omogućava različitim alatima poslovne inteligencije da koriste podatke kroz Spark.

- *MLlib* - zadužena je za razvoj algoritama mašinskog učenja na Sparku i nad Velikim skupovima podataka. Omogućava implementaciju velikog broja poznatih algoritama iz ove oblasti, kao što su klasifikacija, regresija, klasterovanje i mnogi drugi. Spark se posebno ističe po performansama na ovom polju budući da razvijeni algoritmi postižu izuzetnu paralelizaciju, pa je omogućeno efikasno učenje modela nad Velikim skupovima podataka, i korišćenje modela za potrebe različitih aplikacija.

- *GraphX* - omogućava efikasan razvoj grafovskih algoritama i izvršavanje iterativnih graf-obrada. Veliki broj često korišćenih algoritama je već implementiran u okviru biblioteke, poput, na primer, Page Rank algoritma. Jednostavnost razvoja Apache Spark aplikacija se ogleda kroz veoma bogate aplikativne programske interfejse (API) koji su podržani u programskim jezicima Skala (eng. Scala), Pajton (eng. Python), Java i R. Pri implementacijama aplikativnih programskih interfejsa (API) su zadržani neki osnovni koncepti iz ovih programskih jezika koji mogu pomoći pri obradi podataka. Ako se razvija neki algoritam mašinskog učenja koristeći MLlib komponentu u PySparku (Python API za Spark), kalkulacije će se izvršavati koristeći NumPy biblioteku koja je veoma efikasna za ovakve zadatke.

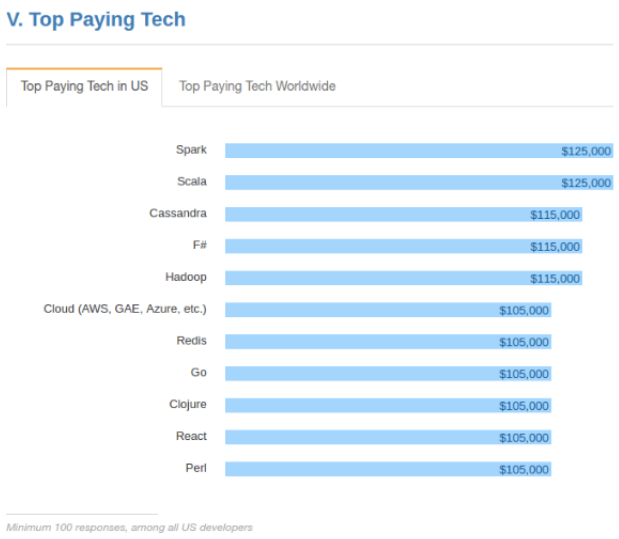
4.1 Primene Apache Spark razvojnog okvira

Ako je deo podataka smešten u JSON fajlovima, deo se nalazi u relacionoj bazi, a deo se nalazi u logovima, Spark je veoma jednostavno rešenje da se ti podaci integrišu, agregiraju i dalje koriste, budući da Spark podržava različite izvore podataka. Još jedan slučaj korišćenja je već pomenuta obrada podataka toka podataka. Spark je veoma zgodno rešenje za ovakve zadatke, i koristan je kada je potrebna obrada, na primer Twitter podataka dobijenih u vidu toka podataka. Budući da se lako integriše sa različitim sistemima za razmenu poruka, kao što je Apache Kafka [Apache Kafka], u Spark aplikaciju za rad sa tokom podataka se mogu ubacivati raznovrsni podaci iz različitih izvora. Spark je dobro rešenje

za specifične (eng. ad hoc) analize, istraživanje podataka i interaktivnu obradu. U takvim situacijama najčešće se koristi Jupyter Notebook [Jupyter Notebook] kao razvojno okruženje koje omogućava interakciju sa podacima. Čuvanje podataka u memoriji omogućava veoma brz odgovor na upite i postiže odlične performanse. Kada se konsoliduju podaci, istraže se i potrebno je da se uoče neke zakonitosti koje su u njima sadržane, Spark MLlib je odlično rešenje za razvoj i treniranje modela mašinskog učenja.

4.2 Istraživanje na veb - sajtu Stack Overflow

O popularnosti i značaju koji Spark ima u svetu nauke o podacima govori i ovogodišnje Stack Overflow istraživanje o stanju u programerskoj oblasti [Developer survey result 2016]. Spark se nalazi na drugoj poziciji po popularnosti u okviru Trending Tech sekcije. Što je možda i značajnije, Spark se nalazi na prvoj poziciji u okviru sekcije tehnologija koje su najplaćenije u Americi, zajedno sa Skalom, a na svetskom nivou je četvrta najplaćenija tehnologija. Na slici 4.1 prikazano je rangiranje tehnologija prema plaćenosti sprovedeno u 2016. godini preuzeto sa [Developer survey result 2016].



Slika 4.1: Rangiranje tehnologija prema plaćenosti

4.3 Otporni distribuirani skupovi podataka (RDD)

Osnovu Sparka čine otporni distribuirani skupovi podataka (eng. Resilient Distributed Datasets), odnosno RDD. RDD predstavlja osnovnu apstrakciju memorije u Sparku, koja programerima omogućava izvršavanje računskih

operacija nad podacima unutar velikih grupa računara koristeći njihovu memoriju, i pritom čuva svojstvo tolerancije greške. Ukoliko neka od mašina u klasteru ne može da završi neki zadatak ili ima nekih problema sa hardverom, samo deo posla koji je bio na toj mašini će se ponovo izvršiti na drugoj mašini, bez uticaja na zadatke koji se izvršavaju na mašinama koje pravilno funkcionišu. RDD predstavljaju particionisane kolekcije objekata rasprostranjene u klasteru, koje se čuvaju u memoriji ili na disku. Neka od osnovnih svojstava RDD su:

Nepromenljivost (eng. immutability) – predstavljaju strukturu podataka koja se ne može izmeniti. Prilikom izvršavanja neke operacije nad RDDovima koja zahteva njihovu izmenu ili generisanje neke nove promenljive, dobija se novi RDD. Na taj način više RDD komponenti oslikava različite verzije seta podataka, pa se dobija promenlivo (eng. "mutable") svojstvo.

Rod (eng. lineage) – za svaki RDD se čuvaju podaci o tome kako je isti dobijen. Ukoliko tokom izvršavanja programa dođe do otkaza neke mašine u grupi, RDDovi se mogu ponovo preračunati od nule, a da to pritom ne utiče na RDDove koji se nalaze na ostalim mašinama. Omogućeno je da ukoliko dođe do gubitka podataka iz bilo kojih razloga tokom izvršavanja nekog programa, ti isti podaci se mogu ponovo efikasno preračunati. Pored toga, rod omogućava lenjo izračunavanje Spark transformacija podataka.

Tolerancija greške (eng. fault tolerance) – omogućena je kroz logovanje svih izmena nad skupovima podataka, odnosno pojedinačnim izmenama koje se izvršavaju nad mnoštvom zapisa. Postiže se kroz rod i logove pisane unapred (veoma značajni za Spark obradu tokova podataka). RDD se može kreirati učitavanjem podataka iz nekog sistema za skladištenje podataka, poput HDFSa ili nekog drugog, transformacijom nekog postojećeg RDD, ili pozivanjem funkcije *parallelize* nad nekom listom podataka u Pajtonu (ukoliko se koristi Python API za Spark, poznatiji kao PySpark).

Dva tipa RDD operacija je podržano:

1. *transformacije* - osnovna karakteristika transformacija je lenjo izračunavanje, što znači da kada se pozove neka transformacija nad nekim RDD, ništa se ne dešava odmah. Zapravo, program tada samo pamti šta je potrebno uraditi, a sama transformacija će se izvršiti tek kada na nju naiđe neka akcija. Izvršavanjem transformacije se dobija novi RDD. Najčešći primeri transformacija su funkcije *map* koja transformiše element skupa po prethodno definisanoj funkciji, *filter* koja iz skupa uklanja elemente koji zadovoljavaju prethodno definisan uslov, *reduce* koja objedinjuje elemente

supa po prethodno definisanoj funkciji, *distinct* koja uklanja element iz skupa ako se ponavlja, itd.

2. *akcije* - Akcija je u Sparku operacija koja se izvšava odmah. Pozivanjem akcije izvršavaju se i sve transformacije nad podacima koje su prethodno pozvane. Predstavljaju mehanizam kojim se uzimaju podaci iz Sparka. Akcije uvek imaju neku povratnu informaciju za krajnjeg korisnika. Najčešći primeri akcija su funkcije *max* vraća najveći element skupa, *min* vraća najmanji element skupa, *count* vraća broj elementata u skupu, *first* vraća prvi element iz skupa, *collect* transformiše RDD u strukturu List definisanu u Javi.

Često korišćena funkcija u Sparku je *cache()*, koja čuva u memoriji RDD nad kojim je pozvana. Na taj način korisnik koji razvija program obezbeđuje da mu se neki podaci, koje će kasnije koristiti, čuvaju u memoriji, kako se ne bi morali ponovo učitavati sa diskova. Dodatna literatura o RDD je dostupna u radu [Matei i sar. 2012] sa Berkli univerziteta, gde je Spark i razvijen.

4.4 Spark okviri podataka

Okvir podataka (eng. data frame) predstavlja često korišćenu apstrakciju podataka u mnoštvu programskih jezika za rad sa podacima. Predstavljaju strukturu podataka poput tabele ili matrice, gde se u svakoj koloni čuvaju merenja neke promenljive, a svaki red predstavlja jednu opservaciju, odnosno jedan zapis. U R jeziku predstavljaju jednu od osnovnih struktura podataka, a u Pajtonu se mogu koristiti kroz Pandas bibilioteku. Spark okvir podataka predstavlja distribuiranu kolekciju podataka koji su organizovani kao tabela ili matrica. Po nekim osnovnim karakteristikama su slični RDDovima, jer se takođe mogu čuvati u memoriji, i podržavaju lenjo izračunavanje. Osnovna razlika između njih se ogleda u tome što Spark može optimizovati operacije nad okvirima podataka, jer svaki okvir podataka sadrži metapodatke o tipovima podataka koji se nalaze u kolonama, što nije slučaj kod RDDa. Spark okvir podataka se može kreirati na više načina, poput učitavanja strukturiranih fajlova sa podacima (poput CSV i JSON fajlova), iz eksternih baza, od postojeće Hive tabele (ukoliko se koristi u kombinaciji sa Hadupom), transformacijom RDDa, itd. Korišćenjem okvira podataka u Spark programu omogućene su sve "konvencionalne" operacije nad podacima koje bismo imali u nekoj relacionoj tabeli, kao što su sečenje tabele, sortiranje redova, agregacije, spajanje sa drugim okvirima podataka, … Okvir podataka kao koncept postoji u Sparku od verzije 1.3. Poput RDDa, i okviri podataka podržavaju lenjo izračunavanje, čime se smanjuju stanja čekanja prilikom izvršavanja programa i omogućava bolji tok procesa. Za rad sa njima je moguće pisati DSL(eng. domain-specific language)

jezik u Javi, Skali ili Pajtonu. Pored toga, njihovim uvođenjem omogućeno je pisanje SQL upita u Spark programu, koristeći klasu SQLContext. Veoma lako se mogu integrisati sa Pandas okvirima podataka ukoliko se koristi PySpark, a na taj način i sa ostalim Pajton bibliotekama za rad sa podacima. Benefiti koji se postižu korišćenjem Spark okvira podataka se odnose na poboljšanje performansi izvršavanja programa i fleksibilnost manipulacije podacima.

4.5 Primer programa u Spark tehnologiji

Program ispod pronalazi broj ponavljanja svake od reči u datoj datoteci.

Algoritam 4.1: Spark program brojanja reči u tekstu

```
public class WordCountSpark
{
        public static void main(String[] args)
         {
                SparkConf sparkConf = new SparkConf()
                                        .setAppName("WordCountExample")
                                        .setMaster("local[2]")
                                        .set("spark.executor.memory",  "1g");
                JavaSparkContext sc = new JavaSparkContext(sparkConf);
                JavaRDD<String> textFile = sc.textFile(".\Text.txt");
                JavaPairRDD<String, Integer> counts = textFile
                .flatMap(s -> Arrays.asList(s.split(" ")).iterator())
                .mapToPair(word -> new Tuple2<>(word, 1))
                .reduceByKey((a, b) -> a + b);
                counts.saveAsTextFile(".\WordCount.txt");
        }
}
```

5. Korišćeni algoritmi

U nastavku je pregled nekih od poznatijih algoritama za generisanje stabala odlučivanja.

- ID3 - smatra se veoma jednostavnim algoritmom stabla odlučivanja [Quinlan 1986];
- C4.5 – predstavlja poboljšanje ID3 [Quinlan 1993];
- CART - predstavlja klasifikaciona i regresiona stabla [Breiman i sar. 1984];
- QUEST - algoritam podržava podele po vrednosti jednog atributa i podele linearne kombinacije [Loh & Shih 1997];
- CAL5 - konstruisan specifično za atribute numeričkih vrednosti [Muller & Wysotzki 1994];
- FACT - ranija verzija QUEST-a [Loh & Vanichsetakul 1988];
- LMDT - konstruiše stablo odlučivanja prema testovima podele po vrednostima više atributa i linearnoj kombinciji atributa. [Brodley & Utgoff 1995];
- T1 - stablo odlučivanja jednog nivoa koje deli instance korišćenjem samo jednog atributa. [Holte 1993];
- PUBLIC - kombinuje rast i pročišćavanje korišćenjem MDL troška da bi umanjio računsku složenost [Rastogi & Shim 2000];
- MARS - funkcija više regresija se aproksimira korišćenjem linearnog splajna i njihovog tenzorskog proizvoda. [Friedman 1991]

U ovom radu će biti razmatrani ID3, C4.5 i CART algoritam. Vršiće se uporedna analiza procenta tačnosti algoritama da bi se otkrio faktor rizika od bolesti kod pripadnika starije populacije. Svaki od algoritama ima sekvencijalnu implementaciju realizovanu u programskom jeziku Java. ID3 i C4.5 algoritam su paralelizovani korišćenjem Spark tehnologije. CART algoritam je paralelizovan kojišćenjem niti u programskom jeziku Java.

5.1 Sekvencijalna implementacija ID3

ID3 koristi informacionu dobit kao kriterijum podele. Rast stabla prestaje kada sve instance pripadaju jednoj vrednosti ciljnih karakteristika ili kada najbolji faktor informacione dobiti (eng. information gain) nije veći od 0. ID3 ne primenjuje nijedan proces potkresivanja, takođe ne obrađuje numeričke atribute niti nedostajuće vrednosti.

Algoritam 5.1 : Sekvencijalna implementacija algoritma ID3

```
StabloOdlucivanja
Ulaz : primeri, vrednostiCiljnogAtribut, atributi;
if primeri is empty or atributi is empty then
        return null ;
ucestalost = icestalost (primeri, vrednostiCiljnogAtributa);
if svi primeri imaju istu vrednostCiljnogAtributa then
        return cvor(primeri.vrednostCiljnogAtributa);
else
        najboljiAtribut = najboljiAtribut(primeri, atributi);
        if najboljiAtribut = null then
                return cvor(najcescaVrednostCiljnogAtributa);
        stablo = cvor(najboljiAtribut);
        for each vrednost v of najboljiAtribut do
            primeri = primeriSaVrednoscuAtributa(primeri, atribut, v);
            podstablo = stabloOdlucivanja(primeri, atributi - najboljiAtribut);
            dodajPodstabloZaVrednost(stablo, podstablo, v);
 return stablo ;
```

Budući da svi atributi algoritma imaju samo 2 vrednosti stablo koje se dobija kao rezultat je binarno. Svaki čvor stabla može imati najviše 2 čvora deteta. Levi čvor je čvor negativnog ishoda, a desni čvor je čvor pozitivnog ishoda. Kada se proverava da li je čvor koji treba kreirati list, potrebno je odrediti koliko elemenata skupa pripada svakoj od vrednosti ciljnog atributa. Funkcija *ucestalost* računa koliko elemenata skupa pripada svakoj od vrednosti atributa. Za brojanje elemenata skupa u algoritmu koriste se mape koje kao ključ imaju vrednosti ciljnog atributa a vrednost je broj elemenata skupa kod kojih je vrednost ciljnog atributa odgovara vrednosti u ključu mape. Na početku se za sve vrednosti ciljnog atributa dodaje ključ u mapu i dodeljuje mu se vrednost 0. Zatim se prolazi kroz skup elemenata, proverava se vrednost ciljnog atributa i u mapi se vrednost odgovarajućeg ključa povećava za 1. Ako samo jedan od ključeva u mapi ima vrednost veću od 0, algoritam vraća čvor list čiji je atribut ciljna vrednost kojoj svi elementi skupa pripadaju.

Funkcija *najboljiAtribut* računa informacionu dobit za svaki od atributa i kao rezultat vraća onaj atribut koji ima najveću vrednost informacione dobiti. Maksimalna informaciona dobit postavlja se na 0. Informaciona dobit se računa za svaki atribut posebno. Za određivanje informacione dobiti potrebno je izračunati učestalost vrednosti ciljnog atributa po vrednosti atributa za koji se računa informaciona dobit. Isto pravimo mapu sa ključevima vrednosti ciljnog atributa a vrednosti se postavljaju na 0. Prolazi se kroz skup elemenata ako se vrednost atributa trenutnog elementa poklapa sa vrednošću atributa za koju se računa učestalost u mapi uvećavamo za 1 vrednost ključa koji se poklapa sa vrednošću ciljnog atributa trenutnog elementa. Postupak se ponavlja za svaku od vrednosti atributa koji se trenutno obrađuje. Zatim se računa informaciona dobit

po formuli, prethodno navedenoj u odeljku 3.2.2 i proverava se da li je informaciona dobit atributa veća od maksimalne do tad izračunate. Ukoliko je uslov za maksimum ispunjen, postavlja se vrednost za novi maksimum i atribut se čuva kao najbolji za podelu. Zatim se prolazi kroz sve vrednosti najboljeg atributa. Pravi se novi skup elemenata, podskup skupa svih elemenata. U podskup se smeštaju oni elementi za koje važi da je vrednost najboljeg atributa jednaka trenutnoj vrednosti. Rekurzivno se poziva algoritam za podskup elemenata i listu atributa iz koje je izbačen najbolji atribut. Rezultat rekurzivnog poziva se smešta u čvor dete. Pošto je dobijeno stablo binarno, u svakom čvoru koji nije list rade se 2 rekurzivna poziva (levo i desno dete). Kao rezultat algoritma vraća se čvor koji je dobijen na osnovu najboljeg atributa.

5.2 Paralelna implementacija ID3

Kod ID3 algoritma u ovom radu je implementirana paralelizacija računanja učestalosti i traženje najboljeg atributa za podelu. Za paralelizaciju je korišćena Spark tehnologija i RDD struktura podataka. U nastavku je prikazano i objašnjeno na koji način je paralelizovano računanje učestalosti vrednosti ciljnog atributa.

```
ucestalost = primeri.mapToPair(i -> new Tuple2<>(i.vrednostCiljnogAtributa(),
1)).reduceByKey((a, b) -> a+b)
```

Funkcionalnosti koje su korišćene za paralelizaciju učestalosti su *mapToPair*, *reduceByKey*. Brojanje elemenata skupa prema vrednosti ciljnog atributa je paralelizovano tako što se svaki element preslikava u par (vrednost ciljnog atributa, 1) funkcijom *mapToPair*. Preslikavanje je ostvareno korišćenjem lambda izraza za funkcije. Ulazni atribut lambda izraza je element strukture RDD a rezultat je struktura Tuple2 programskog jezika Skala koja čuva par vrednosti (vrednost ciljnog atributa elementa, 1) vrednost 1 predstavlja jedno pojavljivanje (preslikava se jedan element skupa u par vrednosti). Funkcija *mapToPair* kao ulaz prima prethodno definisan lambda izraz. Primenjujemo je na RDD strukturu elemenata, a kao rezultat vraća RDDPair strukturu čiji je svaki element Tuple2 oblika (vrednost ciljnog atributa elementa, 1). Time se dobijaju sva pojavljivanja svih vrednosti ciljnog atributa u skupu elemenata. Potrebno je izračunati ukupan broj pojavljivanja svake od vrednosti ciljnog atributa. To se postiže primenom funkcije *reduceByKey* na rezultat primene funkcije *mapToPair*. Funkcija *reduceByKey* primenjuje neku agregaciju vrednosti skupa elemenata koji imaju istu vrednost ključa. Agregacija koja se primenjuje nad vrednostima šalje se kao ulazni atribut funkciji *reduceByKey*. Agregacija koja je korišćena je sumiranje vrednosti (svako ponavljanje je preslikano u vrednost 1,

sumiranjem svih vrednosti 1 dobijamo broj ponavljanja svake od vrednosti ciljnog atributa). Sumiranje vrednosti je takođe definisano pomoću lambda izraza. Lambda izrazi su u programskom jeziku Java definisani od verzije 8. Sličan princip je primenjen kod traženja najboljeg atributa za podelu u funkciji *najboljiAtribut*. U nastavku je fragment koda i objašnjenje.

```
ucestalost = primeri.mapToPair(i -> new Tuple2<>((
(i.getUserFeatures()[trenutniAtribut.getIdx()] == trenutniAtribut.getfOptions()[0])? "0" +
i.getOutputIndicator() : "1" + i.getOutputIndicator()) , 1)) .reduceByKey((a, b) -> a + b);
```

Koristi se funkcija *mapToPair,* element skupa se preslikava u par vrednosti, čiji je ključ oblika „0“ + vrednostCiljnogAtributa ili „1“ + vrednostCiljnogAtributa. „0“ i „1“ predstavljaju preslikane vrednosti atributa za koji se računa informaciona dobit. Ako je vrednost atributa za koju se računa informaciona dobit jednaka „true“ preslikava se u „1“, inače preslikava se u „0“. Na preslikanu vrednost atributa za koji se računa informaciona dobit nadovezuje se vrednost ciljnog atributa. Svaki element se preslikava u par vrednosti (nadovezanString, 1). Na taj način jednostavno se određuju svi koeficijenti potrebni za računanje informacione dobiti atributa. Za računanje inforacione dobiti bitan je broj pozitivnih i broj negativnih vrednosti ciljnog atributa za svaku od vrednosti atributa za koju se računa informaciona dobit. Na rezultat funkcije *mapToPair* primenjuje se funkcija *reduceByKey* na isti način kako je gore opisano da bi se dobio broj primera za svaku kombinaciju vrednosti atributa za koji se računa informaciona dobit i vrednosti ciljnog atributa. Formula po kojoj se računa informaciona dobit navedena je u odeljku 3.2.2 ovog rada. U jednom koraku je podeljen skup elemenata po vrednosti atributa i po vrednosti ciljnog atributa. U Sparku nije dozvoljeno pozivanje funkcije *map* unutar drugog poziva funkcije *map*. Prema tome, da bi se prepoznao atribut sa najvećom vrednošću informacione dobiti, prolazi se kroz listu atributa for ciklusom i traži se maksimalna vrednost informacione dobiti. Rezultat koji vraća funkcija je atribut sa najvećom informacionom dobiti.

5.3 Sekvencijlna implementacija C4.5

C4.5 je evolucija (poboljšanje) ID3, predstavljeno od strane istog autora [Quinlan 1993]. Koristi odnos dobiti kao kriterijum podele. Podela prestaje kada je broj elemenata za podelu ispod određenog praga. Potkresivanje zasnovano na grešci se primenjuje nakon faze rasta. C4.5 može da radi sa numeričkim atributima. Može da zaključuje na osnovu skupa podataka za formiranje stabla koji dozvoljava nedostajuće vrednosti korišćenjem kriterijuma ispravljenog odnosa dobiti definisanog u odeljku 3.2.5.

Algoritam 5.2: Sekvencijalna implementacija algoritma C4.5 potkresivanje (eng. pruning)

```
potkresi
Ulaz : r, testPrimeri;
if r = null or testPrimeri is empty then
        return null;
if r is leaf then
        return r;
for each dete of r.deca() do
        trenutniPrimeri = podskupKojiOdgovaraDetetu(testPrimeri, dete);
        novoDete = potkresi(dete, trenutniPrimeri);
        if novoDete is not null then
                zameni(dete, novoDete);
if r ima dete koje nije list then
        return r;
ucestalost = ucestalostCiljnogAtributa(testPrimeri);
maxUcestalost = max(ucestalost);
maxPoklapanja = poklapanjeCiljnihAtributaSaTestom(testPrimeri);
if maxPoklapanja > maxUcestalost then
        return r;
else
        return Cvor(maxUcestalost);
```

Funkcionalnost *podskupKojiOdgovaraDetetu* kao rezultat vraća podskup skupa *testPrimera* koji pripadaju detetu koje se šalje kao argument funkcije. Rekurzivno se poziva algoritam za dete i podskup *testPrimera*. Ako rezultat rekurzivnog poziva nije null dete se zamenjuje rezultatom rekurzivnog poziva. Funkcionalnost *ucestalostCiljnogAtributa* se računa na isti način kao i u ID3 algoritmu. Biramo onu ciljnu vrednost koja se najviše pojavljuje u test elementima (najveća učestalost).

Funkcionalnost *poklapanjeCiljnihAtributaSaTestom* vraća kao rezultat broj koji predstavlja broj poklapanja testne vrednosti ciljnog atributa i vrednosti ciljnog atributa *testPrimera*. Ako je broj poklapanja veći od najčešće vrednosti ciljnog atributa, čvor se ne menja, inače se pravi čvor list sa najčešćom vrednošću ciljnog atributa. Rekurzivni pozivi u algoritmu su pre same provere da li čvor treba zameniti, tako da algoritam ima pristup sa dna ka vrhu. Algoritam je implementiran prema ideji koja je definisana u odeljku 3.5.5 (potkresivanje zasnovano na grešci). Zaustavljamo se kad je broj poklapanja na trenutnom stablu (funkcija *poklapanjeCiljnihAtributaSaTestom)* veći od najčešće vrednosti ciljnog atributa.

Algoritam 5.3: Sekvencijala implementacija algoritma C4.5 građenje stabla

```
stabloOdlucivanja
Ulaz : ciljniAtribut, atributi, primeri;
if entropija(ciljniAtribut, primeri) = 0 then
        return Cvor(primeri.ciljnaVrednost);
if atributi is empty then
        return Cvor(NajcescaCiljnaVrednost(primeri));
najboljiAtribut = NajboljiAtribut(ciljniAtribut, atributi, primeri);
stablo = Cvor(najboljiAtribut);
for each vrednost v of najboljiAtribut do
        podPrimeri = podskup(primeri, vrednost);
        if podPrimeri is empty then
                list = Cvor(NajcescaCiljnaVrednost(primeri));
                dodajCvor(stablo, list);
        else
                dete = stabloOdlucivanja(ciljniAtribut,
                                         atributi-najboljiAtribut,
                                         podPrimeri);
                dodajCvor(stablo, list);
```

Svi atributi korišćeni u algoritmu tumače se kao nominalni atributi. Algoritam građenja stabla je manje-više sličan ID3 algoritmu. Razlikuje se u načinu računanja informacione dobiti. Nominalni atributi imaju onoliki broj dece koliko ima različitih vrednosti ciljnog atributa (u ovom radu to je 3). Numerički atributi imaju dva deteta. Kod numeričkih atributa uslovu podele dodeljuje se vrednost. Da bi se vršila podela po numeričkom atributu potrebno ga je diskretizovati, odnosno odrediti graničnu vrednost za funkciju podele. Vrednost za podelu se bira tako da atribut ima najveću informacionu dobit. Elementi se dele u dva skupa u odnosu na to da li ispunjavaju uslov podele ili ne. Informaciona dobit za atribute se računa tako što se sortira skup elemenata po vrednosti atributa za koji se računa informacina dobit. Zatim prolazimo kroz skup i računamo informacionu dobit, pod pretpostavkom da trenutni element deli skup elemenata. Postupak se ponavlja za svaki element skupa i čuva se pozicija i vrednost atributa onog elementa kod koga je postignuta najveća vrednost informacione dobiti.

5.4 Paralelna implementacija C4.5

U algoritmu C4.5 u fazi potkresivanja paralelizovano je izvršavanje funkcija:

- *podskupKojiOdgovaraDetetu*,
- *ucestalostCiljnogAtributa*,
- *poklapanjeCiljnohAtributaSaTestom*.

U nastavku je prikazan i objašnjen paralelizovan kod za određivanje podskupa.

```
trenutniPrimeri = primeri.filter(i -> !((tipAtributa.equals("continuous")
&& ((dete.substring(0, 4).equals("less")
&& Double.parseDouble(i.getAttributeValuePairs().get(imeAtributa))
< Double.parseDouble(dete.substring(4)))
||
(dete.substring(0, 4).equals("more")
&& Double.parseDouble(i.getAttributeValuePairs().get(imeAtributa))
>= Double.parseDouble(dete.substring(4))))
||
(!tipAtributa.equals("continuous")
&& dete.equals(i.getAttributeValuePairs().get(imeAtributa)))))
);
```

Za paralelizaciju funkcije *podskupKojiOdgovaraDetetu* koristi se transformacija *filter*. Funkcija *filter* kao rezultat vraća novi skup - podskup skupa elemenata na koji se primenjuje. Funkcija iz skupa na koji se primenjuje uklanja elemente koji zadovoljavaju uslov koji se šalje kao argument funkciji. Bira se podskup skupa *testPrimera* koji se vezuju za čvor dete čvora koji se analizira. Na različite načine se analiziraju numerički i nominalni atributi. Kod numeričkih atributa pravilo po kojem se deli skup čuva se u obliku "manje od vrednosti" ili "veće od vrednosti". Kada se proverava da li element skupa pripada odgovarajućem čvoru detetu upoređuje se vrednost atributa elementa i vrednost koja se čuva u pravilu. Poređenje se vrši po uslovu u pravilu i ako je uslov zadovoljen, smatra se da element pripada čvoru detetu, inače se uklanja iz skupa. Kod nominalnih atributa proverava se da li se vrednost čvora deteta poklapa sa vredošću atributa elementa. Ako se vrednosti ne poklapaju, element se uklanja iz skupa.

Rekurzivno se poziva algoritam za svako dete čvora i podskup. Ako rekurzivni poziv vrati rezultat koji nije null, čvor dete se zamenjuje odgovarajućim potkresanim čvorom. Nakon toga potrebno je odrediti učestalost ciljnog atributa. Implementacija tog dela algoritma je takođe paralelizovana.

```
Comparator<Tuple2<String, Integer>> comparator =
new Comparator<Tuple2<String, Integer>>() {
                        @Override
                        public int compare(Tuple2<String, Integer> x,
                                        Tuple2<String, Integer> y) {
                                if ((x._2).intValue()-(y._2).intValue() > 0)
                                return 1;
                                else if ((x._2).intValue()-(y._2).intValue() < 0)
                                return -1;
                                else return 0;
                        }
                };
ucestalost = primeri.mapToPair(i -> new Tuple2<> (i.getAttributeValuePairs()
.get(ProcessInputData.targetAttribute.getName()), 1))
.reduceByKey((a, b) -> a + b);
maxUcestalost = ucestalost.max(comparator);
```

Funkcija *ucestalostCiljnogAtributa* paralelizuje proces traženja najčešće vrednosti ciljnog atributa na skupu elemenata. Funkcije Sparka koje su korišćenje su *mapToPair*, *reduceByKey*, *max.* Svaki element skupa se funkcijom *mapToPair* preslikava u par (vrednostCiljnogAtributa, 1). Funkcija *reduceByKey* se primenjuje na rezultat funkcije *mapToPair* kako bi izvršila agregaciju nad vrednostima istog ključa. Sumira se vrednost 1 za svako ponavljanje da bi dobili ukupan broj ponavljanja za svaku od vrednosti ciljnog atributa. Pošto je potrebno izračunati onu vrednost ciljnog atributa koja ima najviše ponavljanja primenjuje se funkcija *max* koja kao argument prima komparator 2 elementa skupa da bi dobili element sa najvećim brojem ponavljanja.

Da bi se odredilo da li treba zameniti čvor najčešćom klasom potrebno je uporediti učestalost najčešće klase sa poklapanjem ciljnih atributa na skupu. Računanje broja poklapanja je takođe paralelizovano.

```
maxPoklapanja = testPrimeri.map(i ->
primeriGradjenjaStabla.filter(f -> !((i.getInstanceIndex()==f.getInstanceIndex()
&&
f.getAttributeValuePairs().get(imeCiljnogAtributa)
.equals(f.getAttributeValuePairs().get(testnaVrednostCiljnogAtributa))))
.count())
.reduce((a, b) -> a + b);
```

Funkcija *poklapanjeCiljnihAtributaSaTestom* je paralelizovana korišćenjem funkcija *map, filter, count, reduce*. Za svaki element skupa *testPrimeri* proverava se koliko je elemenata korišćenih za izgradnju stabla kod kojih se testna vrednost ciljnog atributa poklapa sa vrednošću ciljnog atributa *testPrimera*. Prvo se pronalazi podskup skupa elemenata za građenje stabla kod kojih se testna vrednost poklapa sa vrednošću *testPrimera* korišćenjem funkcije *filter* zatim se koristi funkciju *count* koja kao rezultat vraća broj elemenata skupa da bi se dobio broj poklapanja po tom *testPrimeru*. To se paralelno radi za sve

testPrimere korišćenjem funkcije *map* gde se svaki element skupa *testPrimeri* preslikava u broj poklapanja. Nakon toga se primenjuje funkcija *reduce* koja agregira skup vrednosti po funkciji koja se šalje kao argument funkciji *reduce* i kao rezultat vraća agregaciju. Za agregaciju se koristi funkcija sumiranja i kao rezultat se dobija ukupan broj pokalapanja na celom skupu *testPrimera* koji se kasnije koristi za poređenje da li treba zameniti čvor stabla ili ostaje nepromenjen.

U fazi izgradnje stabla u C4.5 algoritmu paralelizovane su funkcije:

- *najboljiAtribut*
- *podskup*.

Paralelizovano je računanje informacione dobiti za atribute. Ako je atribut nominalan prvo je paralelizovan proces podele skupa primera prema vrednosti atributa za koji se računa informaciona dobit.

```
rezultat = primeri.mapToPair(i ->
new Tuple2<>(i.vrednostAtributa(imeAtributa), napraviRDD(i, sc) )
.reduceByKey((a, b) -> a.union(b));
entropijaPodskup = rezultat.mapToPair(i ->
new Tuple2<>(i._1, i._2.count()/((double)velicinaSkupa)
*Entropy.calculate(ciljniAtribut, i._2)))
.values()
.reduce((a, b) -> a+b);
```

Paralelizacija procesa podele skupa elemata pomoću funkcija *mapToPair*, *reduceByKey* i *union*. Funkcija *mapToPair* primenjuje se na skup elemenata i preslikava svaki element skupa u par (vrednostAtributa, RDD(element)). Neophodno je da se svi elementi koji imaju istu vrednost atributa objedine u jedan skup elemenata. Na skup dobijen kao rezultat poziva funkcije *mapToPair* primenjuje se funkcija *reduceByKey* da bi se spojili svi elementi koji se slikaju u istu vrednost atributa u jedan skup. Za spajanje skupova koristimo funkciju *union* koja spaja dva skupa elemenata u jedan novi skup. Rezultat funkcije *reduceByKey* je skup parova (vrednostAtributa, podskupElemenata). Potom je paralelizovan proces računanja informacione dobiti za atribut korišćenjem funkcije *mapToPair* koja svaki par skupa dobijen primenom funkcije *reduceByKey* slika u par (vrednostAtributa, entropija). Potrebno je da se izračuna suma vrednosti entropije. Na tako dobijen skup parova primenjuje se funkcija *values* koja vraća skup vrednosti (entropija). Na dobijenom skupu vrednosti primenjuje se funkcija *reduce* kojoj se kao argument šalje funkcija za računanje sume. Rezultat primene funkcije reduce je ukupna vrednost entropije za atribut.

Kod numeričkih atributa paralelizovano je sortiranje skupa elemenata. Korišćena je funkcija *sortBy*. Skup elemenata se sortira po vrednosti atributa za koji se računa informaciona dobit.

```
primeri=primeri.sortBy(new Function< Instance, Double>() {
   private static final long serialVersionUID = 1L;

   @Override
   public Double call( Instance primer ) throws Exception {
      return Double.parseDouble(
            primer.getAttributeValuePairs().get(imeAtributa));
   }
}, true, 1 );
```

Zatim se prolazi kroz elemente skupa i traži se vrednost koja deli skup tako da atribut ima najveću vrednost informacione dobiti. Čuva se pozicija vrednosti koja je proizvela najveću informacionu dobit. Atribut će imati 2 podskupa skupa elemenata jedan su elementi čija je pozicija manja ili jednaka poziciji koja je sačuvana a drugi su elementi čija je pozicija veća od pozicije koja je sačuvana. Postupak se ponavlja za svaki numerički atribut. For ciklusom se prolazi kroz listu atributa i pamti se onaj koji ima najveću vrednost informacione dobiti. Atribut sa najvećom vrednosti informacione dobiti je povratni rezultat funkcije *najboljiAtribut*. Ako je atribut nominalan funkcija *podskup* kao rezultat vraća podskup elemenata koji je vezan za vrednost atributa koja se obrađuje koji smo već formirali u pozivu funkcije *najboljiAtribut.* Ako je atribut numerički funkcija *podskup* kao rezultat vraća jedan od podskupova elemenata koji su formirani na osnovu vrednosti po kojoj se skup deli koji je već formiran u pozivu funkcije *najboljiAtribut.*

5.5 Sekvencijalna implementacija CART

CART predstavlja klasifikaciona i regresiona stabla [Breiman i sar. 1984]. Karakteriše ih činjenica da se konstruiše binarno stablo - svaki unutrašnji čvor ima tačno 2 izlazna čvora. Podele su izabrane korišćenjem kriterijuma dupliranja i dobijeno stablo je potkresano pomoću potkresivanja složenosti troška. Kada je dostupno, CART može da uzme u obzir trošak pogrešne podele na indukovanom stablu. Bitna prednost CART-a je njegova mogućnost da generiše regresivna stabla. Regresivna stabla su stabla kod kojih čvorovi predviđaju tačan broj, ne klasu. U slučaju regresije, CART traži podele koje minimizuju predviđanje kvadratne greške (najmanje kvadratno odstupanje).

Algoritam 5.4: Sekvencijalna implementacija algoritma CART

```
StabloOdlucivanja
Ulaz: ciljneVrednosti, atributi, primeri, maxBrCvorova, primeriZaUcenje;
razliciteCiljneVrednosti = jedinstveno(ciljneVrednosti);
ucestalost = ucestalost (primeriZaUcenje, ciljneVrednosti);
redZaPodelu = prazanRed();
koren = cvor(max(ucestalost));
poboljsanje = cvorUcenje(koren, primeri, ciljneVrednosti, primeriZaUcenje);
if poboljsanje.imaPodelu() then
        dodajUPodelu(redZaPodelu, poboljsanje);
for i=1 to maxBrCvorova
        cvor = uzmiSledeci(redZaPodelu);
        if cvor = null then
                break;
        cvor.podeli(redZaPodelu);
```

Učestalost vrednosti ciljnog atributa na skupu elemenata računa se isto kao i u druga dva algoritma. Funkcija *cvorUcenje* koristi prethodno utvrđen broj atributa za koje se računa odnos dobiti definisan u odeljku 3.2.5, ako je broj atributa veći od utvrđenog broja bira se slučajni podskup i samo na tom podskupu se bira najbolji atribut (onaj koji ima najveći odnos dobiti). Ako broj atributa nije veći od utvrđenog broja računa se odnos dobiti za sve atribute i bira se najbolji. Stablo koje se gradi je binarno. Funkcija *imaPodelu* proverava da li postoji atribut sa najvećom gini dobiti i čuva atribut koji ima najveća gini dobit kao i vrednost podele koja se bira tako da atribut ima najveću Gini dobit. Formula za računanje Gini dobiti definisana je u odeljku 3.2.3. Kada je atribut nominalni, bira se vrednost tako da atribut ima najveću Gini dobit. Skup elemenata se deli po uslovu da li je vrednost atributa jednaka vrednosti po kojoj se atribut deli ili ne. Kod numeričkih atributa takođe se bira vrednost tako da atribut ima najveću gini dobit. Skup elemenata se deli po tome da li je vrednost atributa manja od vrednosti podele ili ne. Prilikom podele skupa elemenata za svaki čvor dete bira se najčešća ciljna vrednost atributa na podskupu elemenata, zatim se proverava da li postoji *cvorUcenje* koji bi ga zamenio i ukoliko postoji smešta se čvor u red za podelu. Nastavlja se sve dok ima čvorova učenja ili dok stablo ne dostigne dovoljan broj čvorova.

5.6 Paralelna implementacija CART

Implementacija algoritma je preuzeta sa Github platforme otvorenog koda [CART Github]. Implementacija CART algoritma je značajno složenija i različita u odnosu na algoritme ID3 i C4.5. Uglavnom su korišćene strukture podataka poput nizova i matrica. Bilo je teško izvodljivo implementirati paralelizaciju algoritma u Spark platformi. Algoritam je već bio paralelizovan korišćenjem niti u Javi, a preuzeta implementacija nije menjana. Paralelizovano je traženje atributa sa najvećom vrednošću Gini dobiti. Ukoliko se broj atributa za koje treba da se računa Gini dobit poklapa sa prethodno definisanim brojem oni se ubacuju u listu. Zatim se paralelno (na više jezgara) računa Gini dobit za svaki od elemenata liste. Za svaki od atributa dobija se čvor za podelu sa vrednošću Gini dobiti, vrednošću atributa koja se koristi za podelu, i broj pozicije u sortiranom skupu elemenata. Nakon toga prolazi se kroz listu čvorova rezultata i bira se onaj čvor (atribut) koji je imao najveću Gini dobit. Čvor sa najvećom Gini dobiti koristi se kao čvor podele.

6. Eksperimentalni rezultati

Prilikom implementacije paralelizovanih algoritama, delovi koda koji su izmenjeni, odnosili su se na ponavljanje međusobno nezavisnih procesa koji mogu da se izvršavaju istovremeno. Rekurzivni pozivi algoritma u ovom radu nisu paralelizovani iako mogu da se izvršavaju istovremeno. U radu su korišćene tehnologije Java 8 i Spark 2.2.0. Razvojno okruženje koje je korišćeno je Eclipse "Mars" a operativni sistem Windows 7 Ultimate. Izvršavanje algoritama na računarima različitih hardverskih karakteristika pokazalo je blagu razliku u brzini formiranja stabla. Hardverske karakteristike računara na kojima su testirani algoritmi:

- Procesor - Intel® Core™ i5-4210M with Intel HD Graphics 4600 (2.6 GHz, up to 3.2 GHz with Intel Turbo Boost Technology, 3 MB cache, 2 cores), Memory - 8 GB 1600 MHz DDR3L SDRAM (1 x 8 GB), 2 SODIMM Slots
- Procesor-Intel Core i7 - 16GB Memory - NVIDIA GeForce GTX 1050 Ti - 1TB Hard Drive + 256GB Solid State Drive 16GB of DDR4 RAM.

6.1 Podaci

Inicijalna ideja ovog rada je bila da se koriste podaci o vrednostima mera dobijenih praćenjem svakodnevnih aktivnosti za grupu starijih osoba kroz evropski projekat pod nazivom *City4Age*. Projekat je u toku i prikupljeni podaci kontinuirano pristižu. Mere koje su praćene već su objašnjene u odeljku 2.3 ovog rada. Podaci su preuzeti iz baze podataka u obliku: *id_korisnika, vrednost_mere, id_mere, naziv_mere* na koji je nadovezan *id_grupe_aktivnosti, datum i vreme_početka_aktivnosti*. U tabeli 6.1 navedeni su primeri podataka iz baze.

Tabela 6.1 Primer podataka preuzetih iz baze projekta City4Age

id_korisnika	*vrednost_mere*	*id_mere*	*naziv mere i id_grupe_aktivnosti*	*datum početka aktivnosti*	*vreme početka aktivnosti*
9	3000	91	walk_distance5	1/19/2014	17:00
9	529	91	walk_distance5	3/1/2017	0:00
9	0.95	95	walk_speed_outdoor5	3/1/2017	0:00
9	1728	98	walk_time_outdoor5	1/19/2014	17:00
9	3188	98	walk_time_outdoor5	3/1/2017	0:00
10	43	55	public_transport_rides_month3	3/1/2017	0:00
10	38	55	public_transport_rides_month3	4/1/2017	0:00

Prikupljeni podaci su zatim analizirani i praćene su promene u vrednostima mera tokom vremena. Za ovaj rad bitno je da se detektuju značajnije promene vrednosti mera koje se posmatraju. Ako analizom podataka uočimo dovoljan broj promena kod vrednosti neke od mera, zaključujemo da osoba ispoljava porast ili opadanje u vrednostima za tu meru. Kao ulazni atribut za algortime stabla odlučivanja koristi se mera (npr. walk_distance5), a vrednosti atributa su: mera opada, mera raste ili mera ostaje nepromenjena. Imajući u vidu status projekta (u toku je i dalje), količina do sada prikupljenih podataka pokazala se kao nedovoljna za potrebe testiranja algoritama razmatranih u ovom radu. Taj skup podataka nije pružio dovoljno informacija na osnovu kojih bi se moglo zaključiti da li vrednost posmatrane mere raste, opada ili ostaje nepromenjena.

Za manje instance problema korišćene su vrednosti mera generisane od strane gerijatrijskog osoblja za 15 osoba. Mereno je 28 različitih aktivnosti u periodu od jedne godine (360 vrednosti po aktivnosti). Podaci su čitani iz CSV datoteka, gde su zapisi bili u formatu: *r_br, tip, id_korisnika, grupa_mere, tip_mere, mera, mesec, dan, vrednost_mere*. Tabela 6.2 prikazuje primere podataka koji se nalaze u CSV datoteci.

Tabela 6.2 Primer podataka koje je generisalo gerijatrijsko osoblje

r_br	*tip*	*id_korisnika*	*grupa_mere*	*tip_mere*	*mera*	*mesec*	*dan*	*vrednost_mere*
357	type3	1	Contextual	Contextual	Falls	Dec	28	2.06
358	type3	1	Contextual	Contextual	Falls	Dec	29	2.10
359	type3	1	Contextual	Contextual	Falls	Dec	30	2.17
360	type3	1	Contextual	Contextual	Weight loss	Jan	1	3.07
361	type3	1	Contextual	Contextual	Weight loss	Jan	2	2.98
…	…	…	…	…	…	…	…	…
780	type3	1	Contextual	Contextual	Weakness	Mar	1	2.82
781	type3	1	Contextual	Contextual	Weakness	Mar	2	3.09

Za manje instance problema, podataka je bilo dovoljno da bi se pratile promene u vrednostima određene mere. Nakon učitavanja podataka, upoređivane su vrednosti za 2 susedna dana i ako je vrednost drugog dana manja od 90% u odnosu na vrednost prvog dana, to je prepoznato kao opadanje vrednosti, a ako je vrednost drugog dana veća od 110% u odnosu na vrednost prvog dana, to je prepoznato kao rast vrednosti. Ako je broj dana rasta veći od broja dana opadanja i ako je broj porasta bar 20% od ukupnog broja mera, smatra se da osoba ima rast vrednosti mere koja se posmatra na nivou perioda od godinu dana. Analogno važi i za opadanje vrednosti mere. U ovom radu je ispitivano 6 različitih bolesti karakterističnih za starije osobe. Bolesti koji su ispitivane su: dijabetes, demencija, Alchajmerova bolest, hronična oboljenja, kardio – vaskularna oboljenja i astma. Zaključak o tome da li osoba ispoljava rizik od oboljenja i u kojoj meri u radu je donet na osnovu informacija o promenama u vrednostima mera posmatranih aktivnosti. Ciljni atribut algoritma je bolest koja se ispituje, a mere aktivnosti i ostale bolesti koriste se kao ulazni atributi za algoritme stabla odlučivanja. Stabla dobijena na instancama manjih dimenzija bila su najviše visine 3. Prilikom analize manjih instanci (skupove podataka manjih dimenzija) sekvencijalni algoritmi su pokazali dobre performanse u fazi izgradnje stabla (jednostavni algoritmi su dobijeni kao rezultat).

Za uporednu analizu bilo je potrebno testirati algoritme na velikim instancama. Velike dimenzije instanci su formirane na osnovu podataka generisanih od strane gerijatrijskog osoblja, tako što je simulirano kreiranje novih osoba, a za

vrednosti mera za njih korišćene su delimične vrednosti mera osoba za koje su podaci već generisani (podaci za januar uzeti su od jedne osobe, za februar od druge, za mart od treće itd. Za svaki mesec se slučajnim izborom određuje osoba od koje će se izvršiti kopiranje podataka). Isti postupak je primenjen za svaku od mera koja je praćena. Na opisan način generisani su datoteke od 150 osoba (10 puta veća dimenzija) i 1500 osoba (100 puta veća dimenzija). Zbog velike količine podataka i nemogućnosti obrade datoteke tog reda veličine u Javi za dimenziju 1000 puta veću (15000 osoba) korišćen je samo jedan mesec vrednosti (30 vrednosti umesto 360 vrednosti). U datoteku su upisivane vrednosti samo za mesec Januar, a osobe su birane isto slučajnim izborom kao kod datoteka dimenzija 10 i 100.

Podaci transformisani na prethodno opisan način koriste se za algoritme stabla odlučivanja. Za ID3 algoritam mere aktivnosti kao vrednost atributa imaju „true“ - mera aktivnosti opada ili „false“ – mera aktivnosti ne opada. Takođe, bolesti u ID3 algortimu imaju vrednost „true“ – osoba ispoljava faktor rizika od bolesti ili „false“ – osoba ne ispoljava faktor rizika od bolesti. Što se tiče C4.5 i CART algoritama mere aktivnosti imaju vrednost „0“–mera aktivnosti je nepromenjena, „1“ – mera aktivnosti opada ili „2 –mera aktivnosti raste. Bolesti u algoritmima C4.5 i CART takođe imaju vrednost „0“ – osoba ne ispoljava faktor rizika od bolesti, „1“ – osoba ispoljava blagi faktor rizika od bolesti ili „2“–osoba ispoljava ozbiljniji faktor rizika od bolesti.

6.2 Analiza eksperimentalnih rezultata

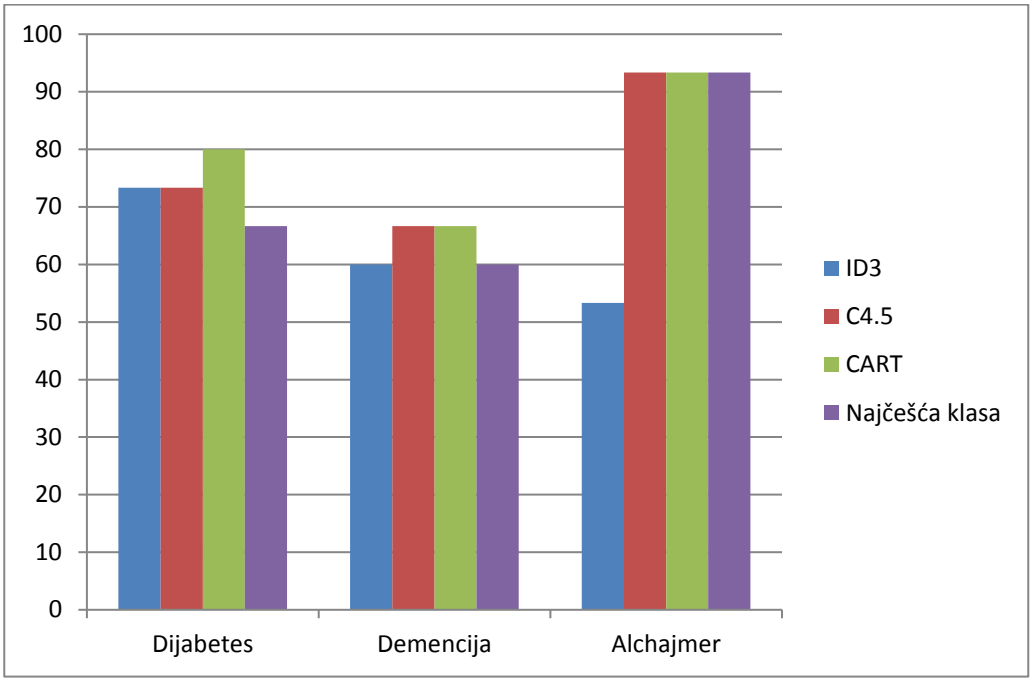
U ovom radu su korišćeni podaci za 15 osoba, 6 osoba muškog pola i 9 osoba ženskog pola. Starost osoba je između 65 i 80 godina, od toga njih četvoro ima 75 i više godina a njih troje ima između 65 i 70 godina. Praćene su njihove svakodnevne aktivnosti u periodu od jedne godine na dnevnom nivou. Prethodno u tekstu je navedeno koje aktivnosti su prepoznavane. U trenutku pokretanja projekta za opservaciju su odabrane zdrave osobe. Nakon perioda od godinu dana njih troje je pokazalo faktor rizika oboljenja od dijabetesa, njih petoro je pokazalo znakove demencije, među kojima je dvoje njih iskazalo faktor rizika za alchajmerovu bolest. Četiri osobe su povremeno ispoljavalo rizik od hroničnih oboljenja, dok je njih dvoje iskazalo rizik od respiratornih oboljenja. Rizik od kardio-vaskularnih oboljenja ispoljila je jedna osoba.

Tabela 6.3 Vremena izgradnje stabla u sekvencijalnoj i paralelnoj varijanti

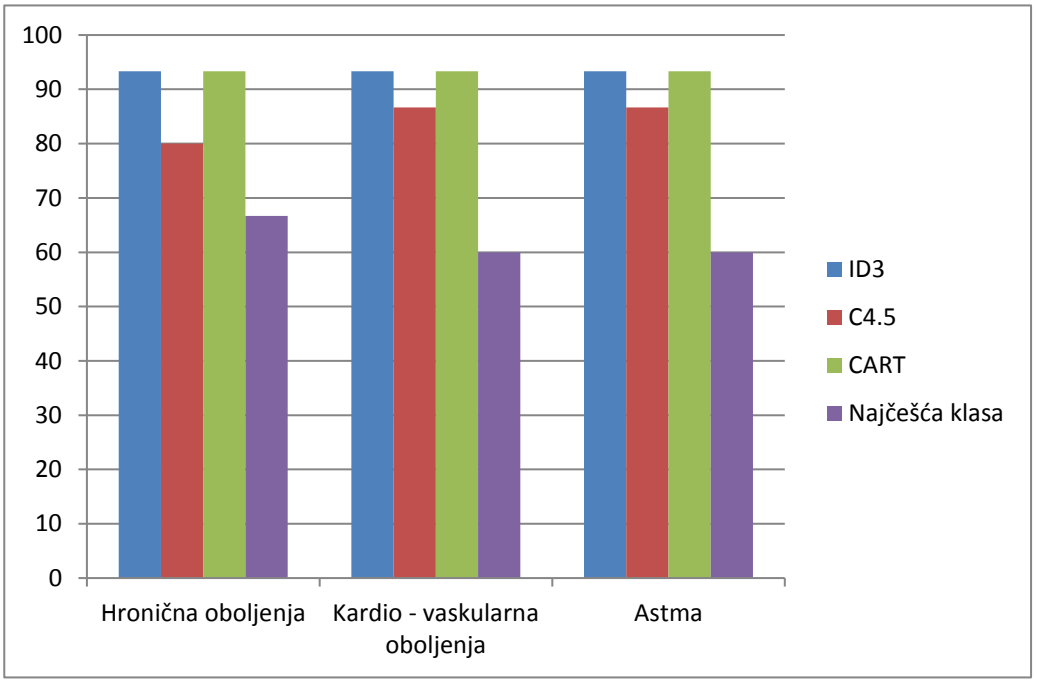
	Dijabetes	Demencija	Alchajmer	Hronična oboljenja	Kard. vask.	Astma
Sekv ID3	8	11	6	13	26	4
Par ID3	60	62	63	40	48	42
Sekv C4.5	66	39	80	63	39	61
Par C4.5	80	55	75	56	55	52
Sekv CART	15	19	17	22	12	23
Par CART	35	44	40	27	22	23

Tabela 6.4 Procenat tačnosti za različite algoritme stabala odlučivanja

	Dijabetes	Demencija	Alchajmer	Hronična oboljenja	Kard. vask.	Astma
Najčešća klasa	66.67	60	93.33	66.67	60	60
ID3	73.33	60	53.33	93.33	93.33	93.33
C4.5	73.33	66.67	93.33	80	86.67	86.67
CART	80	66.67	93.33	93.33	93.33	93.33



Dijagram 6.1 Uporedna analiza tačnosti algoritama



Dijagram 6.2 Uporedna analiza tačnosti algoritama

Za testiranje tačnosti svakog od algoritma primenjivan je proces višestruke izgradnje stabla. Svaki put je jedna osoba korišćena za testiranje stabla, a ostalih 14 osoba su korišćene za izgradnju stabla. Postupak je ponavljan za svih 15 osoba u skupu malih dimenzija. Rezultati testiranja tačnosti algoritama kao i

najčešća klasa za sve bolesti obrađivane u radu prikazani su procentualno u tabeli 6.4. Nemogućnost obrade numeričkih podataka u ID3 algoritmu limitira analize ovog rada. Skup podataka koji je korišćen za formiranje algoritma kao vrednosti za svaki od atributa imao je "true" i "false" i nije bilo moguće odrediti da li su u skupu podataka primećeni porasti ili aktivnosti osoba ostaju nepromenjene. Samim tim je umanjena tačnost rezultata testiranja algoritma. Takođe nije bilo moguće proceniti faktor bolesti neke osobe već samo činjenicu da li osoba ispoljava simptome bolesti ili ne. Iako daje rezultate veoma brzo pokazuje se da kod određenih bolesti ima veći procenat tačnosti a kod drugih manji u odosu na druge algoritme. Uglavnom se pokazao bolje od kriterijuma izbora najbolje klase. Najlošije performanse je pokazao kad se ispitivao faktor rizika od oboljena Alchajmerovom bolešću. Razlog za to su loše generisani podaci za formiranje stabla. Performanse algoritma direktno zavise od količine i raznovrsnosti ulaznih podataka. Veći skup podataka bi obezbedio veću tačnost.

Velika prednost C4.5 algoritma u aspektu ovog rada je mogućnost korišćenja numeričkih vrednosti zato što u velikoj meri poboljšava rezultate analiza. Faza potkresivanja omogućuje da se za kratko vreme provere rezultati ciljnih atributa na skupu za testiranje stabla. Cena uštede vremena je umanjen procenat tačnosti rezultata. Konkretno na podacima koji su korišćeni u radu potkresano stablo daje u proseku 12.22% manju tačnost u odnosu na stablo koje nije potkresano. Mali broj osoba koji je korišćen za eksperimentalne rezultate (15) povećava procenat greške. Zbog svoje složenije implementacije i procesa potkresivanja (smanjenja složenosti) proces formiranja stabla kod algoritma C4.5 traje duže nego kod ID3 algoritma. Algoritam C4.5 pokazuje bolje ili podjednake performanse tačnosti u odnosu na izbor najbolje klase u ispitivanju svake od bolesti.

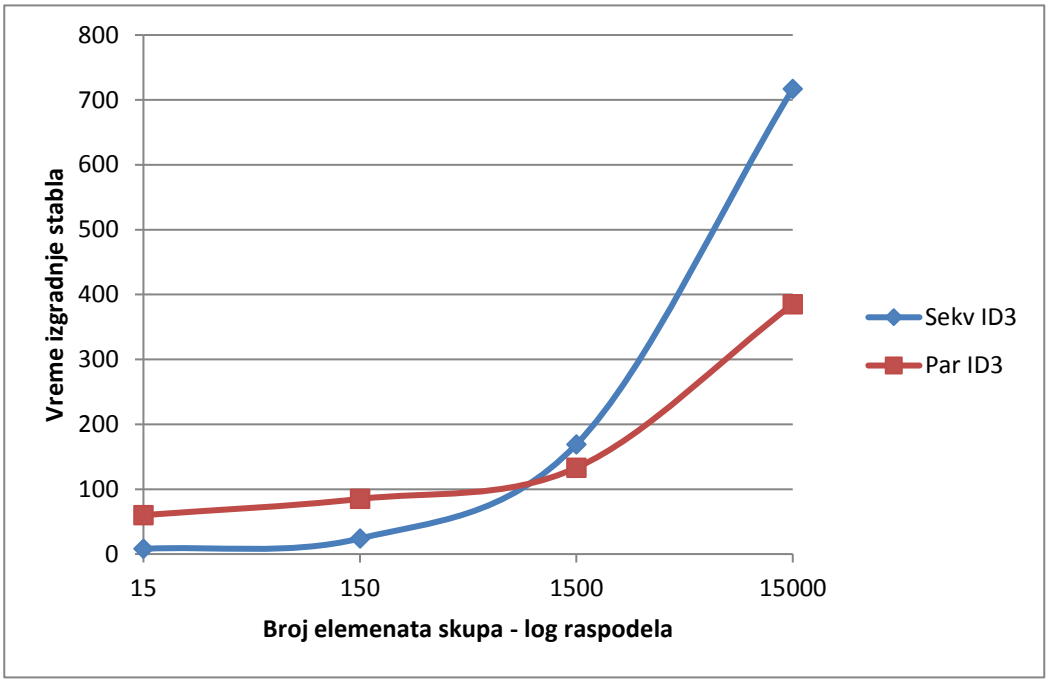
Prednost regresivnih stabla da predviđaju tačan broj a ne klasu obezbeđuje veći procenat tačnosti rezultata kod CART algoritama. Na osnovu analize procenta tačnosti može se zaključiti da se CART algoritam pokazao kao bolji ili podjednako tačan u pogledu druga dva algoritma i u pogledu izbora najbolje klase kod svih bolesti koje su ispitivane u ovom radu.

Mali broj osoba korišćenih za građenje algoritama uzrokuje manju složenost stabla što dovodi do toga da se paralelni algoritmi duže izvršavaju i ne doprinose poboljšanju performansi. Vreme u sekundama koje je bilo potrebno algoritmima da izgrade stablo na instancama male dimenzije prikazano je tabeli 6.3. Zaključak je da će prednost paralelizacije biti izražena kod testiranja na većim skupovima osoba. Izvršavanje algoritma na računarima različitih hardverskih karakteristika pokazalo je blagu razliku u brzini formiranja stabla.

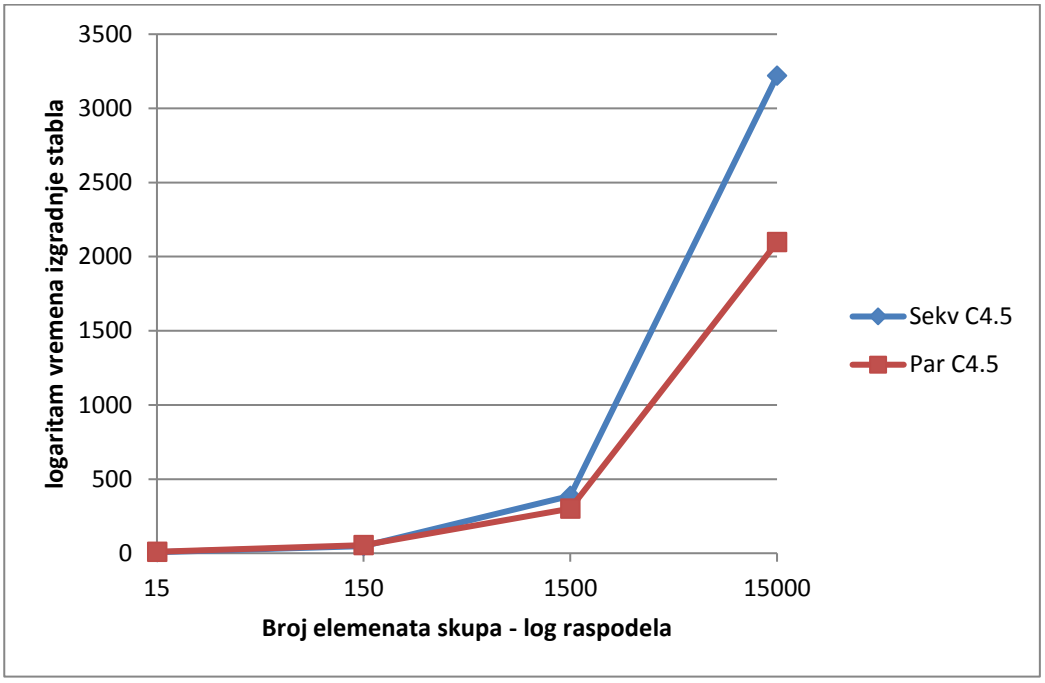
U nastavku su detalji o eksperimentalnoj analizi nad instancama većih dimenzija. U tabeli 6.5 prikazano je vreme u sekundama potrošeno na izgradnju stabla na instancama malih, srednjih i većih dimenzija kod sekvencijalnih i paralelnih implementacija algoritama ID3, C4.5 i CART.

Tabela 6.5 Uporedna analiza vremena izgradnje stabla u sekundama za bolest Dijabetes

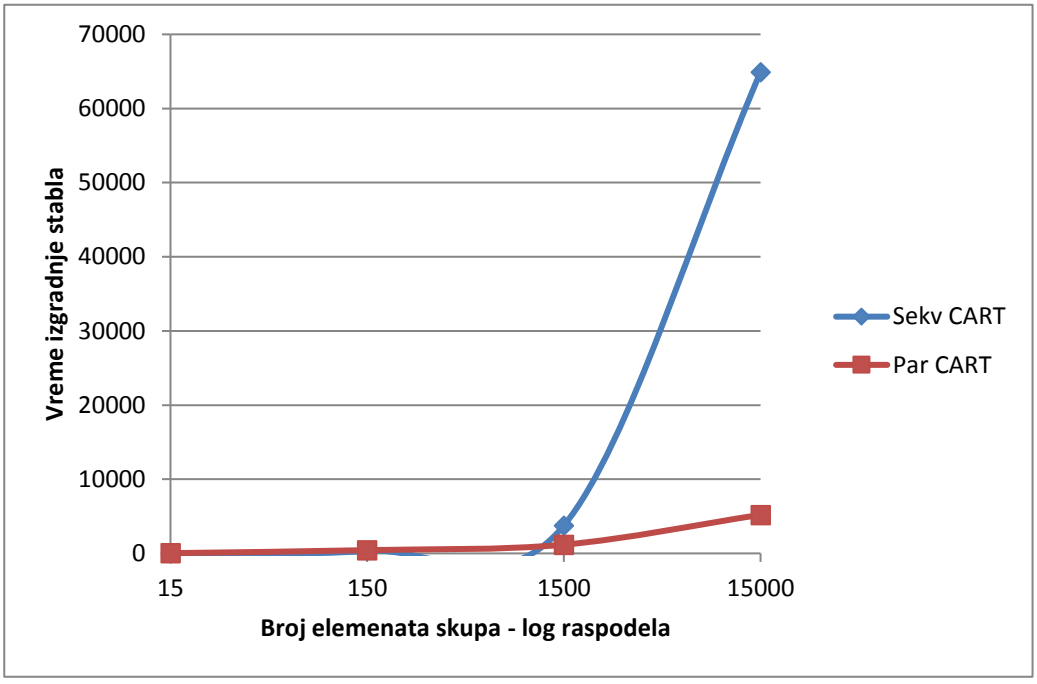
	15 osoba	150 osoba	1500 osoba	15000 osoba
Sekv. ID3	8	24	169	717
Par. ID3	60	85	133	385
Sekv. C4.5	7	48	387	3219
Par. C4.5	10	55	300	2097
Sekv. CART	15	339	3759	64908
Par. CART	35	439	1178	5200



Dijagram 6.3 Uporedna analiza performansi izgradnje stabla kod sekvencijale i paralelne implementacije ID3 algoritma



Dijagram 6.4 Uporedna analiza performansi izgradnje stabla kod sekvencijale i paralelne implementacije C4.5 algoritma



Dijagram 6.5 Uporedna analiza performansi izgradnje stabla kod sekvencijale i paralelne implementacije CART algoritma

Za poređenje performansi izgradnje algoritama korišćene su veće dimenzije podataka koje su dobijene veštačkim generisanjem opisanim u odeljku 6.1

podaci. Korišćene su dimenzije 10 puta veće (150 osoba), 100 puta veće (1500 osoba) i 1000 puta veće (15000 osoba). Kod instanci malih dimenzija, sekvencijalne varijante algoritama su efikasnije, jer inicijalni trošak pokretanja Spark tehnologije i višejezgarne aplikacije prevazilazi uštedu u vremenu koje donosi paralelna obrada. Međutim kako dimenzije podataka rastu, na dijagramima se vidi da paralelne implementacije algoritama pokazuju dosta bolje performanse. Ispitivanjem podataka na još većim dimenzijama bi verovatno doprinelo još izraženijem ubrzanju. Najveće ubrzanje postignuto je u CART algoritmu gde je paraleni algoritam završio izvršavanje za približno 10 puta manje vremena.

7. Zaključak

Predmet istraživanja ovog rada je da se otkrije na koji način promene u svakodnevnom životu starijih ljudi utiču na njihovo zdravstveno stanje. Da bi se to postiglo, urađen je sveobuhvatni pregled bitnih aktivnosti vezanih za zdravlje starijih osoba. Još jedan značajan aspekt prikazanog rada je prepoznavanje nekih važnih istraživačkih izazova, koje treba rešavati kako bi se poboljšao učinak, preciznost i robusnost razmatranih sistema. Ovaj opis doprinosi definisanju budućih istraživačkih projekata i naknadnom napretku u ovom području. U bliskoj vezi sa gore navedenim, od velikog je značaja da se pravilno označe i formalizuju aktivnosti koje se obavljaju, kako bi se uspostavili korisni krajnji zaključci. Ako postoji tačno razumevanje odnosa između raspoređenih senzora i njihovih mogućnosti opažanja aktivnosti, dizajn rešenja i njegovo postavljanje mogu se znatno poboljšati i realizovati lakše i brže. Neophodno je podatke dobijene od senzora prvo transformisati u aktivnosti. Aktivnosti osoba koje učestvuju u projektu korišćene su za izgradnju algoritama. Praćenje aktivnosti iz svakodnevnog života starijih osoba obezbeđuje velike skupove podataka. Nemoguće je da se na jednostavan način analiziraju podaci da bi se uočili šabloni. U radu su korišćeni algoritmi stabla odlučivanja za jednostavniju analizu prikupljenih podataka. Na osnovu rezultata dobijenih u ovom radu ali i korišćenjem raznih drugih analitičkih pristupa u literaturi možemo zaključiti da svakodnevni život utiče na zdravstveno stanje osobe. Kada je stablo odlučivanja već izgrađeno, testiranje instance se izvršava u linearnom vremenu. Brzina dobijanja rezultata zavisi od složenosti stabla koje se testira. Uspeh istraživanja omogućuje novi način za ispitivanje rizika od bolesti kod starijih osoba. Cilj je da se olakša posao lekarima, negovateljima i gerijatrijskim službama. Otkrivanje blagog rizika ispoljavanja bolesti kod osobe pomaže da se veoma brzo reaguje i osobi pruži neophodna pomoć ili preventivna nega koja bi sprečila napredovanje bolesti. Prednost stabla odlučivanja je u tome što veoma brzo testiraju podatke i daju veliki procenat tačnosti. Prilikom testiranja na malim dimenzijama za 6 ispitivanih bolesti CART algoritam je dostigao najveću tačnost za svaku od bolesti, pa možemo zaključiti da je postigao najbolje performanse u domenu tačnosti. Na osnovu eksperimentalnih rezultata možemo zaključiti da ako su dimenzije problema dovoljno velike paralelne implementacije algoritama su efikasnije u odnosu na sekvencijalne. Efikasnost ubrzanja se najbolje vidi kod CART algoritma.

Dalje istraživanje ove oblasti bi moglo da obezbedi uporednu analizu svih dobijenih rezultata na projektima. Uporedna analiza bi mogla da obezbedi pregled:

- aktivnosti koje mogu biti praćene;
- senzora koji bi se koristili;

- algoritama za obradu prikupljenih podataka;
- algoritama za prepoznavanje faktora rizika bolesti kod starijih osoba.

U ovom radu je predstavljena gruba procena i razmatrana je svaka bitnija promena koja je prepoznata na osnovu svakodnevnog života osobe. Za dalju analizu moguća su 4 različita pristupa:

- prepoznavanje da konkretna mera aktivnosti neke osobe kontinualno opada ili raste (svaka vrednost je manja (veća) od one koja je prethodno izmerena);
- prepoznavanje da se padovi ili skokovi vrednosti neke mere aktivnosti vezuju za određenu lokaciju;
- prepoznavanje da se padovi ili skokovi vrednosti neke mere aktivnosti vezuju za određeni vremenski period;
- prepoznavanje da li je promena vrednosti u jedoj od mera aktivnosti povezana sa promenama u vrednosti nekih drugih mera aktivnosti.

Literatura

[Antoniou i sar. 2014] Antoniou, P.E., Konstantinidis, E.I., Billis, A.S., Bamidis, P.D. (2014). Integrating the useful assisted living platform; observation from the field. In Proceedings of the 6th European Conference of the International Federation for Medical and Biological Engineering, Dubrovnik, Croatia; Springer International Publishing AG: Cham, Switzerland; 657–660.

[Apache Spark] Apache Spark - http://spark.apache.org/

[Apache Kafka] Apache Kafka - http://kafka.apache.org/

[Baker & Jain 1976] Baker, E., & Jain, A. K. (1976). On feature ordering in practice and some finite sample effects. In Proceedings of the Third International Joint Conference on Pattern Recognition, pages 45-49, San Diego, CA

[Bao & Intille 2004] Bao, L., & Intille, S. (2004). Activity recognition from user-annotated acceleration data. In L. N. i. C. Science (Ed.), Pervasive Computing (1–17). Vienna, Austria: Springer Berlin Heidelberg.

[BenBassat 1978] BenBassat M. (1978). Myopic policies in sequential classification. IEEE Trans. On Computing, 27(2):170-174

[Billis i sar. 2013] Billis, A.S., Papageorgiou, E.I., Frantzidis, C., Konstantinidis, E.I., Bamidis, P.D. (2013). Towards a hierarchically-structured decision support tool for improving seniors' independent living: The usefil decision support system. In Proceedings of the 6th International Conference on Pervasive Technologies Related to Assistive Environments, Island of Rhodes, Greece; 27.

[Brand i sar. 1997] Brand, M., Oliver, N., & Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. In Proceedings of Computer Vision and Pattern Recognition (994–999). IEEE.

[Bratko & Bohanec 1994] Bratko, I., & Bohanec, M. (1994). Trading accuracy for simplicity in decision trees, Machine Learning 15: 223-250

[Brdiczka 2009] Brdiczka, O. (2009). Learning situation models in a smart home. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 39, 56–63.

[Brdiczka i sar. 2007] Brdiczka, O., Reignier, P., & Crowley, J. (2007). Detecting individual activities from video in a smart home. In Knowledge-Based Intelligent Information and Engineering Systems (363–370). Springer Berlin Heidelberg.

[Breiman i sar. 1984] Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and Regression Trees. Wadsworth Int. Group.

[Brodley & Utgoff] Brodley, C. E., & Utgoff. P. E.(1995). Multivariate decision trees. Machine Learning, 19:45-77.

[Buntine & Niblett 1992] Buntine, W., Niblett, T. (1992). A Further Comparison of Splitting Rules for Decision Tree Induction. Machine Learning, 8: 75-85

[CART Github] CART Github - https://github.com/haifengl/smile (Statistical Machine Intelligence & Learning Engine http://haifengl.github.io/smile/)

[Chahuara i sar. 2014] Chahuara, P., Portet, F., Vacher, M. (2014). Making context aware decision from uncertain information in a smart home: A markov logic network approach. In Ambient Intelligence; Springer International Publishing AG: Cham, Switzerland; Volume 8309, 78–93.

[Chen & Dawadi 2011] Chen, C., & Dawadi, P. (2011). Casasviz: Web-based visualization of behavior patterns in smart environments. In Proceedings of the International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Seattle, WA, USA, 301–303.

[Chen i sar. 2014] Chen, L., Nugent, C., & Okeyo, G. (2014). An Ontology-based Hybrid Approach to Activity Modeling for Smart Homes. IEEE Transaction on Human-Machine Systems, 44, 92–105.

[Chen i sar. 2012] Chen, L., Hoey, J., Nugent, C., Cook, D., & Yu, Z. (2012a). Sensor-based activity recognition. IEEE Transactions on Systems, Man, and Cybernetics-Part C, 42, 790–808.

[Choudhury & Consolvo 2008] Choudhury, T., & Consolvo, S. (2008). The mobile sensing platform: An embedded activity recognition system. IEEE Pervasive Computing, 7, 32–41.

[Cook i sar. 2013a] Cook, D., Feuz, K., & Krishnan, N. (2013). Transfer learning for activity recognition: a survey. Knowledge and information systems, 36, 537–556.

[Cook i sar. 2013b] Cook, D.J., Youngblood, M., Heierman III, E.O., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F. (2013). Mavhome: An agent-based smart home. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom), San Diego, CA, USA, 521–524.

[Cook i sar. 2013c] Cook, D.J., Crandall, A.S., Thomas, B.L., Krishnan, N.C. (2013). CASAS: A smart home in a box. Computer 46, 62–69.

[Cook & Schmitter-Edgecombe 2009] Cook, D., & Schmitter-Edgecombe, M. (2009). Assessing the quality of activities in a smart environment. Methods of information in medicine, 48, 480.

[Developer survey result 2016] Developer survey result - https://insights.stackoverflow.com/survey/2016

[Duda & Hart 1973] Duda, R., & Hart, P. (1973). Pattern Classification and Scene Analysis, New-York, Wiley

[Esposito i sar. 1997] Esposito, F., Malerba, D., & Semeraro, G. (1997). A Comparative Analysis of Methods for Pruning Decision Trees. EEE Transactions on Pattern Analysis and Machine Intelligence, 19(5):476-492

[Fayyad & Irani 1992] Fayyad, U., & Irani, K. B. (1992). The attribute selection problem in decision tree generation. In proceedings of Tenth National Conference on Artificial Intelligence, 104–110, Cambridge, MA: AAAI Press/MIT Press

[Fernandez-Caballero 2012] Fernandez-Caballero, A. (2012). Human activity monitoring by local and global finite state machines.´ Expert Systems with Applications, 39, 6982–6993.

[Ferri i sar. 2002] Ferri C., Flach P., & Hernandez-Orallo, J. (2002). Learning Decision Trees Using the Area Under the ROC Curve. In Claude Sammut and Achim Hoffmann, editors, Proceedings of the 19th International Conference on Machine Learning, 139-146. Morgan Kaufmann

[Fong i sar. 2003] Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. Robotics and autonomous systems, 42, 143–166.

[Friedman 1977] Friedman J. H. (1977). A recursive partitioning decision rule for nonparametric classifiers. IEEE Trans. on Comp., C26:404-408

[Friedman 1991] Friedman, J. H.(1991). "Multivariate Adaptive Regression Splines", The Annual Of Statistics, 19, 1-141.

[Galata i sar. 1999] Galata, A., Johnson, N., & Hogg, D. (1999). Learning structured behaviour models using variable length Markov models. In IEEE International Workshop on Modelling People (95–102). IEEE.

[Gelfand i sar. 1991]Gelfand, S. B., Ravishankar, C. S., & Delp, E. J. (1991). An iterative growing and pruning algorithm for classification tree design. IEEE Transaction on Pattern Analysis and Machine Intelligence, 13(2):163-174

[Gillo 1972] Gillo M. W.(1972). MAID: A Honeywell 600 program for an automatised survey analysis. Behavioral Science 17: 251-252.

[Helal i sar. 2005] Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., Jansen, E. (2005). The gator tech smart house: A programmable pervasive space. 38, 50–60.

[Holte 1993] Holte R. C.(1993). Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11:63-90.

[Hyafil & Rivest 1976] Hyafil, L., & Rivest, R.L. (1976). Constructing optimal binary decision trees is NP-complete. Information Processing Letters, 5(1):15-17

[John 1996] John G. H. (1996). Robust linear discriminant trees. In D. Fisher and H. Lenz, editors, Learning From Data: Artificial Intelligence and Statistics V, Lecture Notes in Statistics, Chapter 36, 375-385. Springer-Verlag, New York

[Jupyter Notebook] Jupyter Notebook – http://jupyter.org/

[Kass 1980] Kass G. V.(1980). An exploratory technique for investigating large quantities of categoricaldata. Applied Statistics, 29(2):119-127.

[van Kasteren i sar. 2010] van Kasteren, T.L.M., Englebienne, G., Kröse, B.J.A. (2010). Activity recognition using semi-markov models on real world smart home datasets. J. Ambient Intell. Smart Environ. 2, 311-325.

[van Kasteren & Krose 2007] van Kasteren, T., & Krose, B. (2007). Bayesian activity recognition in residence for elders. In Proc. of the International Conference on Intelligent

[Laerhoven & Aidoo 2001] Laerhoven, K. V., & Aidoo, K. (2001). Teaching context to applications. Personal and Ubiquitous Computing, 5, 46–49.

[Lester i sar. 2005] Lester, J., Choudhury, T., & Kern, N. (2005). A Hybrid Discriminative/Generative Approach for Modeling Human Activities. In IJCAI ((5)766–772).

[Lim i sar. 2000] Lim, X., Loh, W.Y., & Shih, X. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning 40:203-228

[Lin & Fu 1983] Lin, Y. K. & Fu, K. (1983). Automatic classification of cervical cells using a binary tree classifier. Pattern Recognition, 16(1):69-80

[Logan i sar. 2007]Logan, B., Healey, J., Philipose, M., Tapia, E.M., Intille, S. (2007). A long-term evaluation of sensing modalities for activity recognition. In Ubiquitous Computing; Springer: Berlin/Heidelberg, Germany; Volume 4717, 483–500.

[Loh & Shih 1997] Loh, W.Y., & Shih, X.(1997). Split selection methods for classification trees. Statistica Sinica, 7: 815-840

[Loh & Shih 1999] Loh, W.Y., & Shih, X. (1999). Families of splitting criteria for classification trees.Statistics and Computing 9:309-315

[Loh & Vanichsetakul 1988] Loh, W.Y., & Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant Analysis. Journal of the American Statistical Association, 83: 715-728

[Lubinsky 1993] Lubinsky D. (1993). Algorithmic speedups in growing classification trees by using an additive split criterion. Proc. AI&Statistics93, 435-444

[Matei i sar. 2012] Matei, Z., Mosharaf, C., Tathagata, D., Ankur, D., Justin, M., Murphy, M., Michael, J. F., Scott, Sh., Ion, S. (2012) Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, University of California, Berkeley

[Maurer & Rowe 2006] Maurer, U., & Rowe, A. (2006). Location and activity recognition using eWatch: A wearable sensor platform. In Ambient Intelligence in Everyday Life (86–102). Springer Berlin Heidelberg.

[Mingers 1989] Mingers J. (1989). An empirical comparison of pruning methods for decision tree induction. Machine Learning, 4(2):227-243

[Moeslund i sar. 2006] Moeslund, T., Hilton, A., & Kruger, V. (2006). A survey of advances in vision-based human motion capture and analysis.¨Computer vision and image understanding, 104, 90–126.

[Morgan & Messenger 1973]Morgan J. N. & Messenger R. C.(1973). THAID: a sequential search program for the analysis of nominal scale dependent variables. Technical report, Institute for Social Research, Univ. of Michigan, Ann Arbor, MI.

[Muller & Wysotzki 1994]Muller W., & Wysotzki F.(1994). Automatic construction of decision trees for classification. Annals of Operations Research, 52:231-247.

[Murthy 1998] Murthy S. K. (1998). Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. Data Mining and Knowledge Discovery, 2(4):345-389

[Naumov 1991] Naumov G.E. (1991). NP-completeness of problems of construction of optimal decision trees. Soviet Physics: Doklady, 36(4):270-271

[Oliver i sar. 2004] Oliver, N., Garg, A., & Horvitz, E. (2004). Layered representations for learning and inferring office activity from multiple sensory channels.Computer Vision and Image Understanding, 96, 163–180.

[Olaru & Wehenkel 2003] Olaru, C., Wehenkel, L. (2003). A complete fuzzy decision tree technique, Fuzzy Sets and Systems, 138(2):221–254

[Omar i sar. 2010] Omar, F., Sinn, M., & Truszkowski, J. (2010). Comparative analysis of probabilistic models for activity recognition with an instrumented walker. In Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (392–400).

[Ordonez i sar. 2013] Ordo´nez, J., Iglesias, J., & Toledo, P. D. (2013). Online activity recognition using evolving classifiers.˜ Expert Systems with Applications, 40, 1248–1255.

[Pentney i sar. 2008] Pentney, W., Philipose, M., & Bilmes, J. (2008). Structure Learning on Large Scale Common Sense Statistical Models of Human State. In AAAI (1389–1395).

[Philipose & Fishkin, 2004] Philipose, M., & Fishkin, K. (2004). Inferring activities from interactions with objects. Pervasive Computing, 3, 50–57.

[Quinlan 1986] Quinlan, J.R. (1986). Induction of decision trees, Machine Learning 1, 81-106

[Quinlan 1987] Quinlan, J.R. (1987). Simplifying decision trees, International Journal of Man-Machine Studies, 27, 221-234

[Quinlan 1988] Quinlan, J.R. (1988). Decision Trees and Multivalued Attributes, J. Richards, ed., Machine Intelligence, V. 11, Oxford, England, Oxford Univ. Press, 305-318

[Quinlan 1989] Quinlan J. R. (1989). Unknown attribute values in induction. In Segre, A. (Ed.), Proceedings of the Sixth International Machine Learning workshop Cornell, New York. Morgan Kaufmann

[Quinlan 1993] Quinlan, J. R. (1993). C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos

[Rashidi & Cook 2011a] Rashidi, P., & Cook, D. (2011a). Activity knowledge transfer in smart environments. Pervasive and Mobile Computing, 7, 331–343.

[Rashidi & Cook 2011b] Rashidi, P., & Cook, D. (2011b). Discovering activities to recognize and track in a smart environment. Knowledge and Data Engineering, IEEE Transactions on, 23, 527–539.

[Rashidi & Cook 2013] Rashidi, P., & Cook, D. (2013). COM: A method for mining and monitoring human activity patterns in home-based health monitoring systems. ACM Transactions on Intelligent Systems and Technology (TIST), 4, 64.

[Rastogi & Shim 2000] Rastogi, R., & Shim, K.(2000) PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning, Data Mining and Knowledge Discovery, 4(4):315-344.

[Saeb i sar. 2017] Saeb, S., Lattie, E.G., Kording, K.P., Mohr, D.C.(2017). Mobile Phone Detection of Semantic Location and Its Relationship to Depression and Anxiety JMIR Mhealth Uhealth;5(8):e112

[Sethi & Yoo 1994] Sethi, K., & Yoo, J. H. (1994). Design of multicategory, multifeature split decision trees using perceptron learning. Pattern Recognition, 27(7):939-947

[Sklansky & Wassel 1981] Sklansky, J., & Wassel, G. N. (1981). Pattern classifiers and trainable machines. SpringerVerlag, New York

[Sonquist i sar. 1971] Sonquist, J. A., Baker, E. L., & Morgan, J. N.(1971). Searching for Structure. Institute for Social Research, Univ. of Michigan, Ann Arbor, MI.

[Tapia i sar. 2004] Tapia, E., Intille, S., & Larson, K. (2004). Activity Recognition in the Home Using Simple and Ubiquitous Sensors. Pervasive computing, 3001, 158–175.

[Technical preview] Technical preview - https://databricks.com/blog/2016/05/11/apache-spark-2-0-technical-preview-easier-faster-and-smarter.html

[Utgoff 1989] Utgoff, P. E. (1989). Perceptron trees: A case study in hybrid concept representations. Connection Science, 1(4):377-391

[Vacher i sar. 2011] Vacher, M., Istrate, D., Portet, F., Joubert, T., Chevalier, T., Smidtas, S., Meillon, B., Lecouteux, B., Sehili, M., Chahuara, P. (2011). The sweet-home project: Audio technology in smart homes to improve well-being and reliance. In Proceedings of the Annual International Conference on

Engineering in Medicine and Biology Society, EMBC, Boston, MA, USA; 5291–5294.

[Vail i sar. 2007] Vail, D., Veloso, M., & Lafferty, J. (2007). Conditional random fields for activity recognition. In Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (235). ACM.

[Weinland i sar. 2011] Weinland, D., Ronfard, R., & Boyer, E. (2011). A survey of vision-based methods for action representation, segmentation and recognition.

[Yilmaz i sar. 2006] Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. Acm computing surveys (CSUR), 38, 13.

[Zantema & Bodlaender 2000] Zantema, H., & Bodlaender, H. L. (2000). Finding Small Equivalent Decision Trees is Hard, International Journal of Foundations of Computer Science, 11(2): 343-354

[Zhang i sar. 2013] Zhang, Q., Karunanithi, M., Rana, R., Liu, J. (2013). Determination of activities of daily living of independent living older people using environmentally placed sensors. In Proceedings of the 35th Annual International Conference on Engineering in Medicine and Biology Society (EMBC), Osaka, Japan; 7044–7047.

[Zhang i sar. 2014] Zhang, Q., Su, Y., Yu, P. (2014). Assisting an elderly with early dementia using wireless sensors data in smarter safer home. In Service Science and Knowledge Innovation; Springer: Berlin/Heidelberg, Germany; Volume 426, 398–404.