Usporedba raznih metoda rješavanja problema nedostajućih vrijednosti kod klasifikacije

Kolegij: Strojno učenje

Jadran Berbić; Ivan Ljubičić; Jelena Štimac
Prirodoslovno-matematički fakultet
Sveučilište u Zagrebu
Zagreb, Hrvatska
jberbic@hotmail.com; booraztheone@gmail.com; jelena.stimac14@gmail.com

Sažetak*—U znanstvenoj disciplini rudarenja podataka rijetko se događa da su sve vrijednosti iz skupa podataka prisutne. Vremenske i financijske prilike često ne dozvoljavaju da se nadopune nedostajući podaci ili da se pak postojeći skup podataka proširi novima. Često nedostaju samo pojedine vrijednosti u atributima te se vrijednosti koje postoje mogu iskoristiti za daljnu obradu. U radu je prikazano kakav učinak ima izbacivanje podataka, nasumično umetanje podataka, umetanje podataka po statističkoj raspodjeli te učenje na postojećim podacima na prediktivnu sposobnost modela za učenje.*

Ključne riječi — umetanje podataka, decision table, conjunctive rule, REPTree

I. *UVOD*

Klasifikacijski problemi rješavaju se upotrebom tri različita algoritma za klasifikaciju podataka. Skup podataka provede se kroz odabrani algoritam te se dobije model s određenom prediktivnom sposobnošću. Rijedak je idealni slučaj da su za sve atribute i instance iz skupa podataka prisutne sve vrijednosti, zbog različitih razloga, npr. greške u snimanju i prijenosu podataka [10] ili pak zbog činjenice da ljudi ne žele ustupiti pojedine podatke o sebi. Unatoč tome postojeći podaci mogu se iskoristiti za učenje modela. [10] Umetanje podataka (data imputation) može se riješiti korištenjem statističkih metoda ili pak samih algoritama za učenje koji se mogu nositi s nedostajućim vrijednostima. [2]

Cilj rada je, korištenjem poznatih klasifikacijskih algoritama, saznati koji pristupi rješavanju problema nedostajućih vrijednosti daju najbolje rezultate. U ovom radu korištena su četiri različita pristupa u postupanju s nedostajućim podacima:

1. Izbacivanje svih instanci iz skupa podataka koji imaju i barem jednu nedostajuću vrijednost.

2. Nasumično umetanje nedostajućih vrijednosti (umetnu se nasumične vrijednosti koje se nalaze u rasponu maksimalnih i minimalnih vrijednosti svakog pojedinog atributa).

3. Umetanje nedostajućih podataka po raspodjeli koja prevladava u svakom pojedinom atributu.

4. Umetanje podataka prema očekivanju modela nakon što je izvedeno učenje modela čiji algoritam uključuje mogućnost učenja s nedostajućim podacima.

Drugi i treći pristup spadaju u statističke metode umetanja podataka, a četvrti pristup u metode umetanja podataka korištenjem algoritma za učenje. Hipoteza je da bi prvi pristup trebao dati najlošiju prediktivnu sposobnost algoritma, drugi pristup bi trebao davati bolje rezultate od prvog, a treći i četvrti pristup bi trebali davati podjednaku prediktivnu sposobnost koja je ujedno i veća nego u prva dva pristupa.

II. *MATERIJALI, METODOLOGIJA I ISTRAŽIVANJE*

A. Materijali

Na raspolaganju su tri različita skupa podataka koji imaju različitu strukturu. Prvi skup podataka (*Kontracepcija*) ima ulazne vrijednosti u obliku 9 atributa, jedan izlazni atribut i 1473 instanci. Sve vrijednosti su cijeli brojevi. Drugi skup podataka (*Eye*) ima ulazne vrijednosti u obliku 14 atributa, jedan izlazni atribut i 14976 instanci. Vrijednosti svih atributa su realni brojevi, osim izlaznog koji se sastoji od cijelih brojeva. Treći skup podataka (*Motion*) ima 50 ulaznih atributa s realnim brojevima, i jedan izlazni s cijelim brojevima, s 4833 instanci.

Izabrana su tri algoritma za učenje, od kojih svaki ima predviđenu mogućnost korištenja u slučaju da nedostaju vrijednosti. Za provođenje algoritama za učenje korišten je program *Weka*. *Weka* je kolekcija algoritama strojnog učenja namijenjenih rudarenju podataka. Otvorenog je koda, razvijena je na Sveučilištu Waikato na Novom Zelandu. Algoritmi se primijenjuju izravno sa skupa podataka ili pomoću koda napisanog u programskom jeziku *Java*. [9] Tri algoritma koja su korištena za učenje na danim podacima su *Decision Table*, *Conjunctive Rule* i *REPTree*.

Decision Table (DT), tj. tablica odlučivanja, u programu *Weka* koristi jednostavni DTM (*Decision Table Majority*) klasifikator. DT algoritam koristi klasifikacijska pravila (classification rule algorithm), a rezultat je tablica odluka koja sadrži sve atribute kao i ulazni skup podataka. Izlaz će predstavljati odluke na atributima za sve instance (tj. za dovoljan broj instanci, a da se napravi što bolja klasifikacija). DTM vraća većinu skupa podataka ako ne postoji poklapanje ćelije tablice s novim instancama. Tablica odlučivanja ima dvije komponente: shemu i tijelo. Shema je lista atributa, a

tijelo su skupovi označenih instanci. Svaka instanca sastoji se od vrijednosti za atribut u shemi i vrijednosti za oznaku. Ćelija je skup instanci s istim vrijednostima za atribute u shemi. [1][4][8]

Conjunctive rule (CR) algoritam spada u algoritme za učenje pravila. Pri klasificiranju primjenjuje samo pravilo konjukcije, bilo za brojčanu bilo za nazivnu varijablu. Naziva se još induktivno učenje. Cilj kod induktivnog učenja je pronalaženje pravila iz podataka koja generaliziraju sve znanje o podacima na način da tih pravila bude što manje. Ovakva klasifikacija za vrijednost testne instance uspoređuje slaganje ostalih vrijednosti u toj instanci s ostalim instancama i tako uči pravila. [5][7]

REPTree (*Reduced Error Pruning Tree*) algoritam koristi logiku regresijskih stabala i u različitim iteracijama generira više stabala od kojih opstaje ono najbolje. Koristi porast informacije kao kriterij razdvajanja i reže ih koristeći reduciranu grešku rezanja. S nedostajućim vrijednostima nosi se odvajanjem instanci od nedostajućih vrijednosti. [3][6][9]

Za primarnu obradu podataka, prije korištenja programa *Weka*, napisana su četiri programa u programskim jezicima *Octave* i *Matlab*. Jedan program služi nasumičnom izdvajanju željenog postotka nedostajućih vrijednosti (add_missing), drugi program služi za izbacivanje svih instanci u kojima su nedostajuće vrijednosti (remove_missing). Treći program (add_random_missing) traži u svakom atributu maksimalnu i minimalnu vrijednost te nedostajuće vrijednosti u promatranom atributu nadopuni nasumičnim vrijednostima u tom rasponu. Četvrti program (add_dist_missing) za svaki atribut odredi statističku raspodjelu na temelju postojećih vrijednosti te nedostajuće vrijednosti u tom atributu nadopuni prema toj raspodjeli. Treba napomenuti da se u obradi nije tražila nikakva statistička povezanost vrijednosti između različitih atributa, već se svaki atribut promatrao kao samostalan.

B. *Metodologija i istraživanje*

Prvo se pristupilo učenju modela na potpunim skupovima podataka za učenje, korištenjem programa *Weka* na tri spomenuta algoritma. Potom je svaki skup podataka preoblikovan na načina da se vrijednosti zamijenje s 10%, 20%, 30%, 40% i 50% nedostajućih vrijednosti u odnosu na čitav skup podataka. Daljnji korak bio je izbacivanje instanci u kojima se nalaze nedostajuće vrijednosti te provođenje istih algoritama za učenje na tako oblikovanim podacima (1. pristup, oznaka *removed*).

Zatim su skupovima podataka s nedostajućim vrijednostima te vrijednosti nasumično nadopunjene (2. pristup, oznaka *random*) te su provedeni algoritmi za učenje. Potom su vrijednosti na skupovima s nedostajućim vrijednostima nadopunjene prema raspodjeli postojećih vrijednosti u atributu (3. pristup, oznaka *distributed*). Zatim su algoritmi provedeni i na samim skupovima s nedostajućim vrijednostima (4. pristup, oznaka *algoritam*).

III. REZULTATI

Dane su usporedbe četiri korištena pristupa za svaki algoritam i za svaki skup podataka, pri različitim količinama nedostajućih vrijednosti. Kao parametar za usporedbu korištena je relativna apsolutna greška (u radu je korištena oznaka Error, inače se koristi skraćenica RAE) koja se računa prema jednadžbi:

$$Error = (\ |p_1-y_1|+\ldots+|p_n-y_n|\ ) / (\ |y_{sr}-y_1|+\ldots+|y_{sr}-y_n|\ ) \qquad (1)$$

gdje $p_1,\ldots,p_n$ predstavljaju predviđene ciljne vrijednosti, $y_1,\ldots,y_n$ stvarne ciljne vrijednosti, a y_{sr} aritmetičku sredinu stvarnih ciljnih vrijednosti.

DT algoritam na prvom skupu podataka (*Kontracepcija*) daje za potpuni skup grešku u vrijednosti 91,8%. Za sva četiri pristupa greška u slučaju nedostajućih vrijednosti raste relativno brzo. Najmanju grešku daje korištenje samog algoritma na skupu podataka s nedostajućim vrijednostima (4. pristup). Nadopuna nedostajućih vrijednosti nasumičnim vrijednostima i prema raspodjeli daju najveće greške i to veće nego u slučaju da se uklone instance s nedostajućim vrijednostima. Primijećuje se da nasumično umetnute vrijednosti uzrokuju osjetne oscilacije greške čemu je glavni uzrok upravo način umetanja podataka. U manjem obimu je uzrok tome i priroda umetanja nedostajućih vrijednosti gdje se nije pazilo na međusobnu ovisnost vrijednosti u različitim atributima. Ovaj drugi uzrok može objasniti blaže oscilacije greške kod ostalih pristupa (slika 1).

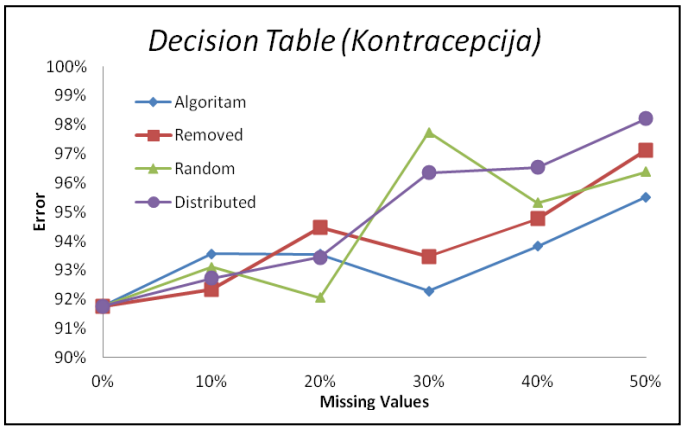

Fig. 1. Greška kod korištenja DT algoritma na skupu podataka Kontracepcija

Kod CR algoritma greška na potpunom skupu podataka *Kontracepcija* podjednaka je kao i u slučaju korištenja DT algoritma. Pri povećanju udjela nedostajućih vrijednosti greška raste nešto sporije nego u prethodnom slučaju. Nasumično i po raspodjeli nadopunjene vrijednosti daju i ovom slučaju najveću grešku, s tim da pristup s uklanjanjem instanci daje nagli rast greške pri 50% nedostajućih vrijednosti pa bi manju grešku pri manjim postocima trebalo uzeti s rezervom. Nedostajuće vrijednosti su ubacivane nasumično pri čemu se moglo dogoditi da u pojedinim instancama nedostaje više vrijednosti nego u drugim instancama pa je pri uklanjanju instanci ostalo više potpunih instanci štp je na kraju rezultiralo manjom greškom pri 10% do 40% nedostajućih vrijednosti kod 1. pristupa. Korištenje 4. pristupa daje najmanju grešku, ali su primjetne blage oscilacije. (Slika 2)

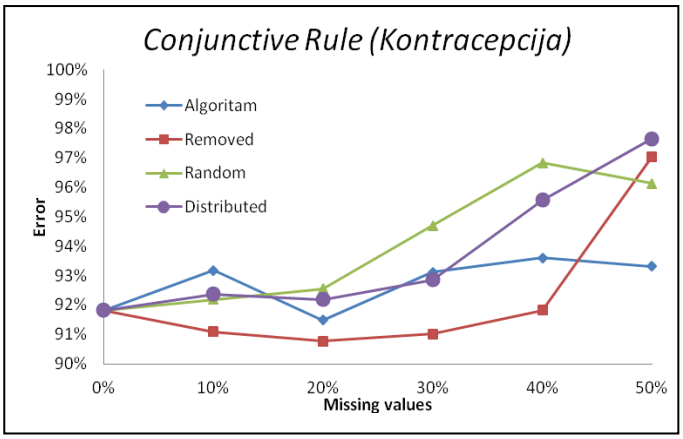

Fig. 2. Greška kod korištenja CR algoritma na skupu podataka Kontracepcija

REPTree algoritam na skupu podataka *Kontracepcija* općenito daje najmanju grešku (koja je i dalje dosta visoka). Za potpuni skup podataka greška iznosi 86,1 %. Pri povećanju udjela nedostajućih vrijednosti greška raste otprilike jednako brzo kao i kod CR algoritma, tj. sporije nego kod DT algoritma. Manju grešku ponovno daju 1. i 4. pristup, dok nadopuna podataka prema raspodjeli daje najveću grešku. Oscilacije se pojavljuju kod prva tri pristupa. (Slika 3)

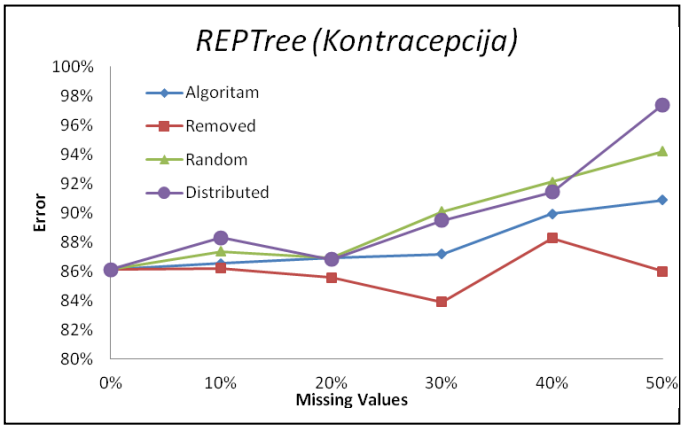

Fig. 3. Greška kod korištenja REPTree algoritma na skupu podataka Kontracepcija

Iz korištenja modela za učenje na prvom skupu podataka može se zaključiti da 1. i 4. pristup daju manje greške nego ostala dva.

DT algoritam na skupu podataka *Eye* daje neobične rezultate. Greška pri korištenju 1. pristupa za 30% i više nedostajućih vrijednosti te 3. pristupa za 10% i više nedostajućih vrijednosti čini kao očekivano rješenje. Ostatak grafa, koji prikazuje grešku od 98% do 100%, definitivno ne izgleda očekivano.

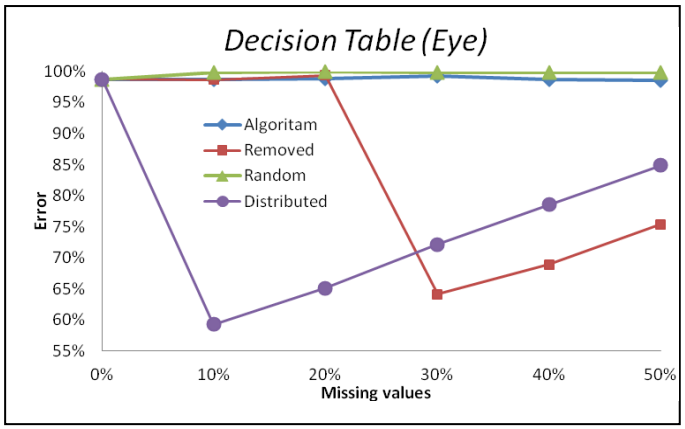

Fig. 4. Greška kod korištenja DT algoritma na skupu podataka Eye

CR algoritam na potpunom skupu podataka *Eye* daje grešku od 93,9 %. S povećanjem nedostajućih vrijednosti greška raste relativno sporo. Četvrti pristup u ovom slučaju daje najveću grešku, dok prvi pristup daje najmanju grešku. Nadopuna podataka prema drugom i trećem pristupu daje podjednake greške koje osciliraju. (Slika 5)

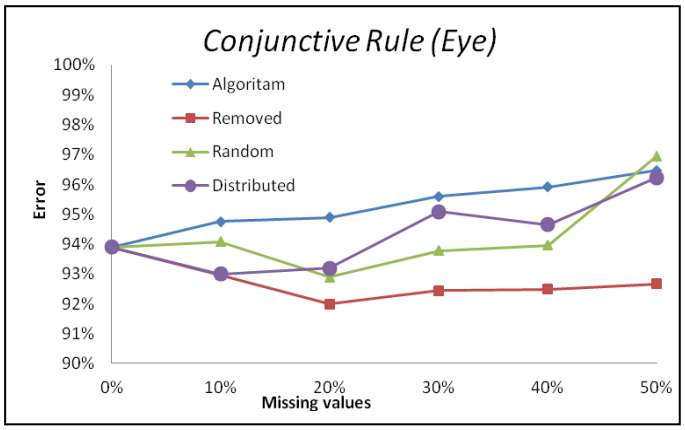

Fig. 5. Greška kod korištenja CR algoritma na skupu podataka Eye

REPTree algoritam na skupu podataka *Eye* daje najmanju grešku u odnosu na prethodna dva algoritma. Na potpunom skupu podataka greška iznosi 46,3%. S povećanjem nedostajućih vrijednosti greška jako brzo raste te doseže vrijednosti od 60% do 80%. Najveću i otprilike podjednako veliku grešku daje korištenje 2. i 3. pristupa. Može se reći da ponovno 1. pristup daje najmanju grešku, ali za 50% nedostajućih podataka počne brže rasti. Četvrti pristup je u ovom slučaju točniji nego kod CR algoritma. Oscilacija nema, već greška raste monotono (Slika 6.)

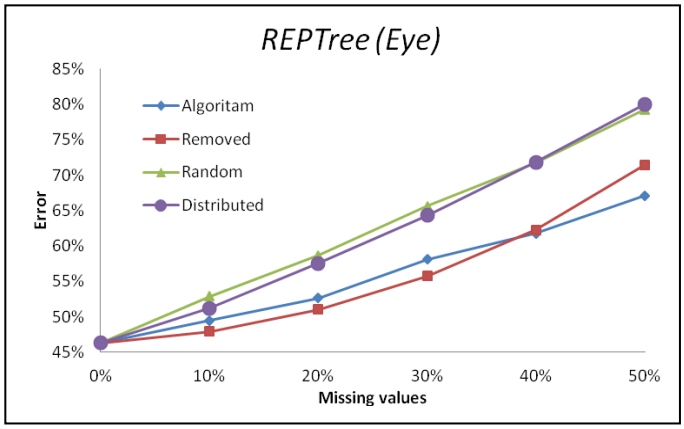

Fig. 6. Greška kod korištenja REPTree algoritma na skupu podataka Eye

Primjena DT algoritma na najvećem od tri skupa podataka (*Motion*) daje grešku od 30,8% u slučaju da nema nedostajućih vrijednosti. Ponovno se može vidjeti da 2. i 3. pristup daju veće greške nego ostala dva, barem kad je riječ o slučajevima 20% i 30% nedostajućih vrijednosti. Također opet se događa, s tim da je u ovom slučaju mnogo jače izraženo, da pristup s uklanjanjem instanci daje brzi porast greške pri povećanju udjela nedostajućih vrijednosti. Kada postoji mnogo atributa, nije dobro ukloniti sve instance kojima nedostaje vrijednost. Najmanju grešku daje 4. pristup. Greške rastu monotono i nema oscilacija. (Slika 7)

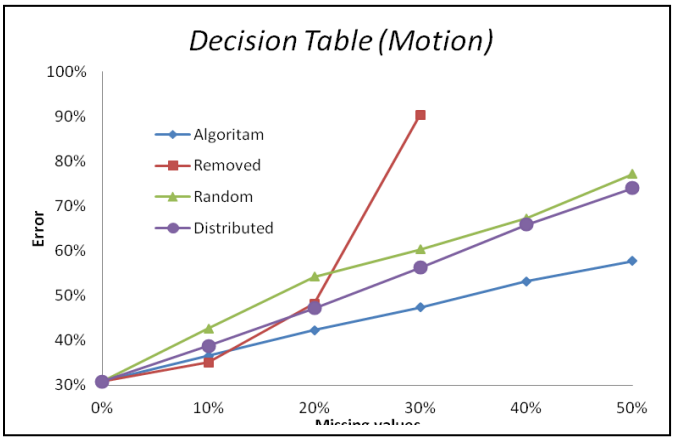

Fig. 7. Greška kod korištenja DT algoritma na skupu podataka Motion

Kod primjene CR algoritma na potpunom skupu podataka *Motion* greška je veća nego u slučaju DT algoritma i iznosi 75,6%. Primjećuje se da greška s povećanjem udjela nedostajućih vrijednosti raste relativno sporo. U ovom slučaju 1. pristup ima podjednaku grešku kao i 2. i 3. pristup. Greška kod pristupa s uklanjanjem instanci naglo raste kod 30% nedostajućih vrijednosti. Najmanju grešku ponovno daje pristup s učenjem na skupu podataka s nedostajućim vrijednostima. Ta razlika kod 30%, 40% i 50% nije toliko izražena, dok nagli pad greške u slučaju 10% i 20% nedostajućih vrijednosti izaziva sumnje. Ako se zanemari taj pad, može se reći da greška raste monotono, bez oscilacija. (Slika 8)

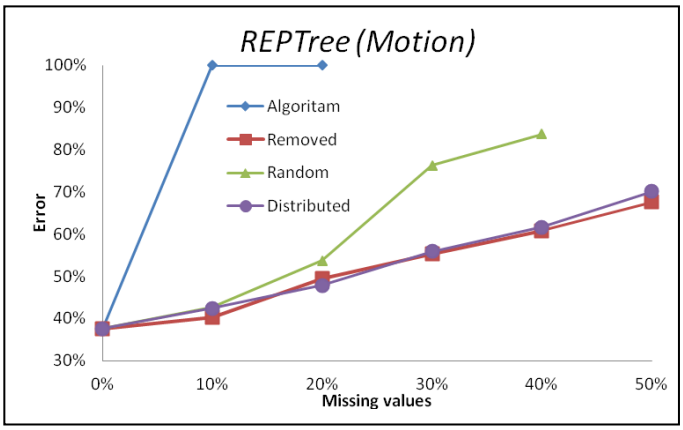

Fig. 8. Greška kod korištenja CR algoritma na skupu podataka Motion

REPTree algoritam je na potpunom skupu podataka *Motion* dao manju grešku nego CR algoritam i nešto veću grešku nego DT algoritam, a iznosi 37,6%. Ugrađeni algoritam za sređivanje nedostajućih vijednosti u programu *Weka* vjerojatno je zbog velike složenosti progutao cijelu memoriju što je zaustavilo funkcioniranje *Weka-e*. Također nedostaje podatak za 50% nedostajućih vrijednosti kod nasumične nadopune podataka. Primijećuje se da greška kod nasumične nadopune podataka raste jako brzo s porastom nedostajućih vrijednosti. Pristupi s uklanjanjem instanci i nadopunom prema raspodjeli daju otprilike podjednaku grešku, a rastu relativno brzo, otprilike kao i kod uporabe DT algoritma. Oscilacija nema. (Slika 9)

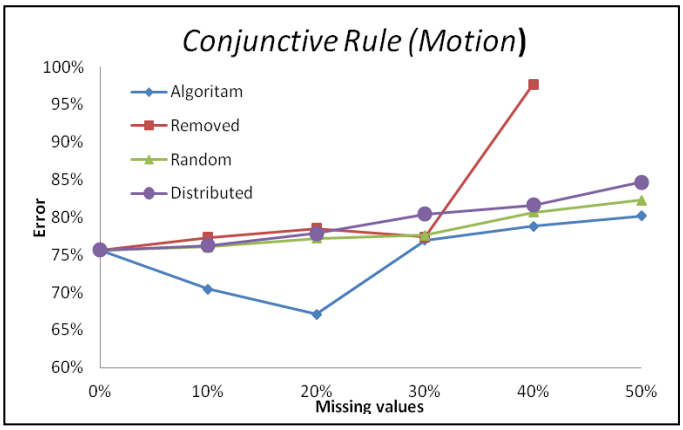

Fig. 9. Greška kod korištenja REPTree algoritma na skupu podataka Motion

IV. ZAKLJUČAK

Umetanje podataka može se vršiti statistički ili algoritmima za učenje. Bitno je u slučajevima kada nedostaju podaci, a poželjno je iskoristiti sve vrijednosti u podacima koje stoje na raspolaganju. Danas se, uz statističke metode umetanja podataka, koriste složenije metode strojnog učenja poput

višeslojnog perceptrona, samoorganizirajućih mapa, *k* najbližih susjeda, koje daju bolje rezultate od statističkih. [2]

Pojedini algoritmi strojnog učenja mogu se nositi s nedostajućim podacima, poput ovih korištenih u radu: *Decision Tree*, *Conjunctive Rule* i *REPTree*.

U radu su korištena četiri različita pristupa u postupanju s nedostajućim podacima. Umetanje nasumičnih vrijednosti i umetanje prema raspodjeli spadaju u statističke metode, korištenje spomenutih algoritama u metode strojnog učenja. Pristup s uklanjanjem instanci nije svrstan u ove dvije metode.

Pristup (4. pristup) u kojem se koristi algoritam za učenje na skupu podataka koji ima nedostajuće vrijednosti daje najmanju grešku. Može se reći da je od promatranih pristupa ovo najbolji pristup, ali valja imati na umu da je računalno zahtjevniji. Također, svi algoritmi koji su korišteni u radu imaju uključenu mogućnost da tretiraju skupove podataka s nedostajućim vrijednostima. Korištena je *10 fold cross validacija* ugrađena u program *Weka*.

Nešto veću grešku u odnosu na 4. pristup daje uklanjanje instanci koje imaju nedostajuće vrijednosti. No, također je primijećeno da kod udjela od 30% i 40% nedostajućih vrijednosti, greška u ovom pristupu počne puno brže rasti. Svakako se može zaključiti da ako nedostaje 30% ili više vrijednosti da uklanjanje instanci nije dobra ideja jer se počinje gubiti velika količina podataka, pogotovo kod podataka koji imaju veću količinu atributa (kao što je skup podataka *Motion* koji ima 51 atribut i 4833 instance). Kod skupa podataka u kojem količina instanci dominira, kao što je skupovi podataka *Eye* (15 atributa i 14976 instanci) i kontracepcija (9 atributa i 1473 instance), uklanjanje instanci manje se primijeti nego kod skupa kao što je *Motion*. Umetanje nedostajućih vrijednosti vršeno je nasumično te uklanjanje instanci iz skupa Motion rezultiralo s premalo preostalih instanci. Osim što algoritam radi sa dosta manjim skupom podataka, i cross validacija se vrši na znatno manjem broju podataka pa bi rezultate pristupa s uklanjanjem instanci trebalo uzeti s dozom rezerve.

Kod skupa podataka *Kontracepcija* izraženije su oscilacije greške. Dojam je da će skup podataka koji ima malo atributa (u ovom slučaju 9) pri nadopuni nedostajućih vrijednosti, posebice nasumičnoj, dati velike oscilacije greške. Uzrok ocilacija je i nasumična priroda umetanja nedostajućih vrijednosti.

Ono što je u prvi mah možda iznenađujuće (a što ni sami autori nisu očekivali) jest da algoritmi učeni na skupovima podataka u kojima su nedostajuće vrijednosti umetane nasumično i prema raspodjeli u atributima daju veće greške nego ostala dva pristupa. Valja imati na umu da su kod nasumičnog umetanja podataka i umetanja podataka po raspodjeli podaci umetani isključivo promatrajući svaki atribut posebno. Dakle, nije promatrano da li postoji ovisnost podataka pojedinog atributa o podacima u drugim atributima. Intuicija (pa i sama ideja da se nešto klasificira) nameće razmišljanje da takva povezanost postoji pa bi pravilniji pristup bio traženje međuovisnosti podataka u različitim atributima, (npr. u vidu statističke korelacije). Nastavak istraživanja mogao bi biti statističko umetanje podataka s time da se uzmu u obzir korelacije među atributima.

Također se za nastavak istraživanja može analizirati utjecaj umetanja nedostajućih vrijednosti po određenom atributu ili po nekoliko određenih atributa. U stvarnosti bi to bili atributi koje ljudi zbog privatnih razloga ili neugode ne žele navesti. Istraživanje se može proširiti na druge metode strojnog učenja, npr. clustering.

Reference

[1] G.Banerji, K.Saxena, An Efficient Classification Algorithm for Real Estate domain, International Journal of Modern Engineering Research, vol2., Issue 4, 2012, pp.2424-2430. Dostupno na: http://www.ijmer.com/papers/Vol2_Issue4/CK2424242430.pdf

[2] J.M.Jerez et al., Missing data imputation using statistical and machine learning methods in a real breast cancer problem, Artificial Intelligence in Medicine, vol.50, 2010, pp.105-115. Dostupno na: http://www.sciencedirect.com/science/article/pii/S0933365710000679

[3] S.Kalmegh, Analysis of WEKA Data Mining Algorithm REPTree, Simple Cart and RandomTree for Classification of Indian News, International Journal of Innovative Science, Engineering & Technology, Vol.2, 2015, pp. 438-436. Dostupno na: http://ijiset.com/vol2/v2s2/IJISET_V2_I2_63.pdf

[4] R.Kohavi, D.Sommerfield, Targeting Business Users with Decision Table Classifiers, American Association for Artificial Intellignence, KDD-98 Proceedings, 1998. Dostupno na: https://www.aaai.org/Papers/KDD/1998/KDD98-043.pdf

[5] M.F.bin Othman, T.M.S.Yau, Comparison of Different Classification Techniques Using WEKA for Breast Cabcer, Biomed 06, IFMBE Proceedings 15, pp.520-523, 2007

[6] Predavanja iz kolegija strojno učenje. Dostupno na http://web.math.pmf.unizg.hr/nastava/su/materijali

[7] Učenje pravila: https://mydatamining.wordpress.com/2008/04/14/ rule-learner-or-rule- induction

[8] S.Vijayarani, M., Muthulakshmi, Evaluating the Efficiency of Rule Techniques for File Classification, International Journal of Research in Engineering and Technology, 203, pp.38-42. Dostupno na http://esatjournals.org/Volumes/IJRET/2013V02/I10/IJRET20130210005.pdf

[9] Weka: http://www.cs.waikato.ac.nz/ml/weka/

[10] S.Zhang, Z.Qin, C.X.Ling, S.Sheng, "Missing is useful": Missing Values in Cost-Sensitive Decision Trees, IEEE Transactions on Knowledge and Data Engineering, vol.17, no.12, 2005. Dostupno na: http://pages.stern.nyu.edu/~ssheng/papers/tkde_missing05.pdf