Algoritamske i implementacijske optimizacije pristupa za traženje redeskripcija CLUS-RM

Jukić, Ivan

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivan Jukić

ALGORITAMSKE I IMPLEMENTACIJSKE OPTIMIZACIJE PRISTUPA ZA TRAŽENJE REDESKRIPCIJA CLUS-RM

Diplomski rad

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivan Jukić

ALGORITAMSKE I IMPLEMENTACIJSKE OPTIMIZACIJE PRISTUPA ZA TRAŽENJE REDESKRIPCIJA CLUS-RM

Diplomski rad

Voditelj rada:
doc. dr. sc. Matej Mihelčić

Zagreb, 2022.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Pretraživanje redeskripcija jest tehnika dubinske analize podataka koja se bavi pronalaskom različitih opisa ili karakterizacija istih ili sličnih skupova instanci, odnosno gledanje na skupove podataka iz više različitih perspektiva. Često se može naići na različite skupove podataka koji opisuju iste instance. To su najčešće podaci prikupljeni iz različitih izvora ili pomoću različitih metodologija. Pretraživanje redeskripcija omogućuje pronalazak različitih karakterizacija istih skupova podataka. Proizvod procesa pretraživanja redeskripcija n-torke su pravila koji se nazivaju *redeskripcije*.

Prvi algoritam nastao sa zadaćom pretraživanja redeskripcija bio je algoritam *CARTwheels* [9] koji se zasniva na ideji naizmjeničnog stvaranja stabala odluke preko jednog skupa podataka koji sadrži samo binarne vrijednosti. Nakon njega, nastali su algoritmi [10] koji su prihvaćali i druge tipove podataka, također bazirani na stablima odluke. Nastavno na njih, pojavljuje se CLUS-RM [12] koji koristi višeciljna prediktivna stabla klasteriranja (PCT) [6] za razliku od stabla odlučivanja.

Cilj je rada implementirati algoritamske optimizacije algoritma CLUS-RM koje je konceptualno osmislio doc. dr. sc. Matej Mihelčić, kao i poboljšati samu implementaciju algoritma. Rad je podijeljen na četiri poglavlja. U prvom se poglavlju definiraju osnovni pojmovi i uvodi se pojam redeskripcije i pretraživanja redeskripcija. U drugom se poglavlju opisuje pristup za traženje redeskripcija CLUS-RM. Opisuje se način rada algoritma CLUS-RM i algoritma CLUS za stvaranje višeciljnih prediktivnih stabala klasteriranja te kako se algoritam CLUS povezuje s algoritmom CLUS-RM. U trećem poglavlju formalno se opisuju algoritamske optimizacije algoritma i računa se vremenska složenost optimizacije. Uz to, opisuje se provedena implementacijska optimizacija algoritma. Zadnje poglavlje opisuje provedene eksperimente i zaključke o optimizacijama.

Poglavlje 1

Pretraživanje redeskripcija

1.1 Uvod

Dubinska analiza podataka područje je računarskih znanosti kojoj je cilj razvoj algoritama za analizu i otkrivanje znanja iz podataka. Pretraživanje redeskripcija dio je dubinske analize podataka i bavi se pronalaskom različitih opisa ili karakterizacija istih ili sličnih skupova instanci, kao i pronalaskom podskupova instanci koji se mogu karakterizirati na više načina. Pomoću otkrivenih opisa moguće je objasniti početni problem iz druge perspektive i time poboljšati razumijevanje početnog problema. S novim saznanjima može se doći i do nove hipoteze istraživanja. Procesom pretraživanja redeskripcija dobivaju se n-torke pravila koje se nazivaju *redeskripcije* [12].

1.2 Podaci

Formalne definicije u ovom poglavlju preuzete su iz [10].

Osnovni skup podataka $\mathcal{E}$ za zadatak traženja redeskripcija sadrži skup instanci i najmanje dva pogleda (disjunktna skupa atributa). Svaka instanca $e \in \mathcal{E}$ je opisana atributima iz skupa $\mathcal{A}$. Vrijednost atributa a za instancu e označava se kao $a(e)$.

Redeskripcije služe prikazivanju podataka na više načina. Atributi se dijele na više pogleda $\mathcal{V} = \{V_1, V_2, ..., V_k\}$. Za poglede vrijedi $V_i \cap V_j = \emptyset$ za sve $i \neq j$ i vrijedi $\bigcup_{i=1}^{k} V_i = \mathcal{A}$. Skup atributa se može podijeliti na dva ili više pogleda. Iako zadatak otkrivanja redeskripcija generalno može sadržavati k pogleda, u radu se koriste **dva pogleda**.

Formalna definicija modela podataka je [10]:

Definicija 1.2.1. *Podaci $\mathcal{D}$ za problem traženja redeskripcije je uređena trojka $\mathcal{D} = (\mathcal{E}, \mathcal{A}, \mathcal{V})$, gdje su instance $e \in \mathcal{E}$ opisane atributima iz $\mathcal{A}$ i atributi su podijeljeni u poglede $V \in \mathcal{V}$.*

Definicija 1.2.2. *U tabličnom prikazu podataka, podaci se sastoje od jedne ili više tablica* $\mathbf{D}_1, \mathbf{D}_2, ...$ *Retci u tablici predstavljaju instance, a stupci predstavljaju atribute. Svaki redak u svakoj tablici mora predstavljati iste instance.*

Tablica 1.1 prikazuje podatke u dva pogleda sadržana u dvije zasebne tablice. Lijeva tablica sastoji se od binarnih podataka, odnosno pokazuje postoji li životinja na određenom području. Druga tablica sastoji se od numeričkih podataka, odnosno pokazuje temperaturu za određeni mjesec u godini za pojedina područja.

instanca	jež	lisica	$\cdots$
Bjelovar	da	da	$\cdots$
Osijek	ne	da	$\cdots$
Rijeka	ne	ne	$\cdots$
Zadar	da	ne	$\cdots$
Zagreb	ne	da	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\cdots$

instanca	t_1	t_2	t_3	t_4	$\cdots$
Bjelovar	-0.2	2.0	6.4	11.4	$\cdots$
Osijek	-0.6	1.4	6.3	11.6	$\cdots$
Rijeka	5.8	6.3	9.0	12.8	$\cdots$
Zadar	7.2	7.6	9.9	13.4	$\cdots$
Zagreb	0.2	2.2	6.4	11.3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

Tablica 1.1: Primjer podataka prikazanih u dva pogleda.

Vidljivo je da se podaci zasnivaju na istim instancama (područja) i da su razdvojeni u dva pogleda, odnosno dvije tablice. Može se primijetiti da se ovaj skup podataka mogao prirodno podijeliti na poglede. Za prvi pogled postoje binarni podaci o postojanju životinja u određenom području, a u drugom su pogledu klimatske specifikacije svakog područja kao numerički podaci. Neke druge skupove podataka može biti teže podijeliti samo na dva pogleda.

1.3 Pravila

Atributi mogu biti kategorijski, numerički ili binarni. Za definiranje pravila, potrebno je povezati instance i atribute pomoću *predikata*. Predikat za atribut $a \in \mathcal{A}$ je funkcija $p_a : \mathcal{E} \rightarrow \{true, false\}$. Skup $\mathcal{L}$ sadrži sve predikate i negacije predikata. Predikati ovise o tipu atributa, a zapisuju se kao $p_a = [P_a(e)]$ gdje je $P_a(e)$ neka logička rečenica. Za binarne atribute, predikat vraća vrijednost instance za taj atribut. U slučaju kategorijskih atributa, predikati mogu izgledati kao $p_a(e) = [a(e) = X]$ gdje je X neka konstanta. Skraćeni način pisanja predikata je $[a = X]$. Za numeričke atribute se u većini slučajeva promatraju predikati kao $[a \leq x]$ ili $[a \geq x]$.

Deskripcija predstavlja pravilo $q : \mathcal{E} \rightarrow \{true, false\}$ koje zadaje vrijednost true ili false za svaku instancu $e \in \mathcal{E}$.

Definicija 1.3.1. *Neka je $q \in Q$ pravilo.* Podrška *$supp(q)$ je skup instanci koje pravilo q opisuje. Odnosno vrijedi $supp(q) = \{e \in \mathcal{E} : q(e) = true\}$. Skup atributa koji se pojavljuju u pravilu* q *označavaju se oznakom $att(q)$. Pogledi pravila q, views(q) je unija svih pogleda od svih atributa u $att(q)$.*

Za primjer tablice 1.1 jedno pravilo može glasiti *jež* $\wedge$ *lisica* (područja u kojima se nalaze i jež i lisica). Skup *att(q)* = {*jež, lisica*}. Može se vidjeti da je to pravilo istina za instancu *Bjelovar*. Iz toga se može vidjeti da je $supp(q) = \{Bjelovar\}$.

1.4 Redeskripcije

Redeskripcije predstavljaju n-torku pravila s disjunktnim pogledima i s dovoljno sličnim podrškama. Pogledi bi trebali biti disjunktni, odnosno skupovi atributa bi trebali biti disjunktni skupovi. Ovaj uvjet postoji da se vide poveznice između atributa jednog pogleda i atributa drugog pogleda. Zato je potrebno definirati način kako mjeriti razliku između skupova. Za to se može koristiti bilo koja funkcija udaljenosti $d : 2^{\mathcal{E}} \times 2^{\mathcal{E}} \rightarrow [0, \infty\rangle$. Funkcija udaljenosti mora zadovoljavati sljedeće uvjete:

(1) $d(X, Y) \geq 0$
(2) $d(X, Y) = 0$ ako i samo ako vrijedi $X = Y$
(3) $d(X, Y) = d(Y, X)$ za sve X, Y $\in 2^{\mathcal{E}}$

Najčešći odabir za udaljenost je Jaccard udaljenost, koja se zasniva na vrijednosti Jaccard indeksa sličnosti.

Definicija 1.4.1. *Jaccard indeks sličnosti J za skupove podrški dva pravila p i q se definira kao:*

$$J(supp(p), supp(q)) = \frac{|supp(p) \cap supp(q)|}{|supp(p) \cup supp(q)|}$$

Jaccard udaljenosti se definira kao:

$$d(supp(p), supp(q)) = 1 - J(supp(p), supp(q)) = 1 - \frac{|supp(p) \cap supp(q)|}{|supp(p) \cup supp(q)|}$$

Funkcijom udaljenosti možemo mjeriti koliko su slične podrške dvaju pravila. Ako su podrške jednake vrijedi $d(p, q) = 0$ pa je ta redeskripcija egzaktna i zapisuje se kao $p \equiv q$. U tom slučaju vrijednost Jaccard indeksa sličnosti iznosi 1.

Definicija 1.4.2. *Redeskripcija je par pravila* (p, q) *tako da vrijedi*

$$d(supp(p), supp(q)) \leq \tau,$$

gdje je τ *pozitivna konstanta zadana od strane korisnika. Vrijedi i*

$$views(p) \cap views(q) = \emptyset$$

da se vide poveznice između atributa jednog pogleda i atributa drugog pogleda

Definicija 1.4.3. *Neka je* (p, q) *redeskripcija za neki skup podataka* $\mathcal{D}$*. Podrška redeskripcije* (p, q) *je podrška pravila* $p \wedge q$ *u* $\mathcal{D}$*, odnosno,*

$$supp(p, q) = supp(p \wedge q) = supp(p) \cap supp(q)$$

Cilj pretraživanja redeskripcija jest pronaći sve redeskripcije (p_i, q_i) koje zadovoljavaju zadane uvjete.

1.5 P-vrijednost

Bitna je odrednica redeskripcija da su statistički značajne, odnosno da pokazuju neke nove informacije o podacima. Redeskripcije ne bi trebale proizlaziti nasumično iz distribucija podataka, odnosno preciznost redeskripcija ne bi trebala ovisiti o podršci pravila redeskripcije. Ako podrške pravila pojedinačno vrijede za velik broj instanci, tada je vjerojatno da će pravila imati veliki presjek međusobnih podrški. Mjerenje statističke značajnosti redeskripcija odvija se tako da se procijeni kolika je vjerojatnost da je redeskripcija nastala slučajno zbog distribucije podataka. Neka su $X \subseteq \mathcal{E}$ i $Y \subseteq \mathcal{E}$ neka dva nasumična neovisna skupa tako da vrijedi $p(e \in X) = \frac{|supp(p)|}{|\mathcal{E}|}$ i $p(e \in Y) = \frac{|supp(q)|}{|\mathcal{E}|}$ za pravila p i q redeskripcije. Neka je α vjerojatnost da je neka nasumična instanca $e \in \mathcal{E}$ sadržan u $X \cap Y$. Dobiva se:

$$\alpha = p(e \in X, q \in Y) = p(e \in X)p(e \in Y) = \frac{|X|}{|\mathcal{E}|}\frac{|Y|}{|\mathcal{E}|} = \frac{|supp(p)||supp(q)|}{|\mathcal{E}|^2}$$

Vjerojatnost da vrijedi $|X \cap Y| \geq |supp(p, q)|$ je:

Definicija 1.5.1.

$$pV(p, q) = \sum_{k=|supp(p,q)|}^{n} \binom{n}{k} \alpha^k (1 - \alpha)^{n-k}$$

Što predstavlja p-vrijednost.

Poglavlje 2

Pristup za traženje redeskripcija CLUS-RM

U ovom poglavlju opisuje se algoritam za pretraživanje redeskripcija CLUS-RM [12]. Ovaj pristup koristi višeciljnu klasifikaciju ili regresiju za pronalazak kvalitetnih i statistički značajnih redeskripcija. Algoritam koristi prediktivna stabla klasteriranja koja mogu istovremeno predviđati i učiti iz više ciljnih labela [6] za kreiranje hijerarhije pravila koja se kasnije transformiraju u redeskripcije. Koristeći slijed testova, stablo grupira instance u klastere koji postaju sve više homogeni po pitanju ciljnih labela, a slični po pitanju vrijednosti atributa prilikom spuštanja po stablu. Takvo stablo omogućava stvaranje modela za pronalazak više redeskripcija koristeći čvorove stabla na svim dubinama. Svaki čvor stabla predstavlja jedno pravilo koje se potom koristi kao ciljna labela za indukciju stabla na suprotnom pogledu.

2.1 Prediktivna stabla klasteriranja PCT

Opis i definicije stabla preuzete su iz [12] i [6].

Prediktivna stabla klasteriranja (PCT) povezuju klasteriranje i klasifikacijska / regresijska stabla odluke. PCT može koristiti informaciju o deskriptivnim atributima, ciljnim atributima ili oba tipa atributa za proizvodnju različitih klastera koji se koriste u predviđanjima (predikcijama).

Definicija 2.1.1. *Funkcija prototipa je funkcija* $p : 2^{\mathcal{E}} \rightarrow E$ *koja preslikava skup instanci* $E \subseteq \mathcal{E}$ *u prosjek koji se može smatrati jednom umjetnom konstruiranom instancom p(E), koja se naziva* ***prototip*** *od E.*

Definicija 2.1.2. *Za zadanu funkciju udaljenosti d za prostor instanci* $\mathcal{E}$*, funkcija prototipa je* ***idealna*** *ako i samo ako vrijedi*

$$\forall E \subseteq \mathcal{E} : \forall x \in \mathcal{E} : \sum_{e_i \in E} d(x, e_i)^2 \geq \sum_{e_i \in E} d(p(E), e_i)^2$$

Idealni prototip p(E) za svaki list za numeričke ciljne labele vraća n-torku predikcija što predstavlja srednju vrijednost svih vektora iz E, a vraća većinski razred (najčešća vrijednost) za kategorijske ciljne labele.

Definicija 2.1.3. *Funkcija klasteriranja je funkcija* $f : \mathcal{E} \times \boldsymbol{C} \to 2^{\mathcal{E}}$ *takva da*

$$\forall x \in \mathcal{E}, \forall E \subseteq \mathcal{E} : \forall C \in \boldsymbol{C} : f(x, C) \in C$$

odnosno, za svako klasteriranje C skupa instanci E, f preslikava svaku moguću instancu iz $\mathcal{E}$ *u klaster C.*

Zadaća prediktivnog klasteriranja je pronalazak klasteriranja C preko E koje maksimizira

$$Q(C) = -\mathbb{E}\left[d(x, p(f(x, C)))^2\right], \tag{2.1}$$

za svaki $x \in \mathcal{E}$.

Maksimiziranje vrijednosti Q odgovara minimiziranju vrijednosti očekivane varijance unutar klastera.

Zadaća prediktivnog klasteriranja može se primijeniti za klasifikaciju, kao i za regresiju. Za prostor instanci E i prediktivni prostor P, ciljna funkcija $\pi : I \to P$ funkcija je koja mapira instance u njihove ciljne vrijednosti. Prediktivna funkcija $pred_C : E \to P$ definira se kao $pred_C(e_i) = \pi(p(f(e_i, C)))$. Klasifikacija se definira kao posebni slučaj prediktivnog klasteriranja za kategorijske ciljne labele i $d(e_i, e_j) = d_1(\pi(e_i), \pi(e_j))$.

$$d_1(\pi(e_i), \pi(e_j)) = \begin{cases} 1, & \pi(e_i) = \pi(e_j) \\ 0, & \pi(e_i) \neq \pi(e_j) \end{cases} \tag{2.2}$$

Regresija je posebni slučaj prediktivnog klasteriranja gdje je π neprekinuta funkcija i ciljne labele su numeričke.

Prediktivna stabla klasteriranja su inducirana na način da izrada modela kreće od korijena stabla (top-down). Algoritam 1 opisuje indukciju prediktivnog stabla klasteriranja. Algoritam počinje tako da se pronalazi najbolji test koji dijeli instance iz skupa E. Test je uređeni par atributa i vrijednosti koji odlučuje kako će se skup podataka podijeliti na manje podskupove. Algoritam prvo pronađe najbolji uvjet podjele cijelog skupa i onda

rekurzivno dijeli te podskupove u manje podskupove dok se ne dostigne neki uvjet prestanka (npr. maksimalna dubina stabla, maksimalan broj instanci itd.). Funkcija *Prototip* koja za svaki list vraća n-torku predikcija (srednje vrijednosti za numeričke ciljne labele ili većinski razred za kategorijske).

Alg. 1 PCT [12]

Input Skup podataka E
Output Prediktivno stablo klasteriranja
1: (t*, h*, P*) = NajboljiTest(E)
2: **if** $t* \neq$ null **then**
3: **for each** $E_i \in P*$ **do**
4: **return** $tr_i = PCT(E_i)$
5: **end for**
6: **return** Čvor($t*, \cup_i tr_i$)
7: **else**
8: **return** List(Prototip(E_k))
9: **end if**

Algoritam 2 (redak 4) koristi heuristiku (h) za odabir najboljeg testa (t). Heuristika (h) predstavlja redukciju varijance zbog dijeljenja instanci na particije (P). Maksimiziranjem redukcije varijance povećava se homogenost klastera te se time poboljšavaju predikcije. Ako se ne može naći najbolji test, odnosno ako test ne smanji varijancu značajno, tada algoritam stvara list i računa prototip instanci koje spadaju u taj list (redak 8 algoritma 1).

Alg. 2 NajboljiTest [12]

Input Skup podataka E
Output Najbolji test (t*), heuristika (h*) i particija (P*) koja se inducira na skup podataka (E)
(t*, h*, P*) $\leftarrow$ (null, 0, $\emptyset$)
1: **for each** mogući test t **do**
2: P = particija koju t inducira na E
3: h = $Var(E) - \Sigma_{E_i \in P} \frac{|E_i|}{|E|} Var(E_i)$
4: **if** $h > H * \wedge Prihvatljivo(t, P)$ **then**
5: (t*, h*, P*) $\leftarrow$ (t, h, P)
6: **end if**
7: **end for**
8: **return** (t*, h*, P*)

Funkcija prototipa koja se računa za svaki list vraća vektor srednjih vrijednosti ako se radi o numeričkim ciljnim labelama, a u slučaju kategorijskih atributa vraća vektor vjerojatnosti da instanca pripada zadanom razredu za svaku ciljnu labelu. Iz toga se može izračunati koje kategorijske vrijednosti ima najviše.

Glavna razlika između prediktivnog stabla klasteriranja i standardnog stabla odlučivanja jest da prediktivna stabla klasteriranja koriste funkciju varijance i funkciju prototipa koje računaju predikcije za svaki list, s time grupiraju podatke i stvaraju pritom klastere.

Višeciljna prediktivna stabla klasteriranja stabla su koja mogu predvidjeti više ciljnih labela. Varijanca se računa kao suma varijanci svih ciljnih labela, $Var(E) = \sum_{i=1}^{T} Var(Y_i)$, gdje je T broj ciljnih labela. Varijance ciljnih labela normalizirane su tako da svaka ciljna labela jednako doprinese varijanci.

2.2 Pretraživanje redeskripcija s višeciljnim prediktivnim stablima klasteriranja

Algoritam CLUS-RM

Alg. 3 Pseudokod CLUS-RM algoritma [12].

Input Dva skupa podataka W_1 i W_2, minimalni Jaccard, maksimalna p-vrijednost, maksimalna podrška, minimalna podrška i broj iteracija κ.

Output Skup redeskripcija $\mathcal{R}$.

1: $\mathcal{R} \leftarrow \emptyset$
2: $[\tau_1, \tau_2] \leftarrow$ pripremanje ciljnih labela za inicijalni PCT od W_1 i W_2
3: $[T_1^{(0)}, T_2^{(0)}] \leftarrow$ induciranje stabala po τ_1 i τ_2
4: $[Q_1^{(0)}, Q_2^{(0)}] \leftarrow$ dobivanje pravila iz $T_1^{(0)}$ i $T_2^{(0)}$
5: **for each** $k \in \{1, 2, ..., \kappa\}$ **do**
6: **for each** $(s, t) \in \{(1, 2), (2, 1)\}$ **do**
7: $[\tau_s, \tau_t] \leftarrow$ pripremanje ciljnih labela od pravila iz $[Q_t^{(k-1)}, Q_s^{(k-1)}]$
8: $[T_s^{(k)}, T_t^{(k)}] \leftarrow$ induciranje stabala po D_s i D_t s dodanim ciljnim labelama τ_s i τ_t
9: $[Q_t^{(k)}, Q_s^{(k)}] \leftarrow$ dobivanje pravila iz $T_t^{(k)}$ i $T_s^{(k)}$
10: $\mathcal{R} \leftarrow$ redeskripcije za par pravila iz $Q_s^{(k)} \times Q_t^{(k)}$ koji zadovoljavaju uvjete
11: **end for**
12: **end for**
13: **return** $\mathcal{R}$

Algoritam CLUS-RM 3 započinje tako da se stvaraju inicijalna stabla za poglede W_1 i W_2 (linija 2 u Alg. 3) na način da se svakom pogledu dodaju negativne instance. Originalne

su instance one koje već postoje u svakom skupu podataka (pogledu), a negativne instance stvaraju se tako da se za svaku instancu u originalnom pogledu stvara po jedna negativna, umjetna (sintetička) instanca (tablice (2.3 i 2.4). Ideja iza ovog načina stvaranja inicijalnog skupa podataka jest da se razbiju korelacije između atributa. Navedeno se postiže nasumičnim miješanjem vrijednosti atributa između instanci za novonastale instance. Da bi se postigla potpuna nasumičnost, broj miješanja mora biti jednak broju atributa u pogledu te se svaka vrijednost atributa treba kopirati u umjetnu instancu za nasumično odabranu originalnu instancu.

instanca	vjetrovito	maglovito
E1	ne	da
E2	ne	da
E3	ne	ne
E4	da	ne
E5	ne	da

Tablica 2.1: Originalni skup podataka za pogled 1

instanca	temperatura	vlažnost zraka
E1	34.0	50.0
E2	30.0	55.0
E3	20.0	70.0
E4	11.0	75.0
E5	20.0	88.0

Tablica 2.2: Originalni skup podataka za pogled 2

Za skupove podataka opisane u tablici (2.1) koja predstavlja prvi pogled i u tablici (2.2) koja predstavlja drugi pogled, vidi se da je skup instanci $\mathcal{E} = \{E1, E2, E3, E4, E5\}$. Prvi se pogled sastoji od binarnih atributa vjetrovito = {da, ne} i maglovito = {da, ne} dok se drugi pogled sastoji od numeričkih atributa temperatura i vlažnost zraka. Svakom se pogledu uz umjetne instance dodaje i ciljna labela (eng. target) τ na način da su vrijednosti originalnih instanci za taj atribut jednake 1.0, a vrijednosti umjetnih atributa jednake 0.0. Tablica (2.3) i tablica (2.4) pokazuju inicijalne skupove podataka za poglede koji se kreiraju u koraku 2 algoritma. Skup instanci za inicijalne poglede je $\mathcal{E}_{inicijalni} = \{E1, E2, E3, E4, E5, E1', E2', E3', E4', E5'\}$ Nakon što se naprave inicijalni skupovi, u koraku 3 regresijom se induciraju stabla na ciljnu labelu koristeći ostale atribute kao deskriptivne. Svaki čvor stabla čini jedan klaster instanci [6].

instanca	vjetrovito	maglovito	ciljna labela
E1	ne	da	1.0
E2	ne	da	1.0
E3	ne	ne	1.0
E4	da	ne	1.0
E5	ne	da	1.0
$E1'$	ne	ne	0.0
$E2'$	da	da	0.0
$E3'$	ne	da	0.0
$E4'$	da	da	0.0
$E5'$	da	ne	0.0

Tablica 2.3: Inicijalni skup podataka za pogled 1

instanca	temperatura	vlažnost zraka	ciljna labela
E1	34.0	50.0	1.0
E2	30.0	55.0	1.0
E3	20.0	70.0	1.0
E4	11.0	75.0	1.0
E5	20.0	88.0	1.0
$E1'$	11.0	70.0	0.0
$E2'$	20.0	55.0	0.0
$E3'$	11.0	50.0	0.0
$E4'$	18.0	50.0	0.0
$E5'$	30.0	95.0	0.0

Tablica 2.4: Inicijalni skup podataka za pogled 2

U koraku 4 svaki se čvor dobivenih stabala pretvara u pravilo i formiraju se skupovi pravila $Q_1^{(0)}$ i $Q_2^{(0)}$ s tim da se eliminiraju umjetne instance iz $supp(q_i)$ za $q_i \in Q_1^{(0)}$ i $supp(q_i')$ za $q_i' \in Q_2^{(0)}$.

Slika 2.1 pokazuje stablo za inicijalni skup podataka za pogled 1 (2.3). Korijen stabla sadrži uvjet $vjetrovito = da$. Korijen se zatim grana na dva čvora s tim da čvor 3 predstavlja klaster instanci za kojeg vrijedi $vjetrovito = da$, a čvor 2 predstavlja klaster za kojeg uvjet ne vrijedi. Treba napomenuti da bi se za kategorijske atribute napravila podjela čvorova za svaku moguću alternativu. Čvorovi 2 i 3 sadrže uvjet po kojem se računaju djeca čvora. Svaki čvor stabla sadrži skup $\mathcal{S}_i$ koji sadrži instance za čvor stabla n_i. U slučaju ovog stabla, može se primijetiti da će skup $\mathcal{S}_i$ čvora n_i biti jednak skupu $supp(q_i)$ pravila q_i koje odgovara čvoru. Ta činjenica neće vrijediti u općenitom slučaju što će se pokazati u odjeljku 3.6. Stoga je potrebno definirati skup instanci čvora:

Definicija 2.2.1. *Za stablo $\mathcal{T}$, čvor $n_i \in \mathcal{T}$ definira se skup $\mathcal{S}_i \subseteq \mathcal{E}$ koji predstavlja skup instanci čvora. U slučaju da čvor n_i nema roditelja, odnosno n_i je korijen stabla $\mathcal{T}$, tada vrijedi $\mathcal{S}_i = \mathcal{E}$.*

Za pravilo q_i koje odgovara čvoru n_i općenito ne vrijedi da je podrška pravila jednaka skupu $\mathcal{S}_i$ čvora. Odnosno vrijedi:

Za stablo $\mathcal{T}$, čvor $n_i \in \mathcal{T}$ i pravilo q_i koje odgovara čvoru n_i vrijedi:

$$supp(q_i) \subseteq \mathcal{S}_i, \tag{2.3}$$

gdje je $\mathcal{S}_i$ skup instanci čvora n_i.

Dobivanje pravila iz stabla radi tako da se prati stablo odluke iz čvora i zapisuju se vrijednosti za svaki uvjet. Na primjer, pravilo za čvor 3 bit će $q_3 = (vjetrovito = da)$, dok će pravilo za čvor 6 biti $q_6 = (vjetrovito = da) \wedge \neg(maglovito = da)$.

Iz slike 2.1 se može vidjeti da će $\mathcal{S}_3 = supp(q_3) = \{E4, E2', E4', E5'\}$, a $\mathcal{S}_6 = supp(q_6) = \{E4, E5'\}$. Vrijednost *sw* predstavlja kardinalni broj skupa instanci čvora. Vrijednosti za čvorove 3 i 6 su $sw_3 = |\mathcal{S}_3| = |supp(q_3)| = 4$, a $sw_6 = |\mathcal{S}_6| = |supp(q_6)| = 2$. Može se definirati vektor vrijednosti atributa za instance kao $\overrightarrow{P_\tau} = (\tau(e_1), \tau(e_2), ...\tau(e_{10}))$ za instance $e_i \in \mathcal{S}_i$ za čvor n_i. Ako se izračunaju vrijednost atributa ciljne labele za svaku instancu, dobit će se $\overrightarrow{P_\tau} = (1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$.

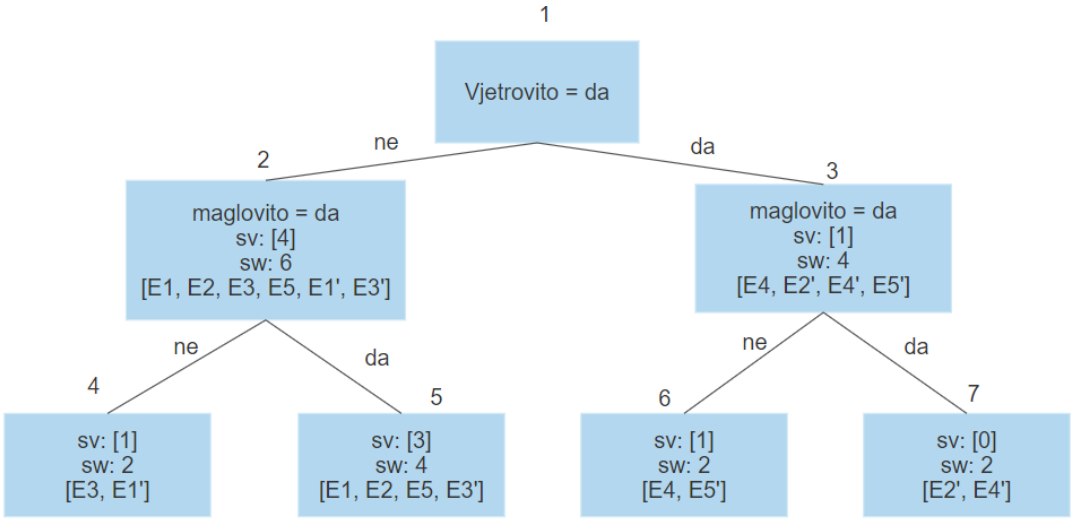
Zatim se može definirati vektor $\overrightarrow{sv}$ koji predstavlja zbroj vrijednosti ciljne labele za instance čvora. Budući da u ovom slučaju postoji samo jedna ciljna labela, dobit će se jednodimenzionalni vektor $\overrightarrow{sv} = (\sum_{i=1}^{sw} \tau(e_i))$ gdje je $sw = |\mathcal{S}|$, kardinalni broj skupa instanci čvora n. Može se vidjeti da su za ovaj primjer $\overrightarrow{sv}_3 = (1+0+0+0) = (1)$, a $\overrightarrow{sv}_6 = (1+0) = (1)$. Analogno se stvara inicijalno stablo za pogled 2 (Slika 2.2).

Sada se može definirati općeniti slučaj kada postoji više ciljnih labela.

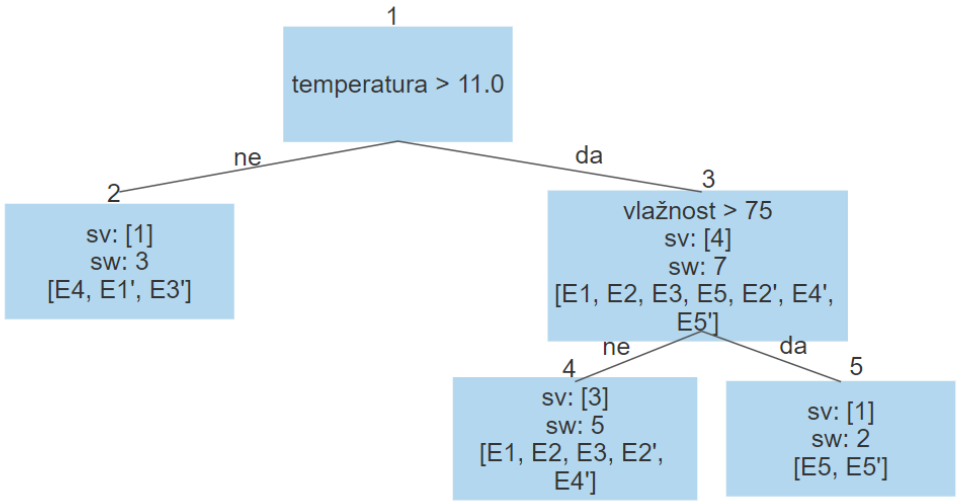
Definicija 2.2.2. *Za stablo $\mathcal{T}$, skup instanci $e_i \in \mathcal{S}$ čvora $n \in \mathcal{T}$ i skup ciljnih labela $\{\tau_1, \tau_2, ..., \tau_N\}$ definira se vektor:*

$$\overrightarrow{sv}_n = \left(\sum_{i=1}^{sw} \tau_1(e_i), \sum_{i=1}^{sw} \tau_2(e_i), ..., \sum_{i=1}^{sw} \tau_N(e_i) \right),$$

gdje je $sw = |\mathcal{S}|$.



Slika 2.1: Inicijalni PCT za pogled 1.



Slika 2.2: Inicijalni PCT za pogled 2.

U koraku 5 počinje iterativni proces gdje se stvaraju ciljne labele na temelju pravila dobivenih iz prethodne iteracije ili iz inicijalnog skupa pravila. Pravila dobivena iz prethodne iteracije $k-1$ za W_1 koriste se kao ciljne labele za W_2 u iteraciji k i obrnuto (korak 7 algoritma). U tablici 2.5 za pogled 2 za iteraciju k mogu se vidjeti dvije ciljne labele T_3 i T_6 koje su nastale iz pravila q_3 i q_6 iz pogleda 1 kojima su vrijednosti podrške $supp(q_3) = \{E4, E2', E4', E5'\}$ i $supp(q_6) = \{E4, E5'\}$. Umjetne instance za inicijalni korak eliminiraju se iz skupova opisanih instanci pravila pa je $supp(q_3) = \{E4\}$,

a $supp(q_6) = \{E4\}$. Sada se za vrijednost svake ciljne labele instance koja je u podršci pravila unosi 1.0, a inače 0.0. Za T_1 i T_2 može se vidjeti da je samo za instancu E4 unesena vrijednost 1.0, a za ostale instance vrijednost 0.0. Analogno se stvaraju ciljne labele za pogled 1.

instanca	temperatura	vlažnost zraka	T_3	T_6
E1	34.0	50.0	0.0	0.0
E2	30.0	55.0	0.0	0.0
E3	20.0	70.0	0.0	0.0
E4	11.0	75.0	1.0	1.0
E5	20.0	88.0	0.0	0.0

Tablica 2.5: Skup podataka za iteraciju k za pogled 2

Stavljanjem ciljnih labela u poglede vodi se konstrukcija stabla. Stvara se klaster koji opisuje isti skup instanci kao i odgovarajući čvor stabla od kojeg je napravljeno pravilo korišteno za konstrukciju ciljne labele. Na primjer, ciljna labela T_3 iz pogleda 2 u tablici 2.5 opisuje pravilo $vjetrovito = da$ iz pogleda 1, a ciljna labela T_3 iz pogleda 1 opisuje pravilo $temperatura > 11.0$.

Dobivena pravila iz pogleda 1 u iteraciji k spajaju se s dobivenim pravilima iz pogleda 2 u redeskripcije ako zadovoljavaju vrijednosti minimalnog Jaccard-a 3.2.1, maksimalne p-vrijednosti 1.5.1 te minimalne i maksimalne podrške. Osim toga, sva pravila iz pogleda 1 spajaju se s negiranim pravilima iz pogleda 2 i obrnuto, uz provjere istih uvjeta. Dodatno, pravila se poboljšavaju koristeći disjunkcije između opisa redeskripcije i pravila konstruiranih na odgovarajućem pogledu.

Poglavlje 3

Optimizacije algoritma CLUS-RM

3.1 Motivacija za algoritamske optimizacije algoritma CLUS-RM

Motivacija za algoritamske optimizacije algoritma CLUS-RM (Alg. 3) zasniva se na liniji 10 algoritma gdje se spajaju pravila koja zadovoljavaju uvjete iz pogleda 1 s pravilima iz pogleda 2. Za svaki par pravila (q_1, q_2) iz Kartezijevog umnoška $Q_1^{(k)} \times Q_2^{(k)}$ provjerava se vrijednost Jaccard indeksa 3.2.1 za koje je potrebno izračunati $|supp(q_1) \cap supp(q_2)|$. U tom slučaju uvjeti se provjeravaju $|Q_1^{(k)}| \cdot |Q_2^{(k)}|$ puta, odnosno za sva pravila iz skupa $Q_1^{(k)}$ provjerava se može li se spojiti sa svakim pravilom iz skupa $Q_2^{(k)}$.

Instance pravila spremaju se u HashSet strukturu podataka [2]. Budući da se za pronalazak presjeka dvaju skupova iterira po prvom skupu i za svaku instancu provjerava sadrži li drugi skup tu instancu, složenost presjeka dviju podrški u algoritmu je $O(|supp(q_1)|)$ u slučaju kada je složenost pretrage instance $O(1)$. Tada se pretpostavlja da se koristi dobra hash funkcija koja nema previše kolizija, odnosno da nije previše instanci pod istim hash indeksom. U najgorem slučaju složenost pretrage je $|supp(q_2)|$ [3] pa je složenost provjere Jaccard indeksa $O(|supp(q_1|) \cdot |supp(q_2|)$. Ideja je optimizacije da se zamijeni takva provjera vrijednosti Jaccard indeksa koristeći statistike koje se računaju prilikom stvaranja prediktivnih stabala. Na taj način bilo bi moguće u potpunosti ukloniti provjeru Jaccard indeksa preko skupova.

3.2 Računanje vrijednosti Jaccard indeksa bez nedostajućih vrijednosti

Računanje vrijednosti Jaccard indeksa

Prilikom rekurzivne indukcije stabla algoritma 3, algoritam CLUS za svaki čvor računa vrijednosti vektora $\overrightarrow{sv} = (s_{\tau_1}, s_{\tau_2}, ..., s_{\tau_n})$ iz definicije 2.2.2 i broj instanci čvora (sw). Za čvor stabla konstruiranog koristeći atribute iz pogleda 1 koji predstavlja pravilo q_1 i ciljne labele nastale iz pravila drugog pogleda iz prethodne iteracije (τ_1) može se vidjeti da vrijednost sv_1 predstavlja broj instanci sadržanih u podršci pravila q_1 koje imaju vrijednost ciljne labele 1. To je onda presjek podrški tih dvaju pravila. Vrijednost sw predstavlja podršku pravila q_1. Uz podršku ciljne labele iz tih vrijednosti, može se izračunati vrijednost Jaccard indeksa.

Sada se može definirati vrijednost Jaccard indeksa.

Definicija 3.2.1. *Za ciljnu labelu τ_i iz skupa ciljnih labela $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$ i vektor $\overrightarrow{sv} = (s_{\tau_1}, s_{\tau_2}, ..., s_{\tau_i}, ..., s_{\tau_n})$ za pravilo q koje odgovara čvoru n stabla definira se Jaccard indeks:*

$$J(supp(q), supp(\tau_i)) \overset{def.3.2.1}{=} \frac{|supp(q) \cap supp(\tau_i)|}{|supp(q) \cup supp(\tau_i)|} = \frac{s_{\tau i}}{sw - s_{\tau i} + |supp(\tau_i)|},$$

gdje je $sw = |\mathcal{S}| = |supp(q)|$ za skup instanci $\mathcal{S}$ čvora n *i pogledi W_1 i W_2* ***nemaju nedostajuće vrijednosti****. Podrška ciljne labele τ_i se računa kao podrška odgovarajućeg pravila iz prijašnje iteracije iz suprotnog pogleda.*

Primjer 3.2.2. *Pretpostavimo da se za skup primjera $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$ za neka dva pogleda induciralo stablo sa ciljnom labelom τ_i. Skup instanci $\mathcal{S}$ čvora* n *je skup $\mathcal{S} = supp(q) = \{e_1, e_2, e_3\}$ gdje je* q *pravilo koje odgovara čvoru* n*. Podrška ciljne labele τ_i je skup $\{e_1, e_3, e_4, e_5\}$.*

Može se vidjeti da je $\overrightarrow{sv} =$ (2) budući da ciljna labela i čvor dijele instance e_1 i e_3. Broj sw *predstavlja broj instanci čvora, što je u ovom slučaju $sw = |\mathcal{S}| = |supp(q)| = 3$. Iz toga se može izračunati vrijednost presjeka pravila q i negirane ciljne labele, odnosno vrijedi $|supp(q) \cap supp(\tau_i)| = s_{\tau_i} = 2$ što je točno broj elemenata u presjeku podrške ciljne labele i skupa $\mathcal{S}$. Unija je onda $sw - s_{\tau_i} + |supp(\tau_i)| = 3 - 2 + 4 = 5$.*

Na ovaj se način vrijednost Jaccard indeksa više ne mora računati iteriranjem po skupovima podrške pravila, nego se mogu iskoristiti već izračunate vrijednosti algoritma CLUS. Na taj se način u rekurzivnoj indukciji stabla može izračunati koji parovi pravila će se spajati.

Osim spajanja pravila iz jednog pogleda s drugim, algoritam CLUS-RM podržava i spajanje negacije pravila sa svake strane. U tom je slučaju potrebno definirati kako se računaju vrijednosti Jaccard indeksa za spajanje pravila s negiranim ciljnim labelama i spajanjem negiranih pravila sa ciljnim labelama.

3.3 Računanje vrijednosti Jaccard indeksa za negirane ciljne labele

Definicija 3.3.1. *Podrška negacije pravila q za skup instanci pogleda $\mathcal{E}$ računa se kao skupovna razlika skupova instanci pogleda i podrške pravila,*

$$supp(\neg q) = \mathcal{E} \setminus supp(q)$$

Ako se uzme u obzir definicija 2.2.2 vektora $\overrightarrow{sv} = (s_{\tau_1}, s_{\tau_2}, ..., s_{\tau_i}, ..., s_{\tau_n})$ može se vidjeti da je presjek negacije ciljne labele τ_i i pravila koje odgovara čvoru jednak $|\mathcal{S}| - s_{\tau i}$ ili $|supp(q)| - s_{\tau i}$ u slučaju kada nema nedostajućih vrijednosti. Ovaj broj predstavlja broj instanci za koje je vrijednost ciljne labele jednaka 0.0, a to se točno može dobiti oduzimanjem podrške pravila koje odgovara čvoru stabla s brojem instanci za koje je vrijednost jedinica odgovarajuće ciljne labele τ_i. U suprotnosti, broj $s_{\tau i}$ predstavlja broj instanci čvora za koje je vrijednost ciljne labele jednak 1.0. S tim se može definirati vrijednost Jaccard indeksa negirane ciljne labele i pravila.

Definicija 3.3.2. *Za negiranu ciljnu labelu $\neg\tau_i$ ciljne labele τ_i iz skupa $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, vektor $\overrightarrow{sv} = (s_{\tau_1}, s_{\tau_2}, ..., s_{\tau_i}, ..., s_{\tau_n})$ za pravilo q koje odgovara čvoru* n *stabla definira se Jaccard indeks:*

$$J(supp(q), supp(\neg\tau_i)) = \frac{sw - s_{\tau i}}{sw - (sw - s_{\tau i}) + |supp(\neg\tau_i)|} = \frac{sw - s_{\tau i}}{s_{\tau i} + |supp(\neg\tau_i)|},$$

odnosno koristeći definiciju 3.3.1:

$$J(supp(q), supp(\neg\tau_i)) \overset{def.3.3.1}{=} \frac{sw - s_{\tau i}}{s_{\tau i} + |\mathcal{E} \setminus supp(q)|},$$

gdje je $sw = |\mathcal{S}| = |supp(q)|$ za skup instanci $\mathcal{S}$ čvora n *i pogledi W_1 i W_2* ***nemaju nedostajuće vrijednosti****.*

Primjer 3.3.3. *Pretpostavimo da se za skup primjera $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$ za neka dva pogleda induciralo stablo sa ciljnom labelom τ_i. Skup instanci $\mathcal{S}$ čvora* n *je skup $\mathcal{S} =$*

$supp(q) = \{e_1, e_2, e_3\}$ gdje je q *pravilo koje odgovara čvoru* n*. Podrška ciljne labele τ_i je skup $\{e_1, e_3, e_4, e_5\}$.*

Može se vidjeti da je $\overrightarrow{sv} = (2)$ budući da ciljna labela i čvor dijele instance e_1 i e_3. Broj sw *predstavlja broj instanci čvora, što je u ovom slučaju $sw = |\mathcal{S}| = |supp(q)| = 3$. Iz toga se može izračunati vrijednost presjeka pravila q i negirane ciljne labele, odnosno po definiciji 3.3.2 vrijedi $|supp(q) \cap supp(\neg\tau_i)| = sw - s_{\tau_i}$.*

Vrijedi $sw - s_{\tau_i} = 3 - 2 = 1$, a s druge strane podrška negirane ciljne labele je po definiciji 3.3.1 jednaka $supp(\neg\tau_i) = \{e_2\}$. Iz toga se može vidjeti da je $|supp(\neg\tau_i)| = 1$, isto što se dobilo prethodnim računom.

Vrijedi $|\mathcal{E} \setminus supp(q)| = 2$, pa je unija po definiciji 3.3.2 jednaka $s_{\tau i} + |\mathcal{E} \setminus supp(q)| = 4$ što je broj elemenata unije podrške pravila q i ciljne labele τ_i.

3.4 Računanje vrijednosti Jaccard indeksa za negirana pravila koja odgovaraju čvorovima stabla

Budući da algoritam CLUS-RM podržava negiranje pravila konstruirana na proizvoljnom pogledu, potrebno je definirati računanje vrijednosti Jaccard indeksa za negirana pravila koja odgovaraju čvorovima stabla.

Definicija 3.4.1. *Za negaciju pravila q koje odgovara čvoru* n *stabla, ciljnu labelu τ_i iz skupa ciljnih labela $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, vektor $\overrightarrow{sv} = (s_{\tau 1}, s_{\tau 2}, ..., s_{\tau i}, ..., s_{\tau n})$, definira se Jaccard indeks:*

$$J(supp(\neg q), supp(\tau_i)) = \frac{|supp(\tau_i)| - s_{\tau i}}{|supp(\tau_i)| + |\mathcal{E}| - sw - (|supp(\tau_i)| - s_{\tau i})} = \frac{|supp(\tau_i)| - s_{\tau i}}{|\mathcal{E}| - sw + s_{\tau i}},$$

gdje je $sw = |\mathcal{S}| = |supp(q)|$ za skup instanci $\mathcal{S}$ čvora n*, $\mathcal{E}$ je skup instanci pogleda i pogledi W_1 i W_2* ***nemaju nedostajuće vrijednosti****.*

Primjer 3.4.2. *Pretpostavimo da se za skup primjera $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$ za neka dva pogleda induciralo stablo sa ciljnom labelom τ_i. Skup instanci $\mathcal{S}$ čvora* n *je skup $\mathcal{S} = supp(q) = \{e_1, e_2, e_3\}$ gdje je* q *pravilo koje odgovara čvoru* n*. Podrška ciljne labele τ_i je skup $\{e_1, e_3, e_4, e_5\}$.*

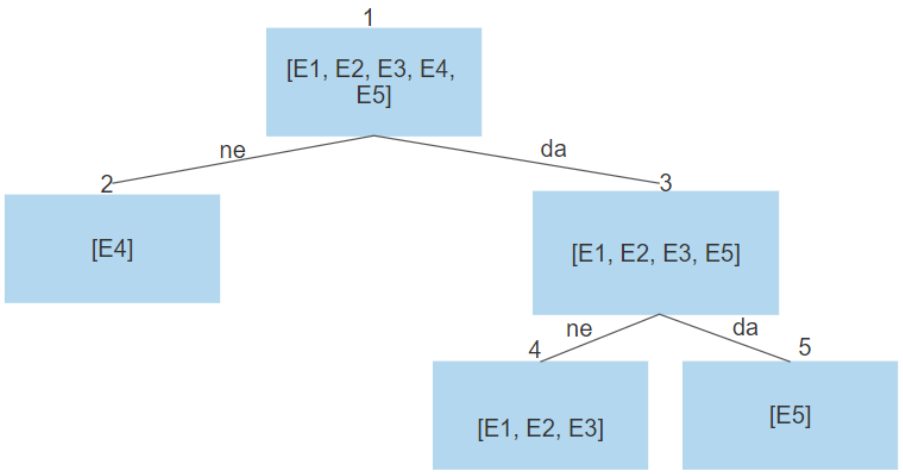
Može se vidjeti da je $\overrightarrow{sv} = (2)$ budući da ciljna labela i čvor dijele instance e_1 i e_3. Broj instanci podrške ciljne labele je 4. Po definiciji 3.4.1 vidljivo je da je veličina presjeka negiranog pravila koje odgovara čvoru n *i ciljne labele jednak $|supp(\tau_i)| - s_i = 4 - 2 = 2$.*

Rezultat se može provjeriti tako da se izračuna podrška negiranog pravila po definiciji 3.3.1. Skupovna razlika skupa $\mathcal{E}$ i broj instanci čvora je skup $\{e_4, e_5\}$. Broj instanci tog skupa je 2 što je jednako računu iz prvog dijela.

Po definiciji 3.4.1 unija se računa kao $|\mathcal{E}| - sw + s_i = 5 - 3 + 2 = 4$ što je jednako broju instanci unije negiranog čvora i ciljne labele.

3.5 Alternativno računanje vrijednosti Jaccard indeksa za negirana pravila koja odgovaraju čvorovima stabla

Za pronalazak presjeka pravila q_k koje odgovara nekom čvoru n_k i neke ciljne labele može se koristiti i alternativni pristup za slučaj kada nema nedostajućih vrijednosti. Broj instanci podrške negiranog čvora može se naći tako da se promatraju svi čvorovi braće tog čvora na istoj razini i svakog lista stabla koji se nalazi na razini iznad čvora ili na istoj razini. Ako se definira skup $Q = \{q_1, q_2, ...q_n\}$ kao skup pravila koja odgovaraju čvorovima na istoj razini kao čvor n_k kojem odgovara pravilo q_k i svakog lista stabla koji se nalazi na razini iznad čvora ili na istoj razini, broj instanci podrške negacije, odnosno $sw_{\neg q} = \sum_{i=1,i\neq k}^{n}(sw_i)$. Sada se može vidjeti da vrijedi $sw_{\neg q} = |\mathcal{E}| - sw_q$ budući da svi čvorovi braće i listovi na istoj razini ili višoj sadrže cijeli skup instanci. Ako se od tog broja instanci oduzme broj instanci podrške za pravilo, dobije se broj instanci podrške negiranog pravila. Na sličan način se računa i vrijednost presjeka ciljne labele τ i pravila $\neg q$. $sv_\tau = \sum_{i=1,i\neq k}^{n}(sv_{\tau i})$



Slika 3.1: Prediktivno stablo s instancama $\mathcal{E}$

Primjer 3.5.1. *Slika 3.1 pokazuje neko stablo za skup instanci* $\mathcal{E} = \{E_1, E_2, E_3, E_4, E_5\}$. *Ako se uzme q kao pravilo koje odgovara čvoru 5, može se vidjeti da je* $supp(q) = \{E_5\}$. *Po definiciji 3.3.1* $supp(\neg q) = \mathcal{E} \setminus supp(q) = \{E_1, E_2, E_3, E_4\}$.
Podrška negiranog pravila q izračunata na opisani način jednaka je $supp(\neg q) = \{E_1, E_2, E_3\} \cup \{E_4\} = \{E_1, E_2, E_3, E_4\}$.

Vrijednost Jaccard indeksa zatim se računa kao $J(\neg q, \tau) = \frac{sv_\tau}{sw_{\neg q} + |supp(\tau)| - sv_\tau}$. Ovaj alternativni pristup sporiji je od prethodnog pristupa po tome što postoji sumiranje po svoj braći i listovima čvora. Osim toga, računanje vrijednosti unije i presjeka moguće je napraviti tek nakon što se inducira stablo jer se vrijednosti $\overrightarrow{sv}$ i sw za svaki čvor računaju kod stvaranja novih čvorova. Implementacija kod ovog pristupa uključuje dva prolaska kroz stablo u suprotnosti s jednim prolaskom kod prvog pristupa.

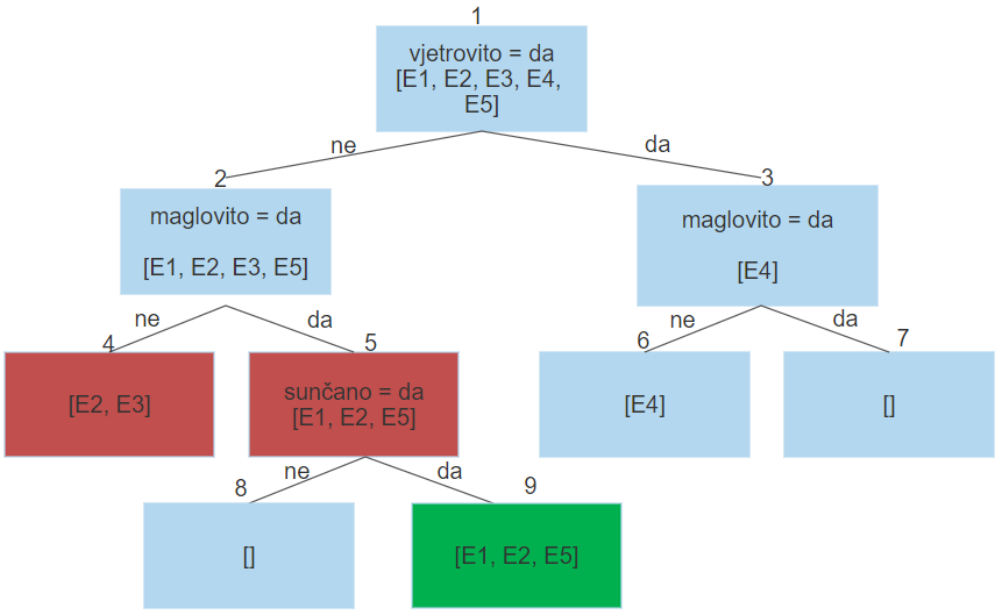
3.6 Računanje vrijednosti Jaccard indeksa na podacima s nedostajućim vrijednostima

Podaci koji nedostaju ili nedostajuće vrijednosti javljaju se kada nije poznata vrijednost atributa za neku instancu. U tablici 3.1 može se vidjeti da za atribut *maglovito* instanca $E2$ nema definiranu vrijednost. Oznaka za nedostajuću vrijednost je upitnik (?).

instanca	vjetrovito	maglovito	sunčano
E1	ne	da	da
E2	ne	?	da
E3	ne	ne	ne
E4	da	ne	ne
E5	ne	da	da

Tablica 3.1: Pogled s nedostajućom vrijednosti

Slika 3.2 prikazuje jedno stablo za pogled iz tablice 3.1. Na stablu je moguće vidjeti da skupovi instanci $\mathcal{S}_4$ i $\mathcal{S}_5$ čvorova 4 i 5 sadrže istu instancu $E2$ za koju vrijednost za atribut *maglovito* nedostaje. Budući da je vrijednost instance $E2$ nedostajuća za atribut *maglovito*, dodaje se u podrške sve djece. U tom slučaju za računanje presjeka pravila koje odgovara čvoru i ciljne labele trebaju se oduzeti nedostajuće vrijednosti od sw i $\overrightarrow{sv}$. Osim toga može se primijetiti da skup instanci čvora 9 $\mathcal{S}_9$ također sadrži instancu $E2$, u tom slučaju se iz podrške odgovarajućeg pravila mora maknuti ta instanca. Pravilo koje odgovara čvoru 9 ne vrijedi za instancu $E2$ jer jedan od roditelja sadrži nedostajuću vrijednost. Ovu instancu potrebno je maknuti iz podrške pravila koje odgovara čvoru 9.



Slika 3.2: Prediktivno stablo s nedostajućom vrijednosti

Za primjer iz slike 3.2 može se vidjeti razlog iza tvrdnje 2.3 koja govori da je podrška pravila koja odgovara čvoru podskup skupa instanci čvora $\mathcal{S}$ definiranog u definiciji 2.2.1. Skup instanci čvora može sadržavati instance koje imaju nedostajuće vrijednosti. Ako pravilo koje odgovara čvoru sadrži atribut za kojeg određena instanca ima nedostajuću vrijednost, ta instanca se neće nalaziti u podršci pravila. Budući da se kod korištenja novog pristupa računanja Jaccard indeksa koristi vrijednost skupa instanci čvora $\mathcal{S}$, potrebno je kod provjere uzeti u obzir da će kod računanja vrijednosti sw i $\overrightarrow{sv}$ biti uključene instance koje imaju nedostajuće vrijednosti za neki atribut. Za čvor 9 vrijedi $\mathcal{S}_9 = \{E1, E2, E5\}$, a za pravilo $\neg$(vjetrovito = da) $\wedge$ (maglovito = da) $\wedge$ (sunčano = da) koje odgovara čvoru 9 podrška je $supp(q_9) = \{E1, E5\}$. Ta dva skupa nisu jednaka, odnosno vrijedi $supp(q_9) \subseteq \mathcal{S}_9$.

Algoritam 4 prikazuje postupak otkrivanja skupa nedostajućih vrijednosti za pojedine čvorove koji se odrađuje za svaki čvor stabla. Za svaki novostvoreni čvor stabla prilikom indukcije provodi se pretraga nedostajućih vrijednosti za skup instanci $\mathcal{S}$ tog čvora. Iterira se po svakoj instanci iz skupa $\mathcal{S}$ (korak 2 algoritma). Ako je vrijednost predikata

za atribut uvjeta čvora nedostajuć, u skup nedostajućih vrijednosti dodaje se odgovarajuća instanca (koraci 3 i 4 algoritma). Inače se pretražuje po skupu nedostajućih vrijednosti svih roditelja pa ako je instanca sadržana u tom skupu, ona se isto dodaje skupu nedostajućih vrijednosti čvora (koraci 8, 9 i 10 algoritma). Vrijednost *sw* čvora ažurira se u retcima 5 i 11 na način da se oduzme jedan ako postoji nedostajuća vrijednost u čvoru. Može se primijetiti da će vrijediti $sw' = sw - |\mathcal{N}|$ čvora.

Alg. 4 Pseudokod pretrage nedostajućih vrijednosti za čvor.

Input Čvor n, skup atributa ciljnih labela $\mathcal{T}$, atribut a za uvjet čvora, kolekcija skupova nedostajućih vrijednosti svih roditelja čvora $\mathcal{N}_{rod}$ dobivenih algoritmom 4, skup instanci $\mathcal{S}$ čvora n, broj *sw* čvora i $\overrightarrow{sv}$.

Output Skup nedostajućih vrijednosti za čvor $\mathcal{N}$.

```
1:  N ← ∅
2:  for each e ∈ S do
3:      if a(e) = ? then
4:          N ← N ∪ e
5:          sw ← sw − 1
6:          ažuririrranjeVrijednostiSV(sv, e, T, a)
7:      else
8:          for each N_i ∈ N_rod do
9:              if e ∈ N_i then
10:                 N ← N ∪ e
11:                 sw ← sw − 1
12:                 ažuririrranjeVrijednostiSV(sv, e, T, a)
13:                 break
14:             end if
15:         end for
16:     end if
17: end for
18: return N
```

Algoritam 5 opisuje funkciju *ažuriranjeVrijednostiSV* za retke 6 i 12 algoritma 4. Algoritam ažurira vrijednosti vektora $\overrightarrow{sv}$ čvora n i vrijednosti *sw* čvora ovisno o tome je li vrijednost instance za atribut ciljne labele jednak jedan.

Alg. 5 Pseudokod ažuriranjeVrijednostiSV

Input $\overrightarrow{sv} = (s_{\tau 1}, s_{\tau 2}, ..., s_{\tau i}, ..., s_{\tau n})$ za čvor n, instanca e, skup atributa ciljnih labela $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$ i atribut a za uvjet čvora.

```
1: i ← 0
2: while i < |T| do
3:     if τ_i(e) = 1 then
4:         s_τi ← s_τi − 1
5:     end if
6: end while
```

S time se formalno može definirati novi vektor $\overrightarrow{sv}'$:

Definicija 3.6.1. *Za skup nedostajućih vrijednosti $\mathcal{N} = \{e_1, e_2, ..., e_m\}$, skup ciljnih labela $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$ i vektor $\overrightarrow{sv} = (s_{\tau_1}, s_{\tau_2}, ..., s_{\tau_i}, ..., s_{\tau_n})$ čvora* n *definira se vektor:*

$$\overrightarrow{sv}' = \left(s_{\tau_1} - \sum_{k=1}^{m} \tau_1(e_k), s_{\tau_2} - \sum_{k=1}^{m} \tau_2(e_k), ..., s_{\tau_i} - \sum_{k=1}^{m} \tau_i(e_k), ..., s_{\tau_n} - \sum_{k=1}^{m} \tau_n(e_k) \right).$$

Skraćeno se može pisati:

$$\overrightarrow{sv}' = (s'_{\tau_1}, s'_{\tau_2}, ..., s'_{\tau_i}, ..., s'_{\tau_n}).$$

Sada se može definirati računanje vrijednosti Jaccard indeksa nad podacima koji sadrže nedostajuće vrijednosti.

Definicija 3.6.2. *Za ciljnu labelu τ_i iz skupa ciljnih labela $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, vektor $\overrightarrow{sv}' \overset{def.3.6.1}{=} (s'_{\tau_1}, s'_{\tau_2}, ..., s'_{\tau_i}, ..., s'_{\tau_n})$ i skup nedostajućih instanci čvora $\mathcal{N}$. Za pravilo q koje odgovara čvoru* n *stabla definira se Jaccard indeks:*

$$J(supp(q), supp(\tau_i)) = \frac{s'_{\tau i}}{sw' - s'_{\tau i} + |supp(\tau_i)|},$$

gdje je $sw' = sw - |\mathcal{N}|$.

Primjer 3.6.3. *Pretpostavimo da se stablo sa slike 3.2 induciralo sa ciljnom labelom τ_i kojoj je podrška skup $\{E1, E2\}$ gdje instanca E2 ima nedostajuću vrijednost za atribut* maglovito. *Vrijedi $\mathcal{S}_5 = \{E1, E2, E3\}$ i $supp(q_5) = \{E1, E5\}$.*

Vrijedi $s_{\tau_i} = 2$ za čvor 5 za ciljnu labelu τ_i budući da ciljna labela dijeli elemente E1 i E2 sa skupom $\mathcal{S}_5$. Za skup nedostajućih vrijednosti vrijedi $\mathcal{N}_5 = \{E2\}$. Budući da je instanca E2 sadržana u podršci ciljne labele, vrijedi $s'_{\tau_i} = 2 - 1 = 1$. Može se vidjeti da je to točno kardinalni broj presjeka podrške ciljne labele i pravila q_5. Vrijedi $sw'_5 = sw_5 - |\mathcal{N}_5| = 3 - 1 = 2$. Što je kardinalni broj podrške pravila q_5.

Može se primijetiti da ako skup ne sadrži nedostajuće vrijednosti, onda će vrijediti $sw' = sw$ i vrijedit će $s'_{\tau i} = s_{\tau i}$ po definiciji 3.6.1. U tom slučaju će definicija 3.6.2 biti jednaka definiciji 3.2.1. Definicija 3.6.2 onda vrijedi kao općeniti slučaj koji podržava skupove s nedostajućim vrijednostima i skupove bez nedostajućih vrijednosti.

3.7 Računanje vrijednosti Jaccard indeksa za negirana pravila koja odgovaraju čvorovima stabla

Kod računanja podrške negacije pravila koji odgovara čvoru stabla, treba u obzir uzeti i nedostajuće vrijednosti. Potrebno je utvrditi postoji li uvjet čvora ili uvjet roditelja za koji instanca ima definiranu vrijednost i kriterij uvjeta nije istina. Takve instance spadaju u negaciju (retci 3 i 4 algoritma 6). Na kraju se računa vrijednost $sw_{negacija}$ negiranog pravila (redak 9 algoritma).

Alg. 6 Pseudokod pretrage nedostajućih vrijednosti za negaciju pravila koje odgovara čvoru.

Input $\mathcal{E}$, Čvor q, skup instanci nedostajućih vrijednosti čvora $\mathcal{N}$, $supp(q)$ i skup atributa uvjeta čvora i svih roditelja čvora $\mathcal{A}$.

Output Skup $\overline{\mathcal{N}}$ instanci koje sadrže nedostajuću vrijednost za barem jedan atribut koji se nalazi u uvjetu čvora ili nekog roditelja ali opisani čvorom i $sw_{negacija}$ bez nedostajućih vrijednosti.

1: $\overline{\mathcal{N}} \leftarrow \emptyset$
2: **for each** $e \in \mathcal{N}$ **do**
3: **for each** $a \in \mathcal{A}$ **do**
4: **if** $p_a(e) \neq ?$ i $p_a(e)$ ne vrijedi za a **then**
5: $\overline{\mathcal{N}} \leftarrow \overline{\mathcal{N}} \cup e$
6: **end if**
7: **end for**
8: **end for**
9: $sw_{negacija} = |\mathcal{E}| - sw' - (|\mathcal{N}| - |\overline{\mathcal{N}}|)$
10: **return** $\overline{\mathcal{N}}$, $sw_{negacija}$

Algoritam 7 pokazuje kako se ažurira vektor $\overrightarrow{sv}$ za negaciju. Iterira se po instancama iz skupa $\mathcal{N}\setminus\overline{\mathcal{N}}$ (redak 1 algoritma) pa ako je vrijednost neke ciljne labele jednaka 1, vrijednost vektora $\overrightarrow{sv}$ se ažurira.

Alg. 7 Pseudokod ažuriranjeVrijednostiSVNegacija

Input $\overrightarrow{sv}_q = (s'_{\tau 1}, s'_{\tau 2}, ..., s'_{\tau i}, ..., s'_{\tau n})$ za čvor q, skup atributa ciljnih labela $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, skup instanci nedostajućih vrijednosti čvora $\mathcal{N}$ i skup $\overline{\mathcal{N}}$ instanci koje sadrže nedostajuću vrijednost za barem jedan atribut koji se nalazi u uvjetu čvora ili nekog roditelja ali opisani čvorom.

```
1: for each e ∈ N \ N̄ do
2:     i ← 0
3:     while i < |T| do
4:         if τ_i(e) = 1 then
5:             s'_τi ← s'_τi + 1
6:         end if
7:     end while
8: end for
```

Za ciljnu labelu τ_i se prolazi kroz skup $\overline{\mathcal{N}}$ i provjerava se je li vrijednost atributa ciljne labele jednak 1.0. Ako je, onda se povećava broj $s'_{\tau i}$ za jedan.
Sada se može definirati vrijednost Jaccard indeksa za negirana pravila koja imaju nedostajuće vrijednosti.

Definicija 3.7.1. *Za negaciju pravila q koje odgovara čvoru* n *stabla, ciljnu labelu τ_i iz skupa ciljnih labela $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, vektor $\overrightarrow{sv'} \overset{def.3.6.1}{=} (s'_{\tau_1}, s'_{\tau_2}, ..., s'_{\tau_i}, ..., s'_{\tau_n})$, skup nedostajućih instanci čvora $\mathcal{N}$, skup $\overline{\mathcal{N}}$ instanci koje sadrže nedostajuću vrijednost za barem jedan atribut koji se nalazi u uvjetu čvora ili nekog roditelja ali opisani čvorom i broj c_i definira se Jaccard indeks:*

$$J(supp(\neg q), supp(\tau_i)) = \frac{|supp(\tau_i)| - s'_{\tau i} - c_i}{sw_{negacija} + |supp(\tau_i)| - (|supp(\tau_i)| - s'_{\tau i} - c_i)} = \frac{|supp(\tau_i)| - s'_{\tau i} - c_i}{sw_{negacija} + s'_{\tau i} + c_i},$$

gdje je $sw_{negacija} = |\mathcal{E}| - sw' - (|\mathcal{N}| - |\overline{\mathcal{N}}|)$ za $sw' = sw - |\mathcal{N}|$ i $c_i = \sum_k \tau_i(e_k)$ za svaki $e_k \in \mathcal{N} \setminus \overline{\mathcal{N}}$.

Primjer 3.7.2. *Pretpostavimo da se za skup primjera $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$ za neka dva pogleda induciralo stablo sa ciljnom labelom τ_i. Skup instanci $\mathcal{S}$ čvora* n *je skup*

$\mathcal{S} = \{e_1, e_2, e_3\}$ *gdje je* q *pravilo koje odgovara čvoru* n. *Podrška ciljne labele* τ_i *je skup* $\{e_1, e_3, e_4, e_5\}$. *Skup nedostajućih vrijednosti čvora* $\mathcal{N} = \{e_2, e_3\}$, *a skup instanci* $\overline{\mathcal{N}} = \{e_2\}$ *koje sadrže nedostajuće vrijednosti za barem jedan atribut koji se nalazi u uvjetu čvora ili nekog roditelja, ali opisani čvorom.*

Počinje se računanjem vrijednosti s'_{τ_i}. *Može se primijetiti da je vrijednost* $s_{\tau_i} = 2$ *jer presjek podrške ciljne labele i skupa instanci čvora* $\mathcal{S}$ *sadrži elemente* e_1 *i* e_3. *Budući da podrška ciljne labele sadrži element* e_3 *koji je sadržan u skupu nedostajućih vrijednosti* $\mathcal{N}$ *čvora, vrijedi* $s'_{\tau_i} = 2 - 1 = 1$.
Zatim se može vidjeti da je $sw = |\mathcal{S}| = 3$, $sw' = sw - |\mathcal{N}| = 3 - 2 = 1$. *Iz toga se može izračunati* $sw_{negacija} = |\mathcal{E}| - sw' - (|\mathcal{N}| - |\overline{\mathcal{N}}|) = 5 - 1 - (2 - 1) = 3$.
Sada se mogu izračunati elementi skupovne razlike, $\mathcal{N} \setminus \overline{\mathcal{N}} = \{e_3\}$. *Podrška ciljne labele sadrži element* e_3 *pa stoga vrijedi* $c_i = 1$. *S ovim izračunima može se izraziti vrijednost Jaccard indeksa:*

$$J(supp(\neg q), supp(\tau_i)) = \frac{|supp(\tau_i)| - s'_{\tau i} - c_i}{sw_{negacija} + s'_{\tau i} + c_i} = \frac{4 - 1 - 1}{3 + 1 + 1} = \frac{2}{5}.$$

Za svako pravilo koje odgovara čvoru čuva se vrijednost $sw_{negacija}$ jer će se ta pravila koristiti u idućoj iteraciji kao negirane ciljne labele za suprotni pogled. Osim toga se za svaki čvor pamti skup $\mathcal{N} \setminus \overline{\mathcal{N}}$. Broj instanci tog skupa koristit će se za provjeru vrijednosti Jaccard indeksa kada se provjerava negacija ciljnih labela.

3.8 Računanje vrijednosti Jaccard indeksa za negirane ciljne labele

Kod skupova koji imaju nedostajuće vrijednosti, algoritam pamti vrijednosti $sw^{(k-1)}_{negacija}$ od čvora svake ciljne labele iz prethodne iteracije. Uz to, svaka ciljna labela sadrži skup $\mathcal{N}^{(k-1)} \setminus \overline{\mathcal{N}}^{(k-1)}$. Za svaki čvor n iterira se po tom skupu i ako skup instanci čvora $\mathcal{S}_n$ sadrži instancu iz skupa $\mathcal{N}^{(k-1)} \setminus \overline{\mathcal{N}}^{(k-1)}$ i instanca nije sadržana u skupu nedostajućih vrijednosti $\mathcal{N}$ čvora, tada se broj $c_i^{'}$ poveća za jedan uz uvjet da je vrijednost ciljne labele instance jednaka nula.
S tim se može definirati vrijednost Jaccard indeksa za negirane ciljne labele i pravila koje odgovara pojedinom čvoru.

Definicija 3.8.1. *Za negiranu ciljnu labelu* $\neg\tau_i$ *ciljne labele* τ_i *iz skupa ciljnih labela* $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_N\}$, *vektor vektor* $\overrightarrow{sv'} \overset{def.3.6.1}{=} (s'_{\tau_1}, s'_{\tau_2}, ..., s'_{\tau_i}, ..., s'_{\tau_n})$, *skup nedostajućih ins-*

tanci $\mathcal{N}$ čvora n*, vrijednosti $sw^{(k-1)}_{negacija}$ za ciljnu labelu te za pravilo q koje odgovara čvoru* n *stabla definira se Jaccard indeks:*

$$J(supp(q), supp(\neg\tau_i)) = \frac{sw' - s'_{\tau_i} - c_i^{'}}{sw' + sw^{(k-1)}_{negacija} - (sw' - s'_{\tau_i} - c_i^{'})},$$

gdje $sw' = sw - |\mathcal{N}|$. $c'_i = \sum_k (1 - \tau_i(e_k))$ za svaki $e_k \in \left((\mathcal{N}^{(k-1)} \setminus \overline{\mathcal{N}}^{(k-1)}) \setminus \mathcal{N}\right) \cap \mathcal{S}_n$ gdje je skup $\mathcal{N}^{(k-1)}$ skup nedostajućih vrijednosti za čvor koji odgovara ciljnoj labeli, a $\overline{\mathcal{N}}^{(k-1)}$ je skup instanci koje sadrže nedostajuću vrijednost za barem jedan atribut koji se nalazi u uvjetu čvora koji odgovara ciljnoj labeli ili nekog roditelja ali opisani tim čvorom. Skup $\mathcal{S}_n$ je skup instanci čvora n*.*
Vrijednost $sw^{(k-1)}_{negacija}$ definirana je u definiciji 3.7.1 i izračunata u prethodnoj iteraciji za čvor koji odgovara ciljnoj labeli.

Primjer 3.8.2. *Pretpostavimo da se za skup primjera $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$ za neka dva pogleda induciralo stablo sa ciljnom labelom τ_i. Skup instanci $\mathcal{S}$ čvora* n *je skup $\mathcal{S} = \{e_1, e_2, e_3\}$ gdje je* q *pravilo koje odgovara čvoru* n*. Podrška ciljne labele τ_i je skup $\{e_1\}$. Skup nedostajućih vrijednosti čvora je $\mathcal{N} = \{e_2, e_3\}$, a skupovi $\mathcal{N}^{(k-1)} = \{e_2, e_3\}$, $\overline{\mathcal{N}}^{(k-1)} = \{e_2\}$ i vrijednost $sw^{(k-1)}_{negacija} = 3$ vrijede za pravilo koje odgovara ciljnoj labeli iz prethodne iteracije.*

Počinje se računanjem vrijednosti s'_{τ_i}. Može se primijetiti da je vrijednost $s_{\tau_i} = 1$ jer presjek podrške ciljne labele i skupa instanci čvora $\mathcal{S}$ sadrži element e_1. Budući da podrška ciljne labele ne sadrži nijedan element koji je u skupu nedostajućih vrijednosti $\mathcal{N}$ čvora, vrijedi $s'_{\tau_i} = 1 - 0 = 1$. Može se vidjeti da vrijedi $\left((\mathcal{N}^{(k-1)} \setminus \overline{\mathcal{N}}^{(k-1)}) \setminus \mathcal{N}\right) \cap \mathcal{S} = \emptyset$, pa je broj $c'_i = 0$. Zatim se može vidjeti da je $sw = |\mathcal{S}| = 3$, $sw' = sw - |\mathcal{N}| = 3 - 2 = 1$.

$$J(supp(q), supp(\neg\tau_i)) = \frac{sw' - s'_{\tau_i} - c_i^{'}}{sw' + sw^{(k-1)}_{negacija} - (sw' - s'_{\tau_i} - c_i^{'})} = \frac{1 - 1 - 0}{1 + 3 - (1 - 1 - 0)} = 0.$$

Definicije 3.6.2, 3.7.1 i 3.8.1 služe kao općenite definicije koje uključuju skupove s nedostajućim vrijednostima i skupove bez nedostajućih vrijednosti. Presjeci i unije podržaka pravila izračunati u tim definicijama koriste se za računanje p-vrijednosti 1.5.1 para pravila.

3.9 Optimizirani algoritam CLUS-RM

Prilikom rekurzivnog stvaranja stabla algoritam (alg 8) računa parove pravila koji zadovoljavaju uvjet zadane minimalne vrijednosti Jaccard indeksa pomoću definicija 3.6.2, 3.7.1 i 3.8.1 i provjere p-vrijednosti. Nakon induciranja stabla, vraćaju se informacije koja pravila će se spajati s pravilima iz prethodne iteracije suprotnog pogleda u obliku HashMap-a. U koracima 11 i 12 se ta pravila spajaju bez računanja uvjeta po definiciji 3.2.1.

Alg. 8 Pseudokod optimiziranog CLUS-RM algoritma

Input Dva skupa podataka W_1 i W_2, minimalni Jaccard, maksimalna p-vrijednost, maksimalna podrška, minimalna podrška i broj iteracija κ.

Output Skup redeskripcija $\mathcal{R}$.

1: $\mathcal{R} \leftarrow \emptyset$
2: $[\tau_1, \tau_2] \leftarrow$ pripremanje ciljnih labela za inicijalni PCT od W_1 i W_2
3: $[T_1^{(0)}, T_2^{(0)}] \leftarrow$ induciranje stabala po τ_1 i τ_2
4: $[Q_1^{(0)}, Q_2^{(0)}] \leftarrow$ dobivanje pravila iz $T_1^{(0)}$ i $T_2^{(0)}$
5: **for each** $k \in \{1, 2, ..., \kappa\}$ **do**
6: **for each** $(s, t) \in \{(1, 2), (2, 1)\}$ **do**
7: $[\tau_s, \tau_t] \leftarrow$ pripremanje ciljnih labela od pravila iz $[Q_t^{(k-1)}, Q_s^{(k-1)}]$
8: $[T_s^{(k)}, T_t^{(k)}] \leftarrow$ induciranje stabala po D_s i D_t s dodanim ciljnim labelama τ_s i τ_t
9: $\left[Mapa\left\langle q_t^{(k)}, Q_s^{(k-1)}\right\rangle, Mapa\left\langle q_s^{(k)}, Q_t^{(k-1)}\right\rangle\right] \leftarrow$ dobivanje mapa pravila koja se trebaju povezati u redeskripcije
10: $[q_t^{(k)}, Q_s^{(k)}] \leftarrow$ dobivanje pravila iz $T_t^{(k)}$ i $T_s^{(k)}$
11: $\mathcal{R} \leftarrow \mathcal{R}$ unija skupa redeskripcija za parove pravila iz $Mapa\left\langle q_t^{(k)}, Q_s^{(k-1)}\right\rangle$
12: $\mathcal{R} \leftarrow \mathcal{R}$ unija skupa redeskripcija za parove pravila iz $Mapa\left\langle q_s^{(k)}, Q_t^{(k-1)}\right\rangle$
13: **end for**
14: **end for**
15: **return** $\mathcal{R}$

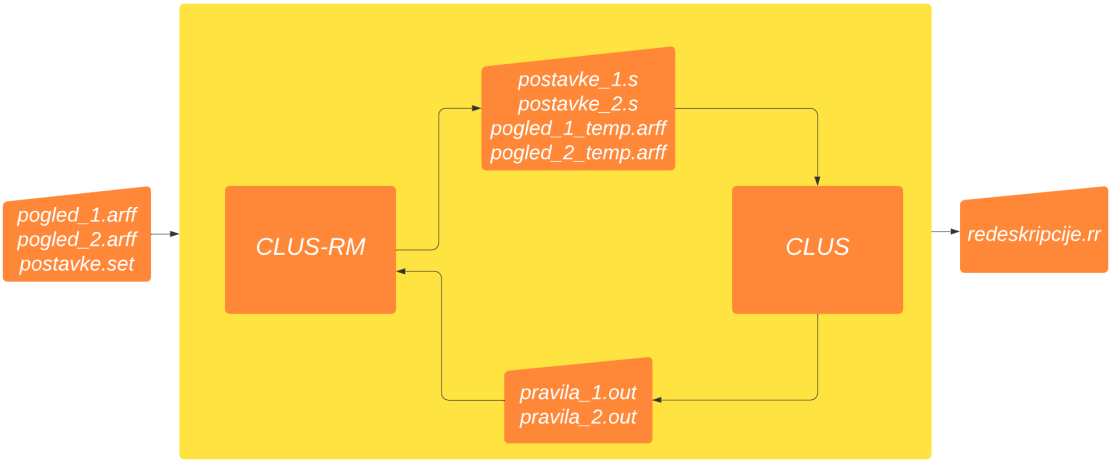
3.10 Analiza složenosti računanja sličnosti pravila

Iz poglavlja 3.1 može se vidjeti da je u koraku 10 algoritma 3 složenost za provjeru vrijednosti Jaccard indeksa za sve parove pravila u iteraciji za najgori slučaj $O\left(|Q_1| \cdot |Q_2| \cdot |supp(q_1)| \cdot |supp(q_2)|\right)$ budući da se radi o provjeri svakog pravila iz pogleda 1 s pravilom iz pogleda 2, a provjera vrijednosti Jaccard indeksa je $|supp(q_1)| \cdot |supp(q_2|$. Optimizacijama opisanim u ovom poglavlju, i uz pretpostavku da je broj čvorova za jedan pogled u svakoj iteraciji približno jednak i u slučaju kada nema nedostajućih vrijednosti,

ova složenost provjere vrijednosti Jaccard indeksa postaje $O(1)$ budući da su eliminirane provjere vrijednosti Jaccard indeksa pomoću iteriranja po skupu podrške pravila.
U slučaju nedostajućih vrijednosti po algoritmu 4 može se vidjeti da se iterira kroz svaku instancu sadržanu u čvoru (uključujući nedostajuće vrijednosti) (redak 2 algoritma). Budući da se tu ne radi o prolasku kroz podrške pravila koja odgovaraju čvoru za svaki par spajanja nego za svaku instancu sadržanu u čvoru ($\mathcal{S}_i$), složenost je onda $O(|Q_1| \cdot |\mathcal{S}_1| + |Q_2| \cdot |\mathcal{S}_2|)$. Vremenska ušteda će se pokazati u dijelu rada s eksperimentima.

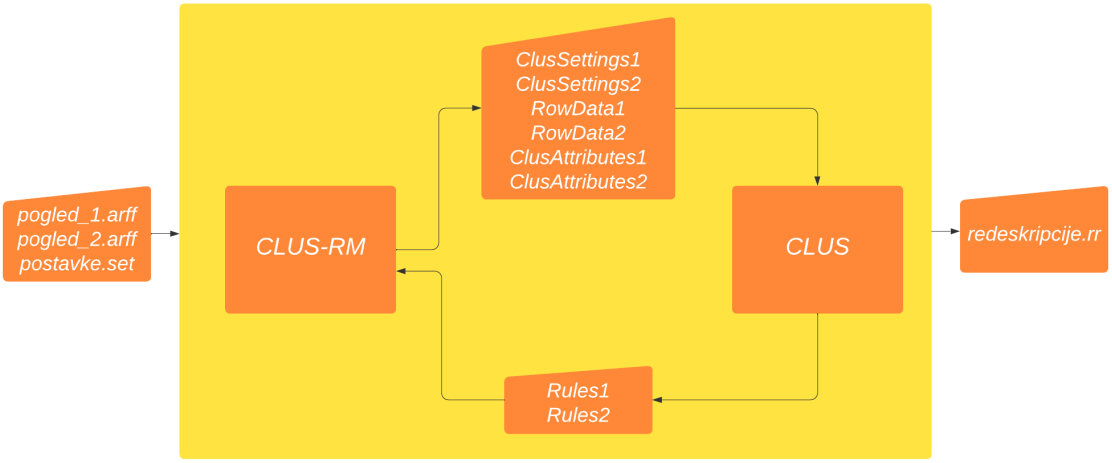
3.11 Implementacijske optimizacije algoritma CLUS-RM

Slika 3.3 pokazuje pojednostavljeni dijagram procesa povezanosti algoritma CLUS-RM i CLUS [1]. Proces počinje s ulaznim datotekama *pogled_1.arff*, *pogled_2.arff* koje sadrže atribute i skup instanci pogleda i datoteka s postavkama *postavke.set*. Datoteka s postavkama sadrži informaciju poput minimalnog Jaccard indeksa, maksimalne p-vrijednosti i maksimalne podrške.
Nakon inicijalizacije gdje se stvaraju umjetne instance, počinje iterativni proces gdje se odvija komunikacija između algoritma CLUS-RM i algoritma CLUS. Komunikacija između algoritama se odvija pomoću datoteka. CLUS-RM stvara datoteke *pogled_1_temp.arff* i *pogled_2_temp.arff* s modificiranim pogledima gdje su dodane dodane ciljne labele. Svaka *arff* datoteka sadrži skup atributa za pogled i skup instanci $\mathcal{E}$ pogleda zajedno s vrijednostima atributa za instance. Datoteke *postavke_1.s* i *postavke_2.s* sadrže postavke za pokretanje algoritma CLUS. Budući da se između iteracija mijenja samo informacija na kojim su pozicijama u pogledu ciljne labele, redundantno je svaki put zapisivati ostale postavke.
Motivacija za prvu implementacijsku optimizaciju je da se eliminira potreba da CLUS-RM stvara ulazne datoteke za inicijalizaciju algoritma CLUS. Budući da oba algoritma dijele interne Razrede *RowData*, koji predstavlja instance s odgovarajućim vrijednostima za svaki atribut, *ClusSettings* koji odgovara datoteci *postavke.s* i *ClusAttributes* koji sadrži sve atribute za pogled u toj iteraciji. Integrirajući dvije aplikacije u jednu, omogućilo se da CLUS sada može prihvatiti navedene objekte umjesto datoteka. S time se postigla jednostruka povezanost algoritama CLUS-RM i CLUS bez korištenja datoteka. Slika 3.4 pokazuje promjenu načina pozivanja algoritma CLUS.



Slika 3.3: Pojednostavljeni dijagram procesa s datotekama.

S druge strane, aplikacija CLUS inducira stablo i vraća skup odgovarajućih pravila stabla i njihovih podršaka. CLUS zapisuje ta pravila u datoteke *pravila_1.out* i *pravila_2.out* koje onda CLUS-RM čita koristeći svoj parser i lekser. Zapisivanje velikog broja pravila i odgovarajućih podrški i zatim ponovno čitanje tih podataka može biti dugotrajan proces pa je zato druga motivacija optimizacije da se eliminira i povezanost algoritama bez korištenja datoteka s te strane. Može se napomenuti da datoteke s pravilima sadrže još i dodatne informacije koje parser iz CLUS-RM mora ignorirati, ali dodatno povećava vrijeme čitanja i pisanja. Zbog toga se napravilo da CLUS više ne generira datoteke, nego da se pravila mapiraju u strukturu *Rules* iz algoritma CLUS-RM. S time se eliminirao potencijalni problem čitanja datoteke u slučaju promjene načina ispisa pravila sa strane algoritma CLUS.
Slika 3.4 pokazuje pojednostavljeni dijagram procesa bez datoteka kod kojeg se vidi da se eliminirala potreba za datotekama, pa se može vidjeti dvostrana komunikacija između CLUS-RM i CLUS bez datoteka.



Slika 3.4: Pojednostavljeni dijagram procesa bez datoteka.

Osim uštede vremena, ova optimizacija omogućava lakše održavanje koda i lakše implementacije potencijalnih promjena u algoritmu CLUS. Bez ovih promjena, svaka komunikacija između algoritama morala bi se odvijati preko datoteka. Ušteda vremena će se pokazati u poglavlju s eksperimentima.

Jedan od potencijalnih načina dodatne implementacijske optimizacije bi se mogao postići uvođenjem višedretvenosti u iterativnom procesu algoritma. Budući da se u algoritmu 8 više ne spajaju pravila iz trenutnih iteracija, nego pravila iz jednog pogleda trenutne iteracije s pravilima drugog pogleda u prethodnoj iteraciji, moguće je napraviti da se ta dva procesa odvijaju paralelno koristeći dvije dretve.

Poglavlje 4

Eksperimenti

4.1 Opis podataka

Eksperimenti su se proveli nad tri različita skupa podataka. Tablica 4.1 opisuje karakteristike skupova i ulaznih parametara.

parametar	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
$\lvert V_1 \rvert$	194	312	35
tip atributa za V_1	kategorijski	numerički	kategorijski
$\lvert V_2 \rvert$	48	49	6
tip atributa za V_2	numerički	numerički	kategorijski
$\lvert \mathcal{E} \rvert$	2575	199	5000
minimalna podrška	10	5	10
maksimalna podrška	2060	120	4500
minimalni Jaccard	0.3	0.3	0.3
broj iteracija	80	80	20
ima nedostajuće vrijednosti	ne	da	ne

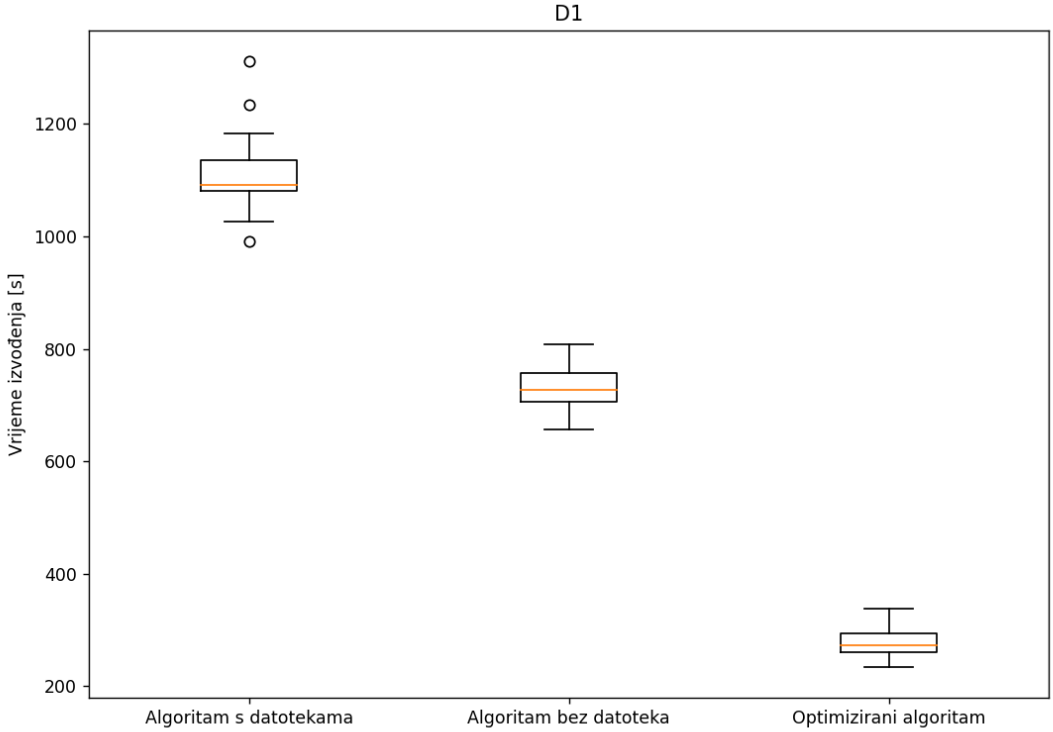
Tablica 4.1: Tablica koja opisuje tri skupa podataka nad kojima se izvode eksperimenti i ulazni parametri za algoritam

Skup $\mathcal{D}_1$ opisan je u [8]. Naziv skupa je *Bio*, a instance su geografska područja. Atributi prvog pogleda predstavljaju postoji li određena vrsta sisavaca na tim geografskim područjima ili ne. Atributi drugog pogleda predstavljaju klimatske karakteristike pojedinih područja. Skup $\mathcal{D}_2$ opisan je u [11], a naziv mu je *Country*. Instance skupa su države svijeta. Atributi prvog pogleda su trgovinske karakteristike, dok drugi pogled opisuje socio-demografske značajke zemalja.

Treći skup dohvaćen je iz repozitorija [4] i opisan je u [7]. Naziv skupa je *Waveform*, a instance su generirani valovi. Atributi prvog pogleda predstavljaju frekvencije valova s dodanom bukom, a atributi drugog pogleda frekvencije sa značajnom bukom i razredom valova.
Može se primijetiti da sva tri skupa imaju različite karakteristike. Skup $\mathcal{D}_2$ sadrži nedostajuće vrijednosti, ali broj instanci u skupu puno je manji od ostala dva skupa. Iz toga se može pretpostaviti da će podrške pravila biti skupovi s manjim kardinalnim brojevima nego za preostala dva skupa.
Skup $\mathcal{D}_3$ sadrži manje atributa za svaki pogled i puno više instanci od druga dva skupa. Za taj skup je moguće da će pravila u algoritmu imati puno veću podršku od ostala dva skupa.
Važno je za napomenuti da su dobiveni skupovi redeskripcija za svaku optimizaciju jednaki skupu redeskripcija koju vraća standardna verzija algoritma. . U eksperimentalnoj evaluaciji će se također provjeravati Mann-Whitney U test [13] koristeći Python paket *pingouin* [5].

4.2 Eksperimentalna evaluacija

Eksperimenti su se provodili tako da se napravilo 10 slučajnih inicijalizacija i mjerilo se vrijeme za svaku optimizaciju te se dobije distribucija koja je opisana brkatim kutijama na slikama. Slika 4.1 pokazuje vremena izvođenja algoritma i optimizacija za skup $\mathcal{D}_1$. Prvi stupac grafa sadrži vremena izvođenja osnovnog algoritma koji koristi datoteke. Drugi stupac sadrži vremena izvođenja algoritma koji ne koristi datoteke, ali spajanje redeskripcija izvodi se računanjem Jaccard indeksa za svaki par pravila iz kartezijevog umnoška skupova pravila. 3.11. Treći stupac predstavlja vrijeme izvođenja algoritma koji sadrži opisane algoritamske optimizacije. 3.2. Važno je za napomenuti da se kod algoritamske optimizacije nadograđuje kod implementacijske optimizacije. Stoga je za analizu uštede vremena algoritamske optimizacije potrebno uspoređivati treći stupac s drugim. Analiza složenosti 3.10 sugerira da bi moglo doći do vremenske uštede kod optimiziranog algoritma.



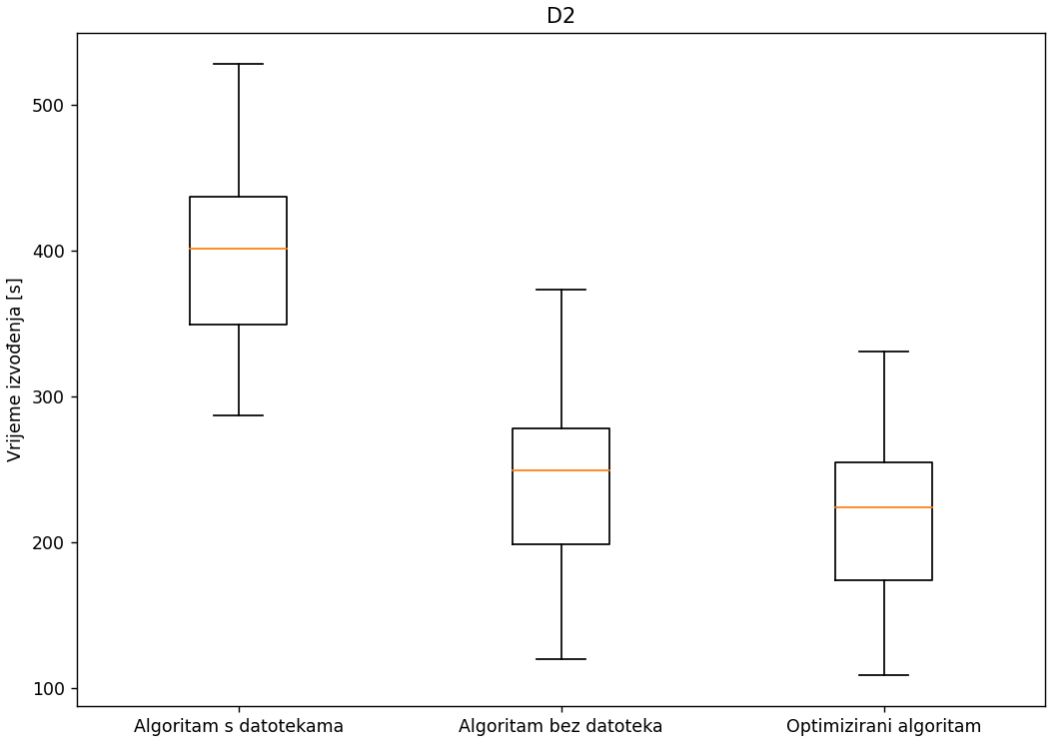
Slika 4.1: Usporedba vremena izvođenja algoritma s korištenjem datoteka i bez korištenja datoteka za skup $\mathcal{D}_1$.

Rezultati Mann-Whitney U testa za eksperimente na skupu podataka $\mathcal{D}_1$ su:

$$pVrijednost_{mwu,\mathcal{D}_1}(\text{Algoritam bez datoteka, Algoritam s datotekama}) = 3.373925 \times 10^{-8}$$
$$pVrijednost_{mwu,\mathcal{D}_1}(\text{Optimizirani algoritam, Algoritam bez datoteka}) = 3.378690 \times 10^{-8}$$

Slika 4.2 pokazuje distribuciju vremena izvođenja algoritma i optimizacija za skup $\mathcal{D}_2$. Graf pokazuje vremensku uštedu za implementacijsku optimizaciju (algoritam gdje se ne koriste datoteke) i za algoritamsku optimizaciju. Uspoređujući ovaj graf s grafom na slici 4.1 može se primjetiti da je ušteda kod ovog grafa manja od grafa na slici 4.1. Iz analize složenosti 3.10 proizlazi da je razlog tome što skupovi $\mathcal{D}_1$ i $\mathcal{D}_2$ imaju različiti broj instanci, ali $\mathcal{D}_2$ ima i nedostajuće vrijednosti, tako da je optimizacija složenija. Skup $\mathcal{D}_1$ ima značajno veći broj instanci pa su sukladno s tim i podrške dobivenih pravila iz stabla veća. Iz toga se može zaključiti da će najveći doprinos algoritamske optimizacije biti za skupove koji imaju veliki broj instanci i guste skupove podataka (razgranata stabla). Inducirano stablo za skup $\mathcal{D}_1$ je razgranato te se indukcijom dobiva puno pravila za svaku iteraciju.

U tom slučaju provjera Jaccard indeksa za svaki par pravila značajno utječe na vrijeme izvođenja.

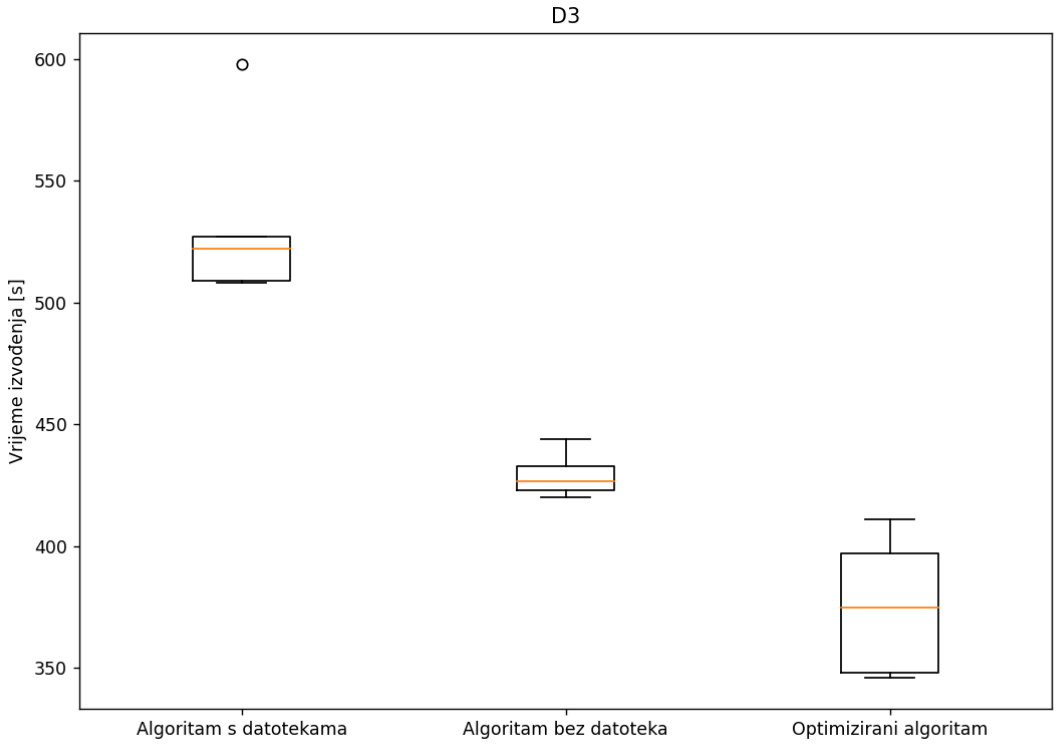


Slika 4.2: Usporedba vremena izvođenja algoritma s korištenjem datoteka i bez korištenja datoteka za skup $\mathcal{D}_2$.

Rezultati Mann-Whitney U testa za eksperimente na skupu podataka $\mathcal{D}_2$ su:

$$pVrijednost_{mwu,\mathcal{D}_2}(\text{Algoritam bez datoteka, Algoritam s datotekama}) = 0.00001$$
$$pVrijednost_{mwu,\mathcal{D}_2}(\text{Optimizirani algoritam, Algoritam bez datoteka}) = 0.127791$$

Slika 4.3 prikazuje vremena izvođenja algoritma i optimizacije za skup $\mathcal{D}_3$. Može se vidjeti da je broj instanci za taj skup znatno veći od ostala dva skupa pa je, unatoč tome što ima manje atributa, utjecaj optimizacije signifikantan.



Slika 4.3: Usporedba vremena izvođenja algoritma s korištenjem datoteka i bez korištenja datoteka za skup $\mathcal{D}_3$.

Rezultati Mann-Whitney U testa za eksperimente na skupu podataka $\mathcal{D}_3$ su:

$$pVrijednost_{mwu,\mathcal{D}_3}(\text{Algoritam bez datoteka, Algoritam s datotekama}) = 0.003968$$

$$pVrijednost_{mwu,\mathcal{D}_3}(\text{Optimizirani algoritam, Algoritam bez datoteka}) = 0.027778$$

Literatura

[1] *Clus: User's Manual*.

[2] `http://trove4j.sourceforge.net/javadocs/gnu/trove/set/hash/TIntHashSet.html`, posjećena 18.02.2022.

[3] `https://programming.guide/hash-tables-open-addressing.html`, posjećena 18.02.2022.

[4] `https://github.com/renatopp/arff-datasets/blob/master/classification/waveform5000.arff`, posjećena 18.02.2022.

[5] `https://pingouin-stats.org/`, posjećena 18.02.2022.

[6] Hendrik Blockeel, Luc De Raedt i Jan Ramon, *Top-down induction of clustering trees*, arXiv preprint cs/0011032 (2000).

[7] Leo Breiman, Jerome H Friedman, Richard A Olshen i Charles J Stone, *Classification and regression trees*, Routledge, 2017.

[8] Esther Galbrun i Pauli Miettinen, *From black and white to full color: extending redescription mining outside the Boolean world*, Statistical Analysis and Data Mining: The ASA Data Science Journal **5** (2012), br. 4, 284–303.

[9] ———, *Redescription Mining: An Overview.*, IEEE Intelligent Informatics Bulletin **18** (2017), br. 2.

[10] ———, *What Is Redescription Mining*, Redescription Mining, Springer, 2017.

[11] Dragan Gamberger, Matej Mihelčić i Nada Lavrač, *Multilayer clustering: a discovery experiment on country level trading data*, International Conference on Discovery Science, Springer, 2014, str. 87–98.

[12] Matej Mihelčić, Sašo Džeroski, Nada Lavrač i Tomislav Šmuc, *Redescription mining with multi-target predictive clustering trees*, International Workshop on New Frontiers in Mining Complex Patterns, Springer, 2015.

[13] B Rosner i D Grove, *Use of the Mann–Whitney U-test for clustered data*, Statistics in medicine **18** (1999), br. 11, 1387–1400.

Sažetak

U ovom radu opisuje se nekoliko optimizacija algoritma CLUS-RM čija je zadaća pronalazak redeskripcija koristeći višeciljna prediktivna stabla klasteriranja. Prvi dio rada opisuje pretraživanje redeskripcija kao tehniku dubinske analize podataka te navodi definicije i izraze vezane uz područje.
Drugi dio rada opisuje algoritam CLUS-RM koji koristi višeciljna prediktivna stabla klasteriranja te opisuje i sam algoritam stvaranja prediktivnih stabala.
Treći dio rada prikazuje algoritamsku optimizaciju provjere Jaccard indeksa dvaju pravila u algoritmu CLUS-RM za podatke koji imaju nedostajuću vrijednost i za podatke koji nemaju nedostajuću vrijednost uz analizu složenosti optimizacije. Uz to se pojašnjavaju implementacijske optimizacije provedene nad algoritmom postignute promjenom načina dvosmjerne komunikacije između algoritama CLUS-RM i CLUS.
U zadnjem dijelu rada, analiziraju se provedeni eksperimenti i ušteda vremena za obje optimizacije algoritma temeljem tri različita skupa podataka.

Summary

This thesis describes several approaches to optimize the CLUS-RM algorithm whose task is to find redescriptions using multi-target predictive clustering trees.
The first part of the thesis describes the search for redescriptions as a technique of data mining and provides definitions and expressions related to the field.
The second part of the thesis describes the CLUS-RM algorithm that uses multi-target predictive clustering trees and describes the algorithm for creating predictive trees.
The third part of the thesis presents the algorithmic optimization of the Jaccard index calculation of the two rules in the CLUS-RM algorithm for data with missing values and data without missing values, also including the complexity analyis of the proposed optimization. In addition, the implementation optimizations performed on the algorithm achieved by changing the way of two-way communication between the CLUS-RM and CLUS algorithms are clarified.
The last part of the thesis analyses the experiments performed and the time savings gained for each proposed algorithm optimization on three different data sets.

Životopis

Rođen sam 23.7.1992. godine u Zagrebu. Završio sam gimnaziju Lucijana Vranjanina u Zagrebu. Nakon toga proveo sam dvije godine na Fakultetu elektrotehnike i računarstva. Potom sam upisao Prirodoslovno matematički fakultet, smjer matematika i fizika. Uz studiranje, nastavio sam se baviti programiranjem u kojem planiram nastaviti karijeru i nakon završetka fakulteta. Od 2020. godine radim kao Java developer u poduzeću TIS.