SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2584

PRIMJENA GENETSKOG PROGRAMIRANJA NA DIJAGNOSTICIRANJE PLUĆNE EMBOLIJE

Maja Kontrec

Zagreb, lipanj 2012.

Sadržaj

1. Uvod

Plućna embolija je stanje u kojem dolazi do začepljenja jedne od plućnih arterija. Začepljenje je najčešće uzrokovano s jednim ili više krvnih ugrušaka, no može ga prouzročiti i mjehurić zraka ili mast u krvotoku. Takvo začepljenje sprječava normalan dotok krvi u pluća, čija posljedica može biti odumiranje plućnog tkiva te naposlijetku smrt.

Budući da je najćešći simptom ove bolesti otežano i ubrzano disanje, njena dijagnoza može se uspostaviti samo primjenom posebnih metoda dijagnosticiranja. Najtočija metoda za dijagnosticiranje plućne embolije je CTA, angiografija kompjuteriziranom tomografijom (engl. *computed tomography angiography*). Ovom metodom dobiva se trodimenzionalni prikaz snimanog objekta koji nastaje spajanjem njegovih slika presjeka. Kako bi dijagnoza bila uspostavljena, svaka slika presjeka mora biti detaljno pregledana u potrazi za plućnom embolijom. Taj posao veoma je kompliciran, iscrpan i dugotrajan za čovjeka, budući da, osim što se svaka snimka sastoji od stotine slika presjeka, mogu se pojaviti slučajevi koji nalikuju na emboliju, a to nisu.

Zbog prirode ovoga problema, idealan alat za njegovo rješenje je računalo. Budući da ne postoji uvriježeni algoritam za dijagnosticiranje plućne embolije, ovaj rad opisuje moguće rješenje problema dijagnosticiranja uz pomoć strojnog učenja realiziranog genetskim programiranjem.

2. Genetsko programiranje

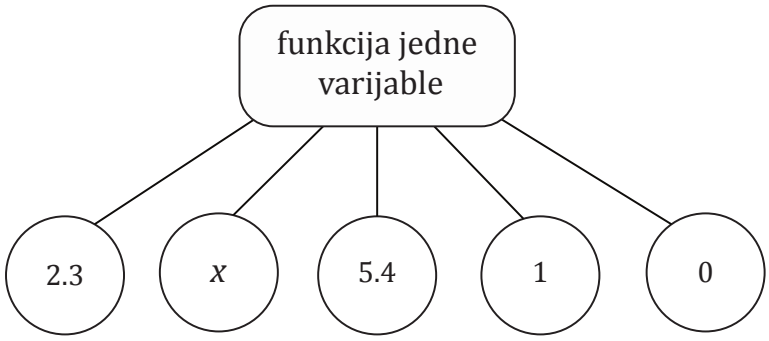
Genetsko programiranje omogućuje nam automatsko rješavanje problema pouzdajući se u principe biološke evolucije. Oponašajući evoluciju, ova metoda gradi populaciju jedinki, programa, koji predstavljaju potencijalna rješenja problema, razvija ih i poboljšava kroz vrijeme međusobnim križanjem, mutacijom te odbacivanjem najlošijih, odnosno preživljavanjem najboljih jedinki.

2.1. Vrste genetskog programiranja

Vrste genetskog programiranja dijele se s obzirom na način prikaza jedinki. Najčešće je genetsko programiranje temeljeno na stablima, no koriste se još i linearno genetsko programiranje, genetsko programiranje temeljeno na grafovima itd.

2.1.1. Genetsko programiranje temeljeno na stablima

U genetskom programiranju temeljenom na stablima, jedinka je prikazana u obliku stabla odluke. Ono se sastoji od nezavršnih, funkcijskih znakova te završnih znakova, koji mogu biti konstante, varijable ili određena akcija. Vrijednosti tih znakova različita su s obzirom na specifičnosti rješavanja određenog problema. Program rješenja iščitava se obilaskom stabla. Kako bi se program interpretirao točno, mora postojati odgovarajući interpreter stabla. Prednost ove vrste genetskog programiranja je jednostavnost izvođenja genetskog operatora križanja.



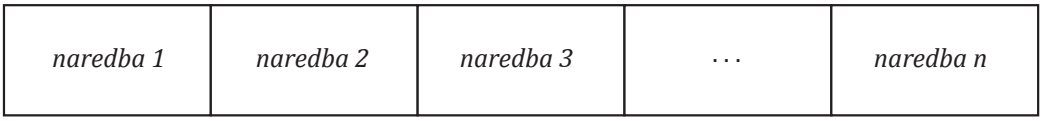
Slika 1.1 prikaz jednostavnog stabla odluke

Na slici 1.1 vidimo prikaz jednostavnog stabla odluke. Funkcijski znak je ovdje čvor *funkcija jedne varijable,* konstante su brojevi 2.3 i 5.4, varijabla čvor *x*, a čvorovi odluke 0 i 1 (0 i 1 reprezentiraju logičku istinu ili laž). Ovo stablo evaluira izraz

2.3 $x \leq 5.4$. Ukoliko je izraz istinit, odlučujemo se za rješenje 1 (prvi čvor odluke). U suprotnom slučaju, tj. ako izraz nije istinit, odlučujemo se za rješenje 0 (drugi čvor odluke).

2.1.2. Linearno genetsko programiranje

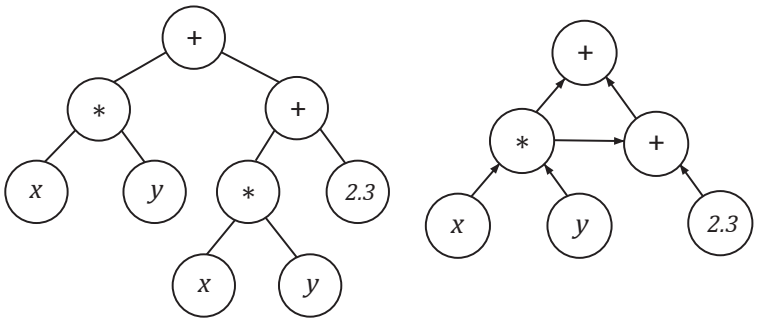
U linearnom genetskom programiranju, jedinka je prikazana kao skup instrukcija (naredbi) određenog programskog jezika jedna za drugom (slika 1.2). Za razliku od stablastog zapisa jedinka, ovakav način je puno lakši za čitati jer nalikuje na program napisan od strane čovjeka. Također je efikasniji pri računalnoj obradi i upotrebi jer se automatski može prevesti u izvršivi program. Mana ovakvog zapisa jedinke je u tome što su genetski operatori mutacije i križanja teže izvedivi zbog same prirode programskih jezika.



Slika 1.2 prikaz jedinke u linearnom genetskom programiranju

2.1.3. Genetsko programiranje temeljeno na grafovima

Genetsko programiranje temeljeno na grafovima pogodno je za evoluciju paralelnih programa. U zapisu jedinke u obliku grafa, za razliku od zapisa u obliku stabla, svaki čvor može biti povezan s bilo kojim drugim čvorom, što omogućuje izbjegavanje evaluacije jednakih izraza na različitim mjestima.

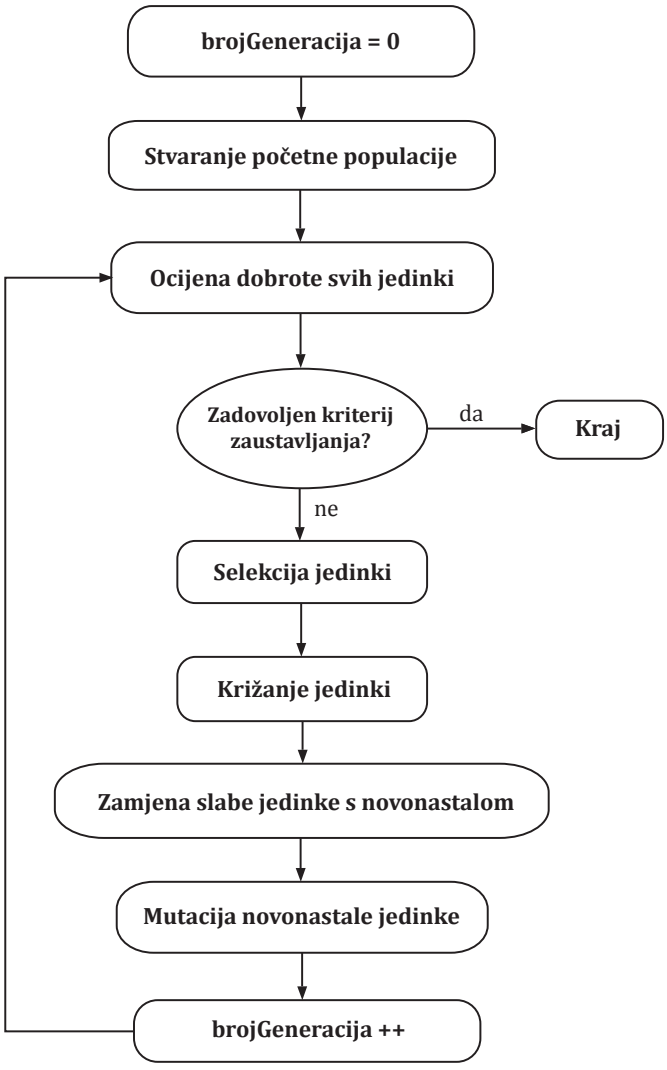


Slika 1.3 jedinka u obliku stabla (lijevo), te ista jedinka u obliku grafa (desno)

Na slici 1.3 vidimo usporedbu zapisa iste jedinke u obliku stabla i u obliku grafa. Jedinka predstavlja izraz $(x * y) + ((x * y) + 2.3)$. Kod zapisa jedinke u obliku stabla, izraz $x * y$ evaluira se dvaput, dok kod se kod jedinke zapisane u obliku grafa taj izraz evaluira jedanput, te se kao takav koristi na mjestima gdje je potrebno.

2.2. Tijek algoritma genetskog programiranja temeljenog na stablima

Na slici 2.1 prikazani su osnovni koraci genetskog programiranja. Algoritam se vrti sve dok nije zadovoljen kriterij zaustavljanja. U svakoj iteraciji algoritma određuju se dobrote svih jedinki te se na osnovu toga vrši selekcija nad populacijom. Nakon toga dolazi do križanja jedinki. Ovaj proces stvara nove jedinke koje se potom mutiraju.



Slika 2.1 dijagram toka genetskog programiranja

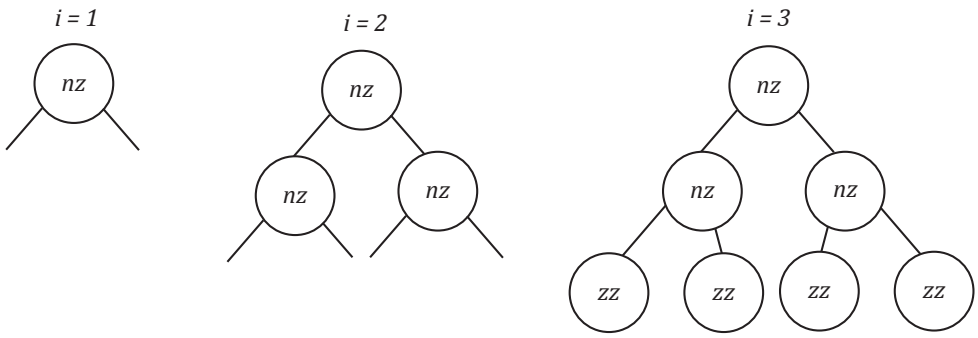
2.2.1. Generiranje početne populacije

Kod generiranja početne populacije, najčešće se stvara određen broj nasumično stvorenih stabala. Postoje tri vrste generiranja stabla, *ramped half-and-half, full* i *grow*, čije se objašnjenje nalazi u nastavku.

Osim nasumičnog generiranja stabla, postoji i metoda koja veći dio populacije generira nasumično, dodavajući u ostatak dobre, već postojeće jedinke (engl. *seeding*). Ova metoda bolja je od poptuno nasumičnog stavaranja populacije zbog toga što algoritmu dajemo naznaku u kojem smjeru bi se trebalo kretati kako bi došao do dobrog rješenja.

2.2.1.1. Full metoda generiranja stabla

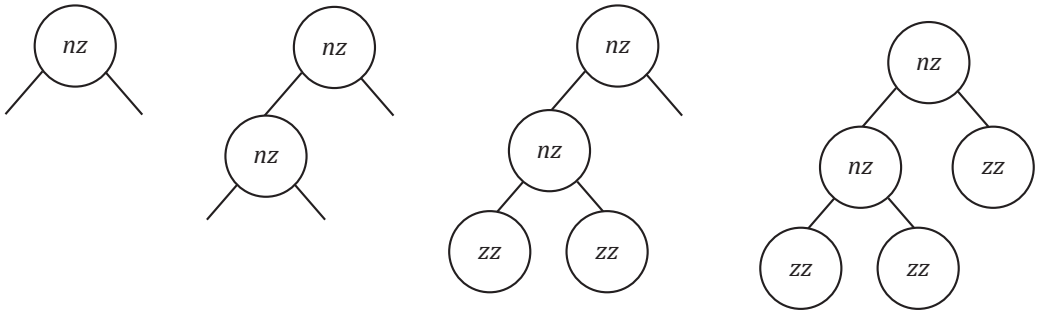
Ovom metodom generiraju se puna stabla; stabla kojima su svi listovi na jednakoj dubini. Sve dok stablo nije postiglo zadanu dubinu, stvarati će se čvorovi s nezavršnim znakovima. Nakon što je postignuta željena dubina, na sve listove mogu se dodati samo završni znakovi (slika 2.2).



Slika 2.2 generiranje stabla dubine 3 metodom *full* - *nz* označava nezavršni znak, a *zz* završni znak

2.2.1.2. Grow metoda generiranja stabla

Grow metodom stvaraju se i dodaju nasumično odabrani, završni ili nezavršni čvorovi sve dok dubina nije jednaka željenoj dubini. Nakon što je postignuta željena dubina stabla na nezavršne listove mogu se dodati samo završni čvorovi (slika 2.3).



Slika 2.3 generiranje stabla dubine 3 metodom *grow* - *nz* označava nezavršni znak, a *zz* završni znak

2.2.1.1. Ramped half-and-half metoda generiranja stabla

Ova metoda je kombinacija *full* i *grow* metode. Početna populacija dijeli se na toliko dijelova kolika je zadana maksimalna dubina stabla, D. Za svaki dio, polovica stabala generira se *full* metodom, a preostala polovica *grow* metodom. Također, svaki dio ima različitu dubinu stabala, u rasponu od 2 do D. Prednost ove metode je ta što se pri generiranju populacije stvaraju stabla različitih dubina što pridonosi raznolikosti populacije.

2.2.2. Dobrota jedinke

Dobrota jedinke predstavlja njenu kvalitetu u odnosu na to koliko dobro rješava zadan problem. Dobrota je jedan od najvažnijih svojstava jedinke, budući da utječe na buduće napredovanje algoritma, odnosno preživljavanje najboljih i odbacivanje najgorih jedinki u populaciji. Izračun dobrote ovisi o prirodi problema kojeg algoritam rješava.

2.2.3. Selekcija

Selekcija je mehanizam koji osigurava prijenos najboljih karakteristika jedinke u sljedeće generacije populacije. Dvije najbitnije vrste selekcija su jednostavna proporcionalna selekcija (engl. *roulette-wheel* selection) i eliminacijska K-turnirska selekcija.

2.2.3.1. Jednostavna proporcionalna selekcija

U ovoj vrsti selekcije, vjerojatnost odabira određene jedinke za postojanje u idućoj generaciji proporcionalna je njezinoj dobroti. Ovakvim načinom selekcije, velika je vjerojatnost da će dobre jedinke preživjeti i ostati u sljedećoj generaciji. No, mana

ovog načina selekcije je ta što može doći do pretjeranog izabiranja nekoliko najboljih jedinki što može poremetiti raznolikost populacije, te posljedično tome izazvati okupljanje rješenja oko jedne točke koja ne mora biti globalni, već samo lokalni optimum.

2.2.3.2. Eliminacijska K-turnirska selekcija

Kod ove selekcije, nasumično se odabire K jedinki iz populacije. Unutar tih K odabranih, pronalaze se dvije najbolje i najlošija jedinka te se elminira. Na mjesto eliminirane jedinke dolazi nova jedinka, nastala križanjem dvije najbolje jedinke iz tog turnira. Ovakvom selekcijom, u iduću generaciju uvijek će ući najbolja jedinka trenutne generacije. Na slici 2.4 nalazi se pseudokod ove selekcije.

```
turnir = nasumično izaberi k jedinki iz populacije;
roditelj1 = najboljaJedinka (turnir);
roditelj2 = drugaNajboljaJedinka (turnir);
dijete = najgoraJedinka (turnir);
izbriši dijete iz populacije;
dijete = križanje (roditelj1, roditelj2);
dodaj novo dijete u populaciju;
```

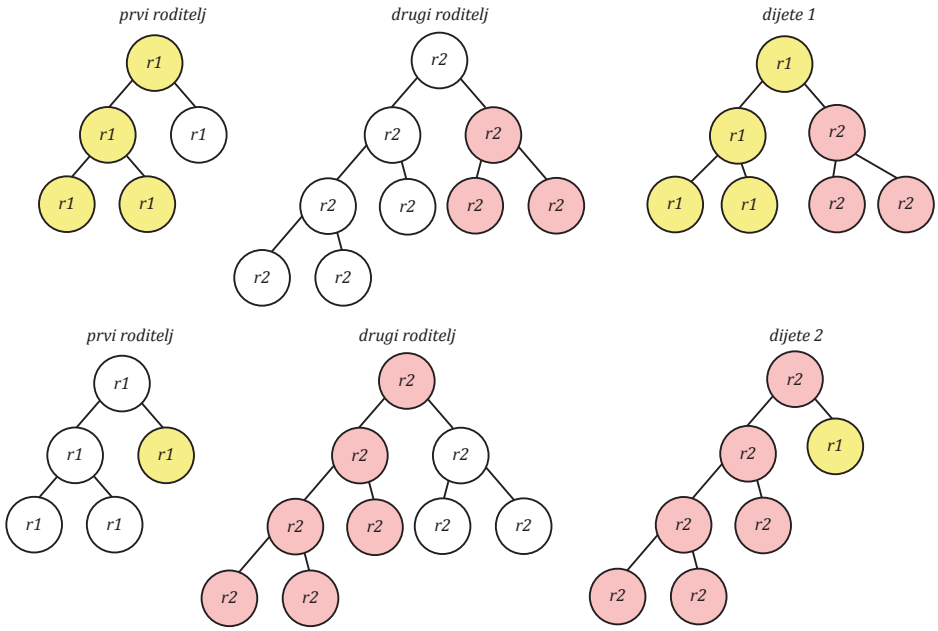
Slika 2.4 pseudokod eliminacijske K-turnirske selekcije

2.2.4. Križanje

Križanje omogućuje kombiniranje karakteristika dvaju roditelja jedinki kako bi dobili novu. Ovim mehanizmom algoritam napreduje sakupljanjem i rekombiniranjem najboljih genetskih materijala. Iako je namjera da jedinka dijete bude bolja od svojih roditelja, nije nužno da će se to dogoditi. Princip križanja analogan je onome u prirodi, geni djeteta kombinirani su geni njegovih roditelja. U nastavku su opisane dvije vrste križanja, križanje podstabala (*engl. subtree-crossover)* i uniformno križanje.

2.2.4.1. Križanje podstabala

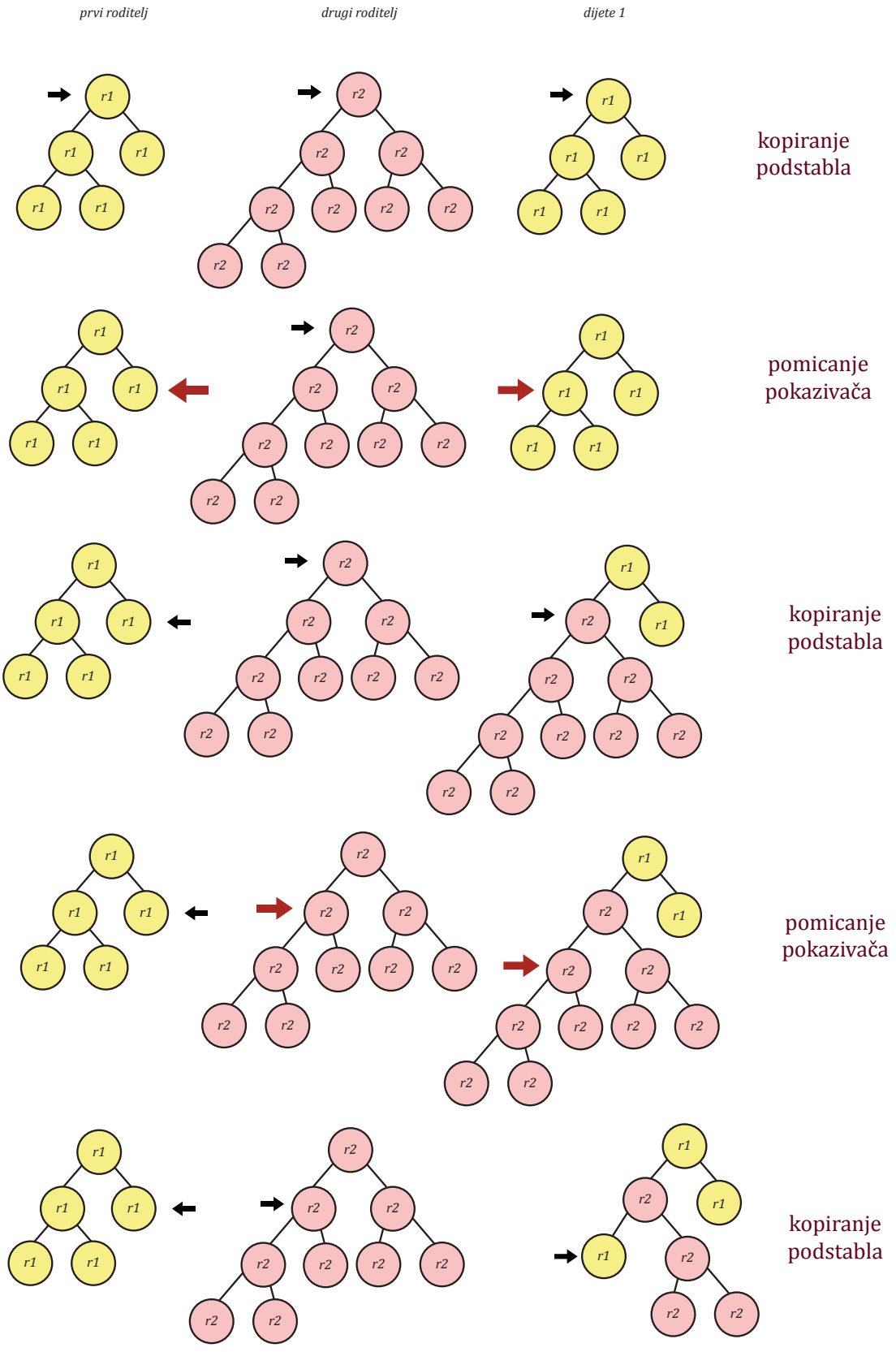
U ovakvom križanju, nasumičnim odabirom izabire se roditelj od kojega će se uzeti početni čvor te djeca tog čvora zaključna s lijevim čvorom odluke. Na mjesto desnog čvora odluke, kopira se desni čvor te njegovo podstablo odluke (ako postoji) početnog čvora drugog roditelja (slika 2.5).



Slika 2.5 križanje podstabala

2.2.4.2. Uniformno križanje

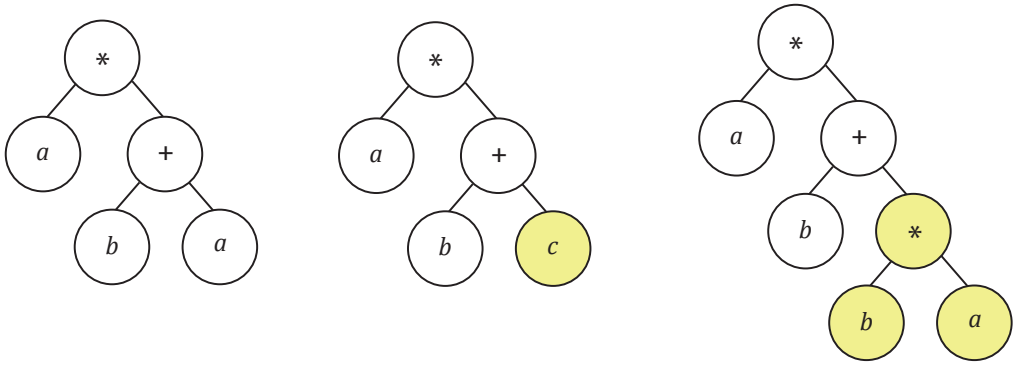
Ovo križanje izvodi se rekurzivno. U svakoj iteraciji u početni čvor djeteta kopira se čvor i pripadajuće podstablo nasumično odabranog roditelja. Nakon kopiranja, pokazivač odabranog roditelja i djeteta nasumično se pomiče na vlastiti lijevi ili desni čvor odluke, te se funkcija ponovno poziva s parametrima roditelj1, roditelj2, dijete. Na slici 2.6 detaljno je prikazan algoritam uniformnog križanja.



Slika 2.6 uniformno križanje

2.2.5. Mutacija

Mutacija se događa u novonastalim jedinkama populacije. Ona može promijeniti nasumično odabran gen, ili pak dodati novi gen u jedinku u cilju njenog poboljšanja (slika 2.7). Budući da može postojati neko drugo dovoljno dobro, ili čak bolje rješenje problema od onoga u kojeg trenutna populacija konvergira, mutacija može biti koristan alat za pomak od točke obustavljanja napredovanja algoritma. Ukoliko mutacija pogorša dobrotu jedinke, ta jedinka ne bi trebala bitno pokvariti populaciju zahvaljujući mehanizmima selekcije, odnosno eliminacije najlošijih jedinki.



Slika 2.7 lijevo - jedinka prije mutacije; sredina - jedinka nakon mutacije izmjenom postojećeg gena; desno - jedinka nakon mutacije dodavanjem novog gena

2.2.6. Ograničavanje pretjeranog rasta jedinki

Pretjerana veličina jedinke može uzrokovati znatno usporenje izvršenja programa. Budući da kroz više generacija populacije može doći do pretjeranog rasta stabala jedinki, moraju postojati mehanizmi koji to sprječavaju. Pretjeran rast jedinke može se sprječiti jednostavnim podrezivanjem stabla na željenu maksimalnu dubinu, tj. zamjenom predubokih grana sa završnim znakovima. Ovakav princip može biti destruktivan prema napredovanju algoritma ukoliko se prečesto pojavljuje potreba za podrezivanjem. Drugi pristup rješenju ovoga problema bio bi da, ukoliko dođe do pretjerane veličine jedinke tijekom operacija u kojoj se može desiti rast stabla (križanje ili mutacija), operacija ponavlja sve dok veličina stabla nakon operacije nije u okvirima zadane maksimalne veličine.

3. Primjena genetskog programiranja u strojnom učenju

Strojno učenje je metoda kojom pronalazimo generalizirane algoritme rješavanja novih specifičnih problema na temelju već rješenih, sličnih problema istog područja. Najčešća primjena strojnog učenja je u klasifikaciji podataka. Klasifikacijom želimo pronaći nepoznata, relevantna i točna svojstva podatka na osnovi njegovih ostalih, poznatih svojstava. Razlikujemo dvije vrste učenja na temelju kojih gradimo algoritme klasifikacije; učenje s nadzorom i učenje bez nadzora. Kod učenja s nadzorom, na skupu podataka za učenje svojstvo za klasifikaciju koje želimo naučiti odrediti već postoji, dok kod učenja bez nadzora, algoritam mora sam pronaći svojstvo prema kojem će klasificirati podatke.

Kako bi klasifikacija uzoraka bila dobra, moramo izgraditi dobar klasifikator koji mora moći ispravno klasificirati još neviđen podatak. Pri građenju klasifikatora, moramo proizvesti mehanizme koji na osnovu otkrivanja poveznica između poznatih svojstava pronalaze klasu kojem taj podatak pripada. Radi toga, klasifikator je najćešće program sastavljen od *if-then-else* blokova. Posljedično tome, lako ga možemo vizualizirati i zapisati u obliku stabla odluke.

U ovome trenutku možemo uočiti poveznicu između klasifikatora prikazanog u obliku stabla odluke i genetskog programiranja temeljenog na stablima. Budući da genetsko programiranje temeljeno na stablima manipulira populacijom stabala odluke, klasifikator možemo izgraditi upravo pomoću tog algoritma, "prevodeći" učenje u izvršavanje genetskog programiranja.

3.1. Stabla odluke

Postoje različiti oblici stabala kojima možemo prikazati jedinku. Najznačajnija su linearna i nelinearna stabla odluke.

3.1.1. Linearna stabla odluke

Kod linearnih stabala odluke, svaki čvor ima više od dvoje djece. Ovakva stabla sadrže čvorove koji predstavljaju aritmetički izraz koji se treba evaluirati u svrhu daljnjeg izvršavanja programa opisanog stablom. Stablo se sastoji od međusobno povezanih čvorova koji sadrže završni ili nezavršni znak. Nezavršni, odnosno funkcijski znak, u ovom slučaju označava koliko varijabli sadrži aritmetički izraz koji

je potrebno evaluirati, a završne znakove dijelimo na konstante, varijable i znakove odluke. Od završnih znakova konstanti i varijabli gradimo aritmetički izraz. Djeca nezavršnog znaka su prvo parovi konstante i varijable, zatim čvor koji označava odluku ako je izraz istinit, te na poslijetku čvor koji označava odluku ako evaluirani izraz nije istinit. Čvorovi odluke mogu biti i završni i nezavršni znakovi. Aritmetički izraz koji se evaluira gradi se kao

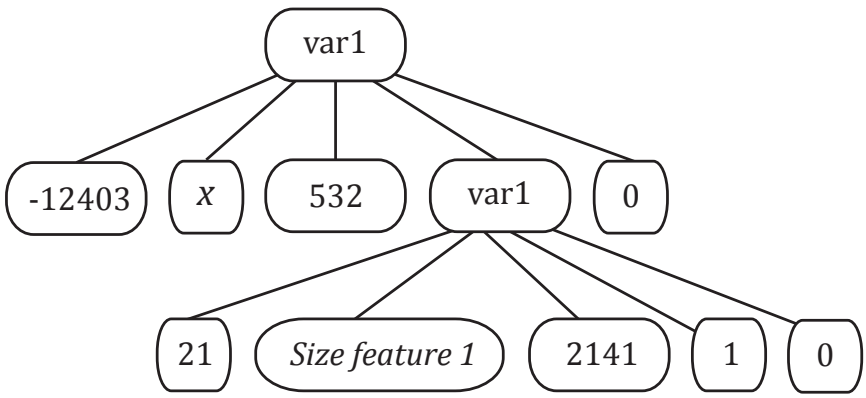
$$\sum_{n} konstanta_n \times varijabla_n \leq konstanta_{uspoređivanje}$$

Ovakvo stablo bi uz drugačiju interpretaciju moglo manipulirati i tekstualnim podacima (slika 3.1).

```
izraz = true;
za svaki ( par varijabla -> konstanta)
{
   ako ( varijabla != konstanta)
         {
               izraz = false;
               break;
         }
}
```

Slika 3.1 moguća interpretacija linearnog stabla odluke za tekstualne podatke

Na slici 3.2. vidimo primjer jednog linearnog stabla odluke, a njegovu interpretaciju na slici 3.3.



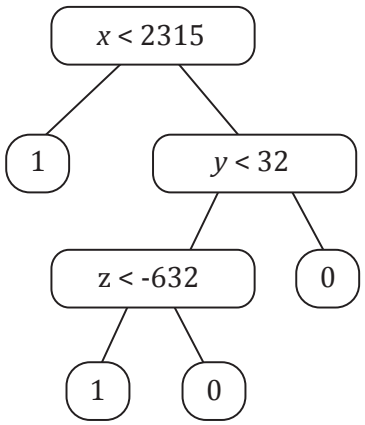
Slika 3.2 linearno stablo odluke

```
ako ( -1243 * x <= 532)
{
   ako ( 21 * SizeFeature1 <= 2141)
         odluka = 1;
   inače
   odluka = 0;
}
inače
{
   odluka = 0;
}
```

Slika 3.3 interpretacija nelinearnog stabla odluke sa slike 3.2

3.1.2. Nelinearna stabla odluke

Nelinearno stablo odluke je specijalizacija linearnog stabla odluke. Naime, nelinearno stablo u evaluiranom aritmetičkom izrazu može koristiti samo jednu varijablu. Izgled ovakvog stabla prikazan je na slici 3.4, a njegova interpretacija na slici 3.5.



Slika 3.4 izgled nelinearnog stabla odluke

```
ako (x < 2135)
   odluka = 1;
inače
{
   ako (y < 32)
   {
         ako (z < -632)
               odluka = 1;
         inače
               odluka = 0;
   }
   inače
         odluka = 0;
}
```

Slika 3.5 interpretacija nelinearnog stabla odluke sa slike 3.4

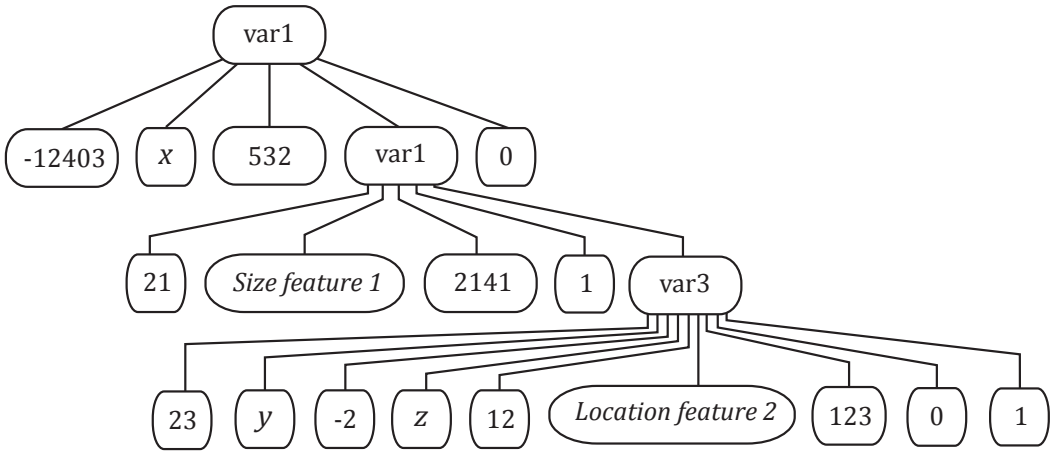
4. Ostvarenje dijagnosticiranja plućne embolije strojnim učenjem realiziranim genetskim programiranjem

Najtočnija metoda za dijagnosticiranje plućne embolije je angiografija kompjuteriziranom tomografijom. Ovom metodom dobivamo stotine slika snimanog objekta u kojem tražimo začepljenje krvne žile. Svaka slika predstavlja jedan sloj trodimenzionalnog prikaza snimljenog područja. U ovom slučaju, te slike su obrađene prikupljajući značajke o svakom objektu na slici koja bi mogla predstavljati začepljenje žile, odnosno emboliju. Skup podataka su dakle, potencijalna začepljenja, a svaki podatak opisuje 116 značajki. Značajke opisuju položaj, oblik i susjedstvo potencijalnog začepljenja te su numeričkog oblika.

Budući da je skup podataka velik, te njegova obrada dugotrajna i teška za čovjeka, rješavanje ovog problema strojnim učenjem realiziranim genetskim programiranjem čini se kao primjereno rješenje.

4.1. Prikaz i generiranje jedinke

Jedinka predstavlja program sastavljen od *if-then-else* blokova implementiranim kao linearno stablo (slika 4.1). Nezavršni čvorovi su oblika varX, gdje X označava broj varijabli evaluiranog aritmetičkog izraza. Završni čvorovi su konstante (cijeli brojevi u rasponu $[-2^{31}, 2^{32} - 1]$), varijable (vrijednosti značajki podataka) te čvorova odluka, koji mogu biti 0 ili 1. Ukoliko je sadržaj čvora odluke 1, kandidata za emboliju klasificiramo kao emboliju, a ukoliko je sadržaj čvora odluke 0, ne.



Slika 4.1 izgled jedinke

Generiranje jedinke implementirano je grow metodom. Dubina generiranog stabla nasumično je odabrana u rasponu od 2 do specificirane maksimalne dubine za čitavu populaciju.

4.2. Izračunavanje dobrote jedinke

Nakon dijagnoze plućne embolije, pacijentu se propisuju lijekovi za sprječavanje nastajanja novih ugrušaka. Ukoliko je pacijentu pogrešno dijagnosticirana embolija, može doći do neželjenog spontanog krvarenja. Ukoliko pacijent boluje od embolije, a ne donese se ispravna dijagnoza, postoji vjerojatnost da će izgubiti život. Zbog ovih razloga, pri računanju dobrote, za svaku točnu dijagnozu, dobroti pribrajamo 3. Za svaki put kada je embolija dijagnosticirana, a nije prisutna, dobroti oduzimamo 1, dok za pogrešno nedijagnosticiranje dobroti oduzimamo 2. Na kraju taj broj dijelimo s ukupnim brojem svih dijagnoza kako bi dobrota bila neovisna o veličini skupa nad kojim se računa. U ovom slučaju, dobrota može biti i negativna.

4.3. Selekcija

Za selekciju u ovoj implementaciji odabrana je eliminacijska K-turnirska selekcija. Iz populacije nasumično se odabire K jedinki. Unutar tih K jedinki turnira, odbacuje se ona s najmanjom dobrotom, te se nadomještava novom jedinkom nastalom križanjem dviju najboljih jedinki turnira.

4.4. Križanje

Implementacija sadrži dvije vrste križanja; križanje podstabala (engl. *subtree-crossover*) i uniformno križanje.

4.5. Mutacija

Mutacija se događa s vjerojatnošću zadanom parametrom faktora mutacije. Implementacija sadrži dvije vrste mutacija, mutaciju *grow* metodom te *point* mutaciju.

4.5.1. Mutacija grow metodom

Kod ove mutacije, nasumično odabran čvor odluke mijenja se novim stablom generiranim *grow* metodom. Maksimalna dubina tog novog stabla dana je parametrom dubine mutacije.

4.5.2. Point mutacija

Ova mutacija mijenja bilo koje čvorove u stablu. Tijekom jedne mutacije, može se dogoditi promjena više čvorova istog stabla. Ukoliko je čvor konstanta, njena vrijednost se mijenja, a ukoliko je čvor varijabla, mijenja se drugom varijablom. Ako je čvor odluke list, tj. ima vrijednost 0 ili 1, vrijednost mu se mijenja u 1, odnosno 0. Ukoliko je čvor odluke nezavršan znak, on postaje završni znak, postavljajući svoju vrijednost u 0 ili 1.

4.6. Ograničavanje prekomjernog rasta stabala

Kontrola rasta stabala implementirana je jednostavnim podrezivanjem. Naime, ukoliko dubina pojedinog stabla postane veća od zadane maksimalne dubine, ono se podreže na željenu dubinu, mijenjajući sve nezavršne znakove na toj dubini u završne znakove odluke 0 ili 1.

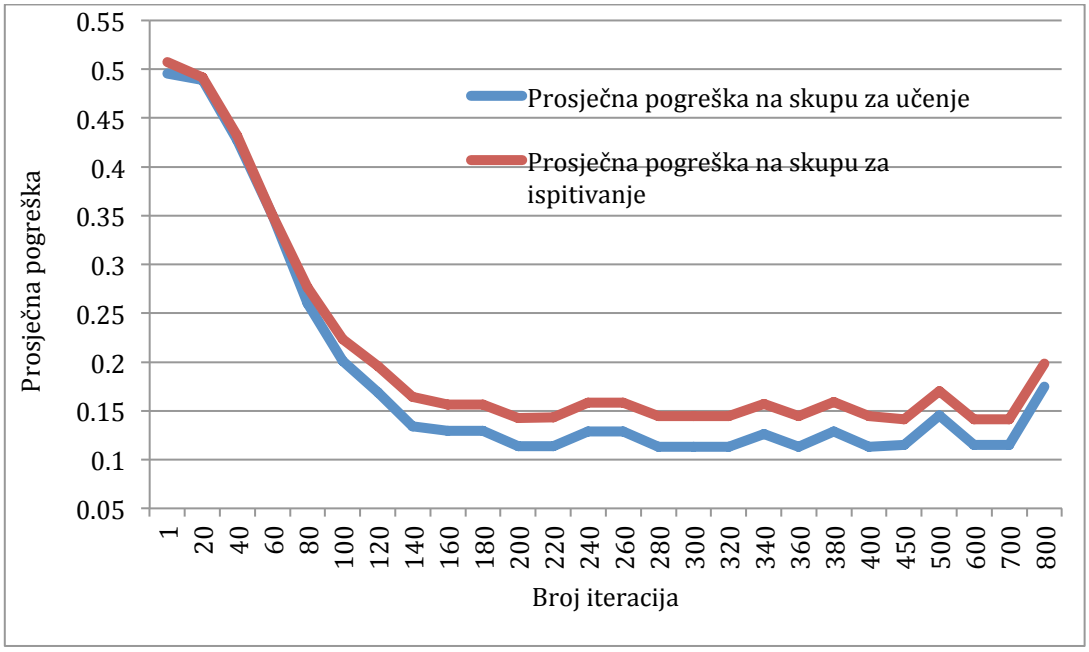
5. Analiza rezultata

Za potrebe ocjenjivanja učinkovitosti ostvarenog rješenja, proveden je niz eksperimenata. Prvi eksperiment proveden je u svrhu pronalaženja učinkovitosti programa s obzirom na broj generacija. Na temelju rezultata ovog eksperimenta pronalazimo optimalni broj generacija u kojem postižemo minimalnu pogrešku na skupu za učenje i skupu za ispitivanje, odnosno dijagnosticiranje embolije.

Skup za učenje i skup za ispitivanje odvojeni su skupovi ukupnog skupa podataka. Skup za učenje služi kako bi program na temelju tih podataka naučio dijagnosticirati bolest, dok skup za ispitivanje predstavlja programu još neviđeni skup podataka. Pomoću skupa za ispitivanje možemo zaključiti da li je pronađeno rješenje dobro za rješavanje još neviđenih problema, odnosno, da li je rješenje općenito. U ovom eksperimentu, skup za učenje sadrži 2493 podatka od kojih 280 predstavlja plućnu emboliju. Skup za ispitivanje sadrži 545 podataka, od kojih 83 predstavlja emboliju.

Pogreška predstavlja postotak netočnih dijagnoza unutar određenog skupa. Netočna dijagnoza je nedijagnosticiranje embolije ukoliko ona postoji ili dijagnosticiranje embolije ako one ne postoji. Zbroj ove dvije vrijednosti predstavlja ukupan broj netočnih dijagnoza. Prosječna pogreška predstavlja aritmetičku sredinu pogrešaka najboljih jedinki dobivenih u svim mjerenjima u sklopu eksperimenta.

Za ovaj eksperiment provedeno je 20 mjerenja. Genetski parametri se nisu mijenjali; veličina populacije iznosila je 40, maksimalna dubina stabla 10, veličina turnira 7, faktor mutacije 0.2, a dubina mutiranja 2. Broj iteracija iznosio je 800, pri čemu se nakon svake desete iteracije provjeravala pogreška na skupu za učenje i na skupu za ispitivanje. Provjeravanje pogreške na skupu za ispitivanje nije utjecalo na daljnji tijek algoritma, već je služilo samo za dokumentiranje napretka.



Slika 5.1 graf vrijednosti pogreške dijagnosticiranja s obzirom na broj generacija

Na grafu na slici 5.1 vidi se napredak algoritma s obzirom na broj generacija. U prvoj generaciji vrijednost pogreške kreće se oko 0.5. Nakon dvadesete generacije pogreška počinje opadati. Minimalna pogreška na skupu za učenje postiže se oko 220. generacije te je njen iznos oko 0.12. Pogreška na skupu za ispitivanje uglavnom prati pogrešku skupa za učenje. Ova pogreška veća je od one na skupu za učenje zbog "specijalizacije" algoritma nad skupom za učenje. Minimum prosječne pogreške na skupu za ispitivanje iznosi oko 0.14 te se, kao i kod pogreške na skupu za učenje, postiže oko 220. generacije. Nakon 220. generacije, obje pogreške se kreću u rasponu od 0.1 do 0.16, sve do oko 700. generacije, kada obje pogreške počinju ponovno počinju rasti.

Budući da su obje vrijednosti pogreške postigle minimum za 220 generacija, taj broj generacija korisiti će se u budućim eksperimentima.

5.1. Učinkovitost s obzirom na veličinu populacije

Za ovaj eksperiment provedena su mjerenja za veličine populacije 10, 50, 100, 200, 500, i 1000. Skup za učenje sadržavao je 2493 podatka, a skup za ispitivanje 545. Veličina turnira bila je 5, faktor mutacije 0.1, a maksimalna dubina stabla 8. Ovaj pokus ponovljen je po 5 puta za svaku veličinu populacije.

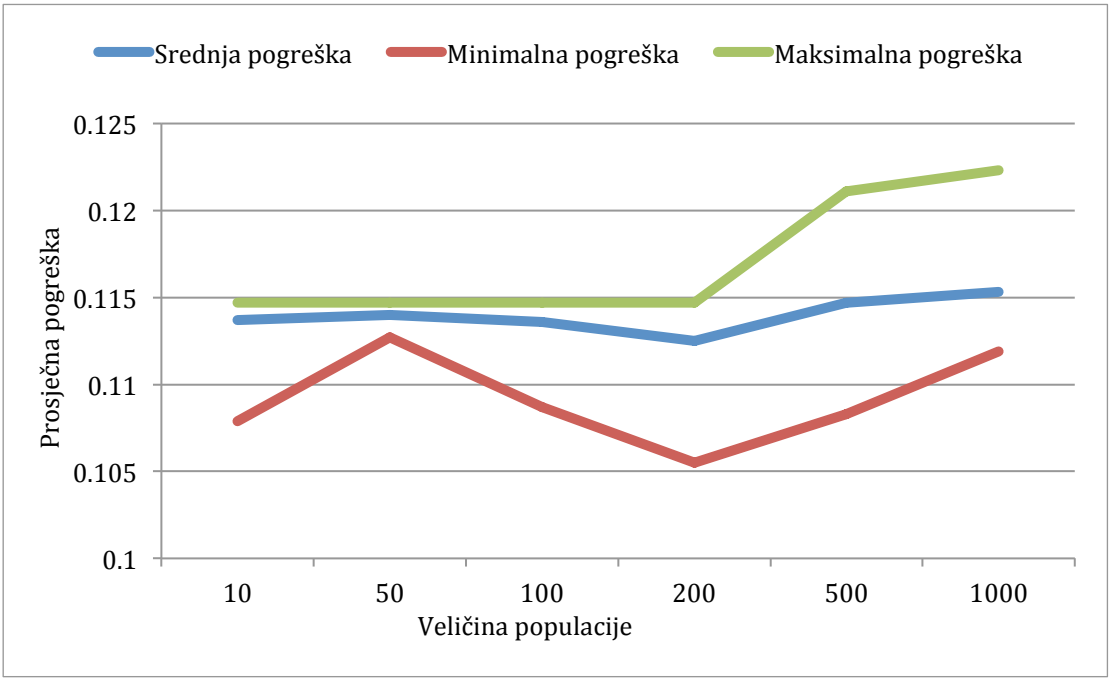
Tablica 5.1 minimalna, srednja i maksimalna pogreška s obzirom na veličinu populacije

	pogreška na skupu za učenje			pogreška na skupu za ispitivanje		
veličina populacije	srednja	minimalna	maksimalna	srednja	minimalna	maksimalna
10	0.1137	0.1079	0.1147	0.1413	0.1321	0.1505
50	0.1140	0.1127	0.1147	0.1415	0.1266	0.1578
100	0.1136	0.1087	0.1147	0.1385	0.1131	0.1449
200	0.1125	0.1055	0.1147	0.1393	0.1284	0.1486
500	0.1147	0.1083	0.1211	0.1489	0.1358	0.1798
1000	0.1153	0.1119	0.1223	0.1492	0.1339	0.1802

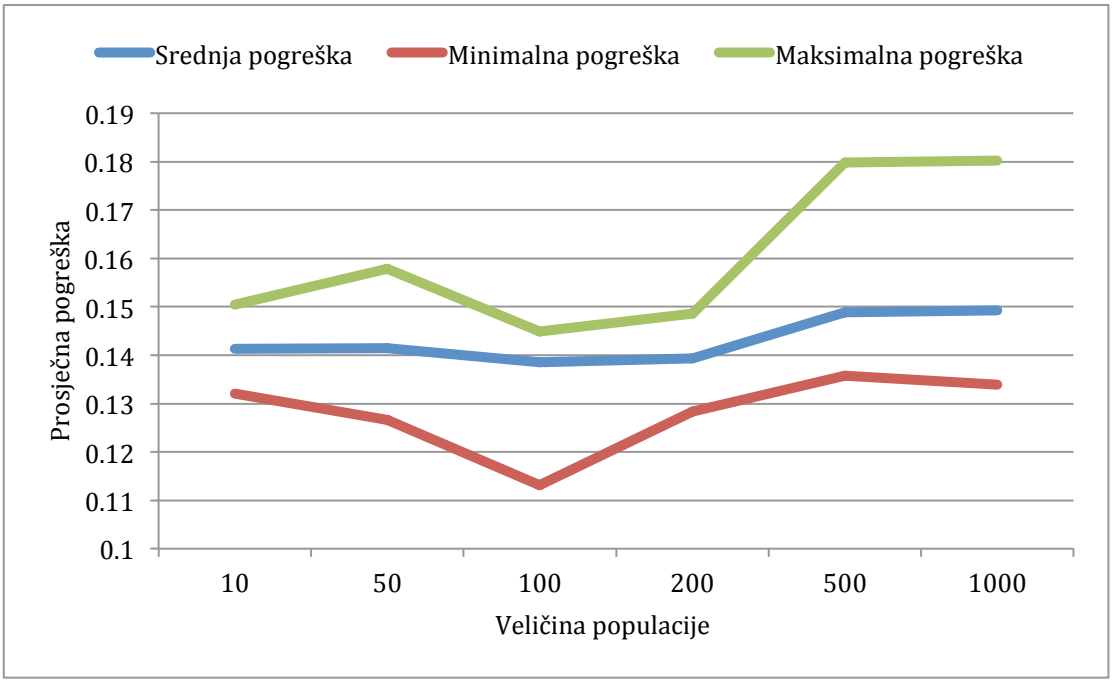
U tablici 5.1 prikazani su iznosi srednje, minimalne i maksimalne vrijednosti pogreške na skupu za učenje i skupu za ispitivanje u ovisnosti o veličini populacije. Kretanje tih vrijednosti vidimo na slikama 5.2 i 5.3.

Na slici 5.2 praćeno je kretanje iznosa pogreški na skupu za učenje. Maksimalna pogreška stoji na vrijednosti od 0.1147 do veličine populacije od 200, te nakon toga raste do vrijednosti od oko 0.12. Minimalna pogreška svoj maksimum postiže za veličinu populacije 50, dok minimum postiže za veličinu populacije 200, kada iznosi 0.1055. Srednja pogreška uglavnom se kreće oko vrijednosti od 0.114, a minimum, kao i minimalna pogreška, postiže za veličinu populacije 200.

Na slici 5.3 promatrano je kretanje pogreški na skupu za ispitivanje. Maksimalna pogreška kreće se između vrijednosti 0.145 do 0.16 do veličine populacije od 200 jedinki, kada naraste na vrijednost od oko 0.18. Minimalna pogreška postiže minimum od 0.1131 za populaciju veličine 100.



Slika 5.2 graf vrijednosti srednje, minimalne i maksimalne pogreške na skupu za učenje s obzirom na veličinu populacije



Slika 5.3 graf vrijednosti srednje, minimalne i maksimalne pogreške na skupu za ispitivanje s obzirom na veličinu populacije

Budući eksperimenti biti će provođeni nad veličinom populacije od 100 jedinki, zbog toga što za tu veličinu populacije minimalna pogreška skupa za ispitivanje postiže svoj minimum.

5.2. Učinkovitost s obzirom na različite parametre genetskih operatora

Sljedeći eksperimenti provedeni su u svrhu uočavanja utjecaja različitih parametara genetskih operatora na uspješnost konačnog rješenja. Promatrani parametri su maksimalna dubina stabla, veličina turnira (parametar K) eliminacijske K-turnirske selekcije, faktor mutacije te dubina mutiranja *grow* metodom. Svaki eksperiment u ovom poglavlju proveden je dvaput.

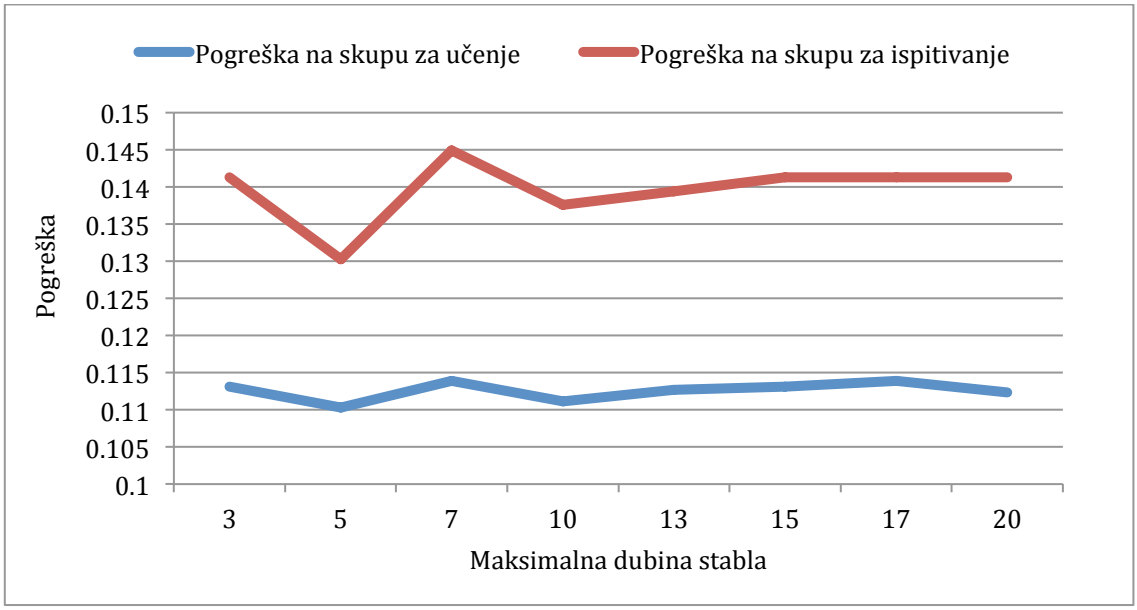
5.2.1. Učinkovitost s obzirom na maksimalnu dubinu stabala odluke

U tablici 5.2 prikazani su iznosi pogrešaka na skupu za učenje i skupu za ispitivanje. U ovim mjerenjima, veličina populacije je 100, veličina turnira 5, faktor mutacije 0.1, a dubina mutiranja *grow* metodom 2. Broj generacija iznosi 220.

Tablica 5.2 pogreške na skupu za učenje u ovisnosti o maksimalnoj dubini stabla populacije

maksimalna dubina stabla	pogreška na skupu za učenje	pogreška na skupu za ispitivanje
3	0.1131	0.1413
5	0.1103	0.1303
7	0.1139	0.1449
10	0.1111	0.1376
13	0.1127	0.1394
15	0.1131	0.1413
17	0.1139	0.1413
20	0.1123	0.1413

Na slici 5.4 prikazan je graf vrijednosti pogrešaka na skupu za učenje i skupu za ispitivanje u ovisnosti o dubini stabla. Krivulja pogreške na skupu za učenje, kao i ona na skupu za ispitivanje, postiže minimum za dubinu stabla 5. Obje krivulje postižu maksimum za istu vrijednost dubine stabla, dubinu 7. Za sve ostale vrijednosti, pogreška na skupu za učenje kreće se između vrijednosti 0.11 i 0.14, dok se pogreška na skupu za ispitivanje kreće između 0.13 i 0.145.



Slika 5.4 graf vrijednosti pogreške na skupu za učenje i skupu za ispitivanje u ovisnosti o maksimalnoj dubini stabla populacije

5.2.2. Učinkovitost s obzirom na parametar K eliminacijske K-turnirske selekcije

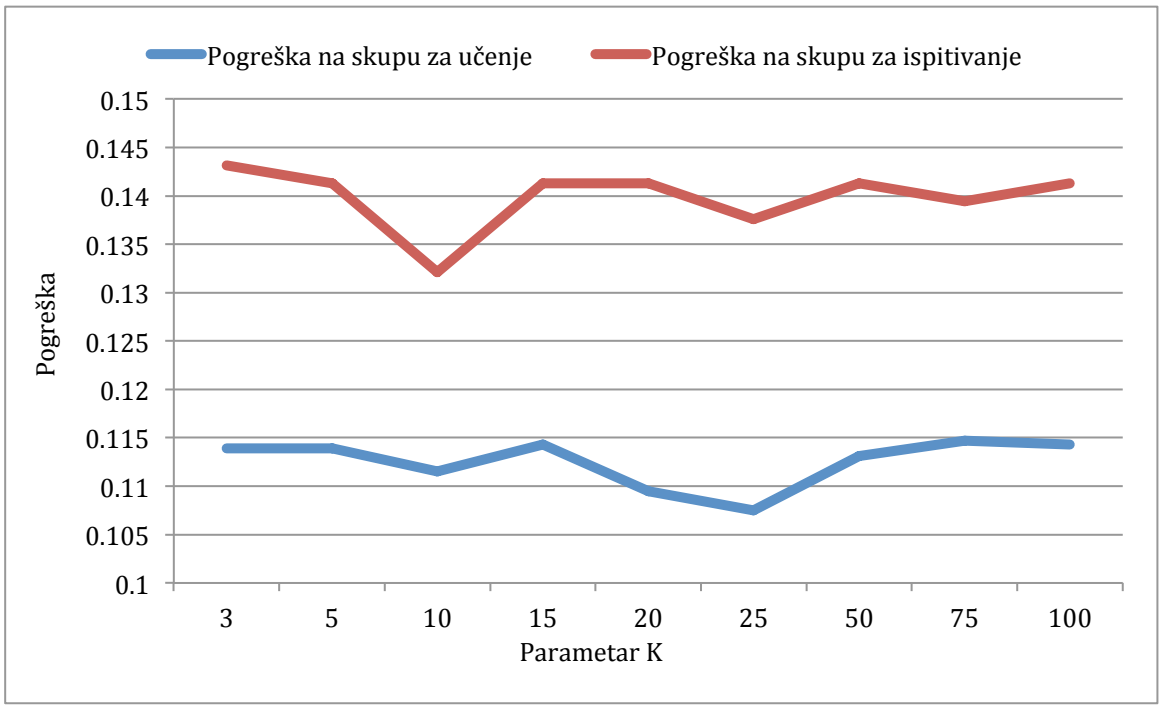
Ovim eksperimentom pokazuje se ovisnost veličine pogreške s obzirom na parametar K eliminacijske K-turnirske selekcije. Vrijednosti ostalih parametara genetskih operatora su: 100 za veličinu populacije, 8 za maksimalnu dubinu stabla, te 2 za dubinu mutacije. Broj generacija iznosi 220.

U tablici 5.2 prikazani su rezultati eksperimenta. Mjerenja su izvršena za parametre K vrijednosti 3, 5, 10, 15, 20, 25, 50, 75 i 100.

Tablica 5.2 pogreške na skupovima za učenje i ispitivanje u ovisnosti o maksimalnoj dubini stabla

parametar K	pogreška na skupu za učenje	pogreška na skupu za ispitivanje
3	0.1139	0.1431
5	0.1139	0.1413
10	0.1115	0.1321
15	0.1143	0.1413
20	0.1095	0.1413
25	0.1075	0.1376
50	0.1131	0.1413
75	0.1147	0.1394
100	0.1143	0.1413

Na slici 5.5 prikazano je kretanje vrijednosti pogrešaka u ovisnosti o parametru K. Pogreška na skupu za učenje potiže minimum za parametar K vrijednosti 25, što je četvrtina cjelokupne populacije. Pogreška na skupu za ispitivanje postiže svoj minimum za K vrijednosti 10.



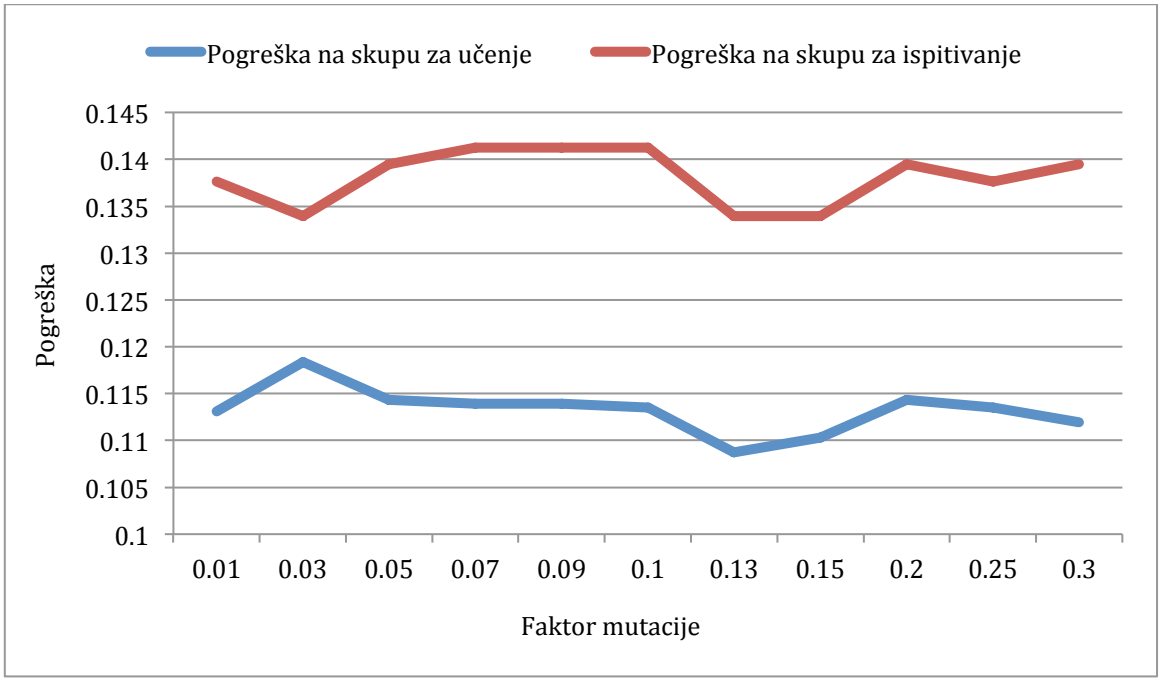
Slika 5.5 graf vrijednosti pogreške na skupu za učenje i skupu za ispitivanje u ovisnosti o parametru K eliminacijske K-turnirske selekcije

5.2.3. Učinkovitost s obzirom na faktor mutacije

Sljedećim eksperimentom prikazuje se utjecaj različitih faktora mutacije na ishod konačnog rješenja. Ostali parametri su fiksni; veličina populacije je 100, maksimalna dubina stabla 8, veličina turnira 5, a dubina mutiranja 2. U tablici 5.3 prikazani su rezultati ovog mjerenja, a na slici 5.6 graf kretanja pogrešaka u ovisnosti o faktoru mutacije.

Tablica 5.3 pogreške na skupovima za učenje i ispitivanje u ovisnosti o faktoru mutacije

faktor mutacije	pogreška na skupu za učenje	pogreška na skupu za ispitivanje
0.01	0.1131	0.1376
0.03	0.1183	0.1339
0.05	0.1143	0.1394
0.07	0.1139	0.1413
0.09	0.1139	0.1413
0.1	0.1135	0.1413
0.13	0.1087	0.1339
0.15	0.1103	0.1339
0.2	0.1143	0.1394
0.25	0.1135	0.1376
0.3	0.1119	0.1394



Slika 5.6 graf vrijednosti pogreške na skupu za učenje i skupu za ispitivanje u ovisnosti o faktoru mutacije

5.2.4. Učinkovitost s obzirom na dubinu grow mutiranja

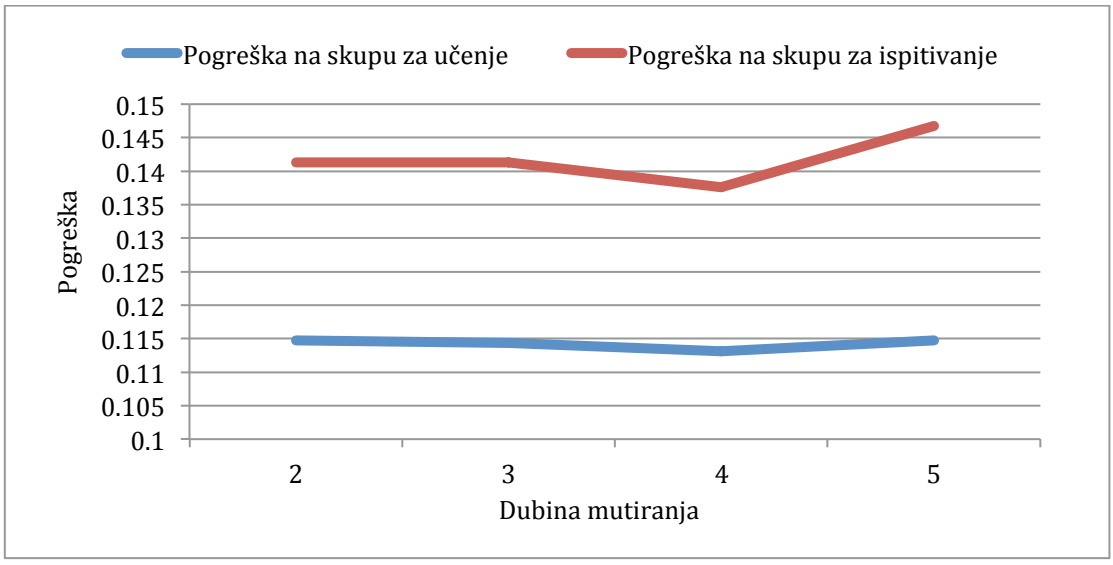
Rezultati ovog eksperimenta pokazuju ovisnost kvalitete konačnog rješenja u ovisnosti o dubini mutiranja. Dok se vrijednosti parametra dubine mutiranja mijenjaju, ostali parametri imaju fiksne vrijednosti. Veličina populacije je 100,

maksimalna dubina stabala 8, veličina turnira 5, faktor mutacije 0.1, a broj generacija 220. U tablici 5.4 prikazani su rezultati ovog mjerenja.

Tablica 5.4 pogreške na skupovima za učenje i ispitivanje u ovisnosti o dubini *grow* mutiranja

dubina mutiranja	pogreška na skupu za učenje	pogreška na skupu za ispitivanje
2	0.1147	0.1412
3	0.1143	0.1413
4	0.1131	0.1376
5	0.1147	0.1468

Na slici 5.7 prikazuje se kretanje pogrešaka u ovisnosti o dubini mutiranja. Pogreška na skupu za učenje za sve vrijednosti parametra dubine mutiranja kreće se oko vrijednosti 0.11. Pogreška na skupu za se ispitivanje također kreće oko iste vrijednosti; u ovom slučaju 0.14. Ovi rezultati pokazuju da dubina mutiranja uglavnom ne utječe na konačno rješenje.



Slika 5.7 graf vrijednosti pogreške na skupu za učenje i skupu za ispitivanje u ovisnosti o dubini mutacije

5.3. Učinkovitost s obzirom na veličinu skupova za učenje i ispitivanje

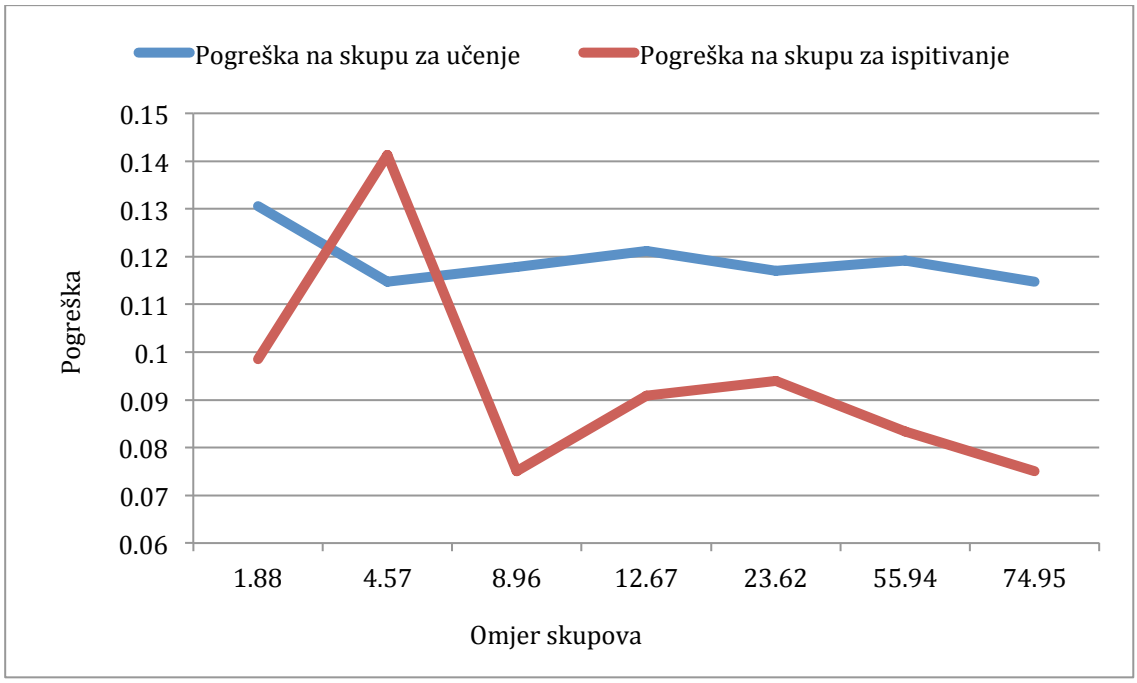
Sljedeći eksperiment pokazuje ovisnost uspješnosti konačnog rješenja s obzirom na različite veličine skupova podataka za učenje i skupova podataka za ispitivanje. U tablici 5.5 prikazani su rezultati mjerenja. U mjerenjima je veličina populacije bila 100, maksimalna dubina stabla 5, veličina turnira 10, faktor mutacije 0.13, dubina

mutacije 2, a broj generacija 220. Ovi parametri jednaki su onima koji su davali najbolja rješenja u prethodnim eksperimentima. U tablici 5.5 prikazani su rezultati ovog mjerenja.

Tablica 5.5 pogreške na skupovima za učenje i ispitivanje u ovisnosti o veličini skupa za učenje i skupa za ispitivanje

omjer veličine skupa za učenje i skupa za ispitivanje	pogreška na skupu za učenje	pogreška na skupu za ispitivanje
1.88	0.13061	0.09858
4.57	0.11472	0.14128
8.96	0.11782	0.075
12.67	0.12120	0.09090
23.62	0.11709	0.093939
55.94	0.11918	0.0833333
74.95	0.11474	0.075

Na slici 5.8 prikazana je ovisnost pogreške s omjerom skupova za učenje i ispitivanje. Pogreška na skupu za ispitivanje doseže jednake minimume za omjere veličine skupova 8.96 i 74.95.

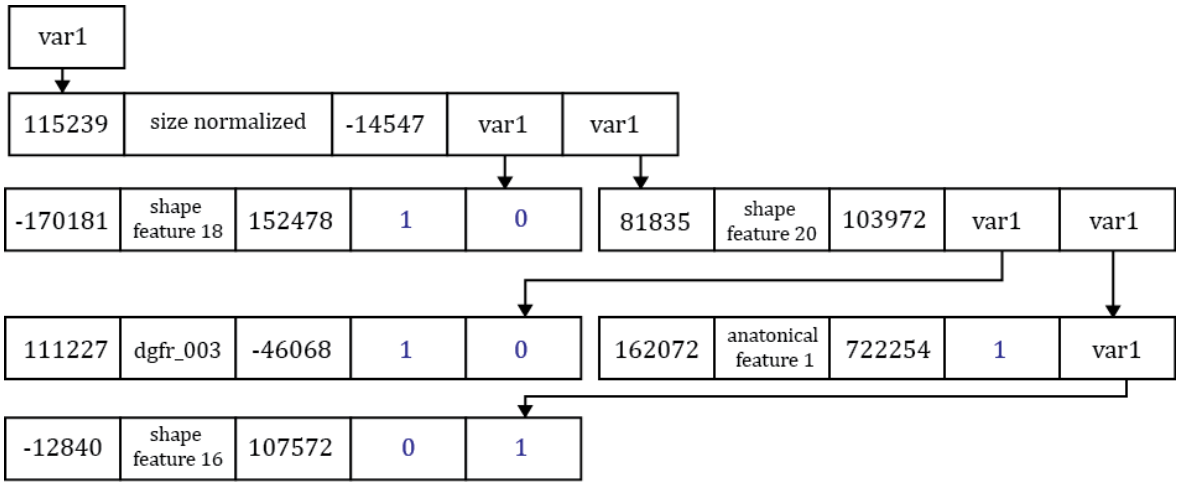


Slika 5.8 graf vrijednosti pogreške na skupu za učenje i skupu za ispitivanje u ovisnosti o veličini skupa za učenje i skupa za ispitivanje

Za omjere skupova 8.96 i 74.95 primjećuju se bitno bolji rezultati nego u drugim mjerenjima. No, razlog rezultata za omjer skupova 74.95 mogao bi biti jako malen skup za ispitivanje (svega oko 1% ukupnog skupa podataka) čime je i puno manja šansa za pogrešku. Radi toga, omjer skupova iznosa 8.96 bolji je od 74.95.

5.4. Primjer generirane jedinke

Na slici 5.9 prikazan je primjer generirane jedinke. Greška te jedinke na skupu za učenje iznosi 0.1149, a greška na skupu za ispitivanje 0.1082 (omjer skupova u ovom slučaju iznosi 8.96). Radi jednostavnijeg prikaza, djeca svakog čvora prikazana su u obliku liste na koju pokazuje strelica.



Slika 5.9 primjer generirane jedinke

6. Zaključak

Zbog prirode plućne embolije, njeno točno dijagnosticiranje moguće je samo pomoću angiografije kompjuteriziranom tomografijom (CTA). Ova metoda daje stotine slike presjeka slikanog objekta, čineći dijagnosticiranje teškim i iscrpnim za čovjeka.

Budući da ne postoji uvriježen algoritam za dijagnosticiranje ove bolesti, strojno učenje bilo je pogodno za njegov pronalazak. Ovim postupkom, automatizirano je učenje dijagnosticiranja, te posljedično tome i samo dijagnosticiranje. Donošenje odluke o dijagnozi reprezentirano je stablom odluke koji se prevodi u izvršivi program.

Samo strojno učenje realizirano je genetskim programiranjem. Jedinku u populaciji algoritma genetskog programiranja predstavlja stablo odluke. Ono će se nakon evolucije upotrijebiti za dijagnosticiranje bolesti.

Strojno učenje realizirano genetskim programiranjem pokazalo se kao dobro rješenje problema dijagnosticiranja plućne embolije.

7. Literatura

Terran Lane, Bharat Rao, Jinbo Bi, Marcos Salganicoff, *2006 KDD Cup Task: Computer Aided Detection of Pulmonary Embolism,* 10.10.2006., http://www.sigkdd.org/kddcup/site/2006/files/KDDCup2006_archive.tgz

Stokić I., Primjena genetskog programiranja u strojnom učenju, diplomski rad, FER, 2011.

Bot M, *Application of genetic programming to induction of linear classification trees*, 10.11.199., http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=DD7B4B642188600347AA944F7D394005?doi=10.1.1.35.8630&rep=rep1&type=pdf

Zhiwei Fu, Bruce L. Golden, Shreevardhan Lele, S. Raghavan, Edward A. Wasil, *A Genetic Algorithm-Based Approach for Building Accurate Decision Trees,* University of Maryland, 2003.

8. Sažetak

U ovom radu predstavljeno je potencijalno rješenje problema automatiziranog dijagnosticiranja plućne embolije pomoću strojnog učenja realiziranog genetskim programiranjem. Opisani su problemi dijagnosticiranja ove bolesti te osnovni principi genetskog programiranja i strojnog učenja. Prikazano je ostvareno rješenje sa svojim algoritmima te detaljnim rezultatima.

Ključne riječi: dijagnosticiranje plućne embolije, genetsko programiranje, strojno učenje

9. Abstract

This paper presents a potential solution to automated diagnosis of pulmonary embolism using machine learning. It describes problems of embolism diagnosis, as well as basic principles of machine learning and genetic programming. It describes implemented solution along with its algorithms and results.

Keywords: diagnosis of pulmonary embolism, genetic programming, machine learning