Prostorni Bloom filter

Bonašin, Daniel

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski sveučilišni studij računarstva

Diplomski rad

Prostorni Bloomov filter

Rijeka, siječanj 2023.

Daniel Bonašin
0069065087

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski sveučilišni studij računarstva

Diplomski rad

Prostorni Bloomov filter

Mentor: prof.dr.sc. Renato Filjar

Rijeka, siječanj 2023.

Daniel Bonašin
0069065087

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 13. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Usluge zasnovane na lokaciji**
Grana: **2.09.04 umjetna inteligencija**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Daniel Bonašin (0069065087)**
Studij: Diplomski sveučilišni studij računarstva
Modul: Programsko inženjerstvo

Zadatak: **Prostorni Bloomov filtar / Spatial Bloom Filter**

Opis zadatka:

Prostorni Bloomov filtar je algoritam za raspodijeljenu klasifikaciju informacija o lokaciji bez njihovog otkrivanja drugim stranama. Algoritam dobiva na značenju za potrebe prostorne kriptografije. U ovom diplomskom radu potrebno je predstaviti i analizirati teorijske osnove i obilježja prostornog Bloomovog filtra. Temeljem razumijevanja teorijske osnove, potrebno je razviti programsku podršku za izvedbu prostornog Bloomovog filtra u programskom okruženju za statističko računarstvo R. Uspješnost izvedbe potrebno je vrednovati putem reprodukcije izvornog istraživanja razvoja prostornog Bloomovog filtra prema zadanoj literaturi, uz korištenje razvijene programske podrške u okruženju R. Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

D. Bonašin

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Prof. dr. sc. Renato Filjar

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, siječanj 2023.

_____

Ime Prezime

Zahvala

Zahvaljujem svojem mentoru prof. dr. sc. Renatu Filjaru na podršci tijekom pisanja ovoga rada i korisnim raspravama i savjetima.

Zahvaljujem se svojoj obitelji što su mi omogućili studiranje te me podržavali cijelim životnim putem.

Na posljetku se zahvaljujem Karolisu Abrutisu i Marku Šegonu što su me konstantno motivirali za rad na ovom diplomskom radu.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Današnja popularizacija pametnih mobilnih telefona i mobilnih aplikacija dovela je do sve većeg razvoja lokacijskih usluga. Porastom broja korisnika lokacijskih usluga raste i količina podataka koju pružatelji usluga trebaju obraditi. U Europskoj uniji, pružatelji usluga moraju sažeti sadržaj identifikacijskih podataka korištenjem funkcije raspršivanja (eng. hashing) kako bi bili u skladu sa zakonskim regulativama.

Prostorni Bloomov filter (PBF), odličan je izbor za ovakvu primjenu s obzirom na to da zauzima malo memorijskog prostora i neovisno o broju točaka u konstantom vremenu može odrediti u kojem području interesa (eng. area of interest) se nalazi prostorna točka. Pritom ne otkriva iz kojeg je smjera korisnik došao i nema podataka o tome gdje se točka nalazi izvan područja interesa. Stoga, PBF može dati dovoljnu količinu podataka za izvedbu neke aplikacije temeljene na lokacijskim uslugama, pritom ne otkrivajući lokacijske detalje.

Ovaj rad će se osvrnuti na teoriju Bloomovog filtera i PBF-a (poglavlje "Teorija prostornog Bloomovog filtera"), izvedbu PBF u R programskom jeziku (poglavlje "Izvedba PBF u programskom okruženju R"), te vrednovanje navedene izvedbe (poglavlje "Vrednovanju izvedbe PBF u programskom okruženju R"). U zaključku je sažeto što je napravljeno u ovom diplomskom radu te se predlažu poboljšanja izvedbe. Uz rad su priložene upute za instalaciju R programskog jezika u prilogu A, upute za instalaciju aplikacije u prilogu B, te link koji vodi na izvorni kod aplikacije u prilogu C.

Poglavlje 2

Teorija prostornog Bloomovog filtera

2.1 Bloomov filter

Bloomov filter (BF) je struktura podataka kojom se određuje nalazi li se neki element u nekom skupu ili ne [1]. Brz je u određivanju, te zauzima malo memorije. BF je stohastička struktura zato što postoji vjerojatnost da klasificirani element zapravo nije dio skupa, no BF može s potpunom sigurnošću klasificirati da element ne pripada skupu.

Matematički izraz 2.1 opisuje BF. U njoj varijabla (B) predstavlja skup S pomoću skupa funkcija raspršivanja (eng. hash function) (H). Funkcije raspršivanja za ulaz imaju elemente (a) iz skupa S [1].

$$B(S) = \bigcup_{a \epsilon S, h \epsilon H} h(a) \tag{2.1}$$

Za izvedbu BF-a koristi se binarni vektor. Dužina m predstavlja duljinu vektora. Prilikom inicijalizacije vektora sve vrijednosti vektora postavljaju se na nulu. Broj funkcija raspršivanja u H, označava se s k. Funkcije raspršivanja kao argument primaju binarni niz (a) koji pripada skupu S. Funkcije raspršivanja kao rezultat daju broj koji se nalazi u intervalu od 1 do m. Rezultat odgovara indeksu polja binarnog vektora, unutar kojeg će se zapisati jedinica.

Funkcije raspršivanja trebaju se ravnati po uniformnoj razdiobi kako bi najbolje raspršile elemente u vektoru. Zbog jednolikog raspršivanja elemenata, manja je mogućnost kolizije, a s manjom mogućnosti kolizije smanjuje se i mogućnost krive kategorizacije.

Provjera pripadnosti elementa skupu započinje tako da sve funkcije raspršivanja iz skupa H za argument uzmu element. Rješenja funkcija raspršivanja koriste se kao indeksi pri određivanju vrijednosti unutar binarnog vektora. Element ne pripada skupu ako je i jedna vrijednost jednaka nuli. U suprotnom, postoji velika vjerojatnosti da je element dio skupa. Vjerojatnost pogrešne klasifikacije pripadnosti skupu označava se slovom p.

$$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \tag{2.2}$$

Kriva klasifikacija je moguća zbog kolizija koje funkcije raspršivanja uzrokuju. Kolizija može nastati tijekom zapisivanja elementa u Bloomov filter i tijekom provjere pripadnosti elementa.

Iz formule 2.2, može se izvesti optimalan broj funkcija raspršivanja 2.3 i optimalna duljina vektora 2.4 [1]. U tim formulama, n predstavlja broj elemenata.

$$k = \frac{m}{n} \ln 2 \tag{2.3}$$

$$m = \left\lceil -\frac{n \ln p}{(\ln 2)^2} \right\rceil \tag{2.4}$$

2.2 Prostorni Bloomov filter

Prostorni Bloomov filter (PBF) može provjeriti nalazi li se element u jednom od više skupova, za razliku od klasičnog BF-a koji može provjeriti nalazi li se element samo u jednom skupu. Prilikom provjere elementa, PBF vraća identifikator skupa, čime je određeno kojem skupu element pripada.

PBF također koristi funkcije raspršivanja, zbog čega postoje kolizije koje uzrokuju lažno pozitivne klasifikacije. Svaki skup također ima pripadnu lažno pozitivnu vjerojatnost. Tako je, primjerice, lažno pozitivna vjerojatnost za područje s najvećim prioritetom približno jednaka formuli 2.5 [1].

$$p_n \approx \left(1 - e^{-\frac{|\Delta_n|}{m}}\right)^k, \tag{2.5}$$

gdje je Δ_n područje s najvećim prioritetom, e Eulerov broj, m duljina vektora, a k broj funkcija raspršivanja. Za izračun lažno pozitivne vjerojatnosti za područje koje je odmah

nakon područja najvećeg prioriteta, vrijedi matematički izraz 2.6 [1].

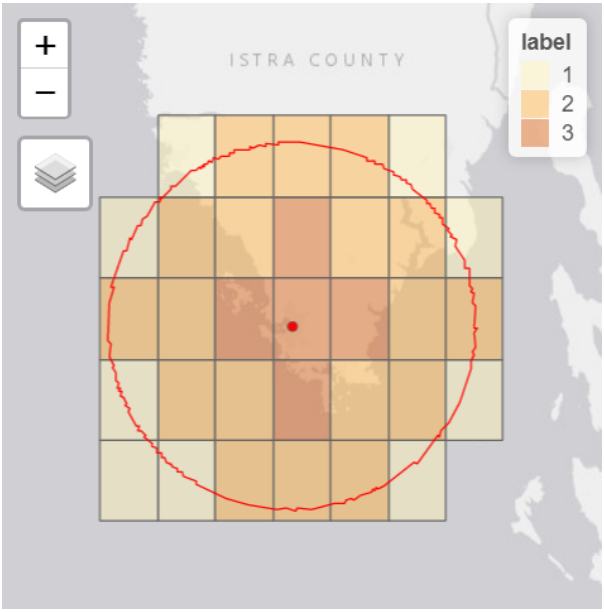
$$p_{n-1} \approx \left(1 - e^{-\frac{|\Delta_n \cup \Delta_{n-1}|}{m}}\right)^k - p_n, \tag{2.6}$$

gdje je Δ_n područje s najvećim prioritetom, Δ_{n-1} područje s prioritetom drugim po redu, m duljina vektora, k broj funkcija raspršivanja, te p_n lažno pozitivna vjerojatnost za područje najvećeg prioriteta.

Zbrojem lažno pozitivnih vjerojatnosti svih područja dobivamo sveukupnu lažnu pozitivnu vjerojatnost. Sveukupna lažno pozitivna vjerojatnost je jednaka lažno pozitivnoj vjerojatnosti Bloomovog filtera (Matematički izraz 2.2) [1].

$$p = \sum_{i=1}^{n} p_i \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \tag{2.7}$$

Kako PBF može imati više skupova, svakome od njih je zadan prioritet. Primjerice, imamo li koncentrična područja interesa čija središta pripadaju točki interesa (eng. point of interest), područjima blizu samog centra mogu se postaviti veći prioriteti od vanjskih područja. (Slika 2.1)



Slika 2.1 PBF s 3 područja interesa, uz točku interesa i radijus označen na mapi

Kako bi se smanjila mogućnost greške u područjima bližim točki interesa, u PBF filter se prvo zapisuju područja manjeg prioriteta, a zatim područja većeg prioriteta. Zapisivanjem elemenata iz područja većeg prioriteta postoji mogućnost promjene vrijednosti u PBF-u za elemente iz područja nižeg prioriteta, no vrijednosti elemenata iz područja većeg prioriteta sigurno neće biti zamijenjene vrijednostima nižih prioriteta.

Poglavlje 3

Izvedba PBF-a u programskog okruženju R

3.1 Programsko okruženje za statističko računarstvo R

R je programski jezik i okruženje za statističko računarstvo i grafički prikaz otvorenog koda. Dostupan je kao razvojno programsko okruženje otvorenog koda, pod uvjetima GNU Opće Javne Licence. Postoje verzije za različite operativne sustave, poput Linux-a, Windows-a i MacOS-a [2].

On pruža širok spektar izvedbi statističkih metoda i postupaka, poput linearnog i nelinearnog modeliranja, klasificiranja, grupiranja te ostalih izvedbi u R programskom okruženju. Navedene izvedbe statističkih metoda mogu se lako primijeniti za potrebe statističke analize, razvoja i provjere statističkih modela, grafičkih prikaza i dr.

Programsko okruženje R koristi posebni programski jezik R, s definiranom sintaksom. Programski jezik R podržava uobičajene funkcionalnosti programskih jezika, uključivo i mogućnost definiranja novih funkcija, kao i proširenje programskog jezika dodatnim knjižnicama (R packages). Programsko okruženje R podržava poziv i izvedbu programa pisanih u drugim programskim jezicima i okruženjima (C/C++, Fortran, Java, MatLab), uz korištenje posebnih API-jeva.

R ima odličnu podršku znanstvene zajednice koja pridonosi izradi dokumentacije i dodatnih materijala koji olakšavaju učenje R programskog jezika poput uputa, knjiga, tečajeva,

vodiča, itd.

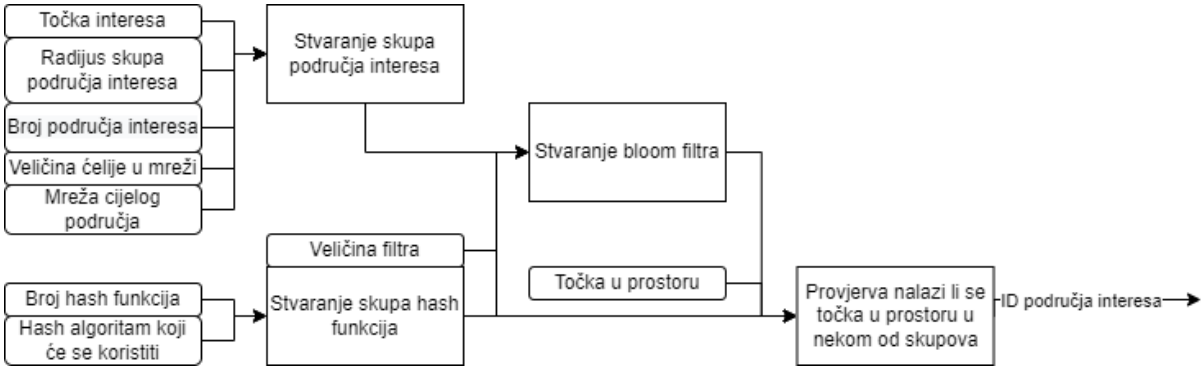
Proširenje funkcionalnosti osnovne konfiguracije okruženja R postignuto je dodatnim R knjižnicama. One omogućavaju izvedbu novih i specifičnih statističkih metoda.

Knjižnice je moguće preuzeti putem Comperhensive R Archive Network (CRAN) skupa internetskih stranica.

Programsko okruženje R pruža sustavnu i potpunu podršku za analizu i modeliranje prostornih opažanja, što ga čini odličnim izborom za ovaj rad.

3.2 Koncept izvedbe

Potpuni postupak kreiranja PBF-a i provjere testne točke prikazan je slikom 3.1.



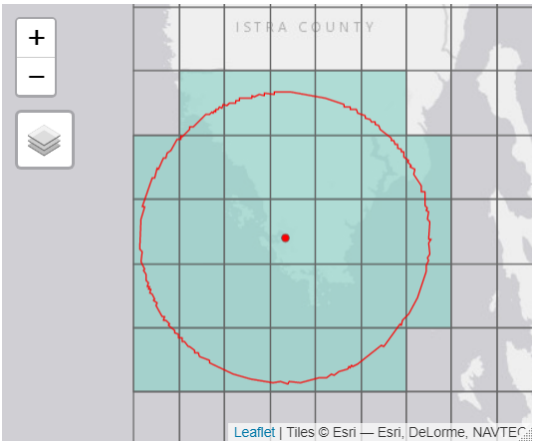
Slika 3.1 Dijagram PBF-a. Kvadrati sa zaobljenim uglovima predstavljaju ulazne podatke dok obični kvadrati predstavljaju korake.

Prije kreiranja PBF-a, potrebno je stvoriti skup područja interesa i skup funkcija raspršivanja. Nakon stvorenog PBF-a, može se provjeriti pripada li testna točka nekom od područja interesa. Za provjeru je potreban isti skup funkcija raspršivanja koji se koristio za kreiranje PBF-a. Provjera daje oznaku područja interesa kao rezultat.

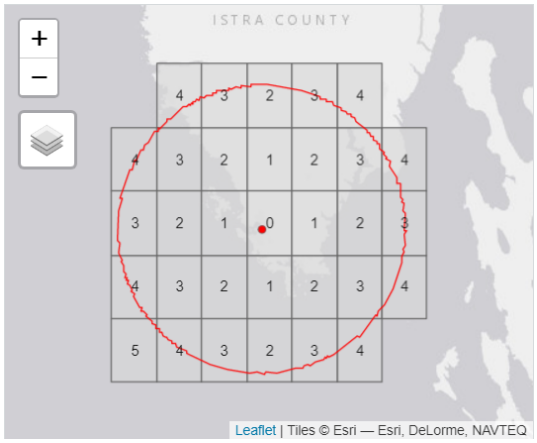
U nastavku slijedi detaljniji pregled elementa potrebnih za izradu filtra.

Prvi korak u dijagramu (Slika 3.1) je kreiranje skupa područja interesa. Za njegovo kreiranje, potrebna je prostorna mreža ε unutar koje će biti elementi δ koji će sadržavati

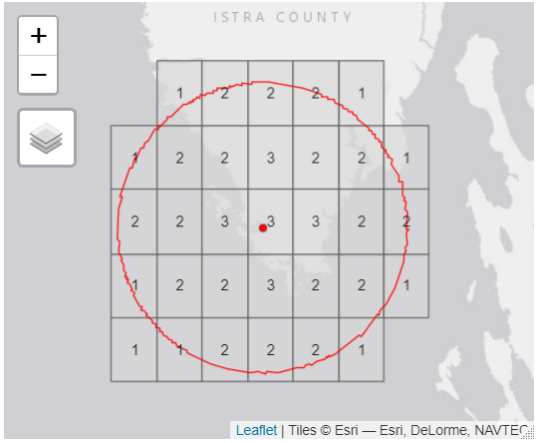
prostorne točke. Osim mreže, potrebni su središte (c), radijus (r) i broj željenih područja (n), koji će s ε biti ulazni podaci za algoritam 1.



(a) Presjek kruga i mreže.



(b) Izračun Manhattanove udaljenosti.



(c) Izračun skupa područja interesa.

Slika 3.2 Prikaz koraka izrade područja interesa

U predstavljenom algoritmu koristi se Manhattanova udaljenost, koja označava fizičku bliskost. No mogu se koristiti i izračuni za kontekstualnu bliskost, tj. za izračunavanje udaljenosti u lokacijskom krajoliku.

Prvi korak algoritma 1 je inicijaliziranje praznog skupa. U drugom koraku, kreira se krug radijusa r sa središtem c. Zatim se pronalazi element δ_c, unutar mreže ε, koji sadržava središte kruga. Element δ_c se dodaje skupu S. U petom koraku, svi se elementi iz mreže

ε dodaju skupu S koje prekriva kružnica C_r. Elementi koji će biti dodani u vektor, kao i sama kružnica, prikazani su slikom 3.2a. U šestom koraku, inicijalizira se vrijednost početne udaljenosti, dok se u sedmom koraku izračunavaju udaljenosti od središta kružnice za preostale elemente u skupu S. Stanje vrijednosti nakon sedmog koraka, prikazano je na slici 3.2b. Najveća udaljenost sprema se u varijablu σ, nakon čega započinje grupiranje elemenata u područja interesa. Varijabla q, koja se nalazi na 10. liniji algoritma 1, predstavlja količinu oznaka koje ćemo svrstati u pojedino područje, dok m predstavlja broj odjeljaka.

Konačno, algoritam vraća skup područja interesa. Na slici 3.2c prikazane su oznake u elementima. Vidljivo je da elementi bliže središtu imaju oznake najvećeg prioriteta, dok se udaljavanjem od središta elementima prioritet smanjuje.

Algoritam 1: Kreiranje područja interesa

Ulaz: c, r, ε, n

Izlaz: $S = \Delta_1 \cup \Delta_1 \cup ... \cup \Delta_n$

$S \leftarrow \emptyset$;

Određivanje kruga C_r pomoću središta c i radijusa r;

Pronalaženje elementa δ_c unutar ε koja sadrži c;

$S \leftarrow S \cup \{\delta_c\}$;

Umetanje svih elemenata u S koji su dio ε, a koje potpuno ili djelomično prekriva krug C_r, počevši od elemenata koji dodiruju δ_c;

Dodavanje vrijednosti udaljenosti 0 elementu δ_c ;

Izračun Manhattan-ove udaljenosti od δ_c za svaki $\delta \in S$ i dodjeljivanje izračunate vrijednosti svakom δ;

Spremanje najveće udaljenosti kao σ;

Grupiranje elemenata unutar S u područja Δ_i kojima će i biti oznaka;

$q \leftarrow \lfloor(\sigma + 1)/n\rfloor$;

$m \leftarrow (\sigma + 1) \mod n$;

for $i \leftarrow 1$ **to** n **do**

  if $m \neq 0$ **then**

    $\Delta_i \leftarrow S_{[\sigma-q,\sigma]}$;

    $\sigma \leftarrow \sigma - (q + 1)$;

    $m \leftarrow m - 1$;

  else

    $\Delta_i \leftarrow S_{[\sigma-q+1,\sigma]}$;

    $\sigma \leftarrow \sigma - q$;

  end

end

return S

R programski jezik nema skup kao strukturu podataka, već se u izvedbi koristi data frame. Zato su dijelovi algoritma drugačiji, no detaljnije objašnjenje bit će dano u odjeljku "3.3 Opis razvijene programske izvedbe PBF-a".

U drugom koraku dijagrama (Slika 3.1), kreira se skup funkcija raspršivanja. Ulazni podaci potrebni za kreiranje funkcija raspršivanja su broj funkcija raspršivanja i koji će se algoritam raspršivanja koristiti za njihovu izradu.

Kako bi svaka funkcija proizvela drugačiji rezultat za isti element, iako sve koriste isti algoritam za sažimanje sadržaja pomoću funkcije raspršivanja, koristi se soljenje (eng. salt). Sol predstavlja niz slučajnih alfanumeričkih znakova. Soljenjem se sol dodaje elementu, nakon čega se sadržaj sažme pomoću funkcije raspršivanja. Svaka funkcija raspršivanja ima svoju jedinstvenu sol, čime se osigurava da za isti element različitim funkcijama nije moguće dobiti isti rezultat.

Nakon sažimanja sadržaja pomoću funkcije raspršivanja, hash se pretvara u heksadecimalni zapis. Dobiveni broj može se raščlaniti u cjelobrojni broj koji će se prilikom kreiranja i provjere PBF-a podijeliti s veličinom PBF vektora, a čiji ostatak predstavlja redni broj polja u PBF vektoru.

Treći korak u dijagramu (Slika 3.1) je kreiranje PBF-a. Po uzoru na algoritam 2, potrebno je kao ulaz proslijediti skup područja interesa ($S = \Delta_1, \Delta_2, ..., \Delta_n$), skup funkcija raspršivanja H i broj područja interesa n.

U početku su sve vrijednosti u vektoru postavljene na nulu. Zatim, počevši od područja koji ima najmanji prioritet, zapisuju se oznake područja unutar vektora. Svako područje sastoji se od skupa elementa. Elementi područja za kojeg se zapisuju oznake područja, predaju se kao argument funkciji raspršivanja. Dobiveni hash se pretvori u redni broj ćelije u PBF vektoru kako je objašnjeno u drugom koraku dijagrama (19). U ćeliju s dobivenim rednim brojem se zapiše oznaka polja interesa. Postupak se ponavlja za sva polja unutar svih područja.

Kao i u funkciji tablice raspršivanja (eng. hash table), može doći do kolizije. Ako se elementi nalaze u istom području vrijednost ostaje ista, što ne predstavlja problem. Međutim, za neke elemente koji spadaju u druga područja, može se dogoditi da promjene vrijednosti od prethodnih područja.

Vjerojatnost da elementi koji se nalaze unutar područja većeg prioriteta u potpunosti

prebrišu oznake područja manjih prioriteta, može se smanjiti odabirom optimalne veličine PBF-a te optimalnim brojem funkcija raspršivanja, baš kao i kod uobičajenog Bloomovog filtera.

Algoritam 2: Kreiranje Prostornog Bloomovog Filtra

Ulaz: $\Delta_1, \Delta_2, ..., \Delta_n, H, n$
Izlaz: $b^{\#}$
for $i \leftarrow 1$ **to** n **do**
  foreach $\delta \in \Delta_i$ **do**
    foreach $h \in H$ **do**
      $b^{\#}[h(\delta)] \leftarrow i$;
    end
  end
end
return $b^{\#}$;

U četvrtom koraku dijagrama (Slika 3.1), odvija se provjera je li točka u prostoru unutar određenog polja interesa. Provjera pripadnosti elementa skupu se osniva na algoritmu 3. Algoritmu je potrebno predati kao argumente PBF vektor, skup funkcija raspršivanja, element koji se treba provjeriti, i broj područja interesa.

Za danu testnu točku odredi se element u mreži ε. Element mora sadržavati testnu točku. Sve funkcije raspršivanja koje se nalaze u skupu H primaju element kao argument. Hashevi iz funkcija raspršivanja se koriste za nalaženje vrijednosti unutar PBF vektora.

Ako je i jedna vrijednost nula, tada element ne pripada niti jednom području interesa. U suprotnom, algoritam pronalazi najmanju vrijednost koja predstavlja područje interesa.

Kako bi svi lokacijski podaci bili usklađeni, korišten je World Geodetic System (WGS)84 referentni sustav unutar cijele aplikacije. Točnije, korišten je European Petroleum Survey Group (dEPSG)4326 2d referentni koordinatni sustav.

Algoritam 3: Provjera elementa unutar Prostornog Bloomovog Filtra

Ulaz: $b^{\#}, H, \delta_u, n$
Izlaz: i
$i = n$;
foreach $h \in H$ **do**
  if $b^{\#}[h(\delta)] = 0$ **then**
    return 0;
  else
    if $b^{\#}[h(\delta)] < i$ **then**
      $i \leftarrow b^{\#}[h(\delta)]$;
    end
  end
end
return i;

3.3 Opis razvijene programske izvedbe PBF-a

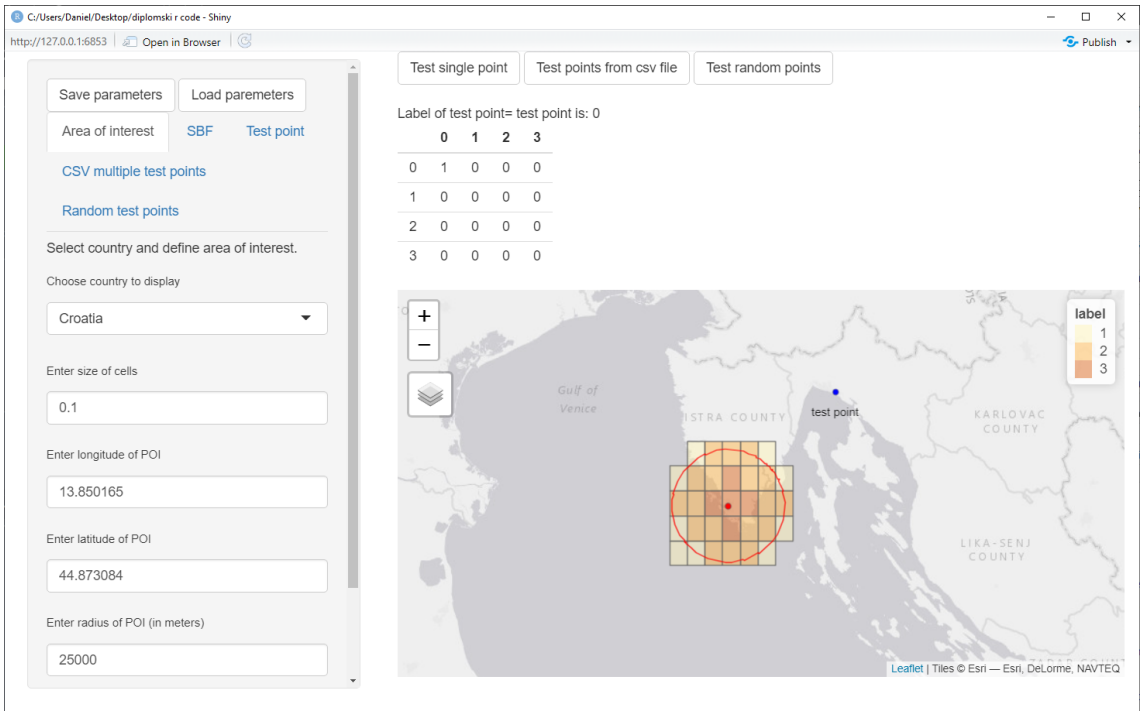
Programski kod podijeljen je na nekoliko programskih blokova. Svaki programski blok predstavlja jedan specifični dio aplikacije. Tako se, na primjer, osnovne funkcije vezane uz PBF nalaze u svojem programskom bloku, dok se funkcije odgovorne za kreiranje područja interesa nalaze u svojem.

U ovom odjeljku, bit će objašnjeno što svaki od tih programskih blokova sadrži, te kako se određenoj funkcionalnosti može pristupiti putem grafičkog sučelja.

Grafičko sučelje izvedeno je zbog lakšeg prikaza evaluacije i testiranja. Knjižnica `shiny` olakšava izradu interaktivnih web aplikacija, te je korištena za definiranje grafičkog sučelja. Izvedba je podijeljena na tri programska bloka `app.r`, `ui.r` i `uilFuncitions.r`.

Unutar `ui.r` programskog bloka, nalazi se kod zaslužan za definiranje elemenata grafičkog sučelja, kao i njegov izgled.

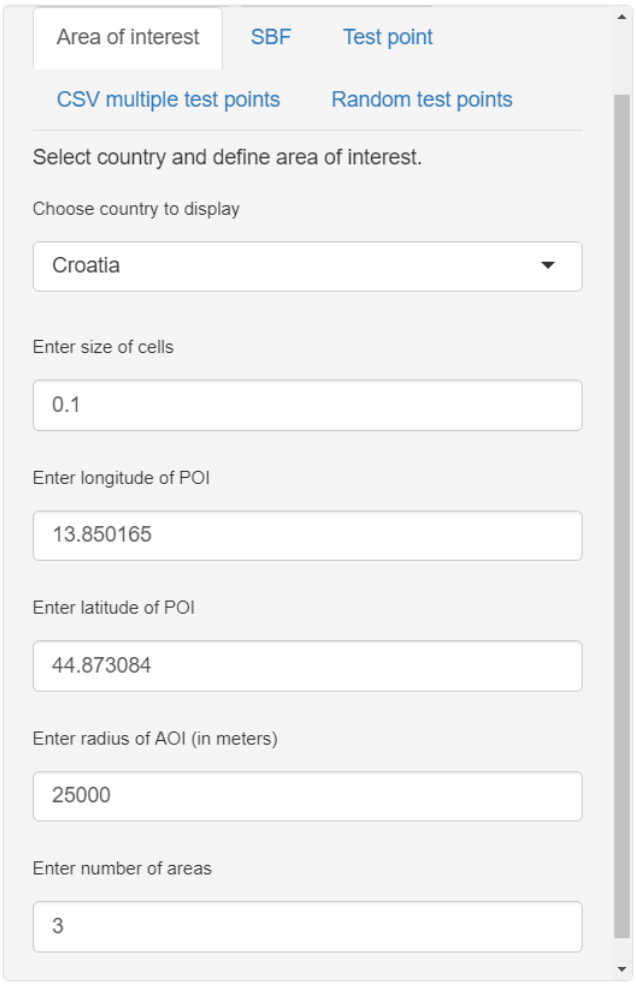
Programski blok `app.r` sadrži kod koji se izvršava na serveru i poziva funkcije koje se nalaze u `uilFuncitions.r` programskom bloku. Pokretanjem ovog programskog bloka, pokreće se web aplikacija. Izgled aplikacije je prikazan na slici 3.3.



Slika 3.3 Prozor aplikacije

Unutar `app.r` bloka učitavaju se i početne postavke koje su spremljene u `parameters.csv` datoteci. Blok `uilFuncitions.r` ima funkcije koje su bitne za izvođenje aplikacije. Programski blok specijaliziran za izračun područja interesa sadrži funkcije za kreiranje područja, Manhattan-ovu udaljenost i za odjeljivanje područja. Svi parametri za ovaj blok mogu se postaviti unutar kartice `Area of interest` (Slika 3.4).

Odabir mreže u aplikaciji određuje se pomoću padajućeg izbornika `Choose country to display` (Slika 3.5). Za popis zemalja i njihovih karakteristika koristi se funkcija `ne_countries` iz knjižnice `rnaturalearth`. Kod za popis zemalja nalazi se unutar programskog bloka `app.r`. Od poligona odabrane zemlje uzmu se najsjevernija, najjužnija, najzapadnija i najistočnija točka na mapi. One određuju veličinu mreže. Točka interesa i testne točke ne smiju izaći iz okvira mreže, jer tada ne postoji element za kojeg možemo ispitati nalazi li se u PBF filteru. Zato aplikacija upozori korisnika ako korisnik pokuša provjeriti neku točku koja se ne nalazi u mreži. Veličina elementa može se podesiti pomoću polja `Enter size of cells` unutar kojeg se upisuje broj koji predstavlja stupnjeve. Točka in-



Slika 3.4 Kartica za podešavanje parametara područja interesa

teresa se definira pomoću zemljopisne širine i dužine izražene u stupnjevima. Zemljopisna širina se uređuje unutar polja `Enter latitude of POI`, a zemljopisna dužina unutar polja `Enter longitude of POI`. Radijus kružnice se definira u polju `Enter radius of AOI`. Na posljetku, poljem `Enter number of areas` se definira broj područja interesa.

Za kreiranje područja koristi se funkcija `coverageAOI` koja za argumente prima mrežu, točku interesa, radijus, veličinu elemenata u mreži, te broj područja interesa. Korištenjem funkcije `st_buffer`, izgrađuje se krug koji će se koristiti za izdvajanje elemenata koje on pokriva.

Slika 3.5 Padajući izbornik kojim se odabire država nad kojom će se izgraditi mreža elemenata.

Funkcija `raster` iz knjižnice `fasterize` transformira mrežu iz vektorskog u rasterski oblik. Unutar rasterskog oblika moguće je dobiti vrijednosti X i Y koje označavaju broj retka i broj stupca unutar mreže. Broj redaka izračunava funkcija `rowFromCell`, dok broj stupaca izračunava funkcija `colFromCell`. Obije funkcije dolaze iz knjižnice `raster`. Dobiveni brojevi redaka i stupaca, kasnije će biti potrebni za izračun Manhattan-ove udaljenosti.

Funkcija `ifelse` zapisuje koji elementi unutar mreže se nalaze unutar kruga. Ako je prvi argument unutar `ifelse` funkcije točan, zapisuje se `Yes` u varijablu `grid$intersects_buffer`. U suprotnom, zapisuje se `No`. Za prvi argument, koristi se funkcija `st_intersects` iz knjižnice `sf` kako bi se saznalo koji se elementi preklapaju s kružnicom.

Istim funkcijama se zapisuje koji element sadrži središte kružnice. Jedina razlika je što se u tom slučaju umjesto kružnice predaje objekt točke u prostoru.

Iz mreže se filtriraju svi elementi koji sadrže krug, te se spremaju u varijablu `S` koja predstavlja skup svih područja interesa. Varijabla `S` predaje se funkciji `ManhattanDistanceAOI`, čiji će rezultat s brojem područja interesa biti proslijeđen funkciji `partitionAOI`. Rezultat funkcije `partitionAOI` bit će ujedno i rezultat funkcije `coverageAOI`.

Manhattan-ova udaljenost računa se u funkciji `ManhattanDistanceAOI`. Kao početne koordinate koriste se koordinate ćelije koje sadrže središte kružnice. Za sve ćelije, vrijednost `manDist` se postavlja na NA. Unutar for petlje, izračunava se Manhattan-ova udaljenost od

ćelije koja sadrži središte kružnice do svake ćelije koja se nalazi unutar `S` varijable. Funkcija vraća promijenjenu varijablu `S` kao rezultat.

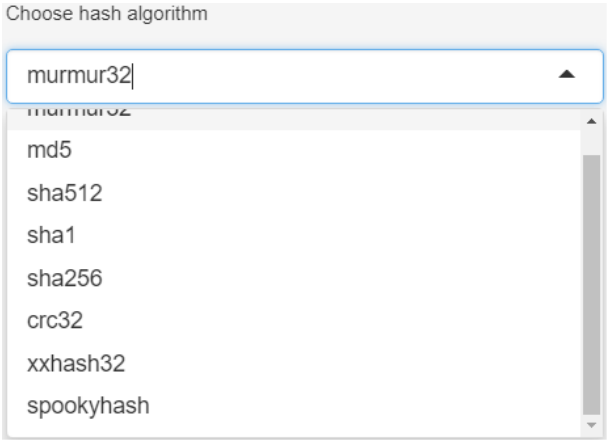
Funkcija `partitionAOI` je rađena po uzoru na dio algoritma 1 od linije 8 do linije 19.

Skripta za sažimanje sadržaja pomoću funkcije raspršivanja sadrži funkcije za čitanje soli, generiranje soli, te kreiranje skupa funkcija raspršivanja.

Funkcija `readSalts` prima za argument broj funkcija raspršivanja. Ako funkcija pronađe datoteku `salts.txt` i ako ima jednak broj zapisanih soli u datoteci kao i potrebnih funkcija raspršivanja, vrati ih kao rezultat. U suprotnom, stvori potrebni broj soli pozivom na funkciju `generateSalts`, te ih zapiše u datoteku i vrati kao rezultat.

Sol se kreira unutar funkcije `generateSalts`. Koristi se knjižnica `stringi` za kreiranje soli. Stvorena sol je duljine pet nasumično odabranih znakova. Znakovi mogu biti velika i mala slova engleske abecede i brojevi. Znakovi su također nasumično raspoređeni prilikom poziva funkcije `stringi`. Broj stvorenih soli jednak je broju funkcija raspršivanja unutar skupa H.

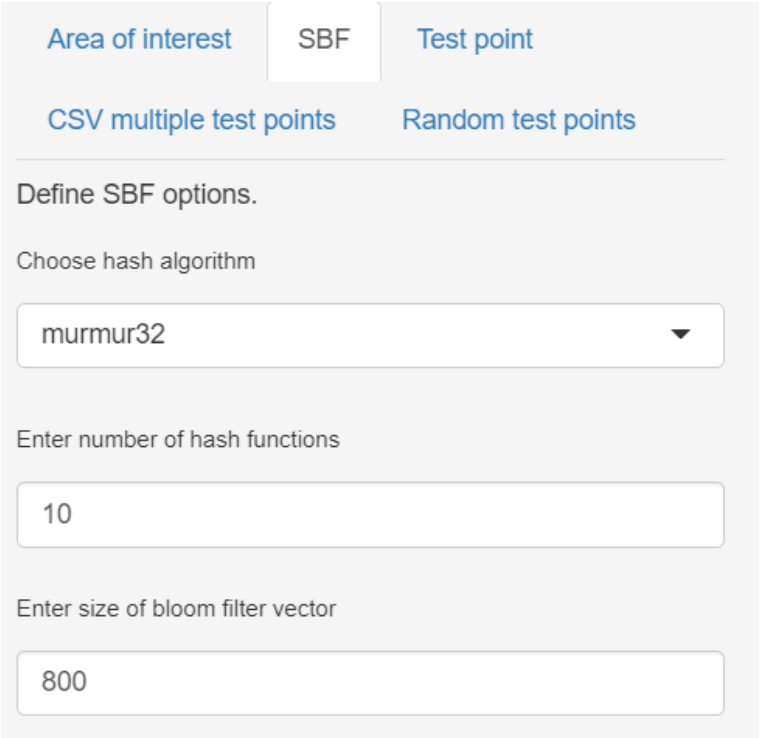
Funkcija `generateHashSetSalts` uzima soli koje su već stvorene te pomoću njih i algoritma raspršivanja (eng. hash algorithm), kreira listu funkcija raspršivanja koje će se spremiti u varijablu H. Vrstu algoritma raspršivanja može se odabrati u padajućem izborniku `Choose hash algorithm` (Slika 3.6) koji se nalazi pod karticom SBF (Slika 3.7).



Slika 3.6 Padajući izbornik kojim se odabire algoritam raspršivanja.

Kod skripte za PBF posebno su bitne funkcija za izradu samog filtra i funkcija za provjeru elementa u filtru.

Funkcija za izradu samog filtra napravljena je po uzoru na algoritam 2. Kartica za uređivanje PBF parametara prikazana je na slici 3.7. Parametri koji se mogu urediti su izbor algoritma raspršivanja, broj funkcija raspršivanja te veličina PBF vektora.



Slika 3.7 Kartica u kojoj se nalaze parametri PBF-a.

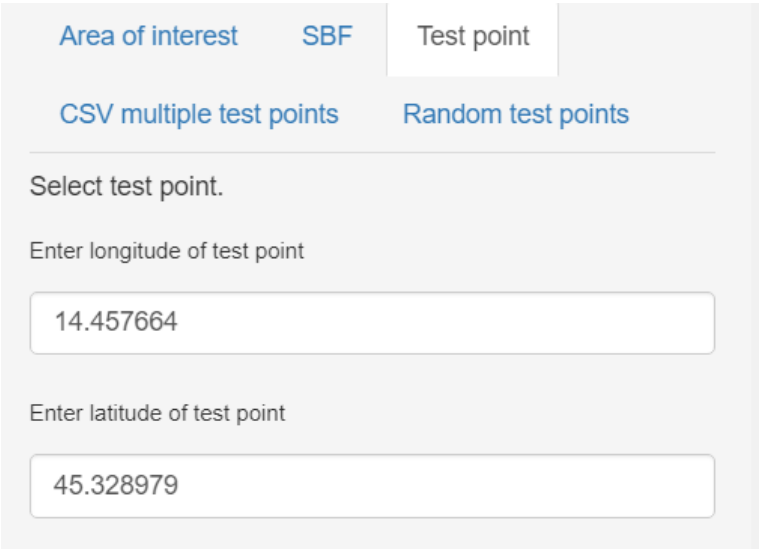
Prvo se sve oznake područja interesa uzmu iz skupa S i sortiraju. U tu svrhu, koriste se funkcije `unique` i `sort` koje su dio osnovne knjižnice. U sljedećem koraku, vrijednosti polja se postavljaju na nulu unutar PBF vektora.

Nakon inicijalizacije vektora, slijedi dio koji je istovjetan algoritmu 2 uz kompresiju hash-a. Ako se hash samo prevede kao cjelobrojni broj, on može biti veći od veličine filtera. Korištenjem metode cjelobrojnog dijeljenja, sprječava se pokušaj unosa u polje koje ne postoji [3].

U izvedbi, povećava se dobiveni broj za 1 jer u R programskom jeziku polja počinju s

jedinicom. Funkcija za provjeru elementa u filtru je istovjetna algoritmu 3 te koristi funkcije raspršivanja za sažimanje sadržaja pomoću metode dijeljenja.

Aplikacija ima tri načina kako se mogu dodijeliti točke za provjeru. Prvim načinom, može se provjeriti samo jedna točka, namještanjem parametara zemljopisne dužine i širine unutar kartice `Test point` (Slika 3.8).

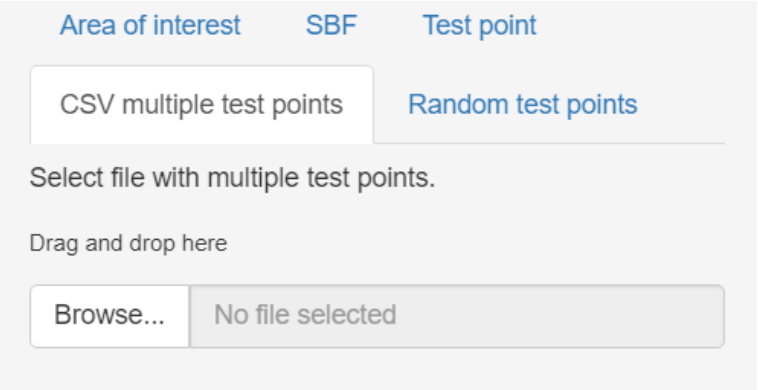


Slika 3.8 Kartica za definiranje položaja testne točke.

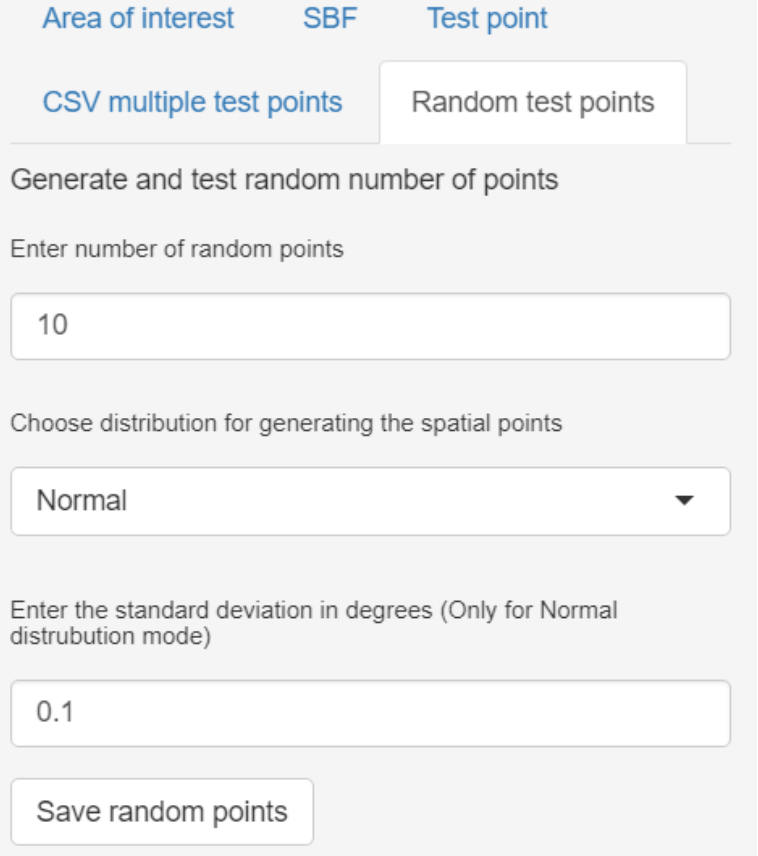
Drugim načinom, može se provjeriti više točaka koje se nalaze u comma-separated values (CSV) datoteci. Datoteku se može definirati u polju `Select file with multiple test points` koje se nalazi unutar kartice `CSV multiple test points` (Slika 3.9). Datoteku je moguće pronaći klikom na tipku `Browse` ili povući mišem i ispustiti u polje.

CSV datoteka koju se želi koristiti za provjeru, mora biti strukturirana na idući način. Zaglavlje mora imati `lat_test`, `lon_test` i `name` nazive stupaca. `Lat_test` stupac treba sadržavati zemljopisne širine, `lon_test` stupac zemljopisne dužine, a stupac `name` imena točaka.

Treći način za testiranje točaka je korištenje funkcija za generiranje pseudo slučajnih brojeva. Parametri za generiranje točaka mogu se podesiti unutar kartice `Random test points` (Slika 3.10).



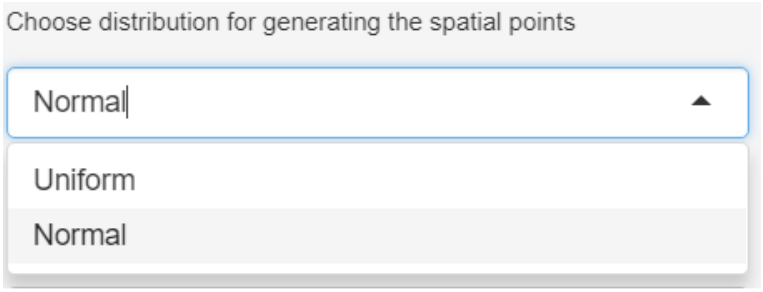
Slika 3.9 Kartica za definiranje položaja testne točke.



Slika 3.10 Kartica za definiranje položaja slučajnih testnih točaka.

Unutar spomenute kartice, može se podesiti broj slučajnih točaka poljem `Enter number of random points`, odabrati razdioba za generiranje slučajnih točaka s padajućim izbornikom `Choose distribution for generating the spatial points`, namjestiti veličina standardne devijacije izražene u stupnjevima s poljem `Enter the standard deviation in degrees`, te spremiti slučajne točke za ponovno korištenje tipkom `Save random points.`

Izvedene su dvije mogućnosti kreiranja testnih točka. Prva se ravna po uniformnoj razdiobi, a druga po normalnoj razdiobi. (Slika 3.11) Kada se koristi uniformna razdioba,

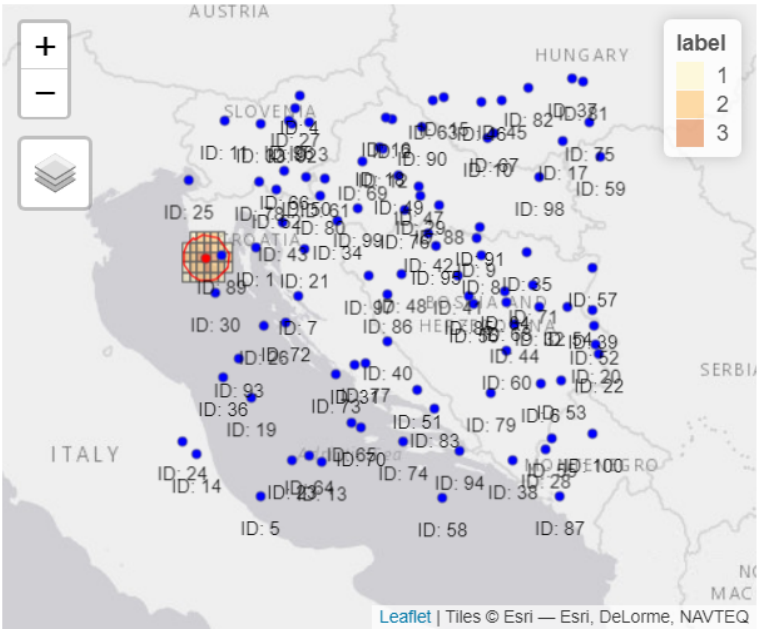


Slika 3.11 Padajući izbornik unutar kojeg se može odabrati koja razdioba će se koristiti za generiranje slučajnih prostornih točaka.

aplikacija generira slučajne točke po cijeloj površini mreže s potpuno ravnomjernom vjerojatnošću da se neka točka pojavi u bilo kojem dijelu mreže. Za takvo raspršivanje točaka zaslužna je funkcija `st_sample` iz knjižnice `sf`.

Prilikom korištenja kreiranja slučajnih točaka odabirom normalne razdiobe potrebna je standardna devijacija i srednja vrijednost. Standardnom devijacijom se određuje koliko daleko će točke biti udaljene od srednje vrijednosti. Za srednju vrijednost koristi se točka interesa. Kako bi točke bile raspršene po mapi koja je prikazana u dvodimenzionalnom prostoru, potrebne su dvije funkcije za normalnu razdiobu, jedna za zemljopisnu širinu i druga za zemljopisnu dužinu.

Generirane slučajne točke mogu se spremiti u CSV datoteku. Ta datoteka će imati prefiks `saved_random_points_"` u nazivu, te datum i vrijeme nastanka. Zaglavlje će biti formatirano s `lat_test`, `lon_test` i `name` nazivima stupaca.
U stupac `name` spremit će se identifikacijski broj točaka. Takav format zaglavlja podržava

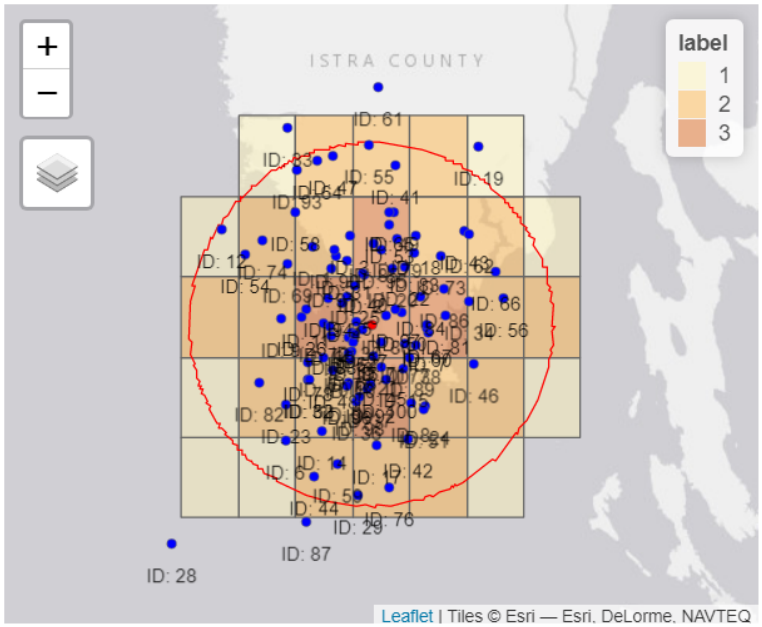


Slika 3.12 Prikaz slučajnih prostornih točaka raspršenih uniformnom razdiobom. Korišteni parametri su 100 točaka i uniformna razdioba.

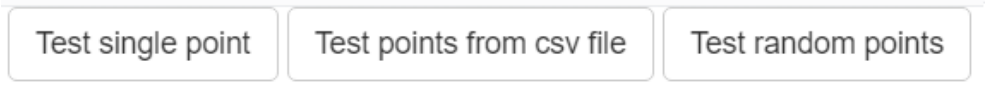
opcija testiranja točaka iz CSV datoteke u aplikaciji, pa se te iste točke mogu testirati nekom drugom prilikom i drugim parametrima.

Pritiskom na jednu od tipki prikazanih na slici 3.14 pokreće se testiranje. Odabirom prve tipke pokreće se testiranje točke koja je definirana u kartici `Test point`. Druga tipka pokreće testiranje točaka koji se nalaze u CSV datoteci. Odabir datoteke je definiran u kartici `CSV multiple test points`. Treća tipka pokreće testiranje slučajno odabranih točaka čiji su parametri definirani u kartici `Random test points`. Neovisno o izboru testa, rezultati će biti prikazani u obliku matrice zabune i na interaktivnoj mapi.

Matrica zabune i njezine osnovne mjere će također biti zapisane kao CSV datoteke u direktoriju `tmp`. Datoteka `Confusion_matrix.csv` ima pohranjenu matricu zabune, a datoteka `F1_score.csv` ima pohranjene osnovne mjere. `Confusion_matrix.csv` je formatirana tako da zaglavlje predstavlja identifikacijski broj područja interesa kojeg je PBF predvidio, dok prvi stupac predstavlja identifikacijski broj područja u stvarnosti. Ako su parametri zadovoljavajući može ih se spremiti kako bi bili dostupni prilikom sljedećeg pokretanja. Također,



Slika 3.13 Prikaz slučajnih prostornih točaka raspršenih normalnom razdiobom. Korišteni parametri su 100 točaka, normalna razdioba i standardna devijacija od 0.1 stupnja.



Slika 3.14 Tipke za pokretanje testiranja.

moguće je i odmah učitati parametre iz datoteke `parameters.csv`. Tipke za spremanje i učitavanje parametara se nalaze iznad kartica (Slika 3.15). Datoteka `parameters.csv` kao zaglavlje sadrži imena svih parametara.

Slika 3.15 Tipke za spremanje i učitavanje parametara.

Poglavlje 4

Vrednovanje izvedbe PBF-a u programskom okruženju R

Za vrednovanje modela koristi se matrica zabune. Izabrana je jer daje odgovore na pitanje koliko je dobar neki klasifikator i jer može dati procjenu pogreške.

		Prognozirane vrijednosti	
		P	N
Stvarne vrijednosti	P	IP	LN
	N	LP	IN

Slika 4.1 Matrica zabune s dvije klase. P označava pozitivnu klasu, N označava negativnu klasu, IP označava istinito pozitivnu kategoriju, LP označava lažno pozitivnu kategoriju, IN označava istinito negativnu kategoriju, a LN označava lažno negativnu kategoriju

U matrici zabune (Slika 4.1), stupci predstavljaju vrijednosti koje je neki klasifikator predvidio, dok redovi predstavljaju stvarne vrijednosti [4]. Klasifikator je algoritam koji pokušava pogoditi klasu nekog ulaznog elementa. Za lakše razumijevanje teorijskog dijela matrice zabune PBF je istovjetan klasifikatoru, dok su područja interesa istovjetna klasama.

Matrica zabune se sastoji od četiri kategorije, istinito pozitivno (IP), lažno pozitivno (LP), istinito negativno (IN) i lažno negativno (LN) [5]. Istinito pozitivna kategorija predstavlja rezultate koji su u stvarnosti pozitivni i za koje je klasifikator točno predvidio da su pozitivni. Lažno pozitivna kategorija predstavlja rezultate koji su u stvarnosti negativni, ali klasifikator je pogrešno predvidio da su pozitivni. Istinito negativna kategorija predstavlja rezultate koji su u stvarnosti negativni i za koje je klasifikator točno predvidio da su negativni. Lažno negativna kategorija predstavlja rezultate koji su u stvarnosti pozitivni, ali klasifikator je pogrešno predvidio da su negativni.

Neke od osnovnih mjera koje možemo dobiti od rezultata matrice zabune su točnost, preciznost, odaziv i F1 mjera. Točnost je omjer točno klasificiranih prognoza klasifikatora i zbroja svih prognoza (Matematički izraz 4.1) [4].

$$T = \frac{IP + IN}{IP + IN + LP + LN} \tag{4.1}$$

Preciznost je omjer točno pozitivnih prognoza i zbroja svih pozitivnih prognoza, istinitih i lažnih (Matematički izraz 4.2) [4].

$$P = \frac{IP}{IP + LP} \tag{4.2}$$

Odaziv je omjer točno pozitivnih prognoza i zbroja svih stvarnih pozitivnih primjera, tj. zbroja istinito pozitivnih i lažno negativnih primjera [4].

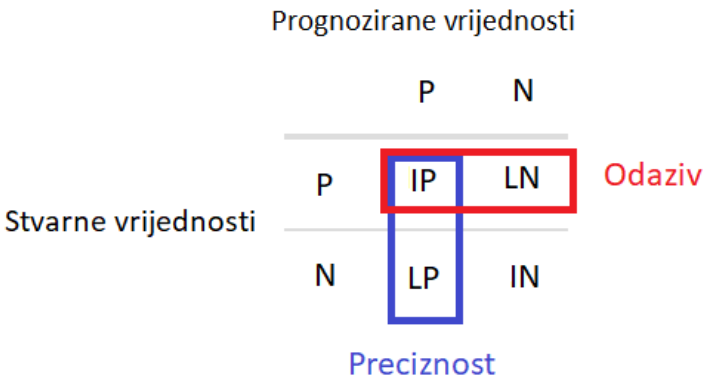
$$O = \frac{IP}{IP + LN} \tag{4.3}$$

Polja koja se koriste za preciznost i odaziv prikazana su na slici 4.2 [4].

F1 je harmonijska sredina između preciznosti i odaziva [4].

$$F1 = \frac{2 * P * O}{P + O} \tag{4.4}$$

Kako u PBF-u može biti više područja interesa, tj. klasa, treba razmotriti matricu zabune s više klasa. (Slika 4.3)



Slika 4.2 Matrica zabune s prikazanim poljima koji se koriste za preciznost i odaziv.

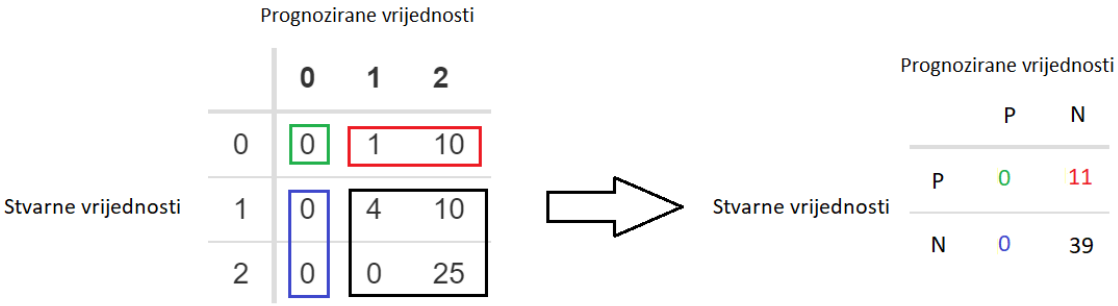
Prognozirane vrijednosti

Stvarne vrijednosti	**0**	**1**	**2**
0	0	1	10
1	0	4	10
2	0	0	25

Slika 4.3 Matrica zabune s više klasa. Brojevi redova i stupaca predstavljaju identifikacijske brojeve područja. U matrici s više klasa, kao i kod matrice zabune s dvije klase, stupci predstavljaju vrijednosti koje je predvidio klasifikator, dok redovi predstavljaju stvarne vrijednosti.

Kako bi se mogle izračunati osnovne mjere za matricu zabune s više klasa, za svaku klasu je potrebno izvesti matricu zabune s dvije klase. Jedna klasa označava sve slučajeve koji su jednaki toj klasi dok druga označava sve slučajeve koji ne pripadaju toj klasi. U procesu izvoda matrice s dvije klase iz matrice s više klasa, prvo se istinito pozitivna vrijednost za

pojedinu klasu stavi u prvu ćeliju. Zatim se zbroje sve vrijednosti osim istinito pozitivne vrijednosti koje su u retku u kojem se nalazi istinito pozitivna vrijednost. Zbroj predstavlja lažnu pozitivnu vrijednost. Sve vrijednosti osim istinito pozitivne vrijednosti koje su u stupcu u kojem se nalazi istinito pozitivna vrijednost se zbroje. Taj zbroj predstavlja lažnu negativnu vrijednost. Preostale ćelije se zbroje i predstavljaju istinito negativnu vrijednost. Opisani proces prikazan je i slikom 4.4.



Slika 4.4 Primjer kreiranja matrice zabune s dvije klase iz matrice zabune s više klasa. Prikazan primjer je za klasu 0.

Od dobivenih matrica zabune s dvije klase, za svaku klasu mogu se izračunati osnovne mjere. No problem je što osnovne mjere pokazuju koliko je dobar klasifikator za pojedinu klasu, a ne koliko je on kvalitetan za sve klase u prosjeku. Kvalitetu klasifikatora za sve klase može se saznati izračunavanjem prosjeka F1 vrijednosti svih klasa. Prosjek F1 vrijednosti svih klasa naziva se $F1^{makro}$, te je dan formulom 4.5. Unutar formule, n predstavlja broj klasa, a $F1_i$ predstavlja $F1$ mjeru i-te klase [6].

$$F1^{makro} = \frac{\sum_{i=1}^{n} F1_i}{n} \tag{4.5}$$

4.1 Provjera ispravnosti izvedbe

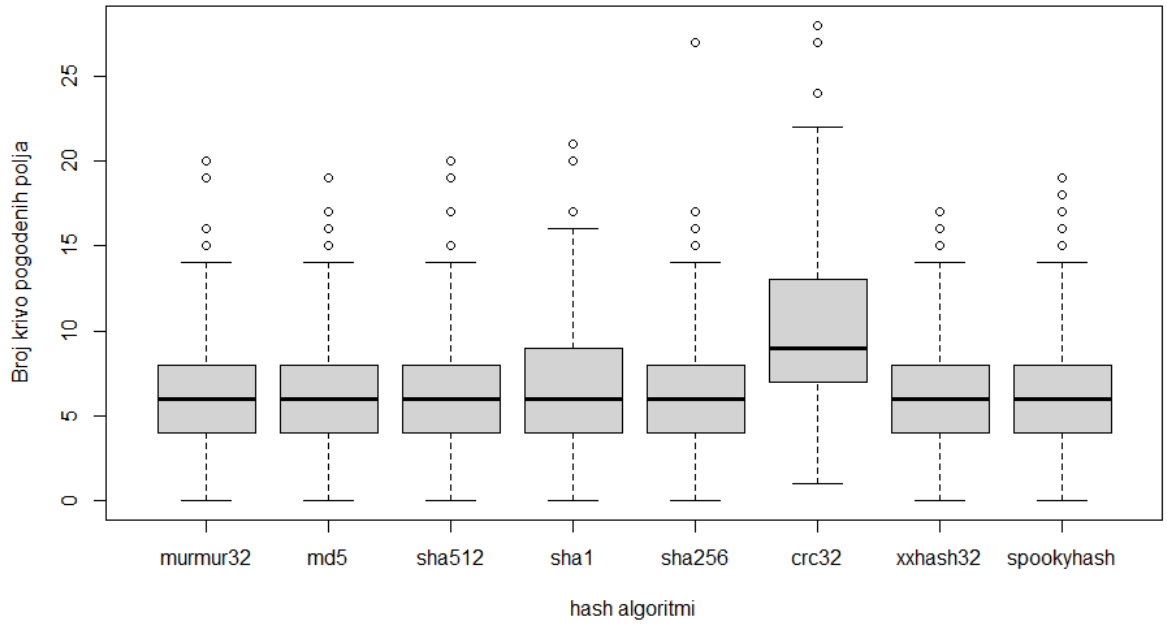
Za sve provjere korištena su ista područja interesa. Točka interesa se nalazi u Hrvatskoj, točnije na poziciji pulskog Amfiteatra koja se nalazi na 44.873084 stupnju zemljopisne širine i 13.850165 stupnju zemljopisne dužine. Udaljenost korištena za polumjer kruga unutar kojeg će se nalaziti 3 područja interesa je 25 kilometra. Veličina ćelije je postavljena na 0.1 stupanj.

Prvi test bio je provjera algoritma raspršivanja. Cilj testa je pokušati naći algoritam koji će u prosjeku dati najveću točnost glspbf-a. Algoritmi korišteni u testu su `murmur32`, `md5`, `sha512`, `sha1`, `sha256`, `crc32`, `xxhash32`, `spookyhash`. Za svaki algoritam je petsto puta kreiran skup funkcija raspršivanja i PBF. Broj funkcija raspršivanja u skupu je postavljen na devet, a veličina filtra na 610 polja u vektoru. Za provjeru algoritma raspršivanja nisu se koristile točke u prostoru, već samo polja u mreži.

Ovi parametri PBF-a za dana područja interesa daju medijan od 6 krivih prognoza za 2478 polja u mreži, što odgovara greški od 0.0024%. Naravno dobiveni rezultat ne znači puno jer ne pokazuje koja su polja i u kojem području interesa bila krivo označena. No za provjeru algoritma raspršivanja bilo je bitno pokušati naći algoritam raspršivanja koji bi i u neoptimalnim parametrima uspio dobiti najbolje rezultate, ne nužno i u potpunosti točne rezultate.

Kako je vidljivo na grafu 4.5, svi algoritmi su gotovo identični osim `sha1` čija je gornja kvartila nešto veća od ostalih algoritama i `crc32` koji ima za medijan 9 pogrešaka i čije su sve kvartile lošije od ostalih algoritama. Stoga se da zaključiti da odabir algoritma nije toliko bitan, već su puno bitniji ostali parametri u PBF-u, što će pokazati testovi koji slijede. U sljedećim testiranjima korišten je algoritam `murmur32` zbog brzine izvođenja [7] i dokazanog dobrog raspršivanja elemenata unutar PBF-a.

Za provjeru ispravnosti korištena su dva skupa točaka. Oba skupa sačinjena su od sto slučajno odabranih točaka generiranih i spremljenih pomoću aplikacije na način opisan u Poglavlju 3.3 "Izvedba PBF-a u programskom okruženju R". Prvi skup sačinjavaju točke generirane uniformnom razdiobom. Drugi skup sačinjavaju točke generirane normalnom razdiobom. Oba skupa testirana su s četiri različite kombinacije broja funkcija raspršivanja i broja polja u PBF vektoru. Prva kombinacija koristi se za početnu točku i izgled PBF vektora. U njoj su veličina vektora i broj funkcija raspršivanja dovoljno veliki da se ne



Slika 4.5 Testirani algoritmi korištenjem identičnih parametara za kreiranje polja i kreiranje PBF filtra.

dogodi niti jedno krivo prognoziranje skupa za testne točke. Druga kombinacija predstavlja najgori slučaj, u kojem je veličina vektora premala za dobro prognoziranje, dok je broj funkcija raspršivanja dovoljno velik. Iako je njihov broj dovoljno velik, prostor na koje one definiraju raspored oznaka u PBF vektoru je premali što na posljetku uzrokuje previše kolizija i gubitak informacija. U trećoj kombinaciji veličina vektora je dovoljno velika, ali broj funkcija raspršivanja je premali za prognoziranje bez greške. Četvrta kombinacija služi kao uvid kada je veličina vektora i količina funkcije raspršivanja premala za dobro prognoziranje.

Svaki test ima svoju matricu zabune i tablicu koja prikazuje točnost, odaziv, preciznost, F1 mikro i makro mjeru te pogreške 1. i 2. tipa za svaku klasu oznake. Svaki test ima i svoj histogram koji prikazuje količinu određenog polja unutar PBF vektora. Histogramom se sažima i prikazuje razdioba jedne slučajne varijable [8]. Uz histogram priložen je i graf koji prikazuje PBF vektor u obliku matrice. Vektor je prebačen u oblik matrice zbog lakšeg

dobivanja dojma raspršenosti i gustoće oznaka u samom vektoru. Transformacija je napravljena na idući način. Nakon svakog desetog elementa sljedećih deset elemenata se stavi u sljedeći stupac. Redni broj elementa u PBF vektoru može se dobiti pomoću funkcije 4.6, gdje i predstavlja redni broj elementa, x broj stupca, a y broj retka.

$$i = (x - 1) * 10 + y \tag{4.6}$$

4.1.1 Provjera PBF-a za vektor veličine 800 polja i 10 funkcija raspršivanja nad testnim točkama slučajno odabranih uniformnom razdiobom

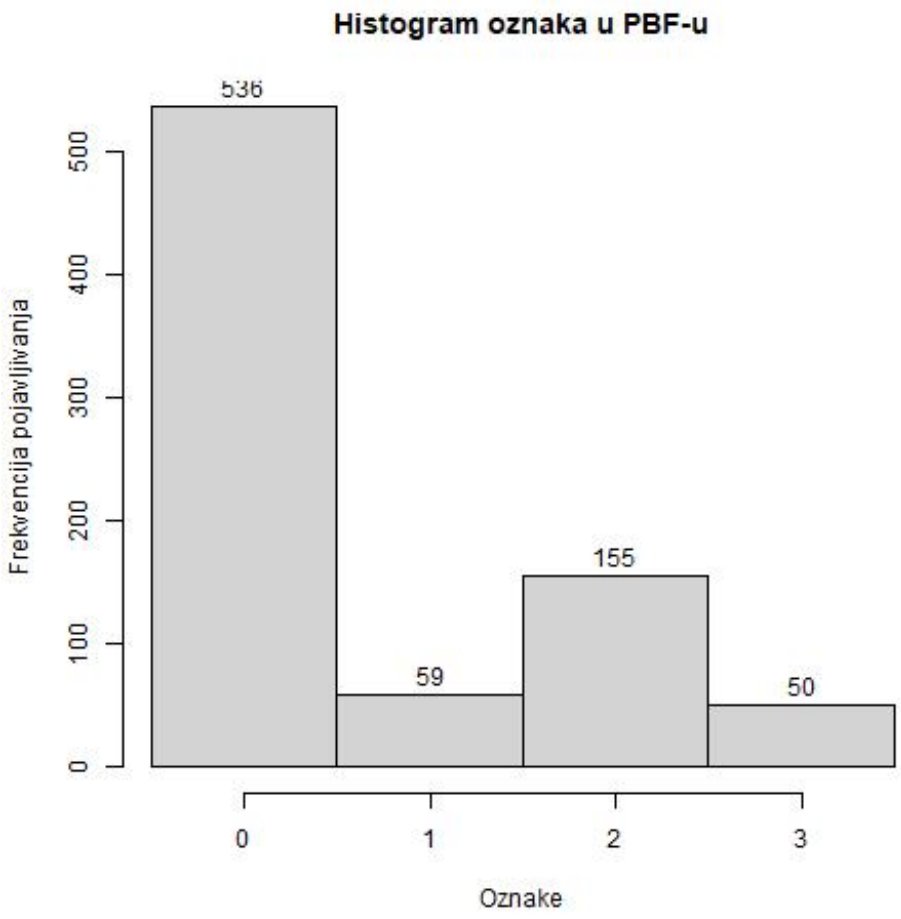
Histogram 4.6 pokazuje kako je u PBF vektoru najviše oznaka s brojem nula. Sljedeća oznaka po količini pojavljivanja u PBF filtru je dva, treća oznaka jedan i naposljetku oznaka tri. U mreži se sveukupno nalazi 2478 polja od kojih 2446 imaju oznaku nula, devet polja ima oznaku jedan, osamnaest ima oznaku dva i pet polja ima oznaku 3. Uspoređujući količinu oznaka u PBF-u s količinom polja određene oznake, može se zaključiti da postoji povezanost između te dvije količine. Ako je neke oznake više u mreži, može se očekivati da bi trebala biti zastupljenija u filtru.

Na slici 4.7 može se vidjeti da su oznake poprilično raspršene unutar vektora, te da nisu grupirane. Treba imati na umu da je drugi stupac nastavak prvog. Zato lijeve i desne ćelije zapravo nisu susjedne promatranoj ćeliji.

Matrica zabune 4.1 za dužinu vektora 800 i količinu funkcija raspršivanja 10 prikazuje da nema ni jedne greške prilikom prognoziranja unutar koje oznake se točka nalazi. U

Tablica 4.1 Matrica zabune za uniformno generirane točke i s PBF parametrima m = 800 i k = 10

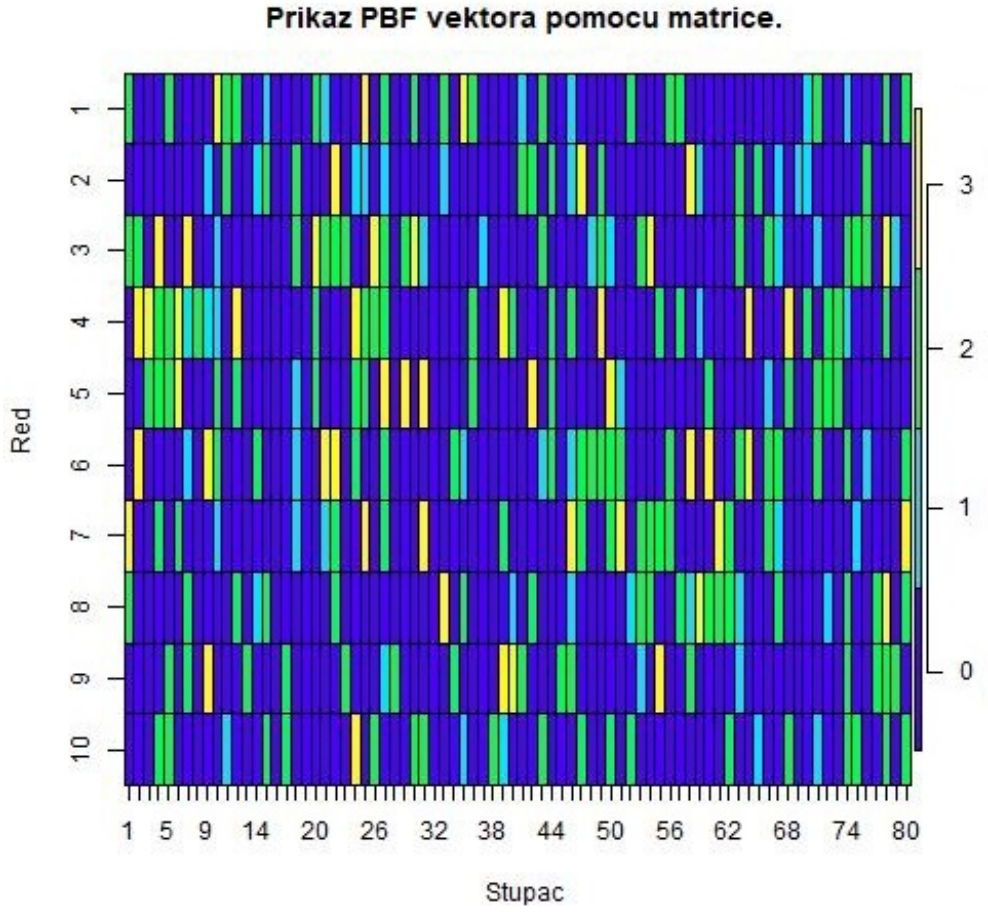
oznaka \ prognoza	0	1	2	3
0	97	0	0	0
1	0	1	0	0
2	0	0	2	0
3	0	0	0	0



Slika 4.6 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 800 i k = 10.

skupu generiranim uniformnom razdiobom većina točaka se nalazi unutar polja s oznakom nula, jedna u polju s oznakom jedan, dvije u poljima s oznakom dva, dok se niti jedna točka se ne nalazi u poljima s oznakom tri.

Vrijednosti unutar tablice 4.2 još jednom potvrđuju da nije bilo niti jedne pogreške, te da je svaka klasa bila točno prognozirana.



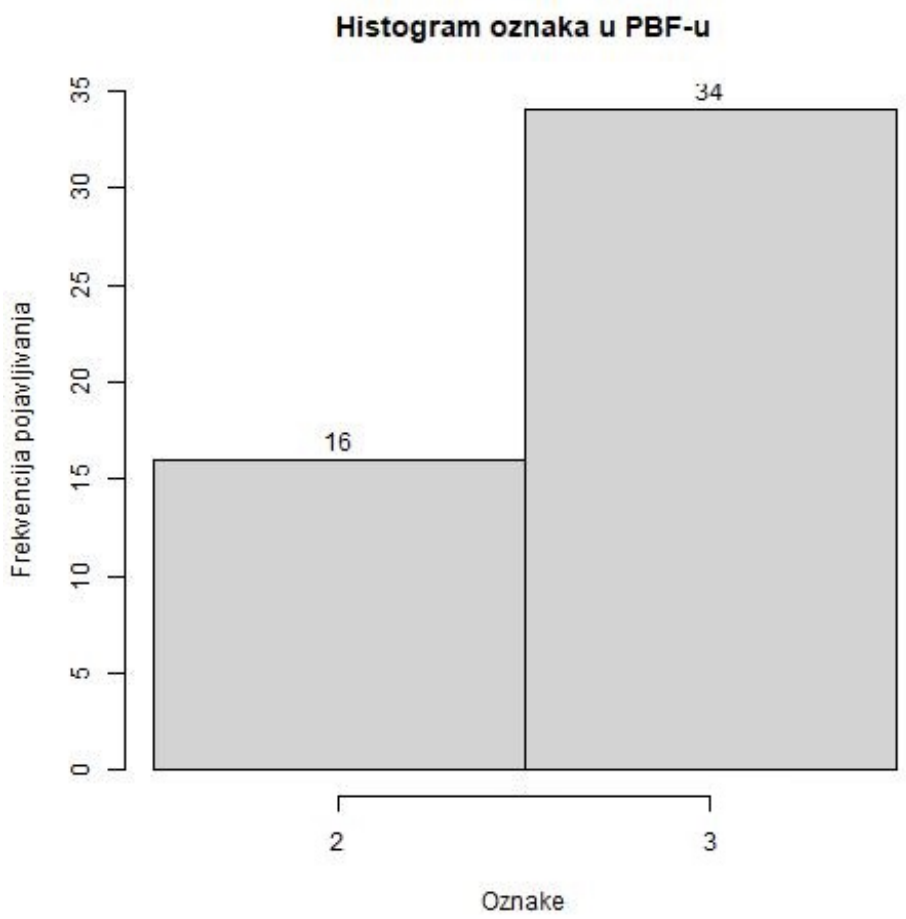
Slika 4.7 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 800 i k = 10. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

Tablica 4.2 F1 vrijednosti za uniformno generirane točke i s PBF parametrima m = 800 i k = 10. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

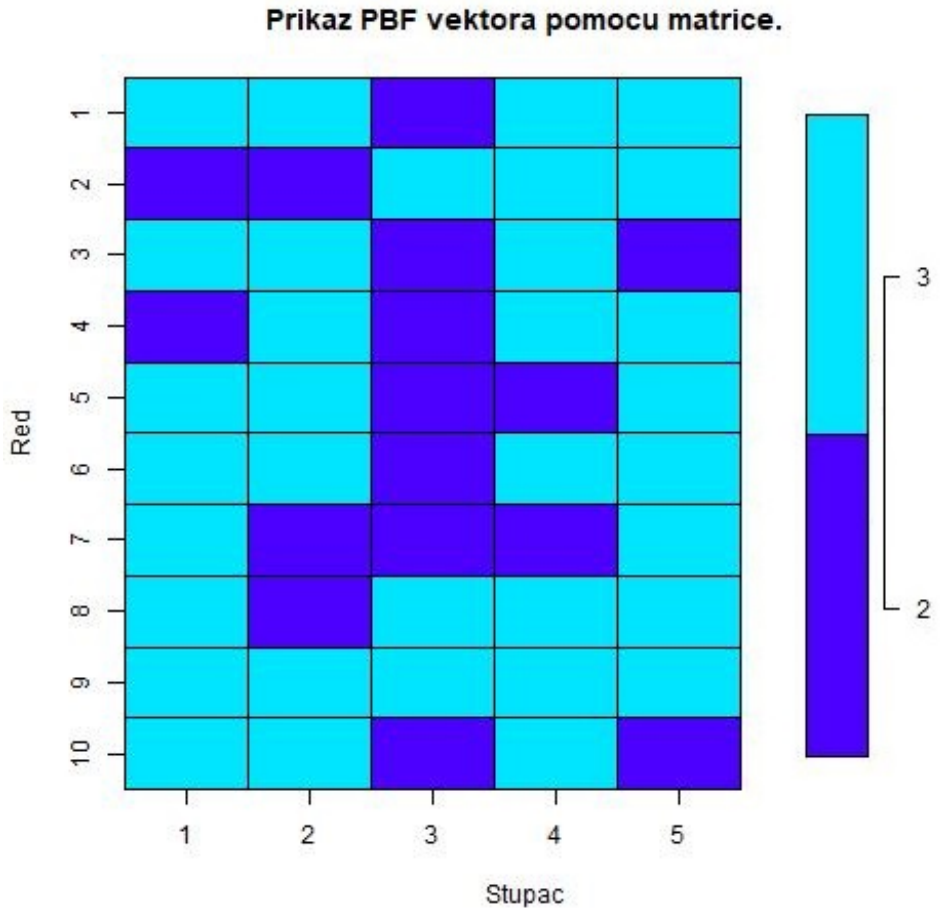
Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	1	1	1	1	0	0	1
1	1	1	1	1	0	0	1
2	1	1	1	1	0	0	1
3	1	1	1	1	0	0	1

4.1.2 Provjera PBF-a za vektor veličine 50 polja i 10 funkcija raspršivanja nad testnim točkama slučajno odabranih uniformnom razdiobom

Kada je veličina PBF vektora premala, a količina funkcija raspršivanja je velika, vjerojatnost za koliziju je velika. Zbog velike vjerojatnosti kolizije, oznake većeg prioriteta su u potpunosti prebrisale oznake jedan i nula. Posljedice kolizije mogu se vidjeti na histogramu 4.8, kao i na slici 4.9 vektora.



Slika 4.8 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 50 i k = 10.



Slika 4.9 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 50 i k = 10. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

U matrici zabune 4.3 može se uočiti da PBF s ovakvim parametrima i testnim točkama ima veliku pristranost prema oznaci dva. Filter niti nije mogao predvidjeti da se neka točka nalazi područjima jedinice i nule jer u PBF vektoru nije niti bilo navedenih oznaka. Tablica 4.4 pokazuje da je za oznake nula i jedan PBF imao za sve točke pogrešku drugog tipa, dok je za oznaku dva u potpunosti imao pogrešku prvog tipa. Obije oznake su imale i potpuni odaziv zbog čega F1 ocjena nije u potpunosti nula. Jedina oznaka koja je u potpunosti pogođena

je oznaka tri. Treba napomenuti kako skup točaka generiran uniformnom razdiobom nije bio dovoljan za provjeru samih područja interesa. Zbog toga i postoji drugi skup, kako bi se bolje ispitale greške između samih područja interesa uz iste parametre. Sveukupna F1 makro ocjena filtera s ovim parametrima je 0.2598, što je veoma loš rezultat koji ukazuje da treba promijeniti parametre.

Tablica 4.3 Matrica zabune za uniformno generirane točke i s PBF parametrima m = 50 i k = 10

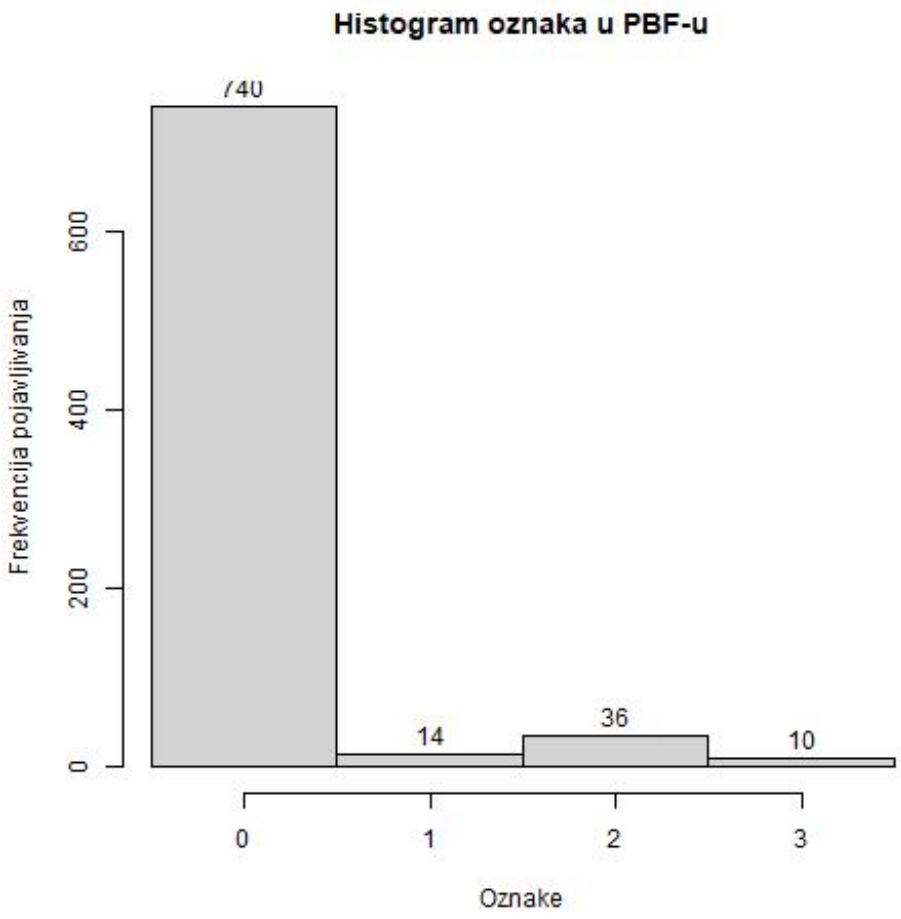
oznaka \ prognoza	0	1	2	3
0	0	0	97	0
1	0	0	1	0
2	0	0	2	0
3	0	0	0	0

Tablica 4.4 F1 vrijednosti za uniformno generirane točke i s PBF parametrima m = 50 i k = 10. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	0.03	0	0	0	0	1	0.2598
1	0.99	0	0	0	0	1	0.2598
2	0.02	1	0.02	0.0392	1	0	0.2598
3	1	1	1	1	0	0	0.2598

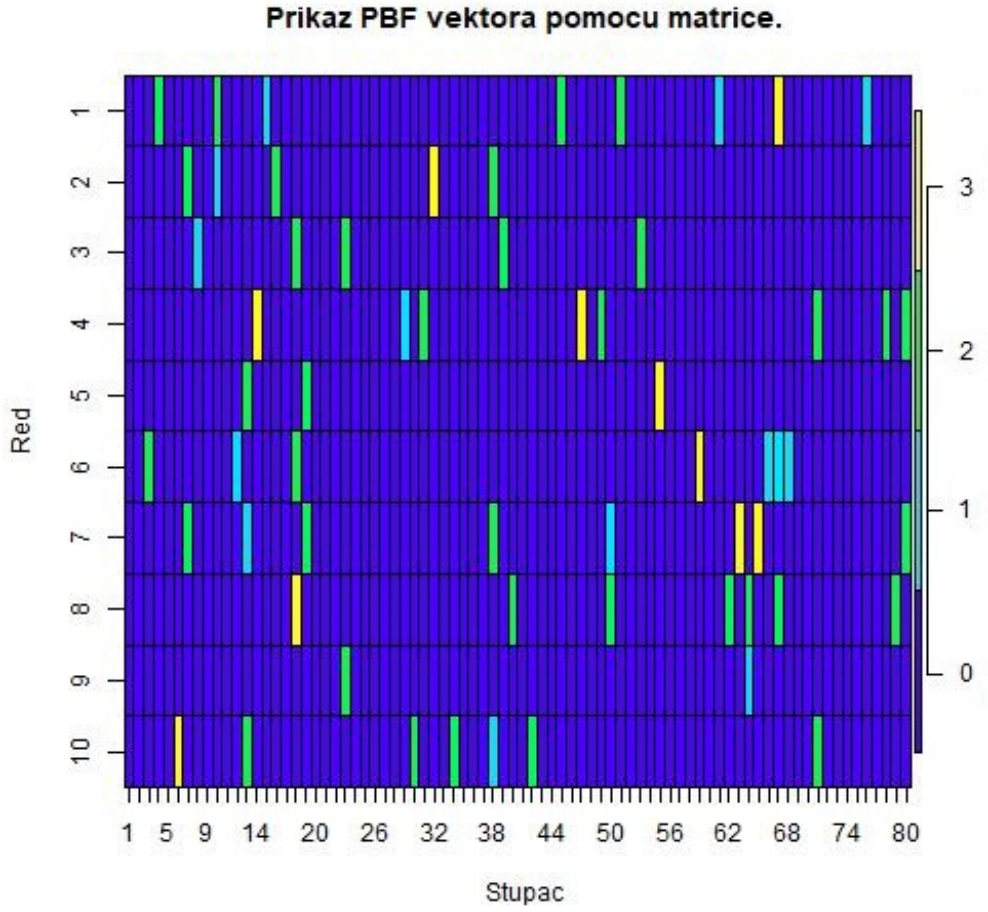
4.1.3 Provjera PBF-a za vektor veličine 800 polja i dvije funkcije raspršivanja nad testnim točkama slučajno odabranih uniformnom razdiobom

Kada je veličina PBF vektora dovoljno velika, a količina funkcija raspršivanja mala, tada nema dovoljan broj oznaka područja interesa u vektoru. Mali broj oznaka područja može se primijetiti na histogramu 4.10. Vidljivo je da je 92.5% oznaka nula kao i na slici vektora 4.11.



Slika 4.10 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 800 i k = 2.

Matrica zabune 4.5 prikazuje da je samo jednu točku koja se nalazi unutar područja s oznakom nula krivo predvidjela da se nalazi u području s oznakom jedan. No zato je ostale točke u područjima s oznakom jedan i dva točno predvidjela. Tablica 4.6 pokazuje naizgled poprilično dobar F1 makro rezultat od 0.9154. Dobiveni rezultat je bolji od očekivanog jer u skupu generiranom uniformnom razdiobom, testne točke su se velikom većinom nalazile u području s oznakom nula što je vjerojatno razlog zbog kojeg je filter dobro prognozirao.



Slika 4.11 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 800 i k = 2. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

Tablica 4.5 Matrica zabune za uniformno generirane točke i s PBF parametrima m = 800 i k = 2

oznaka \ prognoza	0	1	2	3
0	96	1	0	0
1	0	1	0	0
2	0	0	2	0
3	0	0	0	0

Tablica 4.6 F1 vrijednosti za uniformno generirane točke i s PBF parametrima m = 800 i k = 2. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	0.99	0.9897	1	0.9948	0	0.0103	0.9154
1	0.99	1	0.5	0.6667	0.0101	0	0.9154
2	1	1	1	1	0	0	0.9154
3	1	1	1	1	0	0	0.9154

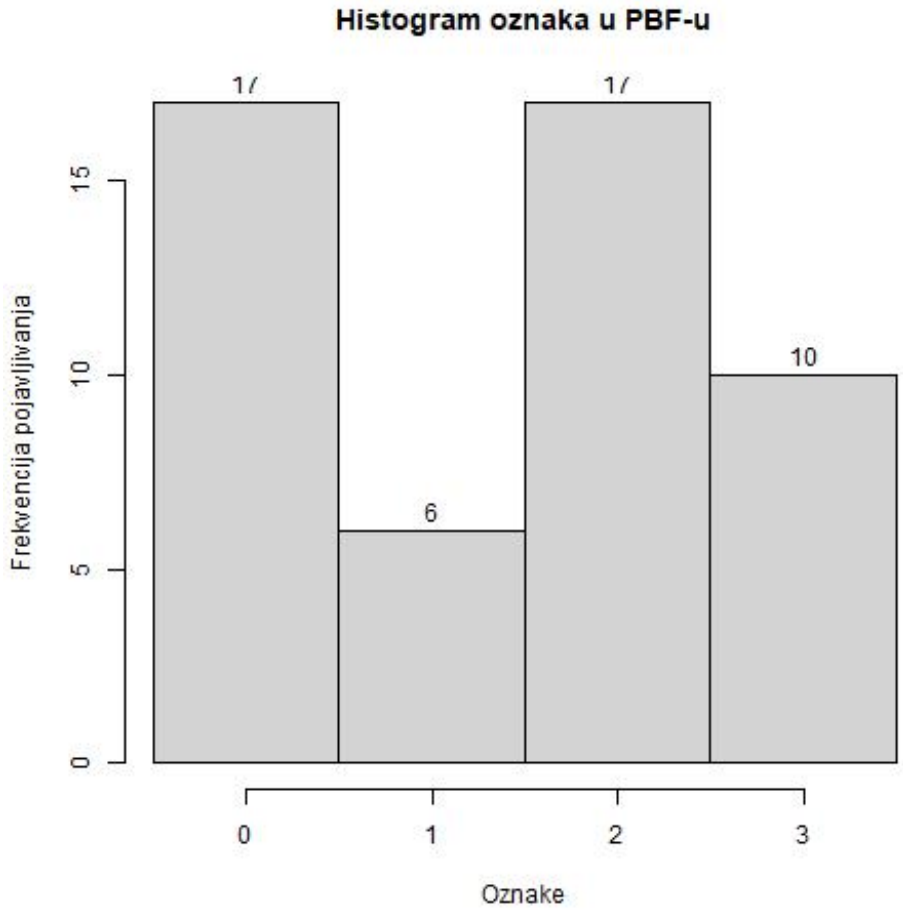
4.1.4 Provjera PBF-a za vektor veličine 50 polja i dvije funkcije raspršivanja nad testnim točkama slučajno odabranih uniformnom razdiobom

U zadnjoj kombinaciji parametara, kada su obije vrijednosti premale, može se pretpostaviti da će rezultati prognoziranja biti loši. Histogram 4.12 prikazuje da su sve oznake zastupljene u PBF vektoru, za razliku od kombinacija parametra gdje je veličina vektora bila 50 i količina funkcija raspršivanja 10. U prethodnoj kombinaciji, zbog količine funkcija raspršivanja samo dvije oznake su bile zastupljene u PBF vektoru. Zbog manje količine funkcija raspršivanja bilo je manje kolizija i oznake većeg prioriteta nisu obrisale oznake manjeg prioriteta. Na PBF vektoru prikazanom slikom 4.13 vidi se da su, iako je vektor manje veličine, oznake u njemu još uvijek dosta raspršene te nema većih grupiranja oznaka.

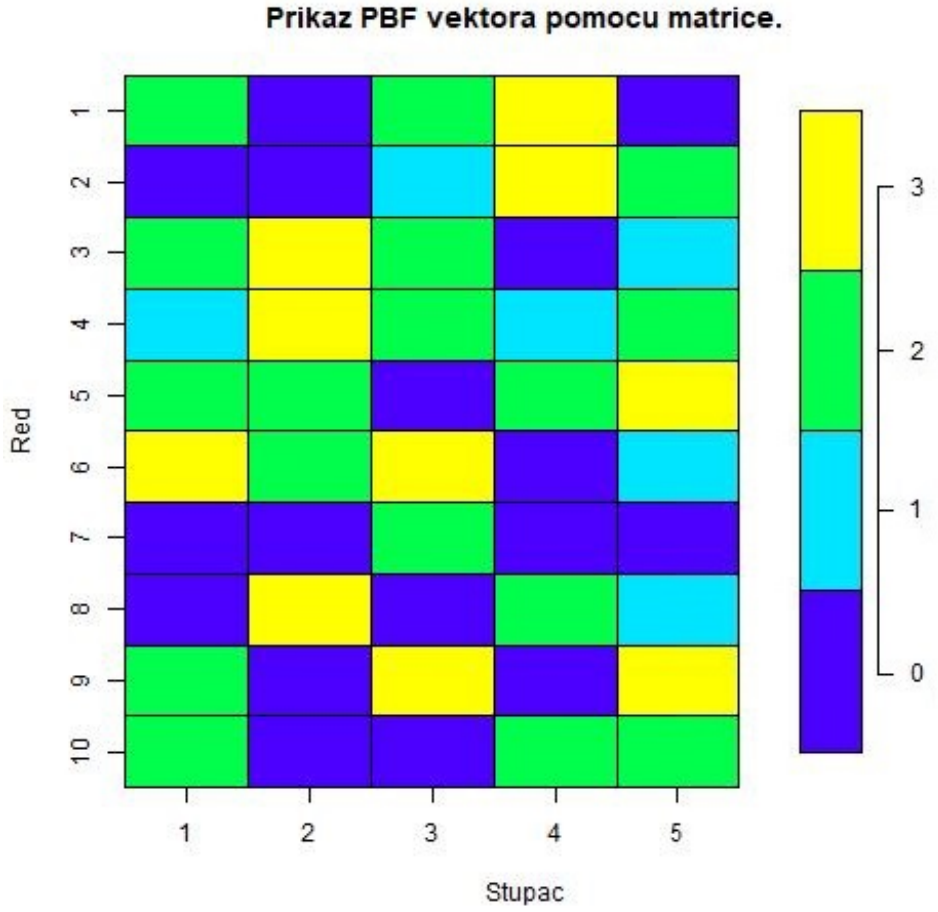
Pomoću matrice zabune 4.7, može se zaključiti kako je četvrta testirana kombinacija parametara za skup generiran uniformnom razdiobom testnih točaka. Čak 49 testnih točaka ima krivu prognozu. Detaljnija analiza grešaka se može vidjeti u tablici 4.8. Najgoru prognozu ima oznaka nula čija je točnost 0.52 i pogreška drugog tipa je 0.4948. To znači da bi filter kategorizirao nešto manje od polovice korisnika koji su izvan područja interesa da se zapravo nalaze unutar nekih od područja interesa. Unatoč tome, ima veću F1 vrijednost od svih ostalih oznaka zbog preciznosti koja je 1 i odaziva koji je 0.5052. Druge oznake imaju veću točnost, no F1 vrijednosti su im značajno lošije. Usporedbom prethodnih F1 makro vrijednosti, četvrta kombinacija parametara je ostvarila najnižu vrijednost. To je indikacija da bi trebalo naći puno bolje parametre.

Tablica 4.7 Matrica za uniformno generirane točke i s PBF parametrima m = 50 i k = 2

oznaka \ prognoza	0	1	2	3
0	49	15	27	6
1	0	1	0	0
2	0	0	1	1
3	0	0	0	0



Slika 4.12 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 50 i k = 2.



Slika 4.13 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su uniformnom razdiobom. Korišteni PBF parametri su m = 50 i k = 2. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

Tablica 4.8 F1 vrijednosti za uniformno generirane točke i s PBF parametrima m = 50 i k = 2. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	0.52	0.5052	1	0.6712	0	0.4948	0.2139
1	0.85	1	0.0625	0.1176	0.1515	0	0.2139
2	0.72	0.5	0.0357	0.0667	0.2755	0.5	0.2139
3	0.93	0	0	0	0.07	0	0.2139

4.1.5 Provjera PBF-a za vektor veličine 800 polja i 10 funkcija raspršivanja nad testnim točkama slučajno odabranih normalnom razdiobom

Nakon što su prikazani svi rezultati za točke slučajno izabrane uniformnom razdiobom, sljedeći na redu su rezultati točaka slučajno izabranih normalnom razdiobom. Zbog toga prvi test ponovno ima parametre iz prve skupine parametara

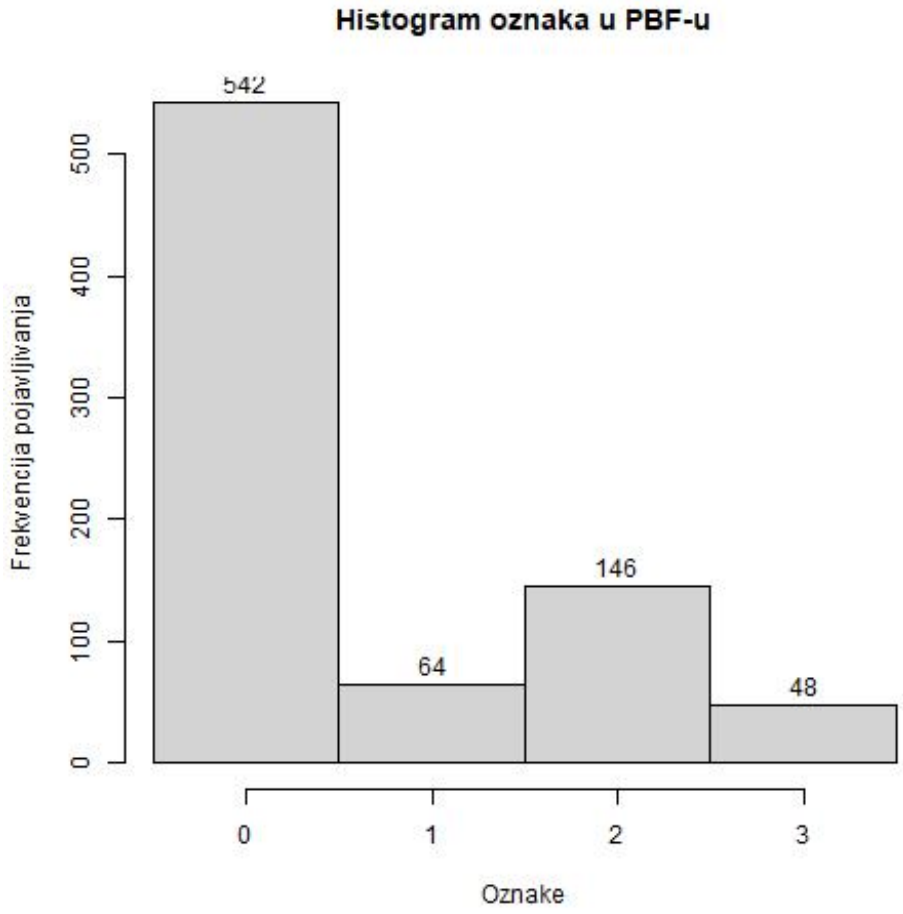
Ako se usporedi histogram 4.6 s histogramom 4.14, može se primijetiti da je raspodjela oznaka unutar vektora poprilično slična, s prosječnom razlikom od 5 elemenata te najvećom razlikom od 9 elemenata. Iako su histogrami isti, usporedbom pozicija oznaka unutar PBF vektora između slike 4.7 i slike 4.15 dolazi se do zaključka da su elementi različito raspršeni što je i očekivano, obzirom da su korištene različite funkcije raspršivanja za oba slučaja.

Kao i kod primjera s istim parametrima ali kod točaka slučajno odabranih uniformnom razdiobom, matrica zabune 4.9 i F1 vrijednosti 4.10 pokazuju da nije došlo do niti jedne greške i s ovim skupom testnih točaka.

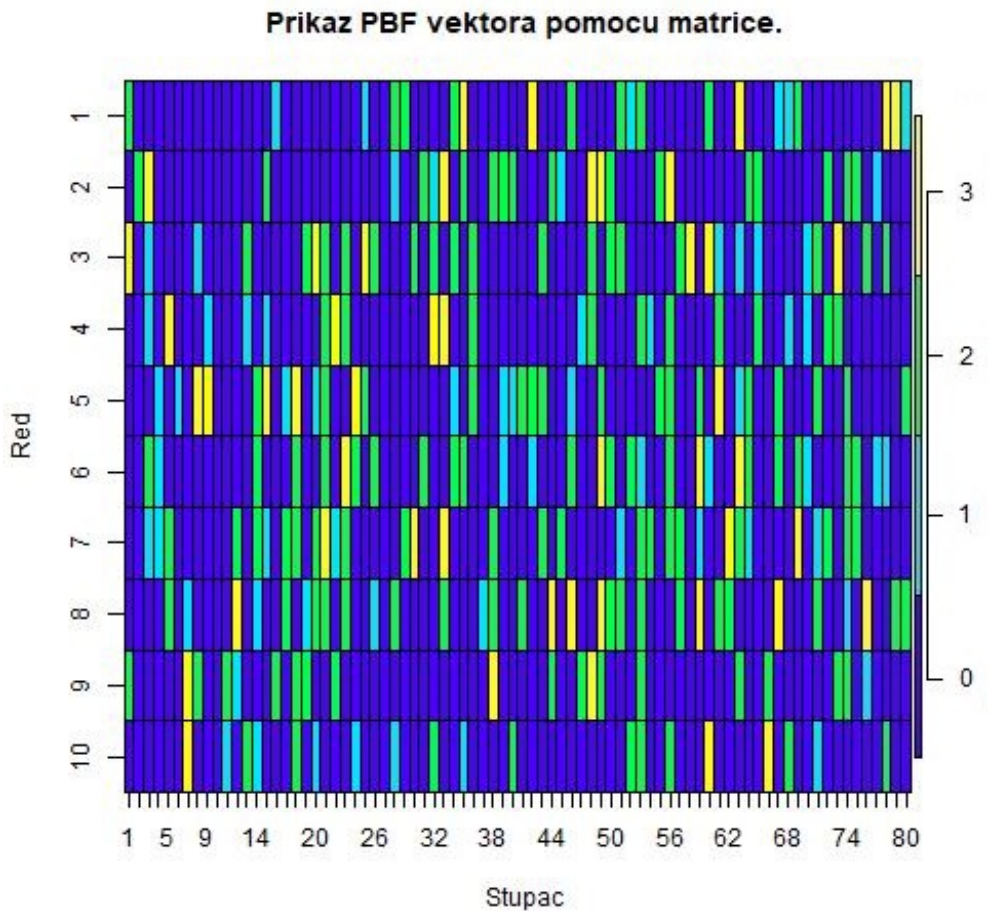
U oba skupa ovi su se parametri pokazali veoma pouzdanima za definirana područja interesa. Iako je zadana kombinacija parametara ostvarila odlične rezultate, u slučaju da se poveća površina područja interesa ili broj područja interesa, vrlo vjerojatno bi trebalo povećati i broj polja u PBF vektoru.

Tablica 4.9 Matrica zabune za normalno generirane točke i s PBF parametrima m = 800 i k = 10

oznaka \ prognoza	0	1	2	3
0	26	0	0	0
1	0	14	0	0
2	0	0	40	0
3	0	0	0	20



Slika 4.14 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 800 i k = 10.



Slika 4.15 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 800 i k = 10. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

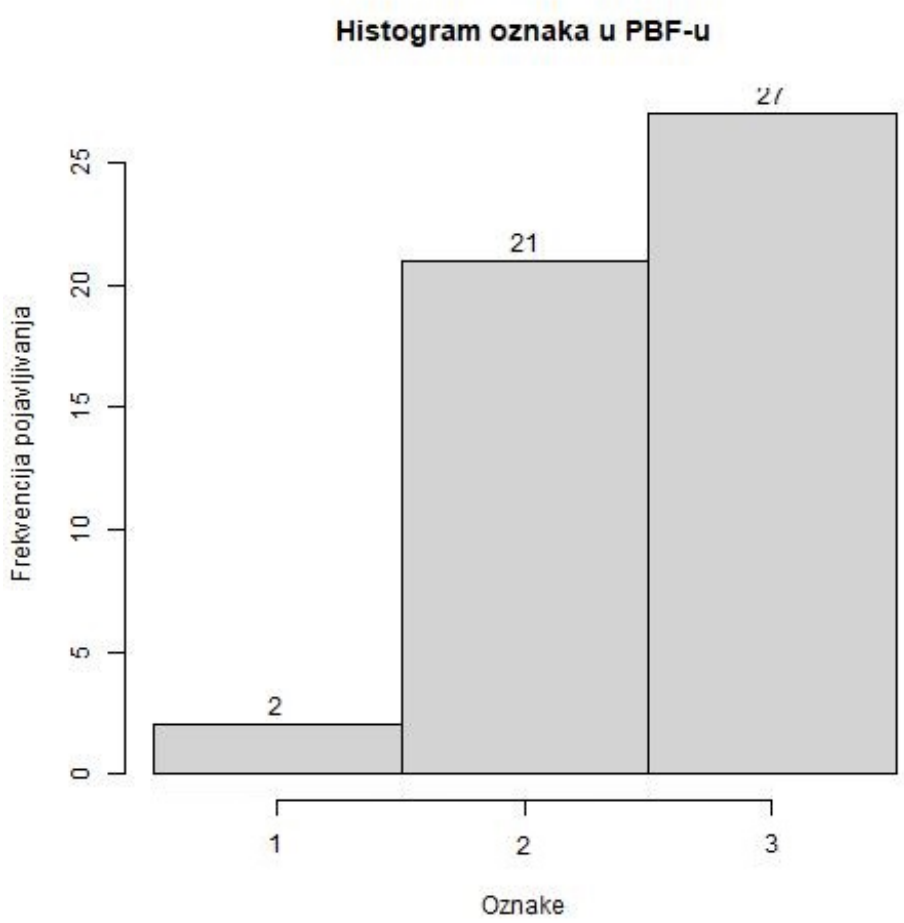
Tablica 4.10 F1 vrijednosti za normalno generirane točke i s PBF parametrima m = 800 i k = 10. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	1	1	1	1	0	0	1
1	1	1	1	1	0	0	1
2	1	1	1	1	0	0	1
3	1	1	1	1	0	0	1

4.1.6 Provjera PBF-a za vektor veličine 50 polja i 10 funkcija raspršivanja nad testnim točkama slučajno odabranih normalnom razdiobom

Histogram 4.16 je sličan histogramu 4.8 gdje je prosječna razlika četiri elementa, a maksimalna sedam elemenata između dva histograma. Jedina veća razlika je da unutar PBF vektora oznaka jedan ima dva predstavnika za točke izabrane normalnom razdiobom. Razlika u količini predstavnika je vrlo vjerojatno izazvana korištenjem različitih funkcija raspršivanja. Ovakav rubni slučaj bi trebalo pomnije istražiti.

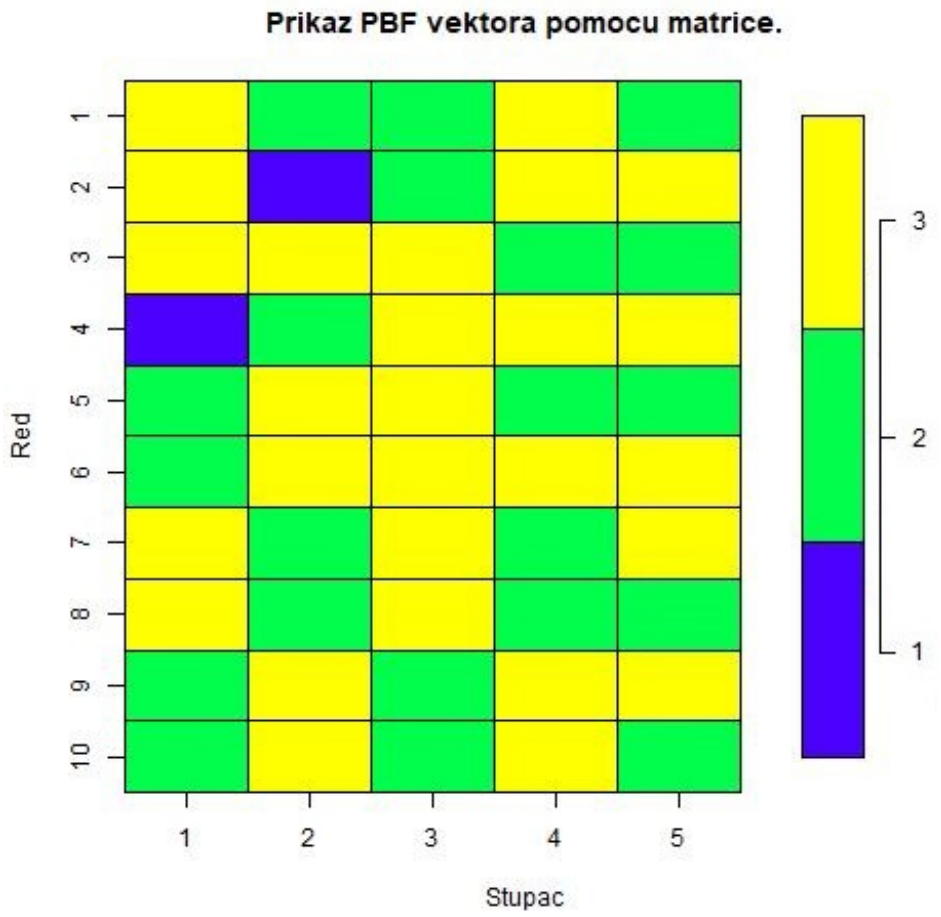
Matrica zabune 4.11 prikazuje da filter nije niti jednom točno predvidio točke koje se nalaze u području nula. PBF nije mogao točno predvidjeti točke koje se nalaze u području s oznakom nula jer nije bilo niti jedne oznake nula unutar PBF vektora. PBF je oznaku jedan predvidio točno u osam slučajeva, a unutar vektora je imao samo dva polja s oznakom jedan. Sve točke su točno prognozirane unutar oznake dva, no zato je filter krivo predvidio točke unutar oznaka jedan i nula da pripadaju oznaci dva. Jedina oznaka koja nije imala nikakve greške je oznaka tri. Tablica 4.12 potvrđuje prethodne tvrdnje brojkama, pa oznaka nula ima sto postotnu pogrešku drugog tipa i mikro F1 vrijednost je 0. Oznaka jedan ima pogrešku prvog tipa 0.0698 i pogrešku drugog tipa 0.4286 uz mikro F1 vrijednost 0.5714. Oznaka dva ima pogrešku prvog tipa 0.4333 i mikro F1 vrijednost od 0.6060. Jedina oznaka koja ima mikro F1 vrijednost 1 je oznaka 3. F1 makro vrijednost je 0.5815, što je značajno više od 0.2598 koliko je imao PBF vektor s istim parametrima, ali drugim skupom slučajno odabranih točaka. Razlog veće F1 makro vrijednosti leži u tome što je skup testnih točaka generiranih uniformnom razdiobom imao 97% točaka unutar oznake nula koja nije bila ni zastupljena u PBF vektoru, te najmanje točaka unutar oznake dva i tri koje su najzastupljenije unutar vektora. Zato skup točaka slučajno odabranih normalnom razdiobom pretežno sadrži točke unutar oznake dva i tri koje su najzastupljenije u filteru te time postiže bolje rezultate. Međutim, u prosjeku kada se uzmu rezultati svih testiranja iz oba skupa ova kombinacija parametara postiže najgori rezultat.



Slika 4.16 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 50 i k = 10.

Tablica 4.11 Matrica zabune za normalno generirane točke i s PBF parametrima m = 50 i k = 10

oznaka \ prognoza	0	1	2	3
0	0	6	20	0
1	0	8	6	0
2	0	0	40	0
3	0	0	0	20



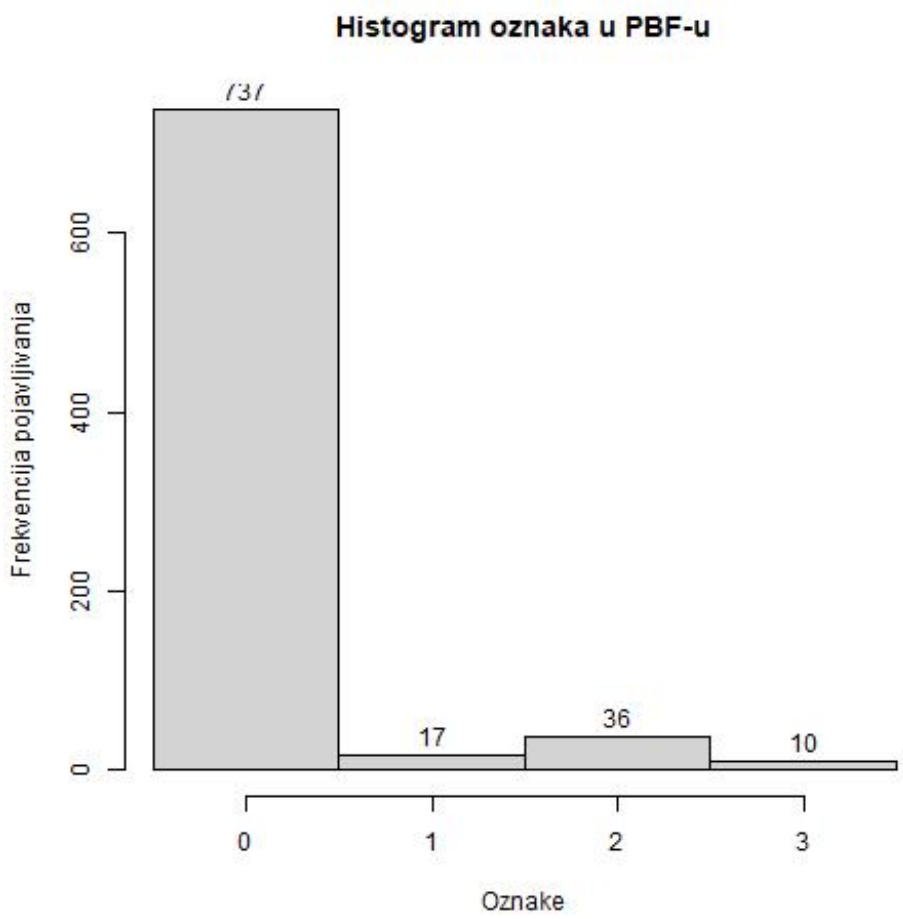
Slika 4.17 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 50 i k = 10. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

Tablica 4.12 F1 vrijednosti za normalno generirane točke i s PBF parametrima m = 50 i k = 10. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

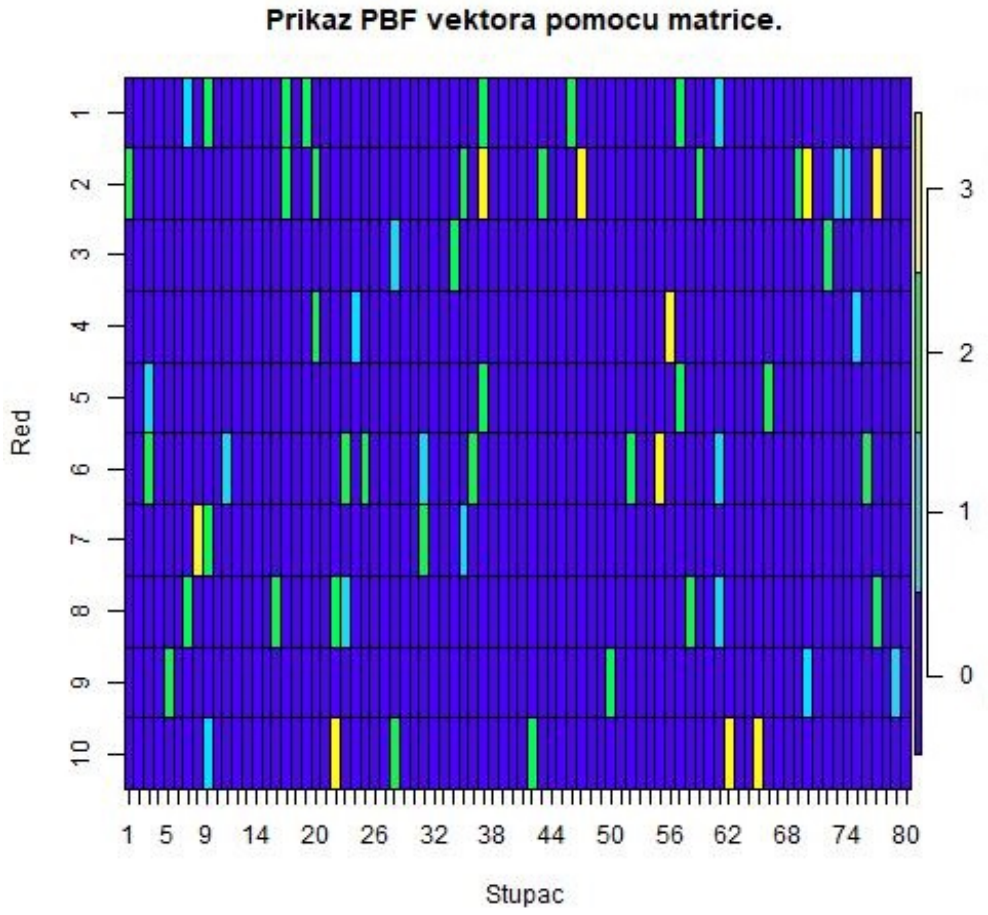
Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	0.74	0	0	0	0	1	0.5815
1	0.88	0.5714	0.5714	0.5714	0.0698	0.4286	0.5815
2	0.74	1	0.6060	0.7547	0.4333	0	0.5815
3	1	1	1	1	0	0	0.5815

4.1.7 Provjera PBF-a za vektor veličine 800 polja i dvije funkcije raspršivanja nad testnim točkama slučajno odabranih normalnom razdiobom.

Prikazan histogram 4.18 PBF vektora je gotovo identičan histogramu 4.10. Jedino se razliku u tome da ima tri elementa više za oznaku jedan dok ima tri elementa manje za oznaku nula. Zbog malog broja funkcija raspršivanja PBF većinom zauzimaju oznake nule, što se može vidjeti u prikazu vektora 4.19.



Slika 4.18 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 800 i k = 2.



Slika 4.19 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 800 i k = 2. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

Unatoč maloj zastupljenosti ostalih oznaka unutar vektora, filter je uspio točno predvidjeti oznake svih testnih točaka što pokazuje matrica zabune 4.13 i tablica F1 vrijednosti. Iako za skup testnih točaka generiran normalnom razdiobom nije bilo niti jedne pogreške, u prethodnom skupu bila je jedna pogreška od sto testnih točaka. Kako dodavanje još funkcija raspršivanja nije računalno zahtjevno omjer dobivenog rezultata i pokušaja optimizacije nije opravdan da bi se riskirala greška od jedan posto.

Tablica 4.13 Matrica zabune za normalno generirane točke i s PBF parametrima m = 800 i k = 2

oznaka \ prognoza	0	1	2	3
0	26	0	0	0
1	0	14	0	0
2	0	0	40	0
3	0	0	0	20

Tablica 4.14 F1 vrijednosti za normalno generirane točke i s PBF parametrima m = 800 i k = 2. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	1	1	1	1	0	0	1
1	1	1	1	1	0	0	1
2	1	1	1	1	0	0	1
3	1	1	1	1	0	0	1

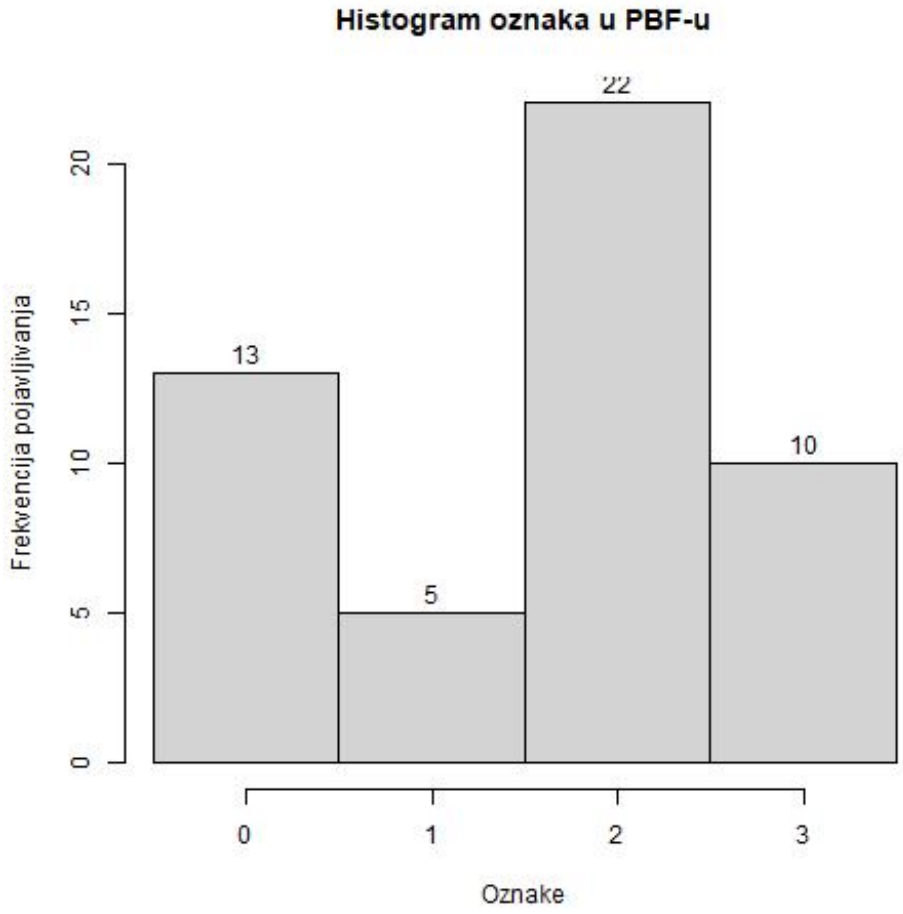
4.1.8 Provjera PBF-a za vektor veličine 50 polja i dvije funkcije raspršivanja nad testnim točkama slučajno odabranih normalnom razdiobom

Zadnji par parametara je za skup testnih točaka odabranih normalnom razdiobom, gdje je veličina vektora pedeset polja i dvije funkcije raspršivanja. Histogram 4.20 je također veoma sličan histogramu PBF vektora koji je imao iste parametre, ali s drugim testnim točkama 4.12. Za oznaku nula razlika je u četiri oznake, za oznaku jedan u jednoj oznaci, oznaka dva ima razliku od pet oznaka više, dok je broj oznake tri isti u oba slučaja.

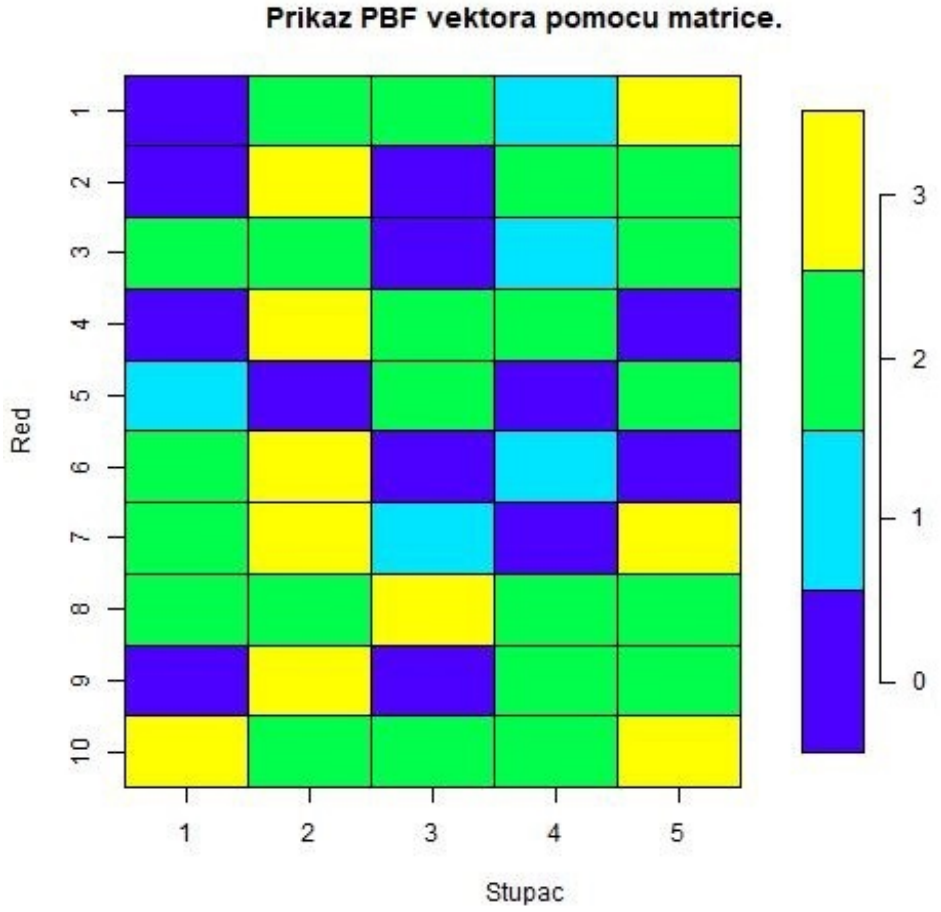
Manja zastupljenost oznake nula unutar PBF filtera je uzrokovala da je dvadeset točaka s tom oznakom krivo prognozirano, što je vidljivo iz tablice 4.15. Četiri točke su prognozirane da pripadaju oznaci jedan dok je šesnaest oznaka prognozirano da pripadaju oznaci dva. Zbog toga je pogreška drugog tipa za oznaku nula poprilično velika i iznosi 0.7692. Filter je najbolje predvidio za oznaku tri što se može vidjeti na F1 mikro vrijednosti od 0.9302. Nakon nje je najbolje predvidio oznaku jedan s F1 mikro vrijednosti 0.8, pa oznaku dva s mikro vrijednosti 0.7789 te na posljetku oznaku tri uz F1 mikro vrijednost 0375. F1 makro vrijednost za cijeli filter je 0.7210 što nije dobar rezultat. Pogotovo uzme li se u obzir da je za iste parametre ali s točkama koje su izabrane uniformnom razdiobom i sadrže puno više točaka s oznakom nula, F1 makro vrijednost bila 0.2139, što je veoma loš rezultat.

Tablica 4.15 Matrica zabune za normalno generirane točke i s PBF parametrima m = 50 i k = 2

oznaka \ prognoza	0	1	2	3
0	6	4	16	0
1	0	12	2	0
2	0	0	37	3
3	0	0	0	20



Slika 4.20 Histogram svih oznaka koje se nalaze u PBF-u. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 50 i k = 2.



Slika 4.21 Prikaz PBF vektora u obliku matrice. Testne točke izgenerirane su normalnom razdiobom. Korišteni PBF parametri su m = 50 i k = 2. Svaka boja u matrici predstavlja određenu oznaku. Odnos boje i oznake je prikazan na legendi s desne strane.

Tablica 4.16 F1 vrijednosti za normalno generirane točke i s PBF parametrima m = 50 i k = 2. U tablici T predstavlja točnost, O odaziv, P preciznost, 1. tip predstavlja pogrešku prvog tipa, a 2. tip pogrešku drugog tipa.

Oznaka	T	O	P	F1	1. tip	2. tip	F1 makro
0	0.8	0.2308	1	0.375	0	0.7692	0.7210
1	0.94	0.8571	0.75	0.8	0.0465	0.1429	0.7210
2	0.79	0.925	0.6727	0.7789	0.3	0.075	0.7210
3	0.97	1	0.8696	0.9302	0.0375	0	0.7210

4.2 Provjera brzine izvođenja PBF-a

Za provjeru brzine izvođenja korištene su funkcije `Sys.time()` i `difftime()`. Rezultati funkcije `Sys.time()` su spremljeni prije i nakon izvođenja dijela koda zaduženog za pronalaženje područja interesa. Funkcija `difftime()` se koristi za dobivanje razlike vremena. Razlikom vremena dobiva se trajanje izvršavanja dijela koda. Osim za izračunavanje trajanja cjelokupnog procesa pronalaženja odgovarajućeg područja interesa, na isti način se računa trajanje izvođenja funkcije `sf::st_intersects()` iz knjižnice `sf`. Prethodno spomenuta funkcija se koristi za izračunavanje unutar kojeg polja u mreži se nalazi prostorna točka. Prosječna vremena izvođenja su prikazana u tablici 4.17. Iz prethodne tablice vidljivo je kako je funkcija `sf::st_intersects()` potrošila u prosjeku 98% vremena izračuna pripadnosti prostorne točke ćeliji.

Tablica 4.17 Prvi stupac prikazuje prosječno vrijeme izračuna unutar kojeg se područja interesa nalazi točka. Drugi stupac prikazuje prosječno vrijeme izračuna funkcije intersects(). Oba prosjeka su dobivena kroz testiranje sto točaka. Treći stupac prikazuje razliku prethodno navedenih vremena. Vremena su izražena u milisekundama. U četvrtom stupcu je postotak ukupnog vremena potrošenog na funkciju intersects().

t	`t_intersects`	t - `t_intersects`	%
111.3904 ms	109.4603 ms	1.9301 ms	98%

Poglavlje 5

Zaključak

U ovom radu PBF je razmotren sa stajališta izvedbe u programskom okruženju R i vrednovanja uspješnosti.

Namjera je bila pokazati koliko dobro PBF može predvidjeti područje interesa za neku zadanu prostornu točku ovisno o zadanim parametrima. Uspješnost PBF-a, provjeravala se pomoću matrice zabune, F1 vrijednosti i interaktivne mape.

U okruženju za statističko računarstvo R, razvijena je programska podrška za izradu i provjeru PBF-a. Programska podrška sastoji se od grafičkog sučelja i algoritama prikazanih u ovom radu.

Svi filteri mogu prouzročiti gubitak informacija, pa tako i PBF. No u ovom radu, dokazano je da gubitak informacija može biti kontroliran pomoću parametara te ga se uz dobre parametre čak svesti na zanemarivu razinu. Pokazano je da PBF nikada neće predvidjeti da točka pripada području manjeg prioriteta ako se nalazi u području većeg prioriteta. U većini slučajeva, oznaka s najvećim prioritetom imala je najmanje grešaka. Gubitak informacijskog sadržaja nije predmet ovog diplomskog rada te će biti razmotren u budućim istraživanjima.

Bibliografija

[1] L. Calderoni, P. Palmieri, and D. Maio, "Location privacy without mutual trust: The spatial bloom filter," *Computer Communications*, vol. 68, pp. 4–16, 2015, security and Privacy in Unified Communications Challenges and Solutions. , s Interneta, https://www.sciencedirect.com/science/article/pii/S0140366415002273

[2] Introduction to R. , s Interneta, https://www.r-project.org/about.html

[3] D. Slavić, "Hash tablice," 2019. , s Interneta, https://urn.nsk.hr/urn:nbn:hr:126:498077

[4] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[5] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Machine Learning – A First Course for Engineers and Scientists.* Cambridge University Press, 2022. , s Interneta, http://smlbook.org/

[6] J. Opitz and S. Burst, "Macro F1 and macro F1," *CoRR*, vol. abs/1911.03347, 2019. , s Interneta, http://arxiv.org/abs/1911.03347

[7] Bloom Filter Tutorial. , s Interneta, https://llimllib.github.io/bloomfilter-tutorial/

[8] P. Biecek and T. Burzykowski, *Explanatory Model Analysis: Explore, Explain, and Examine Predictive Models. With examples in R and Python.* CRC Press, 2020. , s Interneta, https://ema.drwhy.ai/

Pojmovnik

BF Bloomov filter. 2, 3

CRAN The Comprehensive R Archive Network. 64

CSV comma-separated values. 19, 21, 22

dEPSG European Petroleum Survey Group. 12

PBF Prostorni Bloomov filter. x–xiv, 1, 3–5, 7, 11–14, 22, 25, 28–58, 60

WGS World Geodetic System. 12

Sažetak

Sve više ljudi postaje svjesnije važnosti informacija koje stvaraju te želi imati privatnost i kontrolu nad svojim osobnim podacima. Rast mobilnih aplikacija zasnovanih na lokacijskim uslugama ugrožava prethodno navedenu potrebu. No uz pomoć korištenja Prostornog Bloomovog filtera kao osnove za definiranje nalazi li se osoba u području interesa, moguće je sačuvati privatnost bez gubitka kvalitete aplikacije zasnovane na lokacijskim uslugama. U ovom radu predstavlja se i testira navedena struktura podataka. Također se objašnjava korištenje izrađenog grafičkog korisničkog sučelja. Sučelje je izrađeno zbog olakšanog testiranja i prikaza uspješnosti filtra.

Ključne riječi **— PBF, strukture podataka, prostorni podaci**

Abstract

The awareness of the importance of personal information is getting higher and people want to have better privacy and control over their personal data. The growth of mobile applications based on location services threatens the need for keeping personal information private. However, with the help of the Spatial Bloom filter as a basis for defining whether a person is in the area of interest, it is possible to preserve privacy and still have high-quality applications based on location services. In this paper, the mentioned data structure is presented and tested. This paper also provides a user guide of the created graphical user interface. The interface was designed for easy testing and as a display of the filter performance.

Keywords **— SBF, data structures, spatial data**

Dodatak A

Prilog A: instalacija R je programskog jezika i okruženja za statističko računarstvo i grafički prikaz

A.1 Upute za instalaciju R je programskog jezika na Windows operativnom sustavu

Prvo je potrebno otići na stranicu The Comprehensive R Archive Network (CRAN) web stranice: https://cran.r-project.org/. Zatim je potrebno preuzeti verziju za Windows operacijski sustav. Nakon toga je potrebno pokrenuti instalacijsku datoteku i slijediti čarobnjaka za instalaciju. Po završetku instalacije može se pokrenuti RGui program kako bi se provjerilo je li R ispravno instaliran.

A.2 Upute za instalaciju R Studija na Windows operativnom sustavu

Potrebno je preuzeti instalacijsku datoteku sa stranice: `https://www.rstudio.com/products/rstudio/download/#download`. Nakon toga potrebno je pokrenuti instalacijsku datoteku i slijediti upute za instalaciju.

A.3 Upute za instalaciju R je programskog jezika na Ubuntu operativnom sustavu

Prvo je potrebno osvježiti predmemoriju paketa s naredbom: `sudo apt-get update`. Zatim je potrebno pokrenuti naredbu `sudo apt -y install r-base`. Na posljetku se može provjeriti je li R instaliran s pozivom `R --version`.

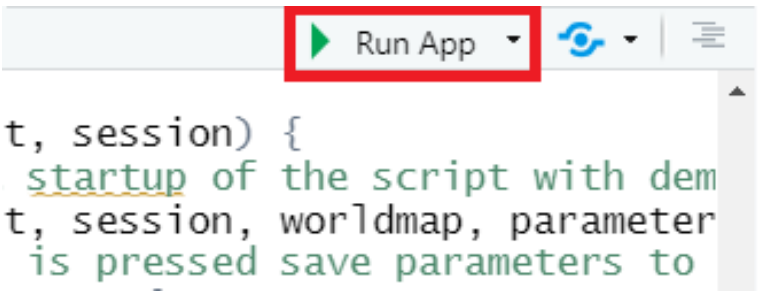
A.4 Upute za instalaciju R Studija na Ubuntu operativnom sustavu

Potrebno je prije instalirati gdebi paket za lakšu instalaciju .deb paketa sa sljedećim pozivima: `sudo add-apt-repository universe`, `sudo apt-get install gdebi-core`. Nakon toga se preuzme instalacijska datoteka s web stranice
`https://www.rstudio.com/products/rstudio/download/\#download`. Unutar terminala treba doći do lokacije gdje se nalazi preuzeta datoteka. Najčešće to bude unutar mape `Downloads`, a do nje se može navigirati pomoću naredbe `cd Downloads/`. Zatim se instalira R Studio pomoću naredbe
`sudo gdebi rstudio-[verzija_R_Studija]-amd64.deb`. Za pokretanje R studija koristi se naredba `rstudio` unutar terminala.

Dodatak B

Prilog B: Upute za instalaciju aplikacije

Prvo je potrebno instalirati R programski jezik te RStudio kako je opisano u prilogu A. Zatim je potrebno preuzeti kod sa sljedeće stranice https://doi.org/10.6084/m9.figshare.21803484. Kod je moguće preuzeti kao zip datoteku, koju je potrebno raspakirati na računalu pomoću programa kao što su 7zip ili WinRar. Kada je kod raspakiran treba otvoriti skriptu `app.R` pomoću RStudija i pokrenuti je pomoću tipke `Run App` koji se nalazi odmah iznad uređivača teksta u gornjem desnom kutu. Tipka je prikazana na slici B.1.



Slika B.1 Tipka za pokretanje aplikcaije.

Prilikom pokretanja aplikacije, sve potrebne knjižnice će se automatski instalirati. Kada aplikacija završi s instaliranjem potrebnih knjižnica, otvorit će se prozor s početnim primjerom. Korištenje same aplikacije je detaljno objašnjeno u poglavlju Izvedba PBF u programskom okruženju R.

Dodatak C

Prilog C: Repozitorij projekta

Izvorni kod može se pronaći na sljedećem linku https://doi.org/10.6084/m9.figshare.21803484 Osim izvornog koda mogu se naći i primjeri spremljenih slučajno generiranih točaka te spremljenih rezultata i grafova koji su korišteni u ovom radu.