Pčelinji algoritmi za odabir podskupa atributa

Puškarić, Lucija

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Lucija Puškarić

PČELINJI ALGORITMI ZA ODABIR PODSKUPA ATRIBUTA

Diplomski rad

Voditelj rada:
doc. dr. sc. Goranka Nogo

Zagreb, 2020.

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

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

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Prijateljima koji su mi pružili pomoć i unijeli vedrinu u studentske dane te cijeloj obitelji koja mi je pružila sigurnost, utjehu i podršku u svakom smislu.
Hvala vam! ♡

Sadržaj

Uvod

Devedesetih godina prošlog stoljeća počinje se sve više obraćati pozornost na podatke i informacije koje se mogu iščitati iz njih. Podaci se opisuju atributima. Broj promatranih atributa početkom 21. stoljeća drastično raste. Skupovi podataka počinju sadržavati desetke tisuća atributa. Stoga se javlja potreba za određivanjem atributa koji su značajni i koji daju određenu informaciju o podacima. Istraživanjem velikih skupova podataka bave se discipline poput strojnog učenja, rudarenja podataka, odnosno općenito podatkovne znanosti.

Odabir značajnih atributa se može promatrati kao optimizacijski problem jer se traži podskup atributa za koji dobivamo najbolju informaciju o podacima. U ovom radu vrši se odabir atributa u svrhu što bolje uspješnosti modela strojnog učenja. Primjena modela strojnog učenja na podacima kod kojih nije izvršen odabir, može dovesti do niza problema npr. loših rezultata, većeg vremena potrebnog za treniranje te do tzv. pretreniranja (eng. *overfitting*). Također, micanjem određenih atributa smanjuje se kompleksnost skupa podataka te on i model učenja postaju jednostavnije objašnjivi.

Prostor svih podskupova nekog skupa može biti jako velik. Stoga su meta-heurističke metode prikladne za pronalaženje dobrih rješenja na takvom prostoru. Naime, takve metode neće vršiti pretragu cijelog prostora već će se zadržavati na područjima gdje se nalaze dobra rješenja. Ne može se garantirati da će optimalno rješenje biti pronađeno, ali može se u prihvatljivom vremenu pronaći prihvatljivo rješenje. Zato, meta-heuristike možemo primijeniti i za rješavanje gore navedenog problema. Konkretno, bit će korištena jedna verzija pčelinjih algoritama.

U radu je sadržan opis problema, zatim je opisan koncept pčelinjih algoritama, a u nastavku slijedi razrada kako je meta-heuristika prilagođena i korištena u svrhu odabira značajki. Na kraju su rezultati dobiveni takvim načinom uspoređeni s rezultatima koji su dobiveni nekim uobičajenim metodama koje se koriste za rješavanje opisanog problema.

Poglavlje 1

Odabir podskupa atributa

1.1 Opis problema

Odabir podskupa atributa se u literaturi najčešće spominje kao odabir značajki (eng. *feature selection*), no ovdje će se najčešće koristiti upravo termin iz naslova. U uvodu je u kratkim crtama rečeno što je odabir podskupa atributa i koja je motivacija za vršenje takvog procesa pri primjeni modela strojnog učenja. Spomenuto je **izbjegavanje pretreniranja**. U slučaju pretreniranja dobivamo model koji je vrlo uspješan na podacima na kojima je treniran, ali na novim podacima ne radi dobre predikcije. To nije dobro zato što je cilj da se model može generalizirati i koristiti na neviđenim podacima. Više detalja o odabiru značajki je u člancima [4] i [6].

U nastavku će biti promatrani **modeli nadziranog učenja** tj. modeli kod kojih na skupu podataka za koje je poznata ciljna varijabla, treniramo model i pomoću njega predviđamo vrijednost ciljne varijable za novi podatak. Ciljna varijabla može imati kontinuiranu brojčanu vrijednost, tada bi se promatrao **regresijski model**, ali ovdje je fokus na ciljnoj varijabli koja ima diskretne vrijednosti. Takva vrijednost predstavlja klasu kojoj pojedini podatak pripada. Drugim riječima, promatrat će se **klasifikacijski modeli strojnog učenja**.

Skupovi podataka koji se u ovom kontekstu koriste, mogu se prikazati tabelarno kao u Tablici 1.1. Svaki od n podataka je opisan s m atributa i s ciljnom varijablom a_{m+1}.

Neki od atributa $a_1, a_2, ..., a_m$ mogu biti **nerelevantni**, odnosno ne doprinose zaključku kojoj klasi će pojedini podatak pripadati. Osim što neki od njih mogu biti nerelevantni, mogu biti i **redundantni**. Da je neki atribut redundantan znači da postoji velika povezanost (korelacija ili neka druga statistička mjera) s nekim drugim bitnim atributom pa jednog od njih možemo izbaciti zbog davanja gotovo iste informacije o ciljnoj varijabli.

Tablica 1.1: Skup podataka u obliku tablice

Podaci	Atributi			Ciljna varijabla
	a_1	...	a_m	a_{m+1}
p_1				
⋮		⋮		⋮
p_n				

Cilj je odrediti koje atribute od $a1, a2, ..., a_m$ ostaviti, a koje izbaciti da primjenom modela strojnog učenja dobijemo najtočnije rezultate.

1.2 Klasifikacijski modeli

Ovaj odjeljak služi za kratke opise korištenih klasifikacijskih modela bez ulaženja duboko u detalje. Svrha je istaknuti da su modeli međusobno različiti te da određeni model, za različite probleme, vrlo vjerojatno neće biti jednako uspješan.

Model k-najbližih susjeda

Model k-najbližih susjeda je jedan od najjednostavnijih klasifikacijskih modela. Pridjeljivanje klase novom podatku vrši se u sljedećim koracima:

1. Izračunavanje udaljenosti između novog podatka i svih pojedinih podataka u skupu za treniranje.

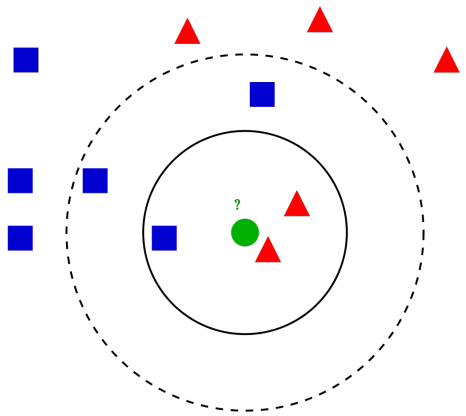
2. Određivanje skupa k-najbližih podataka.

3. Određivanje klase kojoj pripada većina podataka iz skupa k-najbližih.

4. Dodjela tako određene klase novom podatku.

Konkretan primjer je na donjoj slici[1]. Postoje dvije klase označene plavim kvadratima te crvenim trokutima. Pitanje je kojoj će klasi pripasti novi podatak (zeleni krug). Ako koristimo algoritam 3-najbliža susjeda (kružnica pune crte), novi podatak pripada klasi crvenih trokuta, a ako koristimo algoritam 5-najbližih susjeda (crtkana kružnica), podatak će pripasti klasi plavih kvadrata.

[1] Preuzeta s Wikipedije `https://commons.wikimedia.org/wiki/File:KnnClassification.svg`



Slika 1.1: Pridjeljivanje klase novom podatku pomoću algoritma KNN

Vidljivo je da postoje slobodni parametri koji se moraju odrediti prilikom primjene ovog modela. Parametar k, ako problem nije prekompleksan, može se prosto odrediti eksperimentiranjem, a za potrebe ovog rada to je bilo dovoljno. Također, bitno je definirati koja se metrika udaljenosti koristi. Najčešće korištena je euklidska udaljenost.

Naivni Bayesov model

Ovaj probabilistički model slijedi iz Bayesovog teorema čija je osnovna forma navedena u nastavku.

Teorem 1.2.1 (**Bayesov teorem**). *Neka su A i B događaji te neka je $P(B) \neq 0$. Tada vrijedi:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

gdje je:

- $P(A|B)$ *vjerojatnost da se dogodi A uz pretpostavku da je B istinit*
- $P(B|A)$ *vjerojatnost da se dogodi B uz pretpostavku da je A istinit*
- $P(A)$ *i* $P(B)$ *vjerojatnosti za događaje A i B pojedinačno.*

Označimo s $x_1, x_2, ..., x_n$ vrijednosti atributa koje opisuju novi podatak x. Vjerojatnost da taj podatak pripadne klasi C_k se može izraziti pomoću Bayesove formule kao:

$$P(C_k|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|C_k)P(C_k)}{P(x_1, x_2, ..., x_n)}.$$

Model sadrži epitet "naivni" zbog korištenja pretpostavke o uvjetnoj nezavisnosti x_i i x_j za svaki $i \neq j$, pa vrijedi:

$$P(x_1, x_2, ..., x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k).$$

Na kraju, navodimo formulu za određivanje indeksa klase za koju je najveća vjerojatnost da će joj podatak pripasti:

$$y = \underset{k\in\{1,...,K\}}{\arg\max} P(C_k) \prod_{i=1}^{n} P(x_i|C_k). \tag{1.1}$$

Model slučajnih šuma

Za razliku od prijašnjih modela, ovaj model je zapravo **ansambl** što znači da se na podacima trenira više modela te se određenim postupkom odabire najbolji rezultat. Zbog toga je ovo definitivno najkompliciraniji od spomenutih algoritama. U modelu slučajnih šuma stvara se više stabala odlučivanja. Postoji i **model stabla odlučivanja** gdje se stvara samo jedno, no logično je da ovaj ansambl daje u većini slučajeva bolje rezultate te izbjegava neke probleme pa je iz tog razloga i upotrebljavan u ovom kontekstu. U nastavku su opisani osnovni koraci u stvaranju jednog stabla odlučivanja. Pretpostavka je da su svi primjeri iz skupa podataka na početku u korijenu stabla.

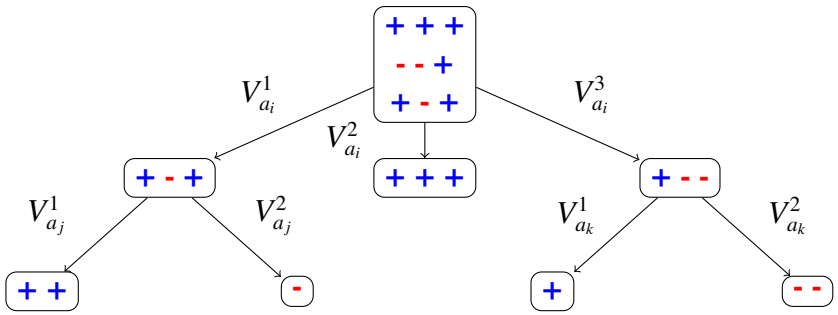
1. Odabiranje jednog atributa iz skupa atributa.
2. Razdjeljivanje primjera po vrijednostima tog atributa.
3. Rekurzivno stvaranje podstabla u čijim su korijenima razdijeljeni podaci iz 2.

   (Uvjet zaustavljanja je kada su u čvoru svi primjeri koji pripadaju istoj klasi.)

Dolje je prikazano jedno vrlo jednostavno stablo odlučivanja. Primjeri u skupu podataka su označeni oznakom klase **+** ili **-**. Odabirom prvog atributa a_i, koji ima tri vrijednosti $V_{a_i}^1, V_{a_i}^2, V_{a_i}^3$, stvorena su tri podstabla. Na sljedećoj razini odabrani su atributi koji imaju dvije vrijednosti u 1. i 3. podstablu s lijeva na desno (2. podstablo zadovoljava uvjet zaustavljanja).

Važno je napomenuti da razvitak stabla ovisi o odabiru atributa na početku i u koracima te postoje točno određeni postupci i kriteriji prema kojima se oni odabiru. Jedan od parametara koji mora biti određen u korištenju ovog algoritma je upravo postupak odabira atributa[2].

[2]Opisano je nekoliko postupaka na web stranici Towards Data Science: `https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4`



Slika 1.2: Stablo odlučivanja

Da bi novi podatak svrstali u klasu, promatramo u koji list će podatak pripasti prema njegovim vrijednostima atributa kroz stablo odluke.

Kao što je već napomenuto, algoritam slučajnih šuma stvara određeni broj stabala odlučivanja i to je također parametar koji se mora zadati. Ono što je specifično baš za ovaj ansambl jest da se stabla odlučivanja stvaraju na skupovima podataka dobivenim *bootstrap* uzorkovanjem. Na određen način, dobivaju se različiti skupovi podataka koji sadrže primjere iz izvornog skupa za treniranje. Također, osim što se skupovi razlikuju po primjerima koje sadrže, razlikuju se i po slučajno odabranim atributima koji su u njima prisutni. Ovaj ansambl je modificirana *bagging* metoda, a više informacija je u člancima autora Lea Breimana [2] i [3] gdje su prvi put opisane spomenute metode.

Evaluacija klasifikacijskog modela

Već je nekoliko puta navedeno da je u ovom radu bitno znati koliko točne rezultate daje klasifikacijski model.

Korišteno je razdvajanje skupa podataka na **skup za trening** na kojemu se trenira model te na **testni skup** gdje za dane podatke predviđamo klasu i uspoređujemo podudara li se predviđena klasa sa stvarnom koja je poznata za taj podatak.

Sve evaluacijske mjere klasifikacije se računaju iz tzv. **konfuzijske matrice** (eng. *confusion matrix*). U donjoj tablici je prikazana konfuzijska matrica za **binarni klasifikacijski problem**. Kod binarnog klasifikacijskog problema, podaci su razvrstani u dvije klase (1 ili 0).

Tablica 1.2: Konfuzijska matrica za binarni klasifikacijski problem

Predviđene klase	Stvarne klase	
	1	0
1	TP	FP
0	FN	TN

Veličina matrice je $2x2$. **TP** (*true positives*) je broj podataka čija je stvarna klasa 1 i predviđena je isto klasa 1, tj. predikcija je točna. **FP** (*false negatives*) je broj podataka čija je stvarna klasa 0, ali predviđena je 1, **FN** (*false negatives*) je broj podataka čija je stvarna klasa 1, ali predviđena je 0. I na kraju, **TN** (*true negatives*) je broj podataka čija je predviđena klasa ista kao i stvarna vrijednosti 0.
Neke često korištene **evaluacijske mjere**:

- **Točnost** = $\frac{TP+TN}{TP+FP+FN+TN}$.
- **Preciznost** = $\frac{TP}{TP+FP}$.
- **Osjetljivost** = $\frac{TP}{TP+FN}$.
- **Specifičnost** = $\frac{TN}{TN+FP}$.

U Poglavlju 3 će biti pojašnjeno zašto je odluka pala na korištenje dolje navedene mjere koja se naziva **balansirana točnost**:

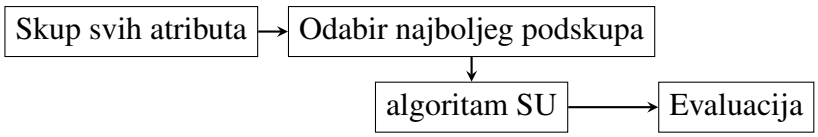
$$\frac{\frac{TP}{TP+FP}+\frac{TN}{TN+FN}}{2}. \tag{1.2}$$

Konfuzijska matrica pa tako i definicije evaluacijskih mjera se mogu poopćiti za klasifikacijske probleme u kojima se javljaju više od dvije klase (eng. *multiclass classification*).

1.3 Metode za odabir podskupa atributa

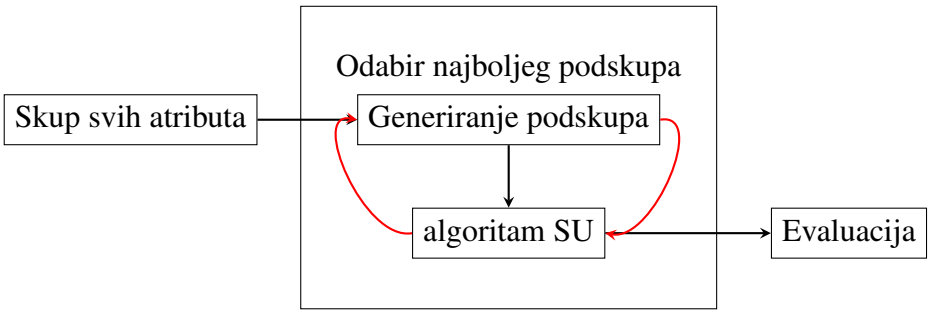
Prije opisa pčelinjih algoritama koji mogu rješavati ovaj problem, u nastavku je spomenuta osnovna podjela metoda koje se koriste u tu svrhu:

- **Filter metode**
  Podskup atributa se odabire neovisno o algoritmu strojnog učenja na temelju vrijednosti raznih statističkih testova između pojedinih atributa i ciljne varijable (Pearsonov koeficijent korelacije, χ^2 test, *mutual information* itd).



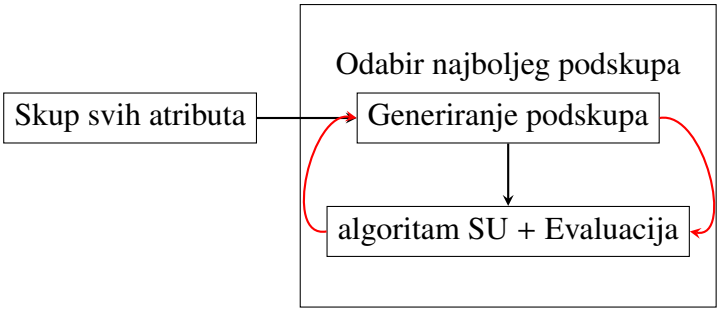
Slika 1.3: Dijagram *filter* metode

- **Wrapper metode**
  Iterativne metode koje se svode na pretraživanje podskupova atributa. Na podskupove se primjenjuje algoritam strojnog učenja i rezultat utječe na sljedeći korak. Neki algoritmi su *forward*, *backward* i *recursive feature elimination* kao i neki metaheuristički npr. genetski algoritmi, mravlji te ovdje opisani pčelinji algoritmi.



Slika 1.4: Dijagram *wrapper* metode

- **Embedded metode**
  Odabir atributa se vrši u samom procesu treniranja kod algoritama strojnog učenja koji imaju ugrađene metode za odabir (Slučajne šume, Metoda potpornih vektora, Lasso regresija itd).



Slika 1.5: Dijagram *embedded* metode

Od nabrojenih metoda, najviše problema s vremenskom složenošću imaju upravo *wrapper* metode jer se svode na pretraživanje prostora svih podskupova atributa. *Filter* metode s tim nemaju problema zbog brzog računanja statističkih mjera i testova. Također, manje je vjerojatno da ćemo uz *filter* metodu doći do pretreniranja, dok je u slučaju *wrapper* metode situacija drukčija jer se trenira model na različitim kombinacijama atributa.

Pčelinji algoritam opisan u sljedećem poglavlju pripada skupini *wrapper* metoda, a na kraju u svrhu usporedbe korištene su i neke druge od gore spomenutih.

Poglavlje 2

Pčelinji algoritmi

2.1 Meta-heuristike

Algoritmi za rješavanje optimizacijskog problema mogu biti **egzaktni**. Kada se egzaktni algoritmi ne mogu primijeniti (npr. u slučaju kada je prostor pretraživanja prevelik), tada se najčešće koriste **heuristike**. Kod heuristika nema garancije da će se pronaći optimalno rješenje, ali se pronalaze rješenja koja su dovoljno dobra tj. bliska optimumu.

Meta-heuristike su viša razina heuristika jer za razliku od njih nisu ovisne o problemu već se općenito primjenjuju na širok spektar NP-teških problema. Takvi algoritmi u iteracijama pokušavaju poboljšati kandidata za rješenje na temelju **funkcije dobrote** koja govori koliko je dobro trenutno rješenje. Rješenja moraju imati svoju **reprezentaciju** kako bi mogli vršiti akcije nad njima i računati njihovu kvalitetu. Mnoge meta-heuristike su tip **stohastičke optimizacije** zbog ubacivanja nasumičnosti odnosno korištenja nasumičnih varijabli. Ono što je izazovno za većinu meta-heuristika jest **izbjegavanje zaglavljivanja u lokalnom optimumu**.

Definicija 2.1.1 (**Optimizacijski problem**). *Neka je $\mathcal{S}$ optimizacijski skup, $\mathcal{F}$ skup dopustivih rješenja te E evaluacijska funkcija. Optimizacijski problem pronalazi $x \in \mathcal{F}$ takav da je $E(x) \leq E(y)$, za svaki $y \in \mathcal{F}$. x se naziva globalni optimum.*

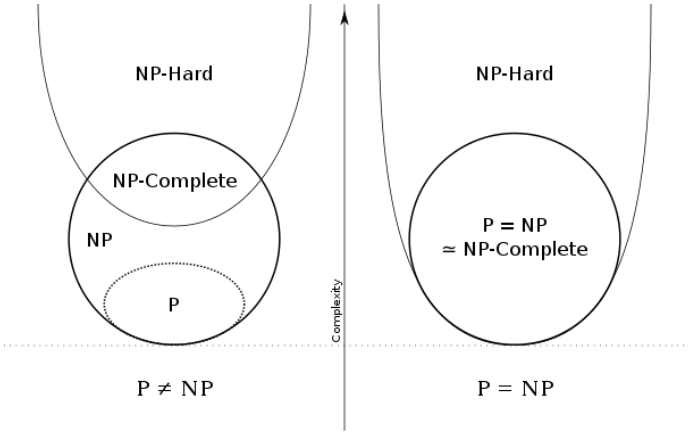
Definicija 2.1.2. *P je klasa svih jezika koji su odlučivi na nekom deterministčkom Turingovom stroju vremenske složenosti $O(n^k)$, za neki $k \in \mathbb{N}$.*

Definicija 2.1.3. *NP je klasa svih jezika koji su odlučivi na nekom nedeterminističkom Turingovom stroju vremenske složenosti $O(n^k)$, za neki $k \in \mathbb{N}$.*

Klase su definirane pomoću Turingovih strojeva u [10], no teorija izračunljivosti povezuje takve strojeve s RAM-strojevima, točnije dokazana je njihova ekvivalencija. Jednostavno govoreći, to znači da problem tj. funkcija koja se može izračunati na Turingovom

stroju, može se izračunati i na RAM-stroju i obratno. RAM-stroj je vrlo jednostavan stroj no vezan je uz današnja računala s mikroprocesorima RISC arhitekture pa se *NP*-teški problemi mogu promatrati i u kontekstu njihovog rješavanja na današnjim računalima. Formalni opis dan je u [7] i [12].

Intuitivno, **klasa *NP*-teških problema** sadrži sve probleme koji su barem jednako teški kao i problemi iz *NP*. **Klasa *NP*-potpunih problema** sadrži najteže probleme iz klase *NP*. Vrijedi da je svaki *NP*-potpun problem u *NP*, a ne mora vrijediti da je svaki *NP*-težak u *NP*. Još i danas ne postoji odgovor na pitanje je li $P = NP$. Odnosi među ovim klasama za oba slučaja, $P = NP$ i $P \neq NP$ najbolje se vide na donjoj slici[1].



Slika 2.1: Eulerov dijagram koji pokazuje odnos klasa

Drugim riječima, ako vrijedi da je $P \neq NP$ onda se *NP*-teški problemi ne mogu riješiti u **polinomijalnom vremenu**, a ako se neki problem ne može riješiti u polinomijalnom vremenu onda je njegovo izvršavanje u praksi na nekom računalu predugo i možda se ne može ni izvršiti do kraja s obzirom na ograničenja postojećih računala. Meta-heuristike pribjegavaju alternativnom pristupu rješavanja u svrhu dobivanja barem nekog dovoljno dobrog rješenja u konačnom vremenu.

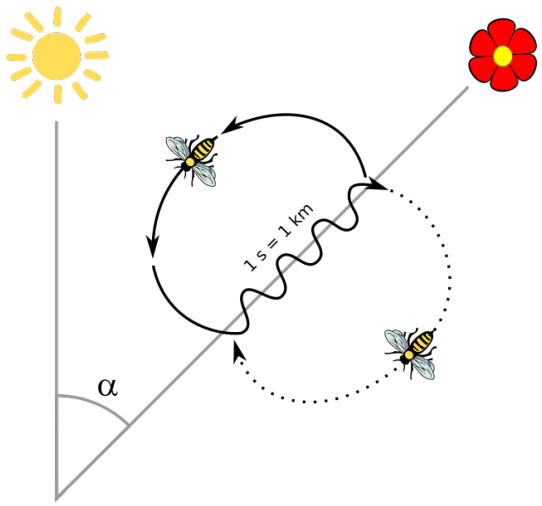
1998. Amaldi i Kann pokazuju da je problem pronalaska optimalnog podskupa atributa *NP*-težak problem. [1]

[1]Preuzeta s Wikipedije `https://commons.wikimedia.org/wiki/File:P_np_np-complete_np-hard.svg`

2.2 Opis pčelinjih algoritama

Pčelinji algoritmi su inspirirani inteligentnim ponašanjem pčela pri prikupljanju hrane, dakle to su **prirodom inspirirani algoritmi**. Budući da se promatra roj pčela, algoritmi se svrstavaju među **populacijske algoritme**. To konkretno znači da se promatra skup kandidata rješenja, a ne samo jedan tijekom izvršavanja algoritma.

Pčele pronalaze mjesta bogata hranom tako da grupa pčela odlazi na teren, a kada se vraćaju u košnicu, obavještavaju ostale pčele o pronađenoj hrani tako da plešu poseban ples koji daje informaciju gdje se nalazi hrana. Ostale pčele na temelju tog plesa odlaze na livadu na mjesta opisana tim plesom. Tako, mjesta bogata hranom bivaju posjećena od sve više pčela, a mjesta siromašna hranom od sve manje pčela. Drugim riječima, pčele pronalaze lokacije koje su bliske optimumu u smislu koncentracije hrane na određenom prostoru. Na slici[2] su prikazane osnovne informacije koje se mogu iščitati iz plesa, smjer određen kutom α te udaljenost do izvora hrane. Kvaliteta izvora hrane iščitava se iz kružnih dijelova plesa.



Slika 2.2: Pčelinji ples

Prvi algoritam zasnovan na ponašanju pčela je predstavljen 2004. godine, a u ovom radu bit će opisan algoritam *Artificial bee colony* (ABC) kojeg je predstavio Karaboga 2005. godine. [5]

[2]Preuzeta s Wikipedije `https://commons.wikimedia.org/wiki/File:Bee_dance.svg`

Algoritam ABC

Kao što je navedeno, pčelinji algoritmi su populacijski pa se tijekom algoritma promatra skup rješenja. Konkretno, u ovom algoritmu postoje tri skupine pčela. Jedna od njih su **zaposlene pčele** i svakoj od njih je pridodano točno jedno rješenje. Druge dvije skupine, **promatrači** i **izviđači** se zajedno nazivaju **nezaposlene pčele**. Svrha podjele pčela na skupine je njihov raspored poslova, a u algoritmu ABC moraju se izvršavati dva bitna zadatka, zadaci eksploatacije tj. **iskorištavanja** te **istraživanja** optimizacijskog skupa.

Zaposlene pčele i pčele promatrači zadužene su za iskorištavanje, a pčele izviđači za istraživanje izvora hrane. Na samom početku izviđači nasumično pretražuju područje. Ta rješenja se pridodaju zaposlenim pčelama, nakon čega, u idućim iteracijama, zaposlene pčele promatraju i iskorištavaju izvore u susjedstvu rješenja koja pamte. Promatrači pregledavaju njihova rješenja te na temelju kvalitete odlučuju na koje područje će se oni uputiti. Kada dođe do iskorištavanja određenog izvora, nastupa pčela izviđač koja pronalazi potpuno novo rješenje neovisno o prethodnom pretraživanju i pridodaje ga zaposlenoj pčeli s iskorištenim izvorom.

Rješenja se reprezentiraju kao konačni vektori nad realnom domenom, dakle rješenje $\vec{x}$ je oblika $(x_1, x_2, ..., x_n)$ gdje je $x_i \in \mathbb{R}$, za sve $i \in \{1, 2, ..., n\}$.

Slijedi jednostavan pseudo-kod i razrada gore navedene ideje.

Algoritam 1: ABC

```
Inicijalizacija;
dok Iteracija < MaxBrIteracija čini
    Faza zaposlenih pčela;
    Faza pčela promatrača;
    Faza pčela izviđača;
    Pamćenje dosad najboljeg rješenja;
    Povećavanje varijable Iteracija;
kraj
```

U pseudo-kodu kao **kriterij zaustavljanja** se gleda izvršavanje unaprijed određenog broja iteracija. Općenito, u meta-heurističkim algoritmima, uvjet zaustavljanja može biti i drugačije definiran, npr. pronalazak rješenja kvalitete veće od unaprijed zahtijevane vrijednosti i slično.

Prilično je jasno što se događa u linijama "Pamćenje dosad najboljeg rješenja" i "Povećavanje varijable *Iteracija*". Ostale faze su pomnije objašnjene u nastavku:

- **Inicijalizacija**
  U ovoj fazi, osim inicijalizacije veličine populacije N, broja pčela promatrača, broja iteracija $MaxBrIteracija$ te vrijednosti *limit* koja će kasnije biti pojašnjena, pčele izviđači nasumično pronalaze N rješenja.

  Neka je $\{\vec{x_1}, \vec{x_2}, ..., \vec{x_N}\}$ skup rješenja i neka je svaki vektor iz prostora dimenzije n. Sljedeća jednadžba može se koristiti za inicijalizaciju rješenja:

  $$x_{ij} = l_j + rand(0,1) * (u_j - l_j). \tag{2.1}$$

  Funkcija *rand* odabire nasumičan realan broj između 0 i 1, l_j i u_j su redom donja i gornja granica x_{ij}.

- **Faza zaposlenih pčela**
  Sada zaposlene pčele traže potencijalno bolje rješenje $\vec{v_i}$ od sadašnjeg $\vec{x_i}$ u njegovom susjedstvu

  $$v_{ij} = x_{ij} + rand(-1,1) * (x_{ij} - x_{kj}). \tag{2.2}$$

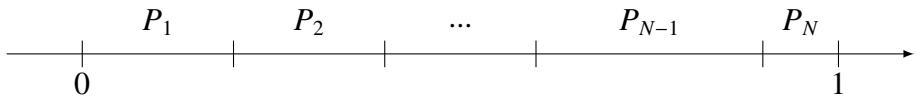
  Gornjom jednadžbom mijenjamo jednu koordinatu starog rješenja i zato se takav $\vec{v_i}$ može smatrati susjednim rješenjem $\vec{x_i}$ jer se razlikuju samo u toj nasumično odabranoj koordinati $j \in \{1, 2, ..., n\}$. Koordinata $k \in \{1, 2, ..., N\}$ je također nasumično odabrana.

  Ipak, dalje pamtimo samo bolje rješenje.

- **Faza pčela promatrača**
  Za svaku zaposlenu pčelu i njeno rješenje računamo vjerojatnost da će pčela promatrač promatrati susjedstvo upravo tog rješenja:

  $$P_i = \frac{kvaliteta(\vec{x_i})}{\sum_{j=1}^{j=N} kvaliteta(\vec{x_j})}. \tag{2.3}$$

  Očito je da je vjerojatnost veća za ona rješenja čija je kvaliteta bolja i da je zbroj svih vjerojatnosti jednak 1.



Slika 2.3: Brojevni pravac s naznačenim vjerojatnostima

Selekcija rješenja se vrši tzv. **jednostavnom proporcionalnom selekcijom**. Nasumično se odabire broj između 0 i 1 i s obzirom na segment kojemu pripada, prema gore navedenom brojevnom pravcu, bira se rješenje.

Često se ova selekcija opisuje pomoću kotača ruleta (eng. *roulette wheel selection*). Ako umjesto brojevnog pravca promotrimo krug i kružne isječke označimo vjerojatnostima, tada se nasumični odabir broja odnosno rješenja može slikovito opisati kao okretanje tog kruga (kotača ruleta) i zaustavljanje pokazivača na određenom kružnom isječku P_i.

Nakon što je rješenje odabrano, promatrač također koristi formulu 2.2 i dalje pamti samo bolje rješenje.

- **Faza pčela izviđača**
  Ranije spomenuta varijabla *limit* označava kriterij da je neki izvor hrane iskorišten, tj. **kriterij napuštanja** (eng. *abandonment criteria*).

  Ako se rješenje neke zaposlene pčele nije promijenilo u *limit* iteracija onda ona napušta to rješenje, a pčela izviđač traži potpuno novo rješenje po formuli 2.1.

Mogući problemi su u usklađivanju slobodnih parametara, vremenska i prostorna složenost, a i inicijalizacija na slučajan način ograničava prostor pretraživanja. Ipak, ovaj algoritam se ističe po svojoj jednostavnosti, a prednost mu je još dobro izbjegavanje zaglavljivanja u lokalnom optimumu zbog balansiranja iskorištavanja i istraživanja.

Poglavlje 3

Implementacija

U ovom poglavlju je objašnjeno kako je iskorištena gore navedena teorija. Čitav kod[1] je napisan u programskom jeziku Python. Osim standardnih biblioteka (*numpy*, *pandas*, *scipy*) korištene su i *matplotlib* i *seaborn* za vizualizacije te *sklearn* za algoritme strojnog učenja.

Specifikacije računala su:

- procesor Intel i5 frekvencije 2.40 GHz
- radna memorija od 8 GB
- 32-bitni operacijski sustav Windows.

3.1 Skupovi podataka

Efikasnost algoritma se promatrala na ukupno tri skupa podataka:

- **skup podataka Cleveland Heart Disease**[2]

  Pojedini primjer predstavlja pacijenta i sadrži informaciju o prisutnosti srčane bolesti u pacijenta.

  - Broj primjera: 303
  - Broj atributa: **13** + 1

[1] Kod je pohranjen u javnom GitHub repozitoriju u Jupyter bilježnicama: `https://github.com/pulucij/Diplomski-rad`

[2] UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`

  - Opis atributa:

    Prva dva atributa su općeniti podaci o pacijentu (dob i spol). Ostali atributi vežu se uz mjerenja karakteristična za srčane bolesnike (bol, vrijednosti nalaza krvi, EKG mjerenja). Zadnji atribut predstavlja zavisnu varijablu koju treba predvidjeti za novi podatak. Pacijent kod kojeg nije prisutna bolest ima vrijednost atributa 0, vrijednosti 1-4 predstavljaju prisutnost srčane bolesti od najmanje do najveće.

- **skup podataka Breast Cancer Wisconsin**[3]

  Pojedini primjer predstavlja pacijenta s tumorom dojke te sadrži informaciju je li tumor maligan ili benigan.

  - Broj primjera: 569
  - Broj atributa: 1 + 1 + **30**
  - Opis atributa:

    Prvi atribut je ID pacijenta kojeg treba izbaciti iz promatranja jer ID nikako ne utječe na benignost tumora. Drugi atribut je zavisna varijabla koju treba predvidjeti i ima vrijednosti ”M” za pacijenta s malignim tj. ”B” s benignim tumorom. Ostali atributi su mjerenja vezana za staničnu jezgru stanica promatrane tvorevine vidljive na snimci dojke.

- **skup podataka Drug consumption**[4]

  Pojedini primjer predstavlja ispitanika na kojemu je vršeno mjerenje osobnosti te sadrži informaciju o konzumiranju alkohola, čokolade, nikotina, kofeina, kanabisa i teških droga.

  - Broj primjera: 1885
  - Broj atributa: 1 + **12** + 1 + 18
  - Opis atributa:

    Prvi atribut je ID ispitanika i njega treba izuzeti iz daljnjeg promatranja. U nastavku je 5 općenitih informacija (dob, spol, obrazovanje, država prebivališta, narodnost). Slijede atributi vezani za psihološka i neurološka mjerenja. Ostalih 19 atributa su zavisne varijable koje se zasebno mogu predvidjeti za novi podatak. U ovom istraživanju, promatrana je samo jedna zavisna varijabla tj. konzumiranje samo jedne droge. Vrijednosti zavisnih varijabli su: ”Never Used”,

[3] UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[4] UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Drug+consumption+%28quantified%29

”Used over a Decade Ago”, ”Used in Last Decade”, ”Used in Last Year”, ”Used in Last Month”, ”Used in Last Week” i ”Used in Last Day”.

3.2 Priprema podataka

Priprema podataka mora prethoditi primjeni algoritama iz strojnog učenja kako bi se otklonile ili popravile neke nepravilnosti u podacima i time izbjegli nepouzdani rezultati i ostali daljnji problemi. Budući da se pčelinji algoritam koristi za odabir značajki te kako se unutar njega koriste klasifikacijski modeli, bitno je provesti postupke sređivanja podataka. Osim toga, korisno je promotriti podatke i steći bolje razumijevanje o njima kako bismo bolje shvatili i krajnje rezultate te benefit korištenja algoritma.

Uvođenjem skupa podataka u Python program, prvo su promatrani podaci i koji su njihovi tipovi, a uobičajeno je da se izvrši prilagodba testnih podataka. Recimo, ako se neka kontinuirana numerička varijabla vodi kao tip `object`, logično je da ju se mora prebaciti u neki tip oblika `float64` kako bismo mogli koristiti algoritme. U navedenim skupovima podataka nije bilo problema takve vrste. Također, promatrano je koji su atributi **kategorički** (vrijednosti su im iz konačnog skupa oznaka kategorija), a koji **kontinuirani** (vrijednosti su im brojevi iz određenog intervala). Potrebno je kategoričke atribute prebaciti u brojevnu vrijednost ako su prikazani kao stringovi, a to se rješavalo samo kod zavisnih varijabli u dva skupa podataka pomoću metode `LabelEncoder`.

Nedostajući podaci

U skupovima podataka može se dogoditi da za neke primjere nedostaju vrijednosti nekih atributa. Razlog može biti da je takav podatak nepoznat, zagubljen ili pak praznina označava neku vrijednost kao što je 0.

U navedena tri skupa, nedostajući podaci su se javili samo u skupu Heart Disease i svi pripadaju atributima tipa realni broj. Problem je riješen tako da su praznine zamijenjene srednjom vrijednosti svih vrijednosti određenog atributa.

Balansiranost skupova

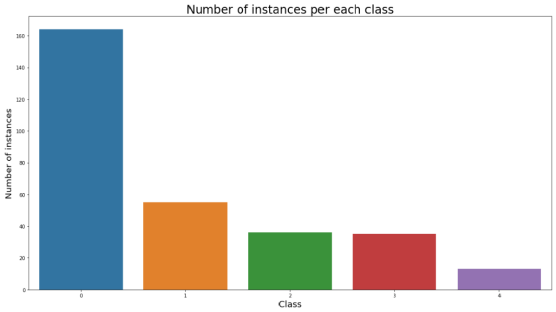
Što uopće znači da je skup podataka balansiran? Najjednostavnije rečeno, skup podataka je balansiran ako je raspodjela primjera iz skupa podataka po klasama uravnotežena tj. ako se podjednak broj primjera nalazi u svakoj pojedinoj klasi.

Razlog zašto je bitno kod klasifikacijskih problema provjeriti balansiranost je taj da koristeći najpopularniju evaluacijsku mjeru točnost, možemo doći do lažno dobrih rezultata.

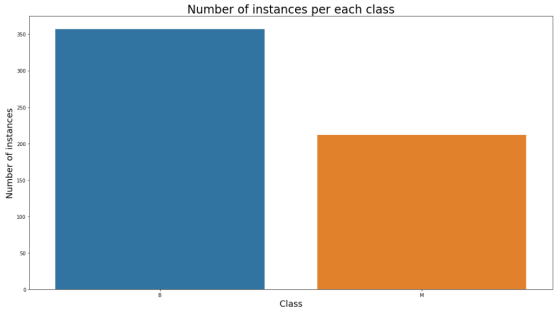
Takva pojava se zove **paradoks točnosti**. Naime, ako je većina primjera u skupu podataka u jednoj klasi, po formuli točnosti automatski ćemo dobiti bolje rezultate nego što to zapravo jesu.

Tom problemu se može doskočiti na više različitih načina. Na primjer, ako je moguće, mogu se prikupiti dodatni primjeri za malobrojne klase. Ako je velik skup podataka, mogu se izbaciti neki primjeri iz klasa gdje ih je "previše", a isto tako mogu se dodati i neki umjetno proizvedeni podaci. Najčešće korištena tehnika koja proizvodi nove podatke je SMOTE (*Synthetic Minority Over-sampling Technique*). Ipak, budući da rješavanje ovog problema nije u fokusu diplomskog rada, pribjeglo se jednostavnijem načinu rješavanja problema koji nikako ne izmjenjuje skup podataka već se koristi drukčija evaluacijska mjera. U odjeljku 1.2 je navedeno više evaluacijskih mjera, a balansirana točnost 1.2 je mjera koja neće davati varljive rezultate u slučaju primjene klasifikacijskih modela na nebalansiranim skupovima podataka.

Na grafovima[5] je prikazan raspored primjera iz skupa podataka o srčanim bolesnicima te o pacijentima s rakom dojke. Očigledno je da ni jedan ni drugi skup podataka nisu balansirani. Ima puno više pacijenata kod kojih srčana bolest nije uopće prisutna, a ostali pacijenti su se razdijelili u klase 1-4. Isto tako, veći je broj pacijenata s benignim tumorom dojke.



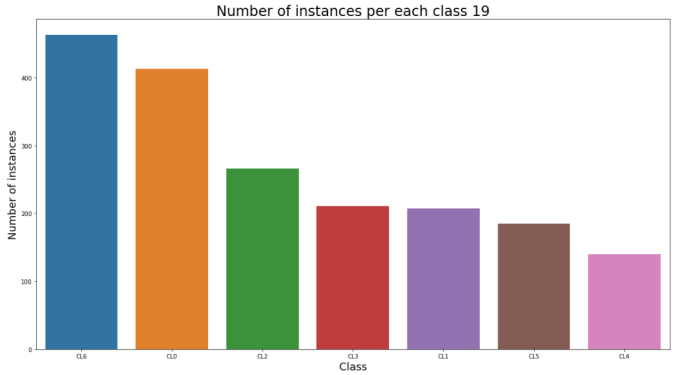
Slika 3.1: Raspodjela srčanih bolesnika po klasama



Slika 3.2: Raspodjela pacijenata s rakom dojke po klasama

Što se tiče skupa podataka o konzumiranju droga, već je ranije navedeno da se promatrala konzumacija samo jedne od droga navedenih u skupu podataka. Odabrana je ona za koju je podjela po klasama bila najviše uravnotežena, a ispostavilo se da je to kanabis.

[5]Grafovi su napravljeni u Pythonu pomoću biblioteke matplotlib.



Slika 3.3: Raspodjela ispitanika o konzumaciji kanabisa po klasama

Vidljivo je da se ni za jedan od tri korištena skupa ne može reći da je balansiran, čak ni posljednji iako se birao najpogodniji slučaj u tom kontekstu. To opravdava korištenje balansirane točnosti.

Netipične vrijednosti (eng. *outliers*)

Netipične vrijednosti predstavljaju one primjere u skupu podataka koji jako odskaču od ostatka. Neki modeli strojnog učenja osjetljivi su na prisutnost takvih podataka te se ne mogu izvršiti ispravno i dati točne rezultate. Stoga, valja proanalizirati postojanje takvih primjera te ih po potrebi ukloniti iz skupa.

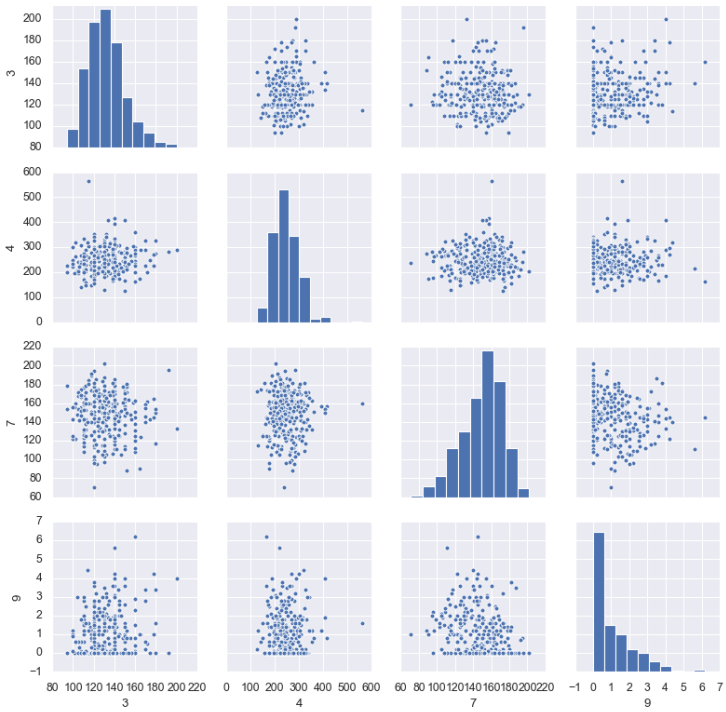
Za utvrđivanje egzistencije takvih vrijednosti u spomenutim skupovima korišteni su kvartili svih vrijednosti u pojedinom atributu te interkvartilni raspon.

Donji kvartil predstavlja vrijednost iz promatranog skupa vrijednosti od koje je manje ili jednako njoj 25 % vrijednosti, a istodobno 75 % vrijednosti veće ili jednako njoj. **Gornji kvartil** je vrijednost iz promatranog skupa vrijednosti od koje je manje ili jednako njoj 75 % vrijednosti, a istodobno je 25 % vrijednosti veće ili jednako njoj. **Interkvartilni raspon** je razlika između gornjeg i donjeg kvartila.

Pomoću definiranih vrijednosti se računaju granice kao donji kvartil - 1.5 $*$ interkvartilni raspon te gornji kvartil + 1.5 $*$ interkvartilni raspon. Sve vrijednosti iznad te ispod tih granica su sumnjive vrijednosti koje treba uzeti u obzir i dalje promotriti. Budući da smo u ovoj proceduri analizirali svaki atribut pojedinačno, takvu analizu svrstavamo u **univarijatnu analizu**. Da budemo detaljniji, dobro je uvrstiti i **multivarijatnu analizu** gdje promatramo odnose između više atributa.

U tu svrhu korišteni su grafovi[6] koji prikazuju sve parove danih atributa.

[6]Grafovi imena pairplot iz biblioteke seaborn.



Slika 3.4: Grafički prikazani odnosi parova atributa u skupu srčanih bolesnika

Nije potrebno promatrati baš sve parove atributa, već je izbor sužen na one atribute koji sadrže neke vrijednosti koje su ispod donje ili iznad gornje granice definirane u univarijatnoj analizi. Iz ovako grafičkog prikaza neki podaci se mogu odrediti kao kandidati za netipične vrijednosti. Recimo podatak s najvećom vrijednosti na x-osi u stupcu atributa 4 te npr. dvije najveće vrijednosti na x-osi u stupcu atributa 9.

U fazi pripreme podataka, kandidati za netipične vrijednosti nisu uklonjeni iz skupa već je prvo algoritam primijenjen na cjelovitom skupu. Nakon toga su uklonjeni kandidati te se promatralo dobiva li se poboljšanje. Ipak, ni u jednom od tri skupa nije došlo do poboljšanja pa su krajnji rezultati vezani za skupove iz kojih nisu uklonjeni kandidati za outliere koji ipak to možda nisu već jednostavno prate određenu raspršenost podataka.

Važnost obavljanja ovog koraka ne opada ovako dobivenim opažanjem jer u nekim skupovima podataka i primjenom nekih modela zaista ne možemo biti sigurni da ćemo dobiti dobre rezultate bez analiziranja i otkrivanja netipičnih vrijednosti.

Skaliranje podataka

Budući da model k-najbližih susjeda koristi udaljenosti između podataka u skupu, bolje je skalirati podatke. U vrijednostima atributa mogu postojati velike razlike. Neki atributi

mogu imati vrlo male vrijednosti, dok neki mogu imati vrlo visoke i to utječe na određene aspekte primjenjivanih algoritama u strojnom učenju.

Skaliranje[7] je napravljeno na svakom atributu zasebno po sljedećoj formuli:

$$z = (x - u)/s$$

gdje je x neskalirana vrijednost atributa, u je srednja vrijednost svih vrijednosti atributa $\{x_1, x_2, ..., x_N\}$ te je s **standardna devijacija** svih vrijednosti atributa

$$s = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - u)^2}.$$

Analiza povezanosti nezavisnih varijabli s ciljnom

Kako bi se dobio uvid u povezanost atributa s ciljnom varijablom, mogu se koristiti statističke mjere koje daju informaciju o tome. To može biti početni korak u odabiru podskupa atributa. Može se doznati postoje li neke značajke koje su vrlo slabo vezane uz zavisnu varijablu i obrnuto. Za mjerenje povezanosti između dvije kontinuirane numeričke varijable gotovo uvijek se koristi Pearsonov koeficijent korelacije i povezanost između varijabli se jednostavno naziva koreliranost. U klasifikacijskom problemu, zavisna varijabla je kategorička, pa se u ovom kontekstu koriste drukčije mjere koje rade s kombinacijama varijabli kategorička-kategorička te kontinuirana-kategorička.

Za mjerenje povezanosti između dvije kategoričke varijable moguće je koristiti Kramerov V koeficijent. Za objašnjenje kako se računa taj koeficijent, potrebna je sljedeća definicija.

Definicija 3.2.1 (**χ^2-statistika**). *Neka su X i Y promatrani kategorički atributi i neka X i Y imaju r i k vrijednosti redom. Neka je n_{ij} broj pojavljivanja para vrijednosti (X_i, Y_j) u skupu podataka. χ^2-statistika se definira kao*

$$\chi^2 = \sum_{i\in\{1,...,r\}, j\in\{1,...,k\}} \frac{(n_{ij} - \frac{n_i n_j}{n})^2}{\frac{n_i n_j}{n}}$$

gdje je n ukupan broj primjera u skupu i n_i broj pojavljivanja i-te vrijednosti atributa X te n_j broj pojavljivanja j-te vrijednosti atributa Y.

Sada se može definirati formula po kojoj se računa **Kramerov V koeficijent**:

$$V = \sqrt{\frac{\frac{\chi^2}{n}}{min(k-1, r-1)}}.$$

[7] StandardScaler iz sklearn biblioteke.

Za mjerenje povezanosti između kontinuirane varijable i kategoričke varijable korišten je **korelacijski omjer** (eng. *correlation ratio*).

Definicija 3.2.2. *Neka je y_{xi} i-ti primjer iz skupa podataka koji pripadaju klasi x te neka je n_x kardinalitet tog skupa. Korelacijski omjer η zadovoljava sljedeću jednakost*

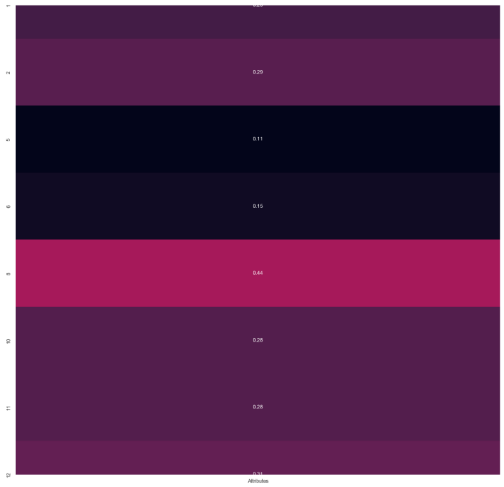
$$\eta^2 = \frac{\sum_x n_x(\overline{y_x} - \overline{y})^2}{\sum_{x,i} n_x(y_{xi} - \overline{y})^2} \tag{3.1}$$

gdje su

$$\overline{y_x} = \frac{\sum_i y_{xi}}{n_x}, \qquad \overline{y} = \frac{\sum_x n_x \overline{y_x}}{\sum_x n_x}.$$

η nije definiran ako svi podaci pripadaju istoj klasi.

Vrijednosti definiranih mjera povezanosti se kreću od 0 do 1, gdje vrijednosti bliske 0 označavaju malenu, a vrijednosti bliske 1 veliku povezanost.

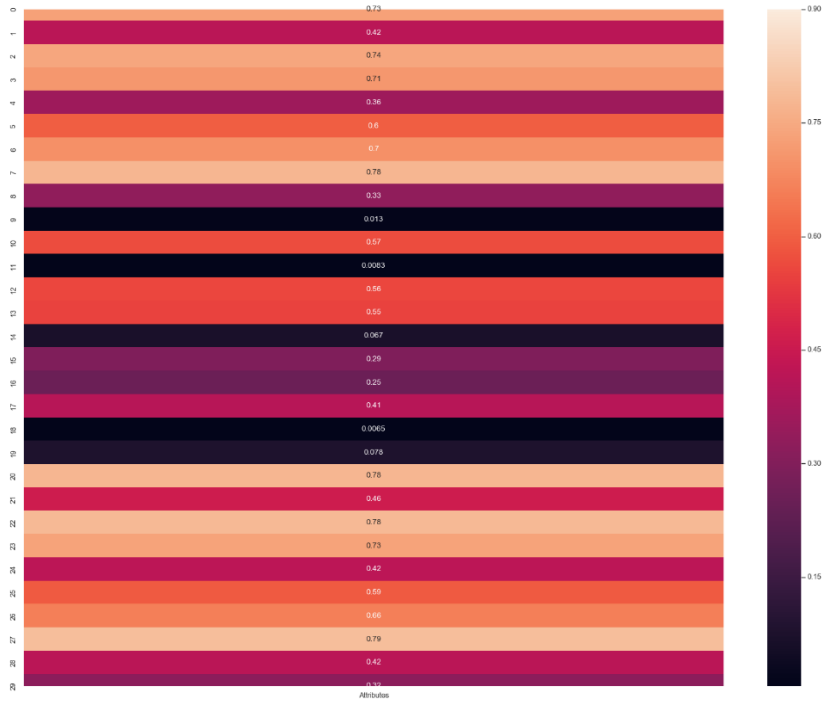


Slika 3.5: Povezanost kategoričkih nezavisnih varijabli sa zavisnom u skupu CHD

Slika 3.6: Povezanost kontinuiranih nezavisnih varijabli sa zavisnom u skupu CHD

Na grafovima, bojama su prikazane povezanosti između nezavisnih i zavisne varijable u skupu podataka vezanom za srčane bolesnike. Tamne boje označavaju nisku, a svijetle visoku povezanost. Očigledno je da ovakva analiza pokazuje da niti jedan od atributa nije visoko povezan. Kod skupa koji sadržava osobe koje boluju od raka dojke je situacija drukčija. Vidi se da su neki atributi dosta povezani s ciljnom varijablom. U tom skupu su sve varijable kontinuirane.

Ova analiza će biti iskorištena kasnije u odlomku 3.4.



Slika 3.7: Povezanost kontinuiranih nezavisnih varijabli sa zavisnom u skupu BCW

Razdvajanje skupa podataka na testni skup i skup za trening

Kako bi se mogla izvršiti evaluacija klasifikacijskog modela, potrebno je imati skup na kojemu se trenira model te skup na kojemu se provjerava uspješnost istreniranog modela.

Korišten je najjednostavniji pristup u kojemu se skup podataka dijeli samo na dva skupa te se provjera uspješnosti izvršava samo jednom. Postoje i kompliciranije metode kao što su tipovi tzv. *cross*-validacije u kojima se skup particionira na više dijelova te se evaluacija izvršava u više koraka. No, u ovako promatranom algoritmu, zadovoljavajuće je koristiti jednostavni pristup.

U skupu podataka vezanom za srčane bolesti je razdijeljeno 15 % primjera u testni skup, a ostalih 85 % u skup za trening. Isto je napravljeno za skup pacijenata s rakom dojke, a budući da je skup podataka o konzumiranju droga veći, u testni skup je pripalo 20 % primjera.

3.3 Implementacija pčelinjeg algoritma

Ranije je općenito opisan algoritam *Artificial bee colony*, a u ovom odjeljku se opisuje kako je taj algoritam korišten na konkretnom problemu opisanom u 1.1.

Označimo s n broj atributa u skupu podataka. Rješenje se može reprezentirati kao konačan niz $\vec{x}$ od n nula ili jedinica. Ako je $x_i = 0$ tada se i-ti atribut ne uključuje u podskup atributa, a uključuje se ako je $x_i = 1$. Dakle, skup dopuštenih rješenja je skup svih takvih vektora, a njegov kardinalitet je $2^n - 1$ (ne ubrajamo niz koji se sastoji samo od nula tj. da bi skup podataka imao smisla, mora postojati bar jedan uključeni atribut). Takav skup postaje jako velik čim n poprimi dvoznamenkastu vrijednost.

Primijetimo da je prostor pretraživanja diskretan, a algoritam je opisan za kontinuirane vrijednosti. To ne predstavlja nikakav problem. Potrebne su samo manje prilagodbe i takva inačica algoritma se naziva algoritam *Binary artificial bee colony* (BABC). Slijedi njegov dodatan opis po fazama:

- **Inicijalizacija**

  Kao što je navedeno u 2.2, inicijalizira se veličina populacije N, broj pčela promatrača, $MaxBrIteracija$, $limit$. Pčele izviđači pronalaze N nasumičnih rješenja tj. stvara se N konačnih nizova duljine n i to na način da se $\forall i \in \{1, ..., n\}$ bira nasumični broj između 0 i 1. Ako je nasumični broj manji ili jednak vrijednosti 0.5 onda je i-ta koordinata niza jednaka 0, a u suprotnom je jednaka 1.

- **Faza zaposlenih pčela**

  Pojedina zaposlena pčela traži potencijalno bolje rješenje od sadašnjeg tako da odabere nasumično dvije koordinate u rješenju te ih zamijeni. Dalje se pamti samo bolje rješenje, tj. ako se zamjenom ne dobije bolje rješenje, pamti se staro, a ako da, pamti se novo.

- **Faza pčela promatrača**

  Koristi se opisana jednostavna proporcionalna selekcija rješenja. Nakon toga se opet konstruira susjedno rješenje kako je opisano u prethodnoj točki te se dalje pamti bolje rješenje.

- **Faza pčela izviđača**

  Ako se rješenje zaposlene pčele nije promijenilo u *limit* iteracija (kriterij napuštanja), onda ona napušta to rješenje, a pčela izviđač traži novo rješenje na nasumični način kako je opisano u prvoj točki.

Funkcija dobrote je zapravo evaluacija primijenjenog klasifikacijskog modela na skup podataka s uključenim određenim atributima i ta procedura se ponavlja nekoliko puta prilikom izvršavanja faza. Dolje je naveden kod tj. funkcija koja za argumente uzima konačan

niz nula i jedinica odnosno jedno rješenje, testni skup i skup za trening te ciljne varijable iz testnog skupa i skupa za trening, oznaku koji će se klasifikacijski model koristiti i vrijednost *n* koja opisuje broj susjeda za algoritam KNN ili broj stabala ako koristimo model slučajnih šuma. U funkcijama `KNN_Classifier`, `RF_Classifier`, `NB_Classifier` su istrenirani i primijenjeni modeli. Na kraju, funkcija vraća balansiranu točnost, što je ujedno i povratna vrijednost funkcije dobrote.

```
def calculate_accuracy(bee, X_train, X_test, y_train, y_test, algorithm,
    n):
    #count number of features
    k=0
    for j in range(0,len(bee)):
        if bee[j]==1:
            k+=1
    X_train_features = np.zeros((X_train.shape[0], k))
    X_test_features = np.zeros((X_test.shape[0], k))
    k=0
    for j in range(0,len(bee)):
        if bee[j]==1:
            X_train_features[:,k]=X_train[:,j]
            X_test_features[:,k]=X_test[:,j]
            k+=1

    if(algorithm == 'KNN'):
        y_pred = KNN_Classifier(X_train_features, X_test_features,
   y_train, y_test, n)
    elif(algorithm == 'RF'):
        y_pred = RF_Classifier(X_train_features, X_test_features,
   y_train, y_test, n)
    elif(algorithm == 'NB'):
        y_pred = NB_Classifier(X_train_features, X_test_features,
   y_train, y_test)
    else:
        raise ValueError('Key word is not recognized.')

    return balanced_accuracy_score(y_test, y_pred)
```

Također, treba voditi računa o broju iteracija u kojima se pojedino rješenje nije promijenilo. To je ostvareno tako da se na početku svake iteracije poveća broj vezan za određeno rješenje, a čim promijenimo to rješenje, taj broj postavimo na nulu. Konkretno, te vrijednosti vezane za svaku pojedinu zaposlenu pčelu pohranjene su u jednom nizu.

U implementaciji algoritma se koriste nizovi, liste i matrica te se često vrše operacije nad njima, a to utječe na prostornu i vremensku složenost pogotovo korištenjem većih vrijednosti parametara u pčelinjem algoritmu.

Prijašnje primjene ovog algoritma su opisane u člancima [9], [11] i [8].

Nakon primjene algoritma BABC, primijenjene su još neke osnovne metode za odabir podskupa atributa, a krajnji rezultati svake od njih služili su za usporedbu i procjenu kolika je uspješnost pčelinjeg algoritma. U 1.3 je napravljena općenita podjela i opis, a u nastavku su detalji.

3.4 Druge metode za usporedbu

Ranije je opisano da postoje tri skupine ovakvih metoda, pa je u ovom radu korištena po jedna iz svake skupine.

- **Filter metode**

  Ovdje se koriste rezultati analize u kojoj je računata povezanost između nezavisnih atributa i ciljne varijable. Atributi i njihove vrijednosti povezanosti su pohranjeni u `DataFrame` i sortirani po vrijednostima od najveće prema najmanjoj. Primjenjivana su i evaluirana sva tri modela prvo na skup podataka samo s jednom najpovezanijom varijablom, zatim s dvije najpovezanije itd. Naposljetku su uključeni svi atributi. U ovoj metodi, bira se onaj podskup atributa sačinjen od prvih k najpovezanijih atributa za koje se dobije najbolji rezultat evaluacije.

- **Wrapper metode**

  Već je spomenuto da primjenjivani pčelinji algoritam spada upravo u ovu kategoriju. Dodatno je promatrana metoda rekurzivne eliminacije značajki (eng. *recursive feature elimination*) i to u kombinaciji s modelom slučajnih šuma jer on jedini od tri navedena pridodaje važnosti atributima prilikom treniranja na skupu podataka. Ova metoda je iterativna. Prvo se promatra cijeli skup podataka, primjeni se model i evaluira, a nakon tog se izbaci najlošiji atribut te se ponavlja postupak i tako sve dok se ne dođe do unaprijed određenog broja atributa kojeg želimo, a taj je skup tada konačni traženi podskup. Ova metoda je primjenjivana onoliko puta koliko ima atributa u skupu. Dakle, unaprijed određeni traženi broj varijabli je prvo bio jedan, pa dva i tako dalje.

- **Embedded metode**

  Kao i u prošloj točki, ova metoda je korištena u kombinaciji s modelom *Random forest* jer on jedini pridružuje važnosti atributima, a te vrijednosti se koriste i kod ovakvog tipa metoda. Prvo se istrenira i primjeni model na cjelovitom skupu podataka te se zapamte važnosti koje je model odredio. Zatim se na određeni način

bira broj atributa s najvećim važnostima koji će pripasti u podskup. Budući da primjena modela i evaluacija ne traju predugo niti za jedan od skupova, model je opet primjenjivan onoliko puta koliko ima atributa u skupu podataka.

Može se primijetiti da se u sve tri metode postupalo tako da se metode primjenjuju iterativno po broju atributa. Razlog je da se provjere svi mogući slučajevi podskupova koji se mogu dobiti ovim metodama, a ne da se nasumično odabire broj atributa koji želimo u podskupu čime bi nam neki dobar rezultat mogao promaknuti.

Poglavlje 4

Rezultati

Cijeli postupak je primijenjen posebno na svaki od skupova podataka pa je najbolje i rezultate i vrijednosti parametara prikazati zasebno za svaki od njih.

Parametri su određeni eksperimentalno i navedeni su oni za koje su se postizali najbolji rezultati. Kod nekih modela, već za parametre manjih vrijednosti se postižu bolji rezultati pa se zato oni razlikuju i navode posebno za svaki model.

U tablicama je navedena najveća vrijednost balansirane točnosti, dobivena nakon tri izvođenja, pomnožena sa sto.

- **Cleveland Heart Disease**

  Parametri vezani za algoritam BABC uz primjenu modela k-najbližih susjeda:

  - Broj zaposlenih pčela: 12
  - Broj pčela promatrača: 12
  - $MaxBrIteracija$: 200
  - *limit*: 10.

  Parametri vezani za algoritam BABC uz primjenu naivnog Bayesovog modela:

  - Broj zaposlenih pčela: 12
  - Broj pčela promatrača: 12
  - $MaxBrIteracija$: 200
  - *limit*: 10.

Parametri vezani za algoritam BABC uz primjenu modela slučajnih šuma:

- Broj zaposlenih pčela: 12
- Broj pčela promatrača: 12
- $MaxBrIteracija$: 100
- *limit*: 10.

Metoda za odabir podskupa atributa	**Klasifikacijski model**	**Br. susjeda/ br. stabala**	**Br. atributa**	**Rezultat**
BABC	KNN	5	8/13	63.60 %
BABC	NB	/	9/13	57.06 %
BABC	RF	10	7/13	**66.61 %**
Filter	KNN	5	13/13	42.83 %
Filter	NB	/	5/13	33.74 %
Filter	RF	100	9/13	32.38 %
RFE	RF	100	2/13	50.07 %
Embedded	RF	100	11/13	34.97 %

Tablica 4.1: Rezultati primjene na skup podataka CHD

Podebljan je najbolji rezultat od 66.61 % koji se postiže pčelinjim algoritmom kombiniranim s modelom 5-najbližih susjeda.

Broj stabala kod ostalih metoda za odabir atributa u tablici je povećan na 100 kako bi se pokazalo da se i povećanjem tog parametra ne dolazi do boljeg rezultata no što se dobije upravo pčelinjim algoritmom. To je napravljeno i za druga dva skupa podataka.

- **Breast Cancer Wisconsin**

  Parametri vezani za algoritam BABC uz primjenu modela k-najbližih susjeda:

  - Broj zaposlenih pčela: 5
  - Broj pčela promatrača: 5
  - $MaxBrIteracija$: 50
  - *limit*: 10.

Parametri vezani za algoritam BABC uz primjenu naivnog Bayesovog modela:

- Broj zaposlenih pčela: 12
- Broj pčela promatrača: 12
- $MaxBrIteracija$: 200
- *limit*: 10.

Parametri vezani za algoritam BABC uz primjenu modela slučajnih šuma:

- Broj zaposlenih pčela: 5
- Broj pčela promatrača: 5
- $MaxBrIteracija$: 30
- *limit*: 10.

Metoda za odabir podskupa atributa	**Klasifikacijski model**	**Br. susjeda/ br. stabala**	**Br. atributa**	**Rezultat**
BABC	KNN	5	13/30	**100.00 %**
BABC	NB	/	12/30	**100.00 %**
BABC	RF	10	15/30	**100.00 %**
Filter	KNN	5	18/30	98.00 %
Filter	NB	/	4/30	95.22 %
Filter	RF	100	18/30	97.61 %
RFE	RF	100	14/30	97.61 %
Embedded	RF	100	17/30	97.61 %

Tablica 4.2: Rezultati primjene na skup podataka BCW

Primjenom algoritma na ovaj skup podataka koji je vezan za binarni klasifikacijski problem se dobiju općenito dobri rezultati, a zanimljivo je da se uz korištenje pčelinjeg algoritma u sva tri slučaja dobije čak stopostotna točnost.

- **Drug Consumption**

  Parametri vezani za algoritam BABC uz primjenu modela k-najbližih susjeda:

  - Broj zaposlenih pčela: 12
  - Broj pčela promatrača: 12
  - $MaxBrIteracija$: 70

– *limit*: 10.

Parametri vezani za algoritam BABC uz primjenu naivnog Bayesovog modela:

– Broj zaposlenih pčela: 12
– Broj pčela promatrača: 12
– $MaxBrIteracija$: 70
– *limit*: 10.

Parametri vezani za algoritam BABC uz primjenu modela slučajnih šuma:

– Broj zaposlenih pčela: 12
– Broj pčela promatrača: 12
– $MaxBrIteracija$: 50
– *limit*: 10.

Metoda za odabir podskupa atributa	**Klasifikacijski model**	**Br. susjeda/ br. stabala**	**Br. atributa**	**Rezultat**
BABC	KNN	7	8/12	31.48 %
BABC	NB	/	8/12	31.32 %
BABC	RF	10	8/12	**32.11 %**
Filter	KNN	7	5/12	27.29 %
Filter	NB	/	12/12	28.57 %
Filter	RF	100	12/12	28.90 %
RFE	RF	100	12/12	28.90 %
Embedded	RF	100	12/12	28.90 %

Tablica 4.3: Rezultati primjene na skup podataka DC

U ovom slučaju je također pčelinji algoritam uz primjenu naivnog Bayesovog modela bio najuspješniji.

Poglavlje 5

Zaključak

Glavno opažanje je da se korištenjem opisanog binarnog pčelinjeg algoritma u kombinaciji s proizvoljnim, ranije spomenutim klasifikacijskim modelom, postižu bolji rezultati nego rezultati dobiveni korištenjem bilo kojeg od ostalih promatranih algoritama. Iz tog razloga se može zaključiti da takav algoritam može biti itekako koristan u rješavanju problema pronalaska podskupa atributa.

Tijekom rada je navedeno nekoliko mana pčelinjih algoritama, uključujući problem usklađivanja slobodnih parametara, inicijalizaciju na slučajan način koja ograničava prostor pretrage te vremensku i prostornu složenost.

U ovom kontekstu, najviše se ističe veća vremenska složenost jer se algoritam izvršava osjetno duže nego uobičajeni algoritmi. Moguća poboljšanja vremenske složenosti te slična poboljšanja nisu uključena u istraživanje jer je naglasak postavljen na tzv. *benchmarking* odnosno na vrednovanje algoritma uspoređujući ga sa standardnim algoritmima.

Bibliografija

[1] E. Amaldi i V. Kann, *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*, Theoretical Computer Science (1998), br. 209, 237–260, https://www.sciencedirect.com/science/article/pii/S0304397597001151?via%3Dihub.

[2] L. Breiman, *Bagging Predictors*, Machine Learning (1996), br. 24, 123–140, https://link.springer.com/content/pdf/10.1007/BF00058655.pdf.

[3] ———, *Random Forests*, Machine Learning (2001), br. 45, 5–32, https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf.

[4] I. Guyon i A. Elisseeff, *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research (2003), br. 3, 1157–1182, http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf.

[5] D. Karaboga, *An idea based on honey bee swarm for numerical optimization*, TECHNICAL REPORT-TR06 (2005), https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf.

[6] R. Kohavi i G. H. John, *Wrappers for feature subset selection*, Artificial Intelligence (1997), br. 97, 273–324, https://ai.stanford.edu/~ronnyk/wrappersPrint.pdf.

[7] Christos H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, Inc., 1994.

[8] M. Schiezaro i H. Pedrini, *Data feature selection based on Artificial Bee Colony algorithm*, EURASIP Journal on Image and Video Processing (2013), br. 47, https://link.springer.com/article/10.1186/1687-5281-2013-47%20.

[9] B. Subanya i R R. Rajalaxmi, *A Novel Feature Selection Algorithm for Heart Disease Classification*, International Journal of Computational Intelligence and Informatics **4** (2014), br. 2, https://www.semanticscholar.org/

paper/A-Novel-Feature-Selection-Algorithm-for-Heart-Subanya/9438186b537468c4aca24817b25cbec3b0fbccbf.

[10] M. Vuković, *Složenost algoritama*, Sveučilište u Zagrebu PMF-Matematički odsjek, 2019, https://www.math.pmf.unizg.hr/sites/default/files/pictures/sa-skripta-2019.pdf.

[11] Z. Yilmaz i F. Basciftci, *Binary Artificial Bee Colony Algorithm to Solve Single Objective Resource Allocation Problem*, International Journal of Future Computer and Communication **7** (2018), br. 1, http://www.ijfcc.org/vol7/514-T026.pdf.

[12] V. Čačić, *Komputonomikon*, Sveučilište u Zagrebu PMF-Matematički odsjek, 2020, https://web.math.pmf.unizg.hr/~veky/izr/Komputonomikon.pdf.

Sažetak

Glavni cilj ovog diplomskog rada je objasniti kako koristiti meta-heuristiku imena *Artificial Bee Colony* u rješavanju problema odabira podskupa atributa u disciplini strojnog učenja. Procedura je opisana i primijenjena na neke standardne skupove podataka koji se inače koriste u sličnim istraživanjima. Prva dva poglavlja opisuju teorijsku stranu teme, a druga dva poglavlja predstavljaju implementaciju i rezultate.

Na početku prvog poglavlja, opisan je problem odabira podskupa atributa. Nakon toga su pojašnjeni korišteni pojmovi, modeli i koncepti strojnog učenja i također standardne metode za rješavanje spomenutog problema.

U drugom poglavlju su definirani osnovni pojmovi vezani za meta-heuristike. U nastavku je objašnjen algoritam *Artificial Bee Colony*.

Treće poglavlje opisuje korištene skupove podataka, pripremu podataka, prilagodbu te primjenu pčelinjeg algoritma. Isto tako je navedeno koje druge metode za odabir podskupa atributa su primijenjene na skupove podataka.

Rezultati pčelinjeg algoritma, usporedba s rezultatima ostalih metoda i nabrojeni parametri su sadržani u četvrtom poglavlju.

Na kraju, zaključeno je da se algoritam *Binary Artificial Bee Colony* pokazao uspješnim u rješavanju opisanog problema.

Summary

The main goal of this diploma thesis is to explain how to use a meta-heuristic named Artificial Bee Colony to solve the problem of feature selection in the field of machine learning. The Procedure is described and applied to some standard data sets which are usually used in similar researches. First two chapters describe the theoretical side of the topic, and second two chapters represent implementation and results.

At the beginning of the first chapter, the problem of feature selection is described. After that, used terms, models, and conceptions of machine learning are depicted, and also some standard methods for solving the mentioned problem.

In the second chapter, basic terms related to meta-heuristics are defined. Artificial Bee Colony is explained afterward.

The third chapter presents data sets that are used, data preprocessing, adjustment, and application of the bee algorithm. It is also presented which other methods for feature selection are applied to data sets.

Results of the bee algorithm, comparison to results of other methods, and listed parameters are contained in the fourth chapter.

In the end, it is concluded that Binary Artificial Bee Colony algorithm proved to be successful in solving the described problem.

Životopis

Lucija Puškarić rođena je 12. 12. 1996. godine u Ogulinu gdje je pohađala Osnovnu školu Ivane Brlić-Mažuranić. Nakon završetka osnovne škole te osnovne glazbene škole, 2011. godine upisuje se u Gimnaziju Bernardina Frankopana gdje završava opću gimnaziju. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja, redovito sudjeluje na natjecanjima iz matematike, a dva puta se plasirala na državno natjecanje.

2015. godine upisuje preddiplomski sveučilišni studij Matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu. Nakon stjecanja akademskog naziva sveučilišna prvostupnica matematike, nastavlja obrazovanje na istom fakultetu upisavši diplomski sveučilišni studij Računarstvo i matematika 2018. godine. Tijekom diplomskog studija, bila je stipendistica jedne informatičke tvrtke te je sudjelovala na natjecanju Copernicus Hackathon.