INTELIGENTNI SISTEMI

as. ms Vladimir Jocović
as. ms Adrian Milaković

MAŠINSKO UČENJE

08

„What we want is a machine that can learn from experience.“
- Alan Turing

MAŠINSKO UČENJE

Šta je mašinsko učenje?

Mašinsko učenje je proces koji omogućava sistemima da automatski uče i unapređuju svoje znanje na osnovu prethodnog iskustva, bez potrebe za eksplicitnim programiranjem.

Gde se koristi mašinsko učenje?

- Kod problema za koje je teško ručno definisati kako se rešavaju (prepoznavanje lica, obrada prirodnih jezika, robotika, igranje igara...) te klasično programiranje nije moguće.
- Kod problema izvlačenja informacija iz velike količine sirovih podataka i predviđanja budućih trendova korišćenjem trenutno dostupnih podataka. (*Data mining*)
- Kod kompleksnih sistema koji se dinamički prilagođavaju okruženju.

MAŠINSKO UČENJE

Kako izgleda proces mašinskog učenja?

U procesu mašinskog učenja moguće je identifikovati sledeće korake (podela ne mora nužno izgledati ovako):

1. Formulisanje problema i čitanje podataka
2. Analiza podataka
3. Čišćenje podataka
4. Modifikacija i transformacija atributa
5. Izbor i treniranje modela mašinskog učenja
6. Validacija i testiranje podataka

MAŠINSKO UČENJE

Tipovi mašinskog učenja

- Nadgledano učenje (*Supervised learning*) – Svakom ulaznom podatku ***x*** je pridružena izlazna vrednost ***y***. Cilj učenja je da se na osnovu datih parova (*x*, *y*) pronađe optimalna funkcija koja mapira ulaz u izlaz. Koristi se u klasifikacionim (medicinska dijagnostika) i regresivnim (cena nekretnina) problemima.
- Nenadgledano učenje (*Unsupervised learning*) – Dati su samo ulazni podaci ***x***. Ne postoji izlazna vrednost ***y***. Potrebno je pronaći pravilnost u ulaznim podacima na osnovu kojih mogu da se generišu izlazne vrednosti. Algoritmi nenadgledanog učenja mogu se koristiti za klasterizaciju ulaznih podataka (pronalaženje sličnih podataka i njihovo grupisanje), detekciju anomalija (sumnjive transakcije na osnovu istorije kupovina), asocijaciju podataka (predikcija vrednosti drugih atributa na osnovu povezanosti datih atributa), itd.

MAŠINSKO UČENJE

Tipovi mašinskog učenja

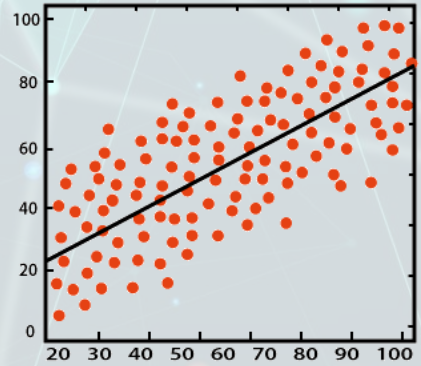
- Polunadgledano učenje (*Semi-supervised learning*) – Predstavlja kombinaciju nadgledanog i nenadgledanog učenja. Delu ulaznih podataka *x* pridružene su izlazne vrednosti *y*. Delu ulaznih podataka nisu pridružene izlazne vrednosti.
- Učenje sa podrškom (*Reinforcement learning*) – Primenjuje se na obučavanje softverskih agenata koji deluju u nekom prostoru akcija. Učenje se vrši na osnovu datih ulaznih podataka i signala podrške koji stiže na kraju nekog skupa akcija agenata i može biti pozitivan ili negativan. Cilj učenja je da iskoristi signal podrške da utvrdi koja tačno akcija ili skup akcija je dovela do pozitivnog signala podrške i shodno tome koriguje ponašanje agenta.

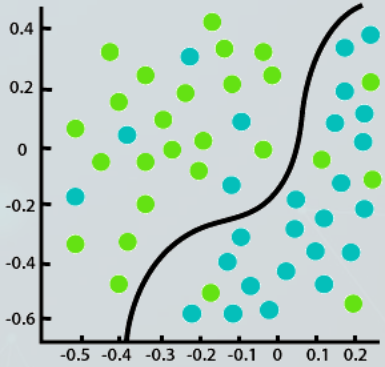
NADGLEDANO UČENJE

Tipovi izlazne vrednosti

Na osnovu tipa izlazne vrednosti ***y*** probleme nadgledanog učenja možemo podeliti na:

- **Probleme regresije** – izlazna vrednost je kontinualnog tipa (realna vrednost)
- **Probleme klasifikacije** – izlazna vrednost je kategoričkog tipa (diskretna vrednost iz skupa)





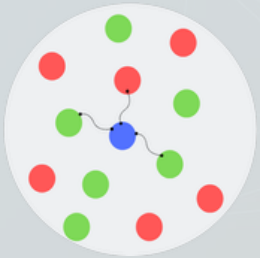
ALGORITAM K NAJBLIŽIH SUSEDA

Kako funkcioniše algoritam k najbližih suseda?

Algoritam k najbližih suseda (*k-nearest neighbors*) je tip nadgledanih algoritama mašinskog učenja. Koristi se kod problema klasifikacije (mada može da se koristi i kod problema regresije).

Algoritam k najbližih suseda polazi od pretpostavke da će se slične instance nalaziti bliže u prostoru. Algoritam klasifikuje određenu instancu na osnovu klasifikacije k najbližih instanci prostim prebrojavanjem svake klase.

U slučaju regresije, algoritam uzima srednju vrednost izlaznog parametra najbližih suseda.



Algoritam započinje biranjem vrednosti za k, potom računa rastojanje od posmatrane instance do svih instanci u skupu podataka, sortira ih i bira k instanci sa namanjim rastojanjem na osnovu kojih daje rezultat.

ALGORITAM K NAJBLIŽIH SUSEDA

Kako odabrati vrednost k?

Da bismo odabrali vrednost k, pokrećemo algoritam više puta sa različitim vrednostima i biramo onu koja najviše smanjuje broj grešaka prilikom testiranja skupa podataka.

Neka opažanja:

- Smanjivanjem vrednosti k do 1, predikcije postaju nestabilnije. Pretpostavimo da određujemo boju nekog oblika oko kojeg se nalazi mnoštvo crvenih oblika i jedan zeleni koji je ujedno i najbliži. Naravno, jasno je da je oblik koji ispitujemo najverovatnije crven, međutim kako je zeleni oblik najbliži, KNN netačno previđa da je oblik zelene boje.
- Sa druge strane, povećanjem vrednosti k, povećavamo i stabilnost predikcije, ali do neke granice. U nekom trenutku, vrednost k će da bude prevelika i obuhvata veći deo skupa podataka čime povećava broj grešaka.
- Najčešće se za vrednost k uzima neparan broj, kako bismo imali *tiebreaker*.
- Jako često se za vresnot k uzima kvadratni koren ukupnog broja podataka u skupu.

ALGORITAM K NAJBLIŽIH SUSEDA

Na koji način odrediti najbliže susede?

Za pronalaženje najbližih suseda između dve tačke P i Q u N-dimenzionalnom prostoru date koordinatama P = (p1, p2, ... , pN) i Q = (q1, q2, ... , qN) koristi se neka od sledećih metrika:

- Euklidska razdaljina - $\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$
- Menhetn razdaljina - $\sum_{i=1}^{n}|p_i - q_i|$
- Čebiševa razdaljina - $\max(|p_i - q_i|)$

i mnoge druge.

Zadatak 1 – Mršava osoba ili ne?

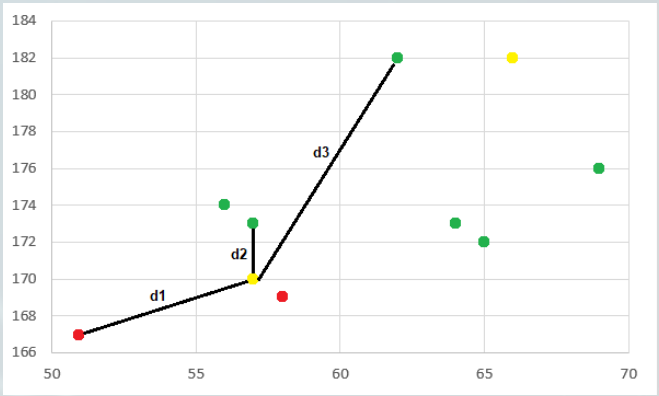
★★☆☆☆

Odrediti da li su posmatrane osobe mršave koristeći *knn* algoritam sa Euklidskim rastojanjem na posmatranom skupu podataka.

Nezavisni atributi		**Atribut odluke**
Masa (kg)	Visina (cm)	Mršava?
51	167	Da
62	182	Ne
69	176	Ne
64	173	Ne
65	172	Ne
56	174	Da
58	169	Ne
57	173	Ne
57	170	?
66	182	?

Zadatak 1 – Rešenje

Na grafiku je vizuelno prikazan skup podataka. Na x osi su prikazane mase osoba, na y osi visine osoba. Crvenim oznakama su prikazane mršave osobe, zelenim oznakama osobe koje nisu mršave, a žutim osobe koje nisu klasifikovane.



Kako računamo rastojanje između posmatrane instance i ostalih? Pr.

d1 = $\sqrt{(170-167)^2+(57-51)^2}$ = 6.71

d2 = $\sqrt{(170-173)^2+(57-57)^2}$ = 3.00

d3 = $\sqrt{(170-182)^2+(57-62)^2}$ = 13.00

Zadatak 1 – Rešenje

Za prvu nepoznatu osobu tražimo Euklidsku razdaljinu do svih drugih tačaka, pa na osnovu faktora *k* određujem koji su susedi najbliži.

Za *k* uzimamo kvadratni koren ukupnog broja podataka u skupu. ($k = \sqrt{8} \approx 3$)

Na osnovu klasifikacije najbližih suseda, određujemo da nepoznata osoba nije mršava.

Masa (kg)	Visina (cm)	Mršava?	Udaljenost
51	167	Da	6.71
62	182	Ne	13.00
69	176	Ne	13.42
64	173	Ne	7.62
65	172	Ne	8.25
56	174	**Da**	**4.12**
58	169	**Ne**	**1.41**
57	173	**Ne**	**3.00**
57	170	**Ne**	

Zadatak 1 – Rešenje

Za drugu nepoznatu osobu tražimo Euklidsku razdaljinu do svih drugih tačaka, pa na osnovu faktora *k* određujem koji su susedi najbliži.

Za *k* uzimamo kvadratni koren ukupnog broja podataka u skupu. ($k = \sqrt{8} \approx 3$)

Na osnovu klasifikacije najbližih suseda, određujemo da nepoznata osoba nije mršava.

Masa (kg)	Visina (cm)	Mršava?	Udaljenost
51	167	Da	21.21
62	182	**Ne**	**4.00**
69	176	**Ne**	**6.71**
64	173	**Ne**	**9.21**
65	172	Ne	10.05
56	174	Da	12.81
58	169	Ne	15.26
57	173	Ne	12.73
66	182	**Ne**	

Zadatak 2 – Crno ili belo vino?

★★★☆☆

Odrediti da li je posmatrano vino crno ili belo koristeći *knn* algoritam sa Euklidskim rastojanjem na posmatranom skupu podataka koristeći za *k* vrednosti: (a) $\sqrt{n}$, (b) 7

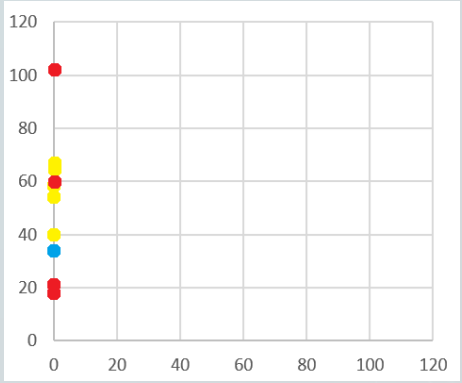
Nezavisni atributi		**Atribut odluke**
Nivo sumpor dioksida	Nivo hlora	Crno/Belo vino
34	0.076	Crveno
67	0.098	Belo
54	0.092	Belo
60	0.075	Crveno
59	0.089	Belo
40	0.075	Belo
59	0.069	Crveno
21	0.065	Crveno
18	0.073	Crveno
102	0.071	Crveno
65	0.097	Belo
34	0.084	?

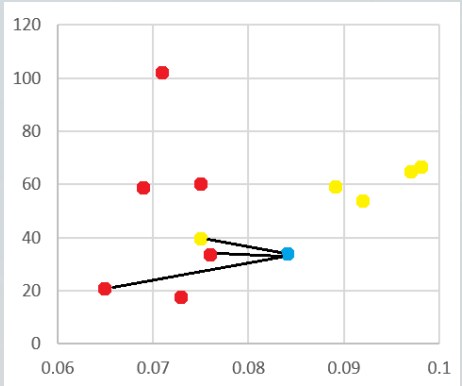
Zadatak 2 – Rešenje

Da li je dobro računati razdaljinu sa ovakvim podacima?

Nivo sumpor dioksida	Nivo hlora	Udaljenost
34	0.076	0.008
67	0.098	33
54	0.092	20
60	0.075	26
59	0.089	25
40	0.075	6.000007
59	0.069	25
21	0.065	13.00001
18	0.073	16
102	0.071	68
65	0.097	31
34	0.084	?

Nivo sumpor dioksida mnogo više utiče na udaljenost nego nivo hlora jer su razlike u vrednostima veće.





Zadatak 2 – Rešenje

Normalizacija je proces reskaliranja svih vrednosti u opseg od 0 do 1. Na taj način sve vrednosti jednako utiču na računanje distance između instanci.

$$x_{novo} = \frac{x_{staro} - x_{min}}{x_{max} - x_{min}}$$

Pr.

x_{min} = 1

x_{min} = 14

$x_{novo1} = \frac{1-1}{14-1} = 0$

$x_{novo2} = \frac{5-1}{14-1} = \frac{4}{13}$

$x_{novo3} = \frac{14-1}{14-1} = 1$

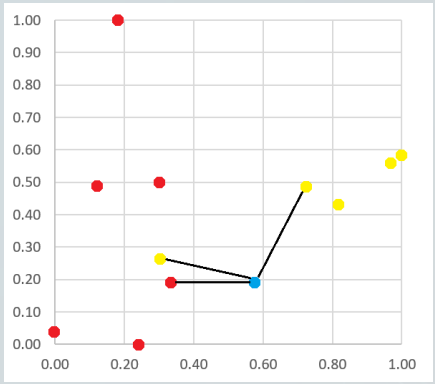
...

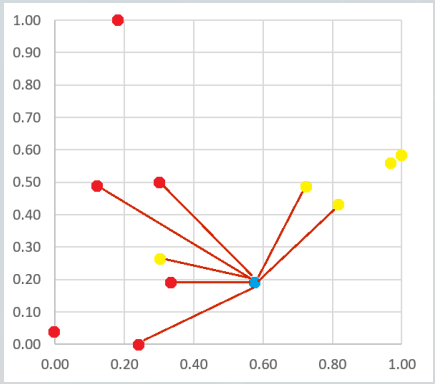
Stara vrednost	Nova vrednost
1	0
5	0.31
5	0.31
14	1
6	0.38
3	0.15
8	0.54
9	0.62
11	0.77
4	0.23
7	0.46

Zadatak 2 – Rešenje

Podaci nakon normalizacije i grafici za $k = \sqrt{11} \approx 3$ i $k = 7$.

Nivo sumpor dioksida	Nivo hlora	Udaljenost	Crno/Belo vino
0.19	0.33	0.242	Crveno
0.58	1	0.578	Belo
0.43	0.82	0.340	Belo
0.50	0.3	0.413	Crveno
0.49	0.73	0.334	Belo
0.26	0.3	0.282	Belo
0.49	0.12	0.543	Crveno
0.04	0	0.596	Crveno
0.00	0.24	0.384	Crveno
1.00	0.18	0.900	Crveno
0.56	0.97	0.540	Belo
0.19	0.58	-	?





Zašto smo dobili različite rezultate za različite vrednosti *k*?

Zadatak 2 – Rešenje

Da li nam normalizacija pomaže u situacijama kada imamo tzv. *outlier*-e?

Nezavisni atributi		Atribut odluke
Nivo sumpor dioksida	Nivo hlora	Crno/Belo vino
34	0.076	Crveno
67	0.098 → 0.980	Belo
54	0.092	Belo
60	0.075	Crveno
59	0.089	Belo
40	0.075	Belo
59	0.069	Crveno
21	0.065	Crveno
18	0.073	Crveno
102	0.071	Crveno
65	0.097	Belo
34	0.084	?

Zadatak 2 – Rešenje

Nivo hlora gubi na značaju zbog prevelikog odstupanja jedne instance.

Nivo sumpor dioksida	Nivo hlora	Udaljenost	Crno/Belo vino
0.19	0.33	0.242	Crveno
0.58	1	0.578	Belo
0.43	0.82	0.340	Belo
0.50	0.3	0.413	Crveno
0.49	0.73	0.334	Belo
0.26	0.3	0.282	Belo
0.49	0.12	0.543	Crveno
0.04	0	0.596	Crveno
0.00	0.24	0.384	Crveno
1.00	0.18	0.900	Crveno
0.56	0.97	0.540	Belo
0.19	**0.58**	-	?

Nivo sumpor dioksida	Nivo hlora	Udaljenost	Crno/Belo vino
0.19	0.01	0.009	Crveno
0.58	**1**	1.055	Belo
0.43	0.03	0.238	Belo
0.50	0.01	0.310	Crveno
0.49	0.03	0.298	Belo
0.26	0.01	0.072	Belo
0.49	0.01	0.298	Crveno
0.04	0	0.156	Crveno
0.00	0.01	0.191	Crveno
1.00	0.01	0.810	Crveno
0.56	0.03	0.369	Belo
0.19	**0.02**	-	?

Zadatak 3 – Filmovi

★★★★☆

Na osnovu skupa podataka, preporučiti korisniku 6 filmova koji su najsličniji filmu *The Imitation Game* koristeći *knn* algoritam sa Menhetn rastojanjem i uporediti rezultate ukoliko: (a) ne koristimo normalizaciju vrednosti atributa, (b) koristimo normalizaciju vrednosti atributa

Film	IMDB ocena	Žanr
The Imitation Game	8	Biografski, Drama, Triler
Ex Machina	7.7	Drama, Misterija
A Beautiful Mind	8.2	Biografski, Drama
Good Will Hunting	8.3	Drama
Forrest Gump	8.8	Drama
21	6.8	Drama
Gifted	7.6	Drama
Travelling Salesman	5.9	Drama, Misterija
Avatar	7.9	-
The Wolf of Wall Street	8.2	Biografski, Komedija
A Time To Kill	7.4	Drama, Triler
Interstellar	8.6	Drama
The Wind Rises	7.8	Biografski, Drama

Zadatak 3 – Rešenje

Neki algoritmi (kao KNN) ne podržavaju rad sa kategoričkim podacima, stoga je prvo potrebno transformisati tabelu. Transformacija kategoričkih vrednosti direktno u numeričke vrednosti ima smisla ukoliko su kategoričke vrednosti uporedive. Transformacija na ovakav način nije dobra jer se nekim kategoričkim vrednostima daje veći prioritet za neke algoritme.

Plata
Niska
Srednja
Visoka
Srednja

Plata
0
1
2
1

Boja
Plava
Zelena
Plava
Crvena

Boja
0
1
0
2

Zadatak 3 – Rešenje

Takve kategoričke vrednosti ćemo pretvoriti u numeričke koristeći *one-hot-encoding* tehniku koja za svaku moguću kategoričku vrednost nekog atributa kreira novi atribut čija je vrednost 0 ili 1.

Boja
Plava
Zelena
Plava
Crvena

Crvena	Zelena	Plava
0	0	1
0	1	0
0	0	1
1	0	0

Zadatak 3 – Rešenje

Za svaki žanr definišemo poseban atribut čija vrednost (1 ili 0) određuje prisustvo žanra u filmu.

Film	IMDB ocena	Biografski	Drama	Triler	Komedija	Misterija
The Imitation Game	8	1	1	1	0	0
Ex Machina	7.7	0	1	0	0	1
A Beautiful Mind	8.2	1	1	0	0	0
Good Will Hunting	8.3	0	1	0	0	0
Forrest Gump	8.8	0	1	0	0	0
21	6.8	0	1	0	0	0
Gifted	7.6	0	1	0	0	0
Travelling Salesman	5.9	0	1	0	0	1
Avatar	7.9	0	0	0	0	0
The Wolf of Wall Street	8.2	1	0	0	1	0
A Time To Kill	7.4	0	1	1	0	0
Interstellar	8.6	0	1	0	0	0
The Wind Rises	7.8	1	1	0	0	0

Zadatak 3 – Rešenje

Tabele najsličnijih filmova ukoliko se ne koristi (levo), odnosno koristi (desno) normalizacija.

Film	Sličnost
The Imitation Game	
Ex Machina	3.30
A Beautiful Mind	**1.20**
Good Will Hunting	**2.30**
Forrest Gump	2.80
21	3.20
Gifted	**2.40**
Travelling Salesman	5.10
Avatar	3.10
The Wolf of Wall Street	3.20
A Time To Kill	**1.60**
Interstellar	2.60
The Wind Rises	**1.20**

Film	Sličnost
The Imitation Game	
Ex Machina	3.10
A Beautiful Mind	**1.07**
Good Will Hunting	**2.10**
Forrest Gump	2.28
21	2.41
Gifted	**2.14**
Travelling Salesman	3.72
Avatar	3.03
The Wolf of Wall Street	3.07
A Time To Kill	**1.21**
Interstellar	2.21
The Wind Rises	**1.07**

Zadatak 4 – Medicinska dijagnostika

★★★★☆

Dat je skup podataka o obavljenoj analizi nad pacijentima i informacije da li boluju od posmatrane bolesti. Zbog privatnosti podataka, originalne vrednosti i opis atributa nije dat.

Pacijent	X1	X2	X3	Bolest
T1	0.74	0.02	0.44	da
T2	0.91	0.13	0.44	ne
T3	0.92	0.07	0.36	ne
T4	0.85	0.13	0.27	ne
T5	0.80	0.03	0.56	da
T6	0.87	0.11	0.17	ne
T7	0.75	0.17	0.39	ne
T8	0.86	0.06	0.41	ne
T9	0.84	0.15	0.47	ne
T10	0.79	0.11	0.50	ne
T11	0.81	0.09	0.54	da
T12	0.69	0.12	0.40	ne
T13	0.99	0.12	0.39	ne
T14	0.95	0.13	0.37	ne
T15	0.94	0.11	0.22	ne

Zadatak 4 – Medicinska dijagnostika

★★★★☆

Za sledeći test skup odrediti tačnost, preciznost i odziv koristeći *knn* algoritam sa *Manhattan* distancom i parametrom k=3.

Pacijent	X1	X2	X3	Bolest
T16	0.76	0.05	0.45	da
T17	0.79	0.06	0.51	da
T18	0.72	0.02	0.39	da
T19	0.82	0.11	0.38	ne
T20	0.88	0.09	0.25	ne
T21	0.91	0.13	0.27	ne
T22	0.87	0.09	0.29	ne
T23	0.86	0.14	0.33	ne

Zadatak 4 – Rešenje

U tabeli su date distance od tačke T16 i obeležena tri najbliža suseda. Analogno bi se radilo za preostale tačke skupa za testiranje.

Pacijent	X1	X2	X3	Bolest	Distanca
T1	0.74	0.02	0.44	da	0.06
T2	0.91	0.13	0.44	ne	0.24
T3	0.92	0.07	0.36	ne	0.27
T4	0.85	0.13	0.27	ne	0.35
T5	0.80	0.03	0.56	da	0.17
T6	0.87	0.11	0.17	ne	0.45
T7	0.75	0.17	0.39	ne	0.19
T8	0.86	0.06	0.41	ne	0.15
T9	0.84	0.15	0.47	ne	0.20
T10	0.79	0.11	0.50	ne	0.14
T11	0.81	0.09	0.54	da	0.18
T12	0.69	0.12	0.40	ne	0.19
T13	0.99	0.12	0.39	ne	0.36
T14	0.95	0.13	0.37	ne	0.35
T15	0.94	0.11	0.22	ne	0.47

Zadatak 4 – Rešenje

Na osnovu stvarnih izlaza i prediktovanih izlaza računamo svu potrebnu metriku.

Pacijent	X1	X2	X3	Bolest	Predikcija
T16	0.76	0.05	0.45	da	ne
T17	0.79	0.06	0.51	da	da
T18	0.72	0.02	0.39	da	ne
T19	0.82	0.11	0.38	ne	ne
T20	0.88	0.09	0.25	ne	ne
T21	0.91	0.13	0.27	ne	ne
T22	0.87	0.09	0.29	ne	ne
T23	0.86	0.14	0.33	ne	ne

True positive – Broj ispravno prediktovanih „da“ primeraka (Ima ih 1)
True negative – Broj ispravno prediktovanih „ne“ primeraka (Ima ih 5)
False positive – Primerci označeni sa „ne“ koji su pogrešno prediktovani kao „da“ (Ima ih 0)
False negative – Primerci označeni sa „da“ koji su pogrešno prediktovani kao „ne“ (Ima ih 2)

Tačnost = *broj_tačnih_predikcija / ukupan_broj_predikcija = (TP+TN) / (TP+TN+FP+FN)*
Tačnost = 6/8 = 0.75 (75%)

Zadatak 4 – Rešenje

Pacijent	X1	X2	X3	Bolest	Predikcija
T16	0.76	0.05	0.45	da	ne
T17	0.79	0.06	0.51	da	da
T18	0.72	0.02	0.39	da	ne
T19	0.82	0.11	0.38	ne	ne
T20	0.88	0.09	0.25	ne	ne
T21	0.91	0.13	0.27	ne	ne
T22	0.87	0.09	0.29	ne	ne
T23	0.86	0.14	0.33	ne	ne

Preciznost (da) = TP / (TP + FP) = 1 / 1 = 1.00 (100%)
Odziv (da) = TP / (TP + FN) = 1 / 3 = 0.33 (33%)

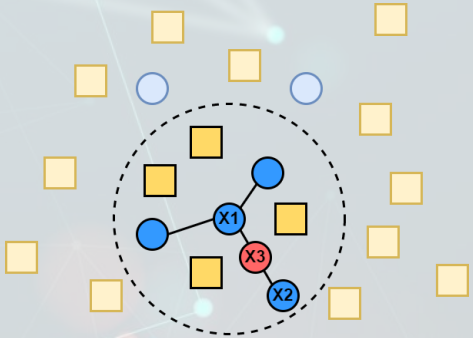
Preciznost (ne) = 5/7 = 0.71 (71%)
Odziv (ne) = 5/5 = 1.00 (100%)

Iako je tačnost važna, ona nije presudna u nekim primerima. U slučaju medicinske dijagnostike, čak i da imamo tačnost od 99%, to nam ništa ne znači ukoliko je greška od 1% prouzrokovana *false negative* primercima, jer nismo uspeli da otkrijemo bolest na vreme kod pacijenata koji su je zapravo imali.
Odziv za pozitivne primerke je vrlo važan u medicinskoj dijagnostici, proveri spama, prevarama u bankarskim transakcijama itd.

Zadatak 4 – Rešenje

Loš odziv za „da“ primerke je posledica neuravnoteženog skupa podataka (premali je procenat stvarno bolesnih u odnosu na one koji nisu bolesni – čest slučaj u praksi).
Da bismo učinili skup uravnoteženim primenjujemo jednu od sledeće dve tehnike:
- *oversampling* – sinteza novih primeraka klase u manjini
- *undersampling* – uklanjanje primeraka klase koja preovladava

Oversampling može da se uradi pomoću SMOTE algoritma. Ideja algoritma jeste da nasumično odabere primerak klase u manjini (x1) i pronađe njegovih *k* najbližih suseda klase u manjini. Potom, nasumično odabere jedan od *k* najbližih suseda (x2) i sintetiše nov primerak (x3) na nasumičnoj poziciji na putanji između odabrana dva primerka (između x1 i x2, najčešće na sredini). Algoritam se ponavlja dok se ne sintetiše željen broj primeraka.



Zadatak 4 – Rešenje

Primenom SMOTE algoritma možemo da povećamo skup pacijenata koji su bolesni i približimo ga broju pacijenata koji nisu bolesni kako bismo potencijalno napravili bolji model sa većim odzivom. Neka je parametar SMOTE algoritma k=1, a svaki novi primerak se sintetiše na sredini.

Pacijent	X1	X2	X3	Bolest
T1	0.74	0.02	0.44	da
T5	0.80	0.03	0.56	da
T11	0.81	0.09	0.54	da

Nasumično je odabran primerak T1. Traži se njegov najbliži sused:
Dist(T1-T5) = 0.06 + 0.01 + 0.12 = 0.19
Dist(T1-T11) = 0.07 + 0.07 + 0.10 = 0.24
Kako je primerak T5 najbliži, sintetiše se nov primerak, L1 na sredini između primeraka T1 i T5.
X1 = (0.74+0.80)/2 = 0.77
X2 = (0.02+0.03)/2 = 0.025
X3 = (0.44+0.56)/2 = 0.50
Bolest = da

Zadatak 4 – Rešenje

Ponavljanjem SMOTE algoritma, možemo da generišemo 9 novih primeraka, kako bi ukupan broj primeraka pacijenata sa i bez bolesti bio po 12 u oba slučaja, čime dobijamo uravnotežen skup.

Pacijent	X1	X2	X3	Bolest
L1	0.77	0.03	0.50	da
L2	0.78	0.06	0.49	da
L3	0.81	0.06	0.55	da
L4	0.77	0.04	0.50	da
L5	0.79	0.04	0.53	da
L6	0.79	0.06	0.52	da
L7	0.77	0.04	0.50	da
L8	0.79	0.06	0.52	da
L9	0.79	0.04	0.53	da

Zadatak 4 – Rešenje

Ponavljamo KNN algoritam sa povećanim k zbog povećanog broja primeraka i određujemo tačnost, preciznost i odziv. U nastavku su date distance od primerka T16.

Pacijent	Bolest	Distanca
T1	da	0.06
T2	ne	0.24
T3	ne	0.27
T4	ne	0.35
T5	da	0.17
T6	ne	0.45
T7	ne	0.19
T8	ne	0.15
T9	ne	0.20
T10	ne	0.14
T11	da	0.18
T12	ne	0.19
T13	ne	0.36
T14	ne	0.35
T15	ne	0.47

Pacijent	Bolest	Distanca
L1	da	0.08
L2	da	0.07
L3	da	0.16
L4	da	0.07
L5	da	0.11
L6	da	0.11
L7	da	0.07
L8	da	0.11
L9	da	0.11

Zadatak 4 – Rešenje

Na osnovu stvarnih izlaza i prediktovanih izlaza računamo svu potrebnu metriku.

Pacijent	X1	X2	X3	Bolest	Predikcija
T16	0.76	0.05	0.45	da	da
T17	0.79	0.06	0.51	da	da
T18	0.72	0.02	0.39	da	da
T19	0.82	0.11	0.38	ne	ne
T20	0.88	0.09	0.25	ne	ne
T21	0.91	0.13	0.27	ne	ne
T22	0.87	0.09	0.29	ne	ne
T23	0.86	0.14	0.33	ne	ne

True positive – Broj ispravno prediktovanih „da“ primeraka (Ima ih 3)
True negative – Broj ispravno prediktovanih „ne“ primeraka (Ima ih 5)
False positive – Primerci označeni sa „ne“ koji su pogrešno prediktovani kao „da“ (Ima ih 0)
False negative – Primerci označeni sa „da“ koji su pogrešno prediktovani kao „ne“ (Ima ih 0)

Tačnost = 8/8 = 1.00 (100%)
Preciznost (da) = TP / (TP + FP) = 3/3 = 1.00 (100%)
Odziv (da) = TP / (TP + FN) = 3/3 = 1.00 (100%)

Preciznost (ne) = 7/7 = 1.00 (100%)
Odziv (ne) = 5/5 = 1.00 (100%)

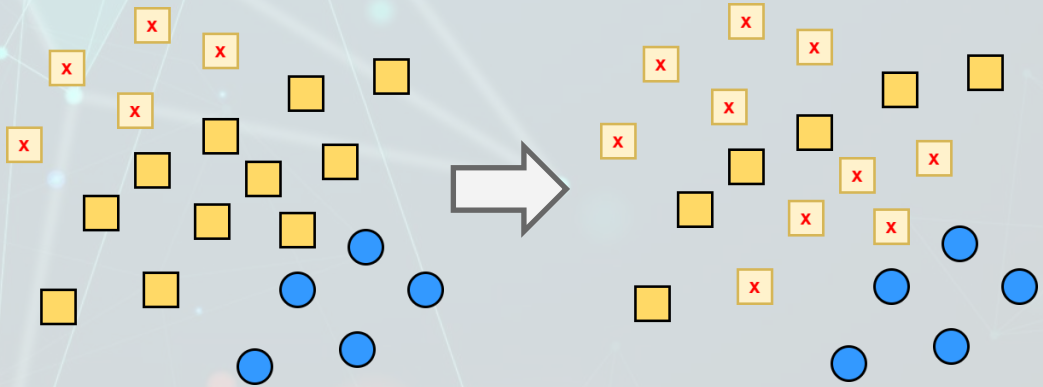
Zadatak 4 – Rešenje

U tabeli su date distance od tačke T16 i obeležena tri najbliža suseda. Analogno bi se radilo za preostale tačke skupa za testiranje.

Pacijent	X1	X2	X3	Bolest	Distanca
T1	0.74	0.02	0.44	da	0.06
T2	0.91	0.13	0.44	ne	0.24
T3	0.92	0.07	0.36	ne	0.27
T4	0.85	0.13	0.27	ne	0.35
T5	0.80	0.03	0.56	da	0.17
T6	0.87	0.11	0.17	ne	0.45
T7	0.75	0.17	0.39	ne	0.19
T8	0.86	0.06	0.41	ne	0.15
T9	0.84	0.15	0.47	ne	0.20
T10	0.79	0.11	0.50	ne	0.14
T11	0.81	0.09	0.54	da	0.18
T12	0.69	0.12	0.40	ne	0.19
T13	0.99	0.12	0.39	ne	0.36
T14	0.95	0.13	0.37	ne	0.35
T15	0.94	0.11	0.22	ne	0.47

Zadatak 4 – Rešenje

Undersampling može da se uradi pomoću *NearMiss* algoritma. Postoje tri verzije algoritma, a ovde će biti iskorišćena verzija 3. Algoritam se izvršava u dva koraka. Ideja algoritma jeste da za svaki primerak klase u manjini pronađe njegovih *k* najbližih suseda klase koja nije u manjini. Od svih takvih primeraka, zadržava se samo *n* primeraka čija je srednja udaljenost do *k*-najbližih suseda klase u manjini najveća, a svi ostali primerci se odbacuju.



Zadatak 4 – Rešenje

Primenom *NearMiss* algoritma možemo da smanjimo skup pacijenata koji nisu bolesni i približimo ga broju pacijenata koji su bolesni kako bismo potencijalno napravili bolji model sa većim odzivom. Neka su parametri *NearMiss* algoritma k=2 i n=3.

Za svaki primerak klase „da“ računamo distancu do svih primeraka klase „ne“ i tražimo k najbližih.

	T2	T3	T4	T6	T7	T8	T9	T10	T12	T13	T14	T15
T1	0.28	0.31	0.39	0.49	0.21	0.19	0.26	0.20	0.19	0.40	0.39	0.51

Kada i od ostalih primeraka (T5, T11) nađemo k najbližih (T2, T7, T9 i T10), završava se prvi korak algoritma.

Zadatak 4 – Rešenje

Od posmatranih primeraka čuva se *n* primeraka čija je srednja udaljenost do *k* najbližih suseda klase „da“ najveća, a svi ostali se odbacuju.

	T1	T5	T11	AVG
T2	0.28	0.24	0.33	0.26
T7	0.21	0.29	0.36	0.25
T8	0.19	0.21	0.24	0.20
T9	0.26	0.16	0.25	0.21
T10	0.20	0.08	0.15	0.12
T12	0.19	0.29	0.36	0.24

Zadatak 4 – Rešenje

Ponavljamo KNN algoritam i određujemo tačnost, preciznost i odziv. U nastavku su date distance od primerka T16.

Pacijent	X1	X2	X3	Bolest	Distanca
T1	0.74	0.02	0.44	da	0.06
T2	0.91	0.13	0.44	ne	0.24
T5	0.80	0.03	0.56	da	0.17
T7	0.75	0.17	0.39	ne	0.19
T11	0.81	0.09	0.54	da	0.18
T12	0.69	0.12	0.40	ne	0.19

Zadatak 4 – Rešenje

Na osnovu stvarnih izlaza i prediktovanih izlaza računamo svu potrebnu metriku.

Pacijent	X1	X2	X3	Bolest	Predikcija
T16	0.76	0.05	0.45	da	da
T17	0.79	0.06	0.51	da	da
T18	0.72	0.02	0.39	da	ne
T19	0.82	0.11	0.38	ne	ne
T20	0.88	0.09	0.25	ne	ne
T21	0.91	0.13	0.27	ne	ne
T22	0.87	0.09	0.29	ne	ne
T23	0.86	0.14	0.33	ne	ne

True positive – Broj ispravno prediktovanih „da“ primeraka (Ima ih 2)
True negative – Broj ispravno prediktovanih „ne“ primeraka (Ima ih 5)
False positive – Primerci označeni sa „ne“ koji su pogrešno prediktovani kao „da“ (Ima ih 0)
False negative – Primerci označeni sa „da“ koji su pogrešno prediktovani kao „ne“ (Ima ih 1)

Tačnost = 7/8 = 0.875 (88%)
Preciznost (da) = TP / (TP + FP) = 2/2 = 1.00 (100%)
Odziv (da) = TP / (TP + FN) = 2/3 = 0.67 (67%)

Preciznost (ne) = 5/6 = 0.83 (83%)
Odziv (ne) = 5/5 = 1.00 (100%)

Zadatak za samostalnu vežbu – Majice

★★★☆☆

Na osnovu sledeće tabele podataka o visini i masi osoba odrediti koja veličina majice odgovara označenim osobama na kraju tabele koristeći *knn* algoritam sa Menhetn rastojanjem i vrednošću k=3. Da li bi se rešenje promenilo ukoliko bi se podaci normalizovali?

Visina	Masa	Veličina majice
168	58	S
168	63	S
170	60	S
173	62	M
173	65	M
175	63	M
178	67	L
180	66	L
182	73	L
172	61	?
181	60	?

STABLA ODLUČIVANJA

Šta su stabla odlučivanja?

Stabla odlučivanja su tip nadgledanih algoritama mašinskog učenja. Najčešće se koriste kod problema klasifikacije (mada mogu da se koriste i kod problema regresije).

Stablo odlučivanja je stablo u kojem je:

- Svakom unutrašnjem čvoru pridružen jedan ulazni parameter (atribut)
- Svakoj grani pridružena jedna vrednost ulaznog parametra čvora od kojeg grana počinje
- Svakom listu pridružen izlazni parametar u zavisnosti od vrednosti ulaznih parametara na datom putu kroz stablo, od korena do posmatranog lista

Ulazni parametri mogu da budu i kontinualnog i kategoričkog tipa.

Prema tipu izlazne vrednosti razlikujemo klasifikaciona i regresivna stabla.

STABLA ODLUČIVANJA

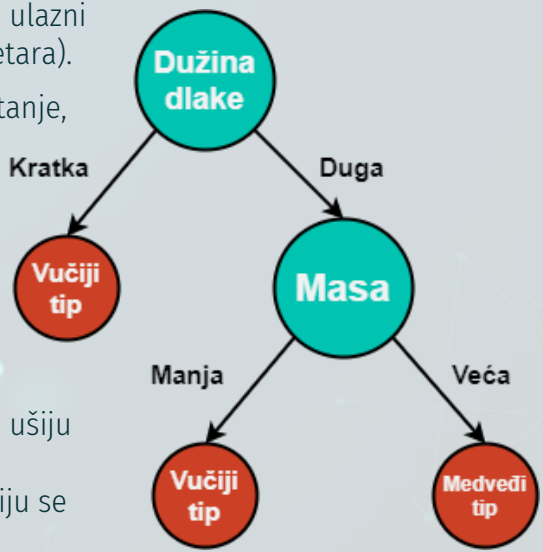
Npr. *Predviđanje tipa samojeda na osnovu nekoliko atributa.*

Stablo se koristi kao klasifikator tako što uzima ulazni primer, dat kao vektor atributa (ulaznih parametara).

Atribut koji je koreni čvor se interpretira kao pitanje, a odgovor se određuje na osnovu vrednosti tog atributa iz ulaznog primera.

Odgovor određuje koji čvor naslednik se posećuje.

Pitanja se ponavljaju dok se ne dođe do lista koji određuje izlaznu vrednost.



Pr. Samojed duge dlake, veće mase i zaobljenih ušiju se klasifikuje kao medveđi tip.
Pr. Samojed kratke dlage, veće mase i oštrih ušiju se klasifikuje kao vučiji tip.

STABLA ODLUČIVANJA

Izgradnja stabla (ID3 algoritam)

Stablo se gradi od korena ka listovima, tzv. pohlepnim (*greedy*) pristupom. Na početku sve instance prostora pripadaju istom skupu, nakon čega se prostor sukcesivno deli na podskupove. Za deljenje kažemo da je pohlepno zato što se pri svakom koraku najbolja podela određuje na osnovu stanja u posmatranom koraku, odnosno ne uzima se u obzir kako će se podela izvršiti u narednim koracima i koja bi podela mogla dovesti do boljih rezultata u narednim koracima.

U svakoj iteraciji se bira najbolji atribut na osnovu kojeg će stablo da se podeli u podskupove. Za svaku moguću vrednost izabranog atributa se prostor deli u podskupve, koji odgovaraju odabranoj vrednosti. Algoritam se rekurzivno ponavlja dok se ne dosegne unapred definisan kriterijum zaustavljanja ili dok se ne dostignu svi listovi.

STABLA ODLUČIVANJA

Kako odabrati najbolji atribut za podelu?

U zavisnosti od toga da li se gradi regresivno ili klasifikaciono stablo, kriterijum na osnovu kojeg se vrši podela može da bude različit.

Postoji nekoliko pristupa pri izboru najboljeg atributa za podelu:

- *Random* – Atribut se bira slučajnim izborom.
- *Least-values* – Bira se atribut sa najmanjim brojem mogućih vrednosti.
- *Most-values* – Bira se atribut sa najvećim brojem mogućih vrednosti.
- *Max-gain* – Bira se atribut koji je po nekom kriterijumu najbolji za podelu (ima najveću dobit informacija ili najmanji Gini indeks).

KLASIFIKACIONA STABLA

Šta je i kako se određuje dobit informacija?

Dobit informacija (*information gain*) računa razliku u entropiji pre i nakon podele skupa podataka na neki način.

Za podelu po svim atributima se računa dobit informacija i bira onaj atribut koji maksimizuje tu dobit.

Npr. Petar je zamislio broj od 1 do 100. Miloš ima pravo da Petru postavlja da/ne pitanja na osnovu kojih treba da pogodi broj koji je Petar zamislio. Koliko najmanje pitanja Miloš treba da postavi da bi pogodio broj koji je Petar zamislio?

Rešenje: 7 pitanja

Sa svakim da/ne pitanjem, optimalno bi trebalo odbaciti najviše ½ od preostalih brojeva ($\log_2 100 = 6.64$).

Ako je dat skup S veličine $|S|$, očekivan broj pokušaja radi određivanja konkretnog elementa je $\log_2 |S|$.

KLASIFIKACIONA STABLA

Šta je i kako se određuje entropija?

Entropija skupa primera, S, za neku binarnu klasifikaciju (ishod ima samo dve klase) je:

$$Entropy(S) = -p_1 * log_2(p_1) - p_2 * log_2(p_2)$$

gde je p_1 procenat pojavljivanja prve klase u skupu S, a p_2 procenat pojavljivanja druge klase u skupu S.

Ako svi primeri pripadaju istoj klasi, entropija je 0 (koristi se jednakost $0 * \log(0) = 0$).

Ako su primeri jednako raspoređeni ($p_1 = p_2 = 0.5$), entropija ima maksimalnu vrednost 1.

Entropiju možemo posmatrati kao prosečan broj bitova potreban za kodiranje klase primera iz skupa S, gde su češćim slučajevima kompresijom dodeljeni kraći kodovi.

Za probleme gde postoji više klasa ishoda, entropija se računa kao:

$$Entropy(S) = \sum_{i=1}^{c} -p_i * \log_2(p_i)$$

gde je *c* broj klasa.

KLASIFIKACIONA STABLA

Dobit informacija atributa *F* je očekivano smanjenje u entropiji koje proizilazi iz podele skupa S po ovom atributu.

$$Gain(S, F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Drugi sabirak predstavlja sumu entropija svakog rezultujućeg podskupa nastalog iz podele skupa S po atributu F za svaku od vrednosti V, pomnoženu svojom relativnom veličinom u odnosu na početni skup S.

Npr. Skup podataka (2 banane, 2 kruške -> Entropija = 1):

[izdužen oblik, žut] -> banana; [izdužen oblik, zelen] -> banana;
[spljošten oblik, žut] -> kruška; [spljošten oblik, zelen] -> kruška;

Podela po obliku:
izdužen oblik: 2 banane, 0 kruški (Entropija = 0)
spljošten oblik: 0 banana, 2 kruške (Entropija = 0)
Gain = 1 – (0.5 * 0 + 0.5 * 0) = 1

Podela po boji:
izdužen oblik: 1 banana, 1 kruška (Entropija = 1)
spljošten oblik: 1 banana, 1 kruška (Entropija = 1)
Gain = 1 – (0.5 * 1 + 0.5 * 1) = 0

Bolja je podela po obliku!

KLASIFIKACIONA STABLA

Šta je i kako se određuje Gini indeks?

Gini indeks skupa primera, S, za neku binarnu klasifikaciju (ishod ima samo dve klase) je:

$$Gini(S) = 1 - p_1^2 - p_2^2$$

gde je p_1 procenat pojavljivanja prve klase u skupu S, a p_2 procenat pojavljivanja druge klase u skupu S.

Ako svi primeri pripadaju istoj klasi, indeks je 0.

Ako su primeri jednako raspoređeni ($p_1 = p_2 = 0.5$), indeks ima maksimalnu vrednost 0.5.

Za probleme gde postoji više klasa ishoda, Gini indeks se računa kao:

$$Gini(S) = 1 - \sum_{i=1}^{c} - p_i^2$$

Ukupan Gini indeks podele skupa na osnovu atributa *F* je dat formulom:

$$Gini(S, F) = \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Gini(S_v)$$

Zadatak 1 – Životinje koje ležu jaja

★★★☆☆

Konstruisati stablo odlučivanja za određivanje da li životinja leže jaja na osnovu sledećeg skupa podataka i njihovih karakteristika.

Nezavisni atributi					Atribut odluke
Životinja	Toplokrvna	Pernata	Ima krzno	Pliva	Leže jaja
Noj	Da	Da	Ne	Ne	Da
Krokodil	Ne	Ne	Ne	Da	Da
Gavran	Da	Da	Ne	Ne	Da
Albatros	Da	Da	Ne	Ne	Da
Delfin	Da	Ne	Ne	Da	Ne
Koala	Da	Ne	Da	Ne	Ne

Zadatak 1 – Rešenje

Prvo ćemo rešiti zadatak korišćenjem Gini indeksa kao kriterijuma podele.

Atribut „toplokrvni“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „toplokrvni“ DA je veličine 5 (noj, gavran, albatros, delfin, koala). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 3 (noj, gavran, albatros), a vrednosti NE je 2 (delfin, koala). Gini indeks takvog podskupa je:

$$Gini(3\ DA, 2\ NE) = 1 - (3/5)^2 - (2/5)^2 = 0.48$$

Podskup skupa S u kojem je vrednost atributa „toplokrvni“ NE je veličine 1 (krokodil). Samo vrednost DA za izlazni atribut „leže jaja“ se u takvom podskupu nalazi. Gini indeks takvog podskupa je:

$$Gini(1\ DA, 0\ NE) = 1 - (1/1)^2 - (0/1)^2 = 0$$

Sada računamo ukupan Gini indeks za podelu po atributu „toplokrvna“:

$$Gini(S, \text{toplokrvna}) = 5/6 * 0.48 + 1/6 * 0 = 0.4$$

Zadatak 1 – Rešenje

Atribut „pernata“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „pernata“ DA je veličine 3 (noj, gavran, albatros). Samo vrednost DA za izlazni atribut „leže jaja“ se u takvom podskupu nalazi. Gini indeks takvog podskupa je:

$$Gini(3\ DA, 0\ NE) = -(3/3)^2 - (0/3)^2 = 0$$

Podskup skupa S u kojem je vrednost atributa „pernata“ NE je veličine 3 (krokodil, delfin, koala). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 1 (krokodil), a vrednosti NE je 2 (delfin, koala). Gini indeks takvog podskupa je:

$$Gini(1\ DA, 2\ NE) = 1 - (1/3)^2 - (2/3)^2 = 0.44444$$

Sada računamo Gini indeks za podelu po atributu „pernata“:

$$Gini(S, pernata) = 3/6 * 0 + 3/6 * 0.44444 = 0.22222$$

Zadatak 1 – Rešenje

Atribut „ima krzno“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „ime krzno“ DA je veličine 1 (koala). Samo vrednost NE za izlazni atribut „leže jaja“ se u takvom podskupu nalazi. Gini indeks takvog podskupa je:

$$Gini(0\ DA, 1\ NE) = 1 - (0/1)^2 - (1/1)^2 = 0$$

Podskup skupa S u kojem je vrednost atributa „ima krzno“ NE je veličine 5 (noj, krokodil, gavran, albatros, delfin). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 4 (noj, krokodil, gavran, albatros), a vrednosti NE je 1 (delfin). Gini indeks takvog podskupa je:

$$Gini(4\ DA, 1\ NE) = 1 - (4/5)^2 - (1/5)^2 = 0.32$$

Sada računamo Gini indeks za podelu po atributu „ima krzno“:

$$Gini(S, \text{ima krzno}) = 1/6 * 0 + 5/6 * 0.32 = 0.26667$$

Zadatak 1 – Rešenje

Atribut „pliva“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „pliva“ DA je veličine 2 (krokodil, delfin). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 1 (krokodil), a vrednosti NE je 1 (delfin). Gini indeks takvog podskupa je:

$$Gini(1\ DA, 1\ NE) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

Podskup skupa S u kojem je vrednost atributa „pliva“ NE je veličine 4 (noj, gavran, albatros, koala). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 3 (noj, gavran, albatros), a vrednosti NE je 1 (koala). Gini indeks takvog podskupa je:

$$Gini(3\ DA, 1\ NE) = 1 - (3/4)^2 - (1/4)^2 = 0.375$$

Sada računamo Gini indeks za podelu po atributu „pliva“:

$$Gini(S, pliva) = 2/6 * 0.5 + 4/6 * 0.375 = 0.41667$$

Zadatak 1 – Rešenje

Tražimo najbolju podelu.

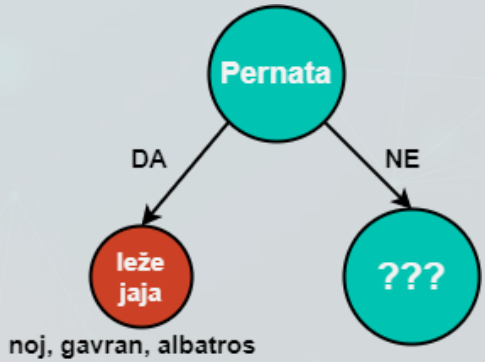
$$Gini(S, \text{toplokrvna}) = 5/6 * 0.48 + 1/6 * 0 = 0.4$$

$$Gini(S, \text{pernata}) = 3/6 * 0 + 3/6 * 0.44444 = 0.22222$$

$$Gini(S, \text{ima krzno}) = 1/6 * 0 + 5/6 * 0.32 = 0.26667$$

$$Gini(S, \text{pliva}) = 2/6 * 0.5 + 4/6 * 0.375 = 0.41667$$

Najmanji Gini indeks ima podela po atributu „pernata“ pa taj atribut biramo za koreni čvor i skup podataka delimo na osnovu tog atributa.



Zadatak 1 – Rešenje

Nakon podele ulaznog skupa podataka po atributu „pernata“, dobijamo sledeći podskup.

Životinja	Toplokrvna	Ima krzno	Pliva	Leže jaja
Krokodil	Ne	Ne	Da	Da
Delfin	Da	Ne	Da	Ne
Koala	Da	Da	Ne	Ne

Za atribut „toplokrvna“ računamo:

$$Gini(0\ DA, 2\ NE) = 1 - (0/2)^2 - (2/2)^2 = 0$$

$$Gini(1\ DA, 0\ NE) = 1 - (1/1)^2 – (0/1)^2 = 0$$

$$Gini(S', \text{toplokrvna}) = 2/3 * 0 + 1/3 * 0 = 0$$

Zadatak 1 – Rešenje

Za atribut „ima krzno“ računamo:

$$Gini(0\ DA, 1\ NE) = 1 - (0/1)^2 - (1/1)^2 = 0$$

$$Gini(1\ DA, 1\ NE) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$Gini(S', \text{ima krzno}) = 1/3 * 0 + 2/3 * 0.5 = 0.33333$$

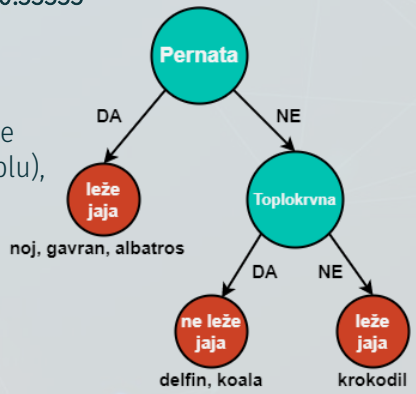
Za atribut „pliva“ računamo:

$$Gini(1\ DA, 1\ NE) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$Gini(0\ DA, 1\ NE) = 1 - (0/1)^2 - (1/1)^2 = 0$$

$$Gini(S', \text{pliva}) = 2/3 * 0.5 + 1/3 * 0 = 0.33333$$

Najbolji Gini indeks ima podela po atributu „toplokrvna“ pa taj čvor biramo za sledeći čvor. Kako skup podataka ne može dalje da se deli (stigli smo do listova po svim putanjama u stablu), konstrukcija stabla je završena.



Zadatak 1 – Rešenje

Sada ćemo isti zadatak da rešimo korišćenjem dobiti informacija kao kriterijuma podele.

U ulaznom skupu S imamo dve vrednosti za atribut „leže jaja“: DA i NE.

Ulazni skup S sadrži četiri reda čija je vrednost izlaznog atributa DA i dva reda čija je vrednost NE. Računamo entropiju takvog skupa.

$$Entropy(4\ DA, 2\ NE) = -\ 4/6 * \log_2(4/6) - 2/6 * \log_2(2/6) = 0.91829$$

Sada je potrebno pronaći dobit informacija za sve atribute: toplokrvna, pernata, ima krzno, pliva (atribut „životinja“ nam ne daje nikakve informacije). Atribut sa najvećom dobiti informacija će biti atribut na osnovu kojeg vršimo prvu podelu ulaznog skupa.

Zadatak 1 – Rešenje

Atribut „toplokrvni“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „toplokrvni“ DA je veličine 5 (noj, gavran, albatros, delfin, koala). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 3 (noj, gavran, albatros), a vrednosti NE je 2 (delfin, koala). Entropija takvog podskupa je:

$$\textit{Entropy}(3\ \text{DA}, 2\ \text{NE}) = -\ 3/5 * \log_2(3/5) - 2/5 * \log_2(2/5) = 0.97095$$

Podskup skupa S u kojem je vrednost atributa „toplokrvni“ NE je veličine 1 (krokodil). Samo vrednost DA za izlazni atribut „leže jaja“ se u takvom podskupu nalazi. Entropija takvog podskupa je:

$$\textit{Entropy}(1\ \text{DA}, 0\ \text{NE}) = -\ 1/1 * \log_2(1/1) - 0/1 * \log_2(0/1) = 0$$

Sada računamo dobit informacija za podelu po atributu „toplokrvna“:

$$\textit{Gain}(\text{S}, \text{toplokrvna}) = 0.91829 - (5/6 * 0.97095 + 1/6 * 0) = 0.10916$$

Zadatak 1 – Rešenje

Atribut „pernata“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „pernata“ DA je veličine 3 (noj, gavran, albatros). Samo vrednost DA za izlazni atribut „leže jaja“ se u takvom podskupu nalazi. Entropija takvog podskupa je:

$$Entropy(3\ DA, 0\ NE) = -\ 3/3 * \log_2(3/3) - 0/3 * \log_2(0/3) = 0$$

Podskup skupa S u kojem je vrednost atributa „pernata“ NE je veličine 3 (krokodil, delfin, koala). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 1 (krokodil), a vrednosti NE je 2 (delfin, koala). Entropija takvog podskupa je:

$$Entropy(1\ DA, 2\ NE) = -\ 1/3 * \log_2(1/3) - 2/3 * \log_2(2/3) = 0.91829$$

Sada računamo dobit informacija za podelu po atributu „pernata“:

$$Gain(S, pernata) = 0.91829 - (3/6 * 0 + 3/6 * 0.91829) = 0.45914$$

Zadatak 1 – Rešenje

Atribut „ima krzno“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „ime krzno“ DA je veličine 1 (koala). Samo vrednost NE za izlazni atribut „leže jaja“ se u takvom podskupu nalazi. Entropija takvog podskupa je:

$$Entropy(0\ DA, 1\ NE) = -\ 0/1 * \log_2(0/1) - 1/1 * \log_2(1/1) = 0$$

Podskup skupa S u kojem je vrednost atributa „ima krzno“ NE je veličine 5 (noj, krokodil, gavran, albatros, delfin). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 4 (noj, krokodil, gavran, albatros), a vrednosti NE je 1 (delfin). Entropija takvog podskupa je:

$$Entropy(4\ DA, 1\ NE) = -\ 4/5 * \log_2(4/5) - 1/5 * \log_2(1/5) = 0.7219$$

Sada računamo dobit informacija za podelu po atributu „ima krzno“:

$$Gain(S, \text{ima krzno}) = 0.91829 - (1/6 * 0 + 5/6 * 0.7219) = 0.3167$$

Zadatak 1 – Rešenje

Atribut „pliva“ ima dve vrednosti: DA i NE.

Podskup skupa S u kojem je vrednost atributa „pliva“ DA je veličine 2 (krokodil, delfin). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 1 (krokodil), a vrednosti NE je 1 (delfin). Entropija takvog podskupa je:

$$Entropy(1\ DA, 1\ NE) = -1/2 * \log_2(1/2) - 1/2 * \log_2(1/2) = 1$$

Podskup skupa S u kojem je vrednost atributa „pliva“ NE je veličine 4 (noj, gavran, albatros, koala). Učestalost pojavljivanja vrednosti DA za izlazni atribut „leže jaja“ u takvom podskupu je 3 (noj, gavran, albatros), a vrednosti NE je 1 (koala). Entropija takvog podskupa je:

$$Entropy(3\ DA, 1\ NE) = -3/4 * \log_2(3/4) - 1/4 * \log_2(1/4) = 0.81127$$

Sada računamo dobit informacija za podelu po atributu „pliva“:

$$Gain(S, pliva) = 0.91829 - (2/6 * 1 + 4/6 * 0.81127) = 0.04411$$

Zadatak 1 – Rešenje

Ostale dobiti smo izračunali ranije. Tražimo najveću od njih.

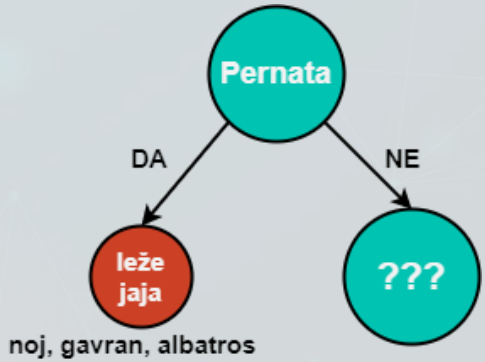
$$Gain(S, \text{toplokrvna}) = 0.91829 - (5/6 * 0.97095 + 1/6 * 0) = 0.10916$$

$$Gain(S, \text{pernata}) = 0.91829 - (3/6 * 0 + 3/6 * 0.91829) = 0.45914$$

$$Gain(S, \text{ima krzno}) = 0.91829 - (1/6 * 0 + 5/6 * 0.7219) = 0.3167$$

$$Gain(S, \text{pliva}) = 0.91829 - (2/6 * 1 + 4/6 * 0.81127) = 0.04411$$

Najveću dobit ima podela po atributu „pernata“ pa taj atribut biramo za koreni čvor i skup podataka delimo na osnovu tog atributa.



Zadatak 1 – Rešenje

Nakon podele ulaznog skupa podataka po atributu „pernata“, dobijamo sledeći podskup.

Životinja	Toplokrvna	Ima krzno	Pliva	Leže jaja
Krokodil	Ne	Ne	Da	Da
Delfin	Da	Ne	Da	Ne
Koala	Da	Da	Ne	Ne

Sada ponavljamo ceo postupak sa novim skupom S’. Računamo entropiju takvog skupa:

$$Entropy(1\ DA, 2\ NE) = -\ 1/3 * \log_2(1/3) - 2/3 * \log_2(2/3) = 0.91829$$

Za atribut „toplokrvna“ računamo:

$$Entropy(0\ DA, 2\ NE) = -\ 0/2 * \log_2(0/2) - 2/2 * \log_2(2/2) = 0$$

$$Entropy(1\ DA, 0\ NE) = -\ 1/1 * \log_2(1/1) - 0/1 * \log_2(0/1) = 0$$

$$Gain(S', toplokrvna) = 0.91829 - (2/3 * 0 + 1/3 * 0) = 0.91829$$

Zadatak 1 – Rešenje

Za atribut „ima krzno“ računamo:

$$Entropy(0\ DA, 1\ NE) = -\ 0/1 * \log_2(0/1) - 1/1 * \log_2(1/1) = 0$$

$$Entropy(1\ DA, 1\ NE) = -\ 1/2 * \log_2(1/2) - 1/2 * \log_2(1/2) = 1$$

$$Gain(S', \text{ima krzno}) = 0.91829 - (1/3 * 0 + 2/3 * 1) = 0.25162$$

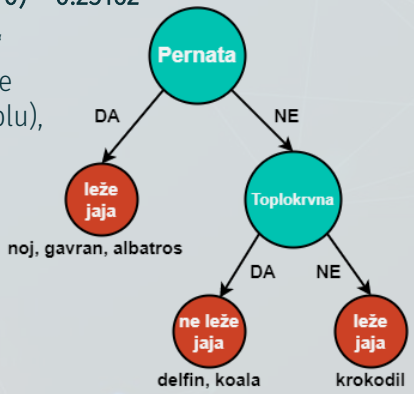
Za atribut „pliva“ računamo:

$$Entropy(1\ DA, 1\ NE) = -\ 1/2 * \log_2(1/2) - 1/2 * \log_2(1/2) = 1$$

$$Entropy(0\ DA, 1\ NE) = -\ 0/1 * \log_2(0/1) - 1/1 * \log_2(1/1) = 0$$

$$Gain(S', \text{pliva}) = 0.91829 - (2/3 * 1 + 1/3 * 0) = 0.25162$$

Najbolju dobit informacija ima podela po atributu „toplokrvna“ pa taj čvor biramo za sledeći čvor. Kako skup podataka ne može dalje da se deli (stigli smo do listova po svim putanjama u stablu), konstrukcija stabla je završena.



Iako je dobijeno isto stablo, u opštem slučaju to ne mora da važi. Gini indeks generalno brže formira stablo zbog lakšeg matematičkog računa, dok dobit informacija daje za nijansu bolje rezultate prilikom testiranja. Češće nam je bitnija brzina treniranja modela, pa je Gini indeks danas popularniji izbor.

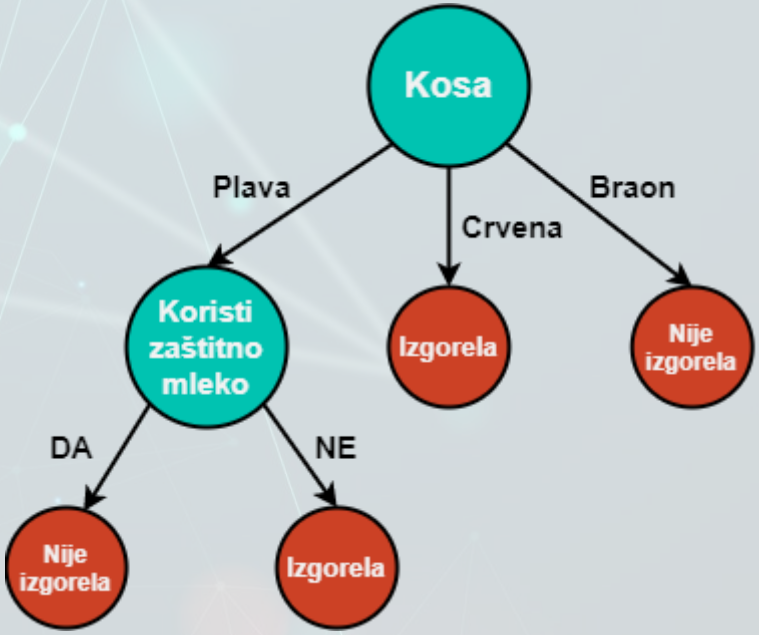
Zadatak za samostalnu vežbu – Sunčanje

★★★☆☆

Konstruisati stablo odlučivanja za određivanje da li će osoba izgoreti u toku sunčanja na osnovu sledećeg skupa podataka i njihovih karakteristika.

Ime	Kosa	Visina	Masa	Koristi zaštitnu kremu	Izgoreo/la
Aleksa	Plava	Prosečna	Laka	Ne	Da
Bojan	Plava	Visoka	Prosečna	Da	Ne
Ceca	Braon	Niska	Prosečna	Da	Ne
Darko	Plava	Niska	Prosečna	Ne	Da
Ema	Crvena	Prosečna	Teška	Ne	Da
Filip	Braon	Visoka	Teška	Ne	Ne
Goran	Braon	Prosečna	Teška	Ne	Ne
Helena	Plava	Niska	Laka	Da	Ne

Zadatak za samostalnu vežbu – Rešenje



Zadatak 2 – Simpsonovi

★★★☆☆

Konstruisati stablo odlučivanja za određivanje da li je osoba muškog ili ženskog pola na osnovu sledećeg skupa podataka i njihovih karakteristika.

Ime	Dužina kose	Masa	Starost	Pol
Homer	0	113	36	M
Mardž	30	68	34	Ž
Bart	5	40	10	M
Lisa	15	35	8	Ž
Megi	10	9	1	Ž
Ejb	3	77	70	M
Selma	20	72	41	Ž
Oto	30	81	38	M
Krasti	15	90	45	M

Zadatak 2 – Simpsonovi

★★★☆☆

Nakon konstrukcije stabla, izvršiti testiranje nad sledećim skupom podataka i izračunati tačnost, preciznost i odziv.

Ime	Dužina kose	Masa	Starost	Pol
Barni	5	120	33	M
Apu	15	74	32	M
Ned	8	77	45	M
Edna	15	74	40	Ž

Zadatak 2 – Rešenje

Skup podataka transformišemo i prilagođavamo radu algoritma.

Ime	Kosa veća od 12	Masa veća od 75	Starost veća od 40	Pol
Homer	NE	DA	NE	M
Mardž	DA	NE	NE	Ž
Bart	NE	NE	NE	M
Lisa	DA	NE	NE	Ž
Megi	NE	NE	NE	Ž
Ejb	NE	DA	DA	M
Selma	DA	NE	DA	Ž
Oto	DA	DA	NE	M
Krasti	DA	DA	DA	M

Zadatak 2 – Rešenje

Kako deliti skup kada je atribut kontinualnog tipa?

Jedna varijanta: naći granične vrednosti na osnovu koje bi se skup podelio na podskupove takve da je dobit informacija za takvu podelu najbolja moguća (problem – koliko „sitno“ tražiti granične vrednosti?).

Npr. Dužina kose

Isprobavamo vrednosti od 0.5 do 29.5. Koji korak uzeti? Ne previše mali, a ne ni previše veliki.

Naći sve granične vrednosti u kojima se menja izlazni atribut „pol“ (problem preobučavanja).

Druga varijanta: naći graničnu vrednost takvu da se podelom skupa dobiju podskupovi sa jednakim (ili skoro jednakim) brojem elemenata (bolje performanse – potencijalno lošija podela).

Npr. za dužinu kose bismo mogli da izaberemo graničnu vrednost 12.

Zadatak 2 – Rešenje

Računamo entropiju početnog skupa:

$$Entropy(4\ Ž, 5\ M) = -\ 4/9 * \log_2(4/9) - 5/9 * \log_2(5/9) = 0.9911$$

Za atribut „dužina kose“ biramo graničnu vrednost 12 i računamo:

$$<= 12 : Entropy(1\ Ž, 3\ M) = -\ 1/4 * \log_2(1/4) - 3/4 * \log_2(3/4) = 0.8113$$

$$> 12 : Entropy(3\ Ž, 2\ M) = -\ 3/5 * \log_2(3/5) - 2/5 * \log_2(2/5) = 0.9710$$

$$Gain(S, \text{dužina kose} <= 12) = 0.9911 - (4/9 * 0.8113 + 5/9 * 0.9710) = 0.0911$$

Da li bi nam izbor neke druge granične vrednosti dao veću dobit?

Za atribut „masa“ biramo graničnu vrednost 75 i računamo:

$$<= 75 : Entropy(4\ Ž, 1\ M) = -\ 4/5 * \log_2(4/5) - 1/5 * \log_2(1/5) = 0.7219$$

$$> 75 : Entropy(0\ Ž, 4\ M) = -\ 0/4 * \log_2(0/4) - 4/4 * \log_2(4/4) = 0$$

$$Gain(S, \text{masa} <= 75) = 0.9911 - (5/9 * 0.7219 + 4/9 * 0) = 0.5900$$

Da li bi nam izbor neke druge granične vrednosti dao veću dobit?

Zadatak 2 – Rešenje

Za atribut „starost“ biramo graničnu vrednost 40 i računamo:

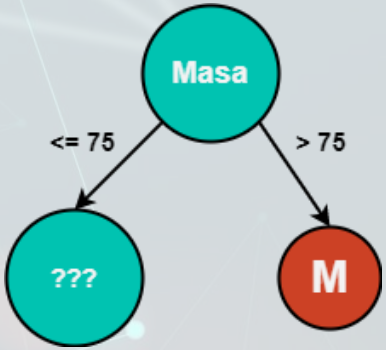
$<= 40$: $Entropy(3\ Ž, 3\ M) = -\ 3/6 * \log_2(3/6) - 3/6 * \log_2(3/6) = 1$

> 40 : $Entropy(1\ Ž, 2\ M) = -\ 1/3 * \log_2(1/3) - 2/3 * \log_2(2/3) = 0.9183$

$Gain(S, \text{starost} <= 40) = 0.9911 - (6/9 * 1 + 3/9 * 0.9183) = 0.0183$

Da li bi nam izbor neke druge granične vrednosti dao veću dobit?

Najveću dobit nam daje podela po atributu „masa“.

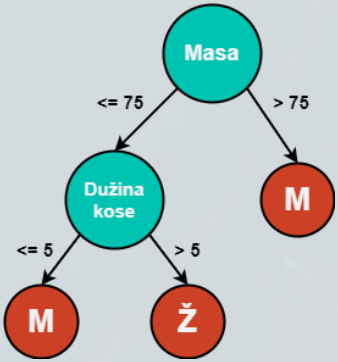


Zadatak 2 – Rešenje

Nastavljajući proceduru, podelu možemo na dalje izvršiti po atributu „dužina kose“ sa granicom 5 čime dolazimo do kraja.

Vrši se testiranje nad sledećim skupom podataka.

Ime	Dužina kose	Masa	Starost	Pol	Predikcija
Barni	5	120	33	M	M
Apu	15	74	32	M	Ž
Ned	8	77	45	M	M
Edna	15	74	40	Ž	Ž



Barni, Ned i Edna su ispravno klasifikovani, dok je Apu pogrešno klasifikovan.

Tačnost modela je stoga 75%.

Preciznost za klasu M je 100%, dok je odziv 67%.

Preciznost za klasu Ž je 50%, dok je odziv 100%.

REGRESIVNA STABLA

Šta je i kako se određuje standardno kvadratno odstupanje?

Ako je dat skup A veličine |A| sa ulaznim vrednostima x, tada je srednja vrednost skupa A

$$x_{sr} = \frac{\sum x}{|A|}$$

Standardno kvadratno odstupanje S(A), skupa A je:

$$S(A) = \sqrt{\frac{\sum(x - x_{sr})^2}{|A|}}$$

Koeficijent varijacije CV(A), skupa A je:

$$CV(A) = \frac{S(A)}{X_{sr}} * 100\%$$

REGRESIVNA STABLA

Šta je i kako se određuje standardno kvadratno odstupanje?

Standardno odstupanje se koristi pri izboru atributa za podelu.

Kako regresivna stabla imaju kontinualnu izlaznu vrednost, potrebna je vrednost na osnovu koje se određuje kriterijum zaustavljanja konstrukcije stabla. U tu svrhu se koristi koeficijent varijacije, veličina skupa ili oba.

Srednja vrednost se koristi za vrednost listova.

Kombinovano standardno kvadratno odstupanje za dva atributa (ulazni X i izlazni T) koje se koristi kao entropija kod klasifikacionih stabala se računa na sledeći način:

$$S(T, X) = \sum_{c \in X} P(c) S(c)$$

gde je *c* jedna od vrednosti atributa X, *P(c)* učestalost pojavljivanja vrednosti *c* u skupu, a *S(c)* standardno kvadratno odstupanje podskupa glavnog skupa dobijenog uzimanjem onih redova čije su vrednosti atributa X jednake *c*.

REGRESIVNA STABLA

Šta je i kako se određuje redukcija standardnog odstupanja?

Redukcija standardnog odstupanja se dobija kao razlika standardnih odstupanja pre i posle podele po nekom atributu i odgovara očekivanoj dobiti kod klasifikacionih stabala.

Atribut koji se koristi za podelu se bira kao onaj sa maksimalnom redukcijom.

Proces se iterativno ponavlja dok nije ispunjen kriterijum zaustavljanja. Uglavnom je kriterijum zaustavljanja maksimalan koeficijent varijacije koji podskup može da ima ili maksimalan broj instanci u podskupu.

Zadatak 3 – Igranje golfa

★★★★☆

Konstruisati stablo odlučivanja za određivanje koliko vremena će društvo igrati golf na osnovu sledećeg skupa podataka i njihovih karakteristika. Kriterijumi zaustavljanja su 10% za koeficijent varijacije i 3 za veličinu skupa.

Vreme	Temperatura	Vlažnost	Vetar	Trajanje igre
Kiša	Vruće	Visoka	Ne	25
Kiša	Vruće	Visoka	Da	30
Oblačno	Vruće	Visoka	Ne	46
Sunčano	Blago	Visoka	Ne	45
Sunčano	Hladno	Normalna	Ne	52
Sunčano	Hladno	Normalna	Da	23
Oblačno	Hladno	Normalna	Da	43
Kiša	Blago	Visoka	Ne	35
Kiša	Hladno	Normalna	Ne	38
Sunčano	Blago	Normalna	Ne	46
Kiša	Blago	Normalna	Da	48
Oblačno	Blago	Visoka	Da	52
Oblačno	Vruće	Normalna	Ne	44
Sunčano	Blago	Visoka	Da	30

Zadatak 3 – Rešenje

Veličina skupa |trajanje igre| = 14

Srednja vrednost izlaza X_{sr} = 39.8

Standardno odstupanje izlaza S(trajanje igre) = 9.32

Koeficijent varijacije izlaza CF(trajanje igre) = 23%

I veličina skupa i koeficijent varijacije su veći od kriterijuma zaustavljanja te se algoritam nastavlja.

Sada je potrebno pronaći standardno odstupanje za sve podskupove dobijene podelom svakog od atributa: vreme, temperatura, vlažnost, vetar. Atribut sa najvećom redukcijom će biti atribut na osnovu kojeg vršimo prvu podelu ulaznog skupa.

Trajanje igre
25
30
46
45
52
23
43
35
38
46
48
52
44
30

Zadatak 3 – Rešenje

Atribut „vreme“ ima tri vrednosti: Oblačno, Kiša i Sunčano.

Podskup skupa S u kojem je vrednost atributa „vreme“ Oblačno je veličine 4.
Srednja vrednost izlazne vrednosti takvog podskupa je 46.25.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 3.49.

Podskup skupa S u kojem je vrednost atributa „vreme“ Kiša je veličine 5.
Srednja vrednost izlazne vrednosti takvog podskupa je 35.2.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 7.78.

Podskup skupa S u kojem je vrednost atributa „vreme“ Sunčano je veličine 5.
Srednja vrednost izlazne vrednosti takvog podskupa je 39.2.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 10.87.

Kombinovano standardno odstupanje za atribute „trajanje igre“ i „vreme“ je:

$$S(\text{trajanje igre, vreme}) = 4/14 * 3.49 + 5/14 * 7.78 + 5/14 * 10.87 = 7.66$$

Redukcija standardnog odstupanja za ovakvu podelu je:

$$SDR(\text{trajanje igre, vreme}) = S(\text{trajanje igre}) - S(\text{trajanje igre, vreme})$$

$$SDR(\text{trajanje igre, vreme}) = 9.32 - 7.66 = 1.66$$

Vreme	Trajanje igre
Oblačno	46
Oblačno	43
Oblačno	52
Oblačno	44

Vreme	Trajanje igre
Kiša	25
Kiša	30
Kiša	35
Kiša	38
Kiša	48

Vreme	Trajanje igre
Sunčano	45
Sunčano	52
Sunčano	23
Sunčano	46
Sunčano	30

Zadatak 3 – Rešenje

Atribut „temperatura“ ima tri vrednosti: Hladno, Vruće, Blago.

Podskup skupa S u kojem je vrednost atributa „temperatura“ Hladno je veličine 4.
Srednja vrednost izlazne vrednosti takvog podskupa je 39.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 10.51.

Podskup skupa S u kojem je vrednost atributa „temperatura“ Vruće je veličine 4.
Srednja vrednost izlazne vrednosti takvog podskupa je 36.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 8.95.

Podskup skupa S u kojem je vrednost atributa „temperatura“ Blago je veličine 6.
Srednja vrednost izlazne vrednosti takvog podskupa je 42.67.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 7.65.

Kombinovano standardno odstupanje za atribute „trajanje igre“ i „temperatura“ je:

$$S(\text{trajanje igre, temperatura}) = 4/14 * 10.51 + 4/14 * 8.95 + 6/14 * 7.65 = 8.83$$

Redukcija standardnog odstupanja za ovakvu podelu je:

$$SDR(\text{trajanje igre, temperatura}) = S(\text{trajanje igre}) - S(\text{trajanje igre, temperatura})$$

$$SDR(\text{trajanje igre, temperatura}) = 9.32 - 8.83 = 0.49$$

Temperatura	Trajanje igre
Hladno	52
Hladno	23
Hladno	43
Hladno	38

Temperatura	Trajanje igre
Vruće	25
Vruće	30
Vruće	46
Vruće	44

Temperatura	Trajanje igre
Blago	45
Blago	35
Blago	46
Blago	48
Blago	52
Blago	30

Zadatak 3 – Rešenje

Atribut „vlažnost“ ima dve vrednosti: Visoka, Normalna.

Podskup skupa S u kojem je vrednost atributa „vlažnost“ Visoka je veličine 7.
Srednja vrednost izlazne vrednosti takvog podskupa je 37.57.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 9.36.

Podskup skupa S u kojem je vrednost atributa „vlažnost“ Normalna je veličine 7.
Srednja vrednost izlazne vrednosti takvog podskupa je 42.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 8.73.

Kombinovano standardno odstupanje za atribute „trajanje igre“ i „vlažnost“ je:

S(trajanje igre, vlažnost) = 7/14 * 9.36 + 7/14 * 8.73 = 9.05

Redukcija standardnog odstupanja za ovakvu podelu je:

SDR(trajanje igre, vlažnost) = S(trajanje igre) – S(trajanje igre, vlažnost)

SDR(trajanje igre, vlažnost) = 9.32 – 9.04 = 0.28

Vlažnost	Trajanje igre
Visoka	25
Visoka	30
Visoka	46
Visoka	45
Visoka	35
Visoka	52
Visoka	30

Vlažnost	Trajanje igre
Normalna	52
Normalna	23
Normalna	43
Normalna	38
Normalna	46
Normalna	48
Normalna	44

Zadatak 3 – Rešenje

Atribut „vetar“ ima dve vrednosti: Da, Ne.

Podskup skupa S u kojem je vrednost atributa „vetar“ Da je veličine 6.
Srednja vrednost izlazne vrednosti takvog podskupa je 37.67.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 10.59.

Podskup skupa S u kojem je vrednost atributa „vetar“ Ne je veličine 8.
Srednja vrednost izlazne vrednosti takvog podskupa je 41.375.
Standardno odstupanje izlazne vrednosti u takvom podskupu je 7.87.

Kombinovano standardno odstupanje za atribute „trajanje igre“ i „vetar“ je:

S(trajanje igre, vetar) = 6/14 * 10.59 + 8/14 * 7.87 = 9.04

Redukcija standardnog odstupanja za ovakvu podelu je:

SDR(trajanje igre, vetar) = S(trajanje igre) – S(trajanje igre, vetar)

SDR(trajanje igre, vetar) = 9.32 – 9.04 = 0.28

SDR(trajanje igre, vlažnost) = 0.28

SDR(trajanje igre, temperatura) = 0.49

SDR(trajanje igre, vreme) = 1.66

Najveći SDR ima atribut „vreme“ pa će nad njim da se vrši podela.

Vetar	Trajanje igre
Da	30
Da	23
Da	43
Da	48
Da	52
Da	30

Vetar	Trajanje igre
Ne	25
Ne	46
Ne	45
Ne	52
Ne	35
Ne	38
Ne	46
Ne	44

Zadatak 3 – Rešenje

Vreme	Temperatura	Vlažnost	Vetar	Trajanje igre
Kiša	Vruće	Visoka	Ne	25
Kiša	Vruće	Visoka	Da	30
Kiša	Blago	Visoka	Ne	35
Kiša	Hladno	Normalna	Ne	38
Kiša	Blago	Normalna	Da	48

Veličina skupa $|\text{trajanje igre}| = 5$

Srednja vrednost izlaza $X_{sr} = 35.2$

Standardno odstupanje izlaza $S(\text{trajanje igre}) = 7.78$

Koeficijent varijacije izlaza $CF(\text{trajanje igre}) = 22.10\%$

I veličina skupa i koeficijent varijacije su veći od kriterijuma zaustavljanja te se algoritam nastavlja po ovom podskupu.

Zadatak 3 – Rešenje

Vreme	Temperatura	Vlažnost	Vetar	Trajanje igre
Oblačno	Vruće	Visoka	Ne	46
Oblačno	Hladno	Normalna	Da	43
Oblačno	Blago	Visoka	Da	52
Oblačno	Vruće	Normalna	Ne	44

Veličina skupa **|trajanje igre| = 4**

Srednja vrednost izlaza X_{sr} = 46.25

Standardno odstupanje izlaza **S(trajanje igre) = 3.49**

Koeficijent varijacije izlaza **CF(trajanje igre) = 7.54%**

Koeficijent varijacije je manji od kriterijuma zaustavljanja te se algoritam ne nastavlja po ovom podskupu. Vrednost lista odgovara srednjoj vrednosti izlaza 46.25.

Zadatak 3 – Rešenje

Vreme	Temperatura	Vlažnost	Vetar	Trajanje igre
Sunčano	Blago	Visoka	Ne	45
Sunčano	Hladno	Normalna	Ne	52
Sunčano	Hladno	Normalna	Da	23
Sunčano	Blago	Normalna	Ne	46
Sunčano	Blago	Visoka	Da	30

Veličina skupa $|\text{trajanje igre}| = 5$

Srednja vrednost izlaza $X_{sr} = 39.2$

Standardno odstupanje izlaza $S(\text{trajanje igre}) = 10.87$

Koeficijent varijacije izlaza $CF(\text{trajanje igre}) = 27.73\%$

I veličina skupa i koeficijent varijacije su veći od kriterijuma zaustavljanja te se algoritam nastavlja po ovom podskupu.

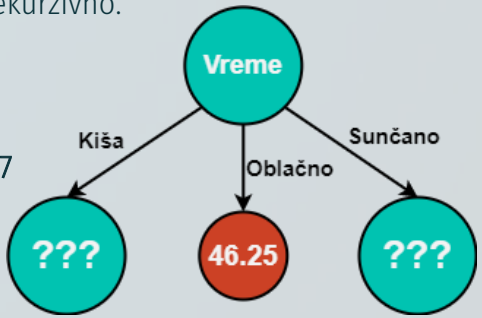
Zadatak 3 – Rešenje

Algoritam se za preostala dva podskupa nastavlja rekurzivno.

Levi podskup (Vreme == Sunčano):

Veličina skupa |trajanje igre| = 5

Standardno odstupanje izlaza S(trajanje igre) = 10.87



SDR(trajanje igre, temperatura) = 0.07

SDR(trajanje igre, vlažnost) = 0.37

SDR(trajanje igre, vetar) = 7.62

Za podelu se dalje bira podela po vetru. Svi podskupovi imaju veličinu manju od 3 te je podela u tom delu završena. Listovi dobijaju vrednosti koje odgovaraju srednjim vrednostima odgovarajućih podskupova.

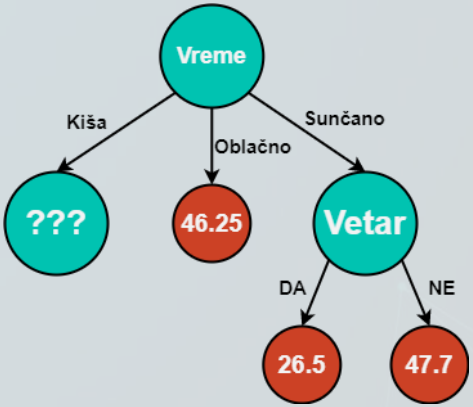
Zadatak 3 – Rešenje

Algoritam se za preostala dva podskupa nastavlja rekurzivno.

Levi podskup (Vreme == Kiša):

Veličina skupa **|trajanje igre| = 5**

Standardno odstupanje izlaza **S(trajanje igre) = 7.78**



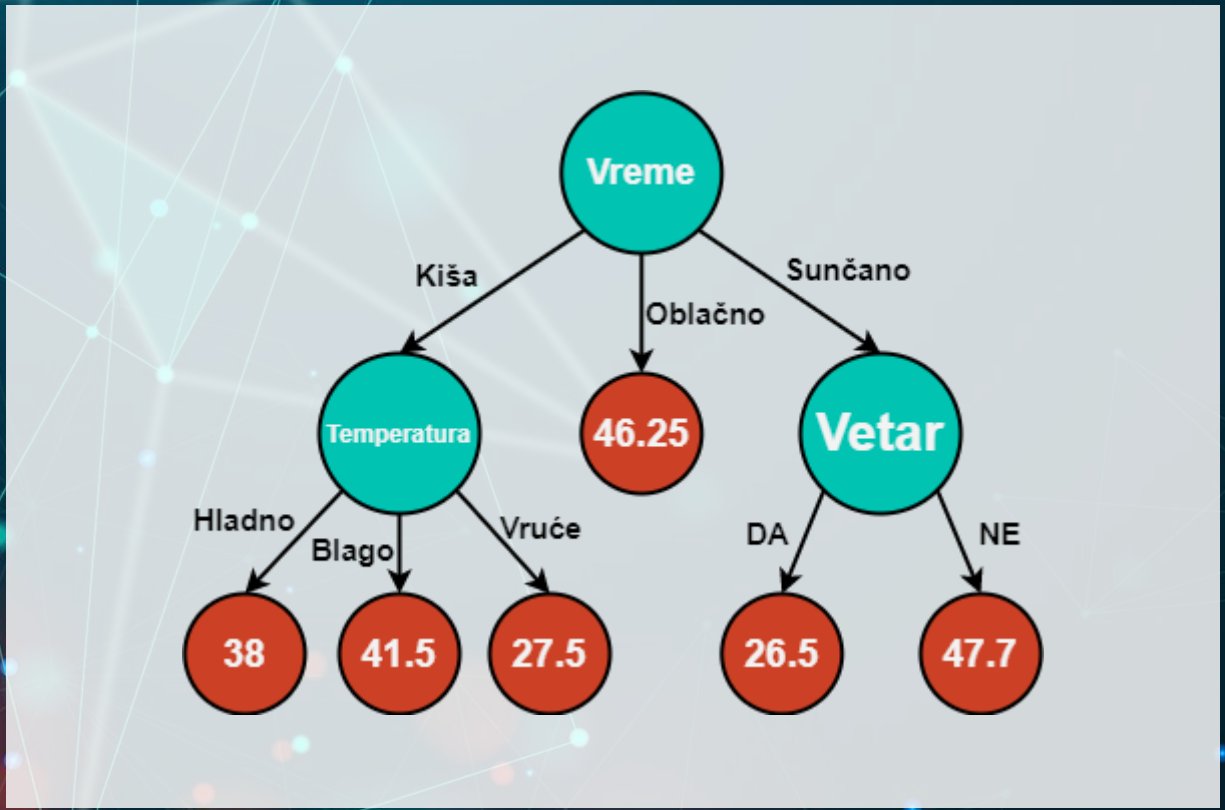
SDR(trajanje igre, temperatura) = **4.18**

SDR(trajanje igre, vlažnost) = 3.32

SDR(trajanje igre, vetar) = 0.82

Za podelu se dalje bira podela po temperaturi. Svi podskupovi imaju veličinu manju od 3 te je podela u tom delu završena. Listovi dobijaju vrednosti koje odgovaraju srednjim vrednostima odgovarajućih podskupova.

Zadatak 3 – Rešenje



Zadatak za samostalnu vežbu – Nekretnine

★★★★☆

Konstruisati stablo odlučivanja za određivanje cene stanova u Beogradu na osnovu sledećeg skupa podataka i njihovih karakteristika. Kriterijumi zaustavljanja su 10% za koeficijent varijacije i 2 za veličinu skupa.

Kvadratura	Broj soba	Opština	Cena
110	3	Palilula	210.000
73	3	Novi Beograd	129.000
65	2	Novi Beograd	171.000
74	3.5	Palilula	159.000
70	2.5	Novi Beograd	139.500
30	1	Palilula	35.000
43	1.5	Palilula	105.000
110	4	Stari grad	219.000
52	2	Stari grad	160.000

NASUMIČNA ŠUMA (RANDOM FOREST)

Iako su brza za komputaciju, Stabla odlučivanja daju generalno lošije rezultate i podložna su problemu preobučavanja (engl. *overfitting* - nepoželjno svojstvo koje se javlja kada model mašinskog učenja daje tačna predviđanja za trening podatke, ali ne i za nove podatke).

Nasumična šuma je algoritam mašinskog učenja koji kombinuje rezultate nekoliko stabala odlučivanja kako bi donela konačan rezultat, povećavajući vreme komputacije, ali smanjujući šanse za preobučavanjem.

Primer: Marko nije siguran koji fakultet da upiše pa se savetuje sa roditeljima, bratom, sestrom i prijateljima. Marko im je izložio prednosti i mane oba fakulteta koja su mu u užem izboru i na osnovu toga svako mu je dao savet šta da upiše. Na osnovu svih saveta, Marko donosi svoju odluku. Ovakav način donošenja odluke daje bolje rezultate od donošenja odluke samo na osnovu jednog saveta (npr. savet brata).

NASUMIČNA ŠUMA (RANDOM FOREST)

Postoje dva načina agregiranja više odluka u jednu:

- *Bagging* – Iz početnog skupa podataka se izvlače podskupovi (koji smeju da imaju deljene primerke ili primerke koji se ponavljaju) na osnovu kojih se treniraju modeli paralelno. Svaki model se koristi za predikciju i daje svoj izlaz, a konačan izlaz se dobija kao najzastupljeniji izlaz/klasa (klasifikacija) ili kao prosečna vrednost svih izlaza (regresija).

- *Boosting* – Modeli se ne kreiraju paralelno već sekvencijalno, tako što se kreira prvi model (uglavnom nasumičan) nad kojim se vrši predikcija. Iz skupa podataka se izdvajaju svi primerci koji su pogrešno prediktovani prvim modelom i koriste se zajedno sa novim podskupom podataka da se istrenira drugi model, itd. Nakon određenog vremena, poslednji model će biti najnapredniji, jer je koristio informacije o greškama svih prethodnih modela.

Nasumična šuma koristi *bagging* tehniku pri čemu za treniranje svakog podstabla ne koristi nužno isti skup ulaznih atributa, već se i za njih bira nasumični podskup (Marko nije svima dao iste informacije o fakultetima).

Zadatak 4 – Životinje koje ležu jaja

★★★☆☆

Konstruisati nasumičnu šumu sa tri stabla za određivanje da li životinja leže jaja na osnovu sledećeg skupa podataka i njihovih karakteristika. Nasumičnim odabirom skup je podeljen na tri podskupa S1(noj, krokodil, gavran, delfin), S2(noj, albatros, delfin, koala) i S3(noj, gavran, albatros, delfin). Stabla koriste atribute A1(toplokrvna, pernata, ima krzno), A2(pernata, ima krzno, pliva) i A3(toplokrvna, ima krzno, pliva) i Gini podelu.

Nezavisni atributi					**Atribut odluke**
Životinja	Toplokrvna	Pernata	Ima krzno	Pliva	Leže jaja
Noj	Da	Da	Ne	Ne	Da
Krokodil	Ne	Ne	Ne	Da	Da
Gavran	Da	Da	Ne	Ne	Da
Albatros	Da	Da	Ne	Ne	Da
Delfin	Da	Ne	Ne	Da	Ne
Koala	Da	Ne	Da	Ne	Ne

Zadatak 4 – Rešenje

S1

Životinja	Toplokrvna	Pernata	Ima krzno	Leže jaja
Noj	Da	Da	Ne	Da
Krokodil	Ne	Ne	Ne	Da
Gavran	Da	Da	Ne	Da
Delfin	Da	Ne	Ne	Ne

$Gini(toplokrvna=DA) = 1 - (2/3)^2 - (1/3)^2 = 0.44$

$Gini(toplokrvna=NE) = 1 - (1/1)^2 - (0/1)^2 = 0$

$Gini(S1, toplokrvna) = 3/4 * 0.44 + 1/4 * 0 = 0.33$

$Gini(pernata=DA) = 1 - (2/2)^2 - (0/2)^2 = 0$

$Gini(pernata=NE) = 1 - (1/2)^2 - (1/2)^2 = 0.5$

$Gini(S1, pernata) = 2/4 * 0 + 2/4 * 0.5 = 0.25$

Po krznu nije moguće podeliti ovaj skup podataka.

Podela se vrši po atributu „pernata“, a potom po preostalom atributu „toplokrvna“.

Zadatak 4 – Rešenje

S2

Životinja	Pernata	Ima krzno	Pliva	Leže jaja
Noj	Da	Ne	Ne	Da
Albatros	Da	Ne	Ne	Da
Delfin	Ne	Ne	Da	Ne
Koala	Ne	Da	Ne	Ne

$Gini(pernata=DA) = 1 - (2/2)^2 - (0/2)^2 = 0$

$Gini(pernata=NE) = 1 - (0/2)^2 - (2/2)^2 = 0$

$Gini(S1, pernata) = 2/4 * 0 + 2/4 * 0 = 0$

$Gini(krzno=DA) = 1 - (0/1)^2 - (1/1)^2 = 0$

$Gini(krzno=NE) = 1 - (2/3)^2 - (1/3)^2 = 0.44$

$Gini(S1, krzno) = 1/4 * 0 + 3/4 * 0.44 = 0.33$

$Gini(pliva=DA) = 1 - (0/1)^2 - (1/1)^2 = 0$

$Gini(pliva=NE) = 1 - (2/3)^2 - (1/3)^2 = 0.44$

$Gini(S1, pliva) = 1/4 * 0 + 3/4 * 0.44 = 0.33$

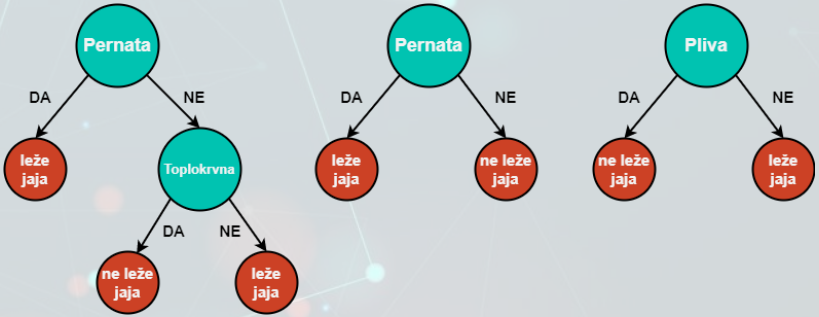
Podela se vrši po atributu „pernata“ čime dobijamo čiste skupove.

Zadatak 4 – Rešenje

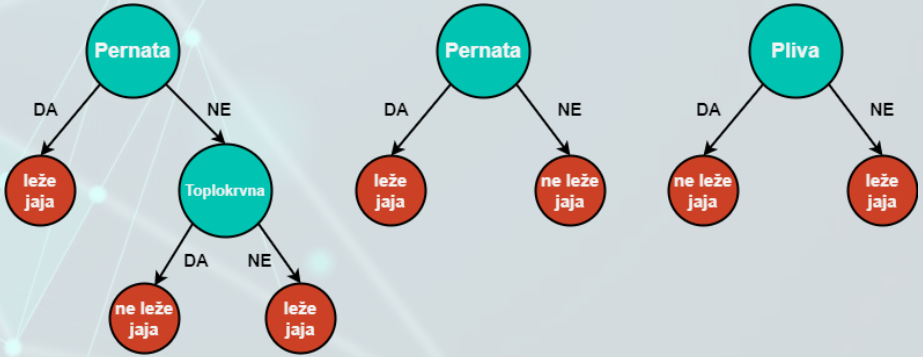
S3

Životinja	Toplokrvna	Ima krzno	Pliva	Leže jaja
Noj	Da	Ne	Ne	Da
Gavran	Da	Ne	Ne	Da
Albatros	Da	Ne	Ne	Da
Delfin	Da	Ne	Da	Ne

Podela se vrši po atributu „pliva“, jer je to jedini atribut koji može da napravi podelu. Takođe, ovakvom podelom dobijamo čiste skupove. Time dobijamo sledeću nasumičnu šumu.



Zadatak 4 – Rešenje



Konstruisana je sledeća nasumična šuma. Sada možemo da izvršimo predikciju:

Životinja	Toplokrvna	Pernata	Ima krzno	Pliva	Leže jaja		
					S1	S2	S3
Čekić ajkula	Ne	Ne	Ne	Da	Da	Ne	Ne
Gušter	Ne	Ne	Ne	Ne	Da	Ne	Da

Da li bismo iste rezultate dobili koristeći stablo iz Zadatka 1?

LINEARNA REGRESIJA

Šta je linearna regresija i u kojim oblicima postoji?

Linearna regresija (*Linear regression*) je tip nadgledanih algoritama mašinskog učenja. Koristi se kod problema regresije.

Algoritam kreće od pretpostavke da zavisnost izlaznog podatka *y* od ulaznog podatka *x* ima oblik linearne kombinacije odlika i njihovih težina.

$$h(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

Cilj algoritma je pronaći težinske parametre modela w_0, w_1, ... tako da hipoteza h(x) bude bliska stvarnim izlaznim podacima *y* za sve ulazne parove (*x*, *y*).

U zavisnosti od broja ulaznih podataka razlikujemo jednostruku i višestruku linearnu regresiju. U zavisnosti od broja izlaznih podataka razlikujemo univarijantnu i multivarijantnu linearnu regresiju.

LINEARNA REGRESIJA

Na koji način izabrati parametre modela?

Parametre modela je potrebno izabrati tako da hipoteza što manje „odskače“ od stvarnih vrednosti *y*. U tu svrhu se definiše funkcija gubitka ***L(h(x), y)*** i funkcija greške ***J(w)*** koja predstavlja prosek vrednosti funkcije gubitka na svim podacima iz posmatranog skupa.

Obučavanje modela se svodi na izbor parametara takvih da je funkcija greške minimalna.

Kao funkcija greške najčešće se koristi prosek kvadrata odstupanja *h(x)* od *y* (*Mean Squared Error*):

$$L(h(x), y) = \frac{1}{2}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

LINEARNA REGRESIJA

Kako naći minimum funkcije greške?

Analitički pristup nalasku rešenja nalaženja minimuma funkcije greške može biti skupa operacija.

U praksi se najčešće koristi metoda gradijentnog spusta koji iterativno traži minimum funkcije računajući izvode za veći broj promenljivih (gradijente). Gradijent funkcije greške možemo definisati sa:

$$\nabla J(w) = \left(\frac{\partial J(w)}{\partial w_0}, \frac{\partial J(w)}{\partial w_1}, \dots, \frac{\partial J(w)}{\partial w_n}\right)$$

Parametre u svakoj iteraciji treba menjati tako da funkcija greške opada:

$$w_i = w_i - \alpha \frac{\partial J(w)}{\partial w_i}$$

Gde je α pozitivan ceo broj koji predstavlja korak učenja i određuje koliko veliki skokovi će se praviti niz gradijent i mora biti pažljivo odabran.

LINEARNA REGRESIJA

Kako naći minimum funkcije greške?

Ukoliko je korak učenja premali, postoji preveliki broj iteracija i moguće je zaglaviti se u lokalnom minimumu funkcije.

Ukoliko je korak učenja preveliki, moguće je preskočiti globalni minimum funkcije.

Gradijentni spust konvergira ka minimumu smanjenjem vrednosti parcijalnih izvoda. Kako je vrednost izvoda u minimumu jednaka nuli, u nekom trenutku se parametri neće više menjati. Međutim, u praksi, često se gradijentni spust obustavlja kada promena vrednosti parametra u jednom koraku bude jako mala (približna nuli), odnosno manja od unapred definisane vrednosti.

JEDNOSTRUKA LINEARNA REGRESIJA

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1 x^{(i)} - y^{(i)}\right)^2$$

$$\frac{\partial J(w)}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}\left(w_0 + w_1 x^{(i)} - y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial J(w)}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m}\left(w_0 + w_1 x^{(i)} - y^{(i)}\right)x^{(i)} = \frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}$$

JEDNOSTRUKA LINEARNA REGRESIJA

Ukoliko je skup podataka mali, rešavanjem parcijalnih izvoda i izjednačavanjem sa nulom, možemo analitički pronaći parametre modela:

$$w_1 = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

$$w_0 = \frac{\sum y - w_1(\sum x)}{n}$$

Zadatak 6 – Zavisnost uspeha od učenja

★★★☆☆

Na osnovu posmatranog skupa podataka o zavisnosti uspeha na ispitu od broja sati provedenih u učenju izvesti zaključak o uspehu ukoliko je student učio 4.5 sata koristeći linearnu regresiju i izračunati grešku.

Nezavisni atribut	Atribut odluke
Sati učenja	Broj poena na ispitu
1	30
2	45
3	51
4	57
5	60
6	65
7	70
8	71

Zadatak 6 – Rešenje

Zadatak ćemo rešiti analitičkim putem. Potrebno je izračunati sve varijacije suma koje su pojavljuju u formulama za parametre modela.

	Sati učenja	Broj poena na ispitu	xy	x^2	y^2
	1	30	30	1	900
	2	45	90	4	2025
	3	51	153	9	2601
	4	57	228	16	3249
	5	60	300	25	3600
	6	65	390	36	4225
	7	70	490	49	4900
	8	71	568	64	5041
Σ	**36**	**449**	**2249**	**204**	**26541**

Zadatak 6 – Rešenje

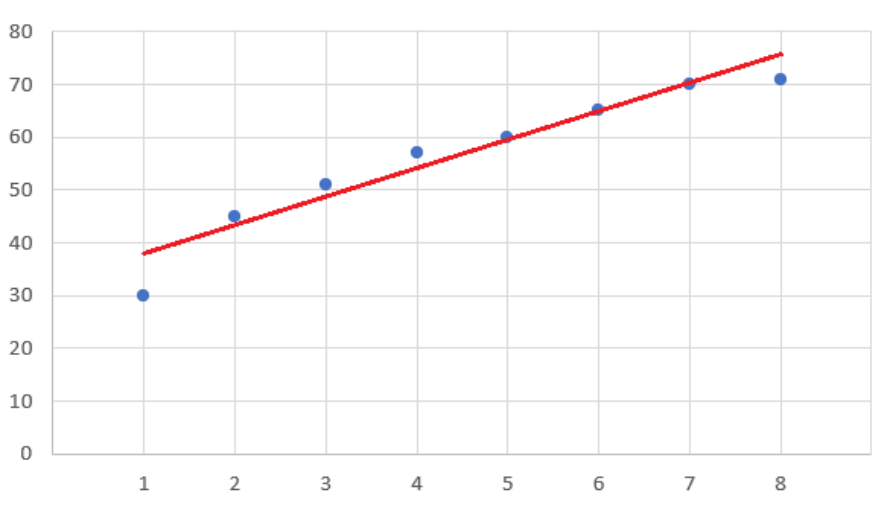
	Sati učenja	Broj poena na ispitu	xy	x^2	y^2
	1	30	30	1	900
	2	45	90	4	2025
	3	51	153	9	2601
	4	57	228	16	3249
	5	60	300	25	3600
	6	65	390	36	4225
	7	70	490	49	4900
	8	71	568	64	5041
Σ	36	449	2249	204	26541

$$w_1 = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} = \frac{8*2249 - 36*449}{8*204 - 36^2} = 5.4405$$

$$w_0 = \frac{\sum y - w_1(\sum x)}{n} = \frac{449 - 5.4405 * 36}{8} = 31.6429$$

Zadatak 6 – Rešenje

$$h(x) = 5.4405x + 31.6429$$



h(4.5) = 5.4405 * 4.5 + 31.6429 = 56.125

Zadatak 6 – Rešenje

$$h(x) = 5.4405x + 31.6429$$

	Sati učenja	Broj poena na ispitu	$h(x)$	$h(x) - y$	$(h(x) - y)^2$
	1	30	37.08	7.08	50.17
	2	45	42.52	-2.48	6.13
	3	51	47.96	-3.04	9.21
	4	57	53.40	-3.60	12.92
	5	60	58.85	-1.15	1.33
	6	65	64.29	-0.71	0.51
	7	70	69.73	-0.27	0.07
	8	71	75.17	4.17	17.36
Σ					97.72

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{16} * 97.72 = 6.11$$

Zadatak 6 – Rešenje

Drugi način za proveru valjanosti modela jeste koeficijent korelacije. Ako skup podataka nije linearan, model neće dati dobre rezultate. Koeficijent korelacije meri jačinu linearne veze između dva atributa. Uzima vrednosti u opsegu [-1, 1], pri čemu granične vrednosti ukazuju na potpunu linearnost, dok vrednost 0 ukazuje na potpuno odsustvo linearnosti. Koeficijent korelacije je dat formulom:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{n(\sum x^2) - (\sum x)^2} * \sqrt{n(\sum y^2) - (\sum y)^2}}$$

	Sati učenja	Broj poena na ispitu	xy	x^2	y^2
Σ	36	449	2249	204	26541

$$r = \frac{8(2249) - (36)(449)}{\sqrt{8(204) - (36)^2} * \sqrt{8(26541) - (449)^2}} = 0.9629$$

Vrednost koeficijenta je približna jedinici te možemo da kažemo da je korelacija jaka i da je model dobar.

Zadatak za samostalnu vežbu - Maratonci

★★★☆☆

Na osnovu posmatranog skupa podataka o zavisnosti rezultata na maratonu od starosti maratonca, proceniti rezultat osobe koja ima 50 godina koristeći model linearne regresije i izračunati grešku.

Nezavisni atribut	Atribut odluke
Starost (godina)	Rezultat (h:m)
23	2:56
25	3:01
35	3:23
47	3:27
52	3:17
60	3:57
62	4:13

Zadatak za samostalnu vežbu - Plata

★★★☆☆

Na osnovu posmatranog skupa podataka o zavisnosti plate na osnovu godina radnog iskustva izvesti zaključak o plati zaposlenog sa 2.5 godina radnog iskustva koristeći linearnu regresiju i izračunati grešku.

Nezavisni atribut	Atribut odluke
Sati učenja	Broj poena na ispitu
1.1	39434
1.3	46205
1.5	37731
2.0	43525
2.2	39891
2.9	56642
3.0	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218

LOGISTIČKA REGRESIJA

Šta je logistička regresija i u kojim oblicima postoji?

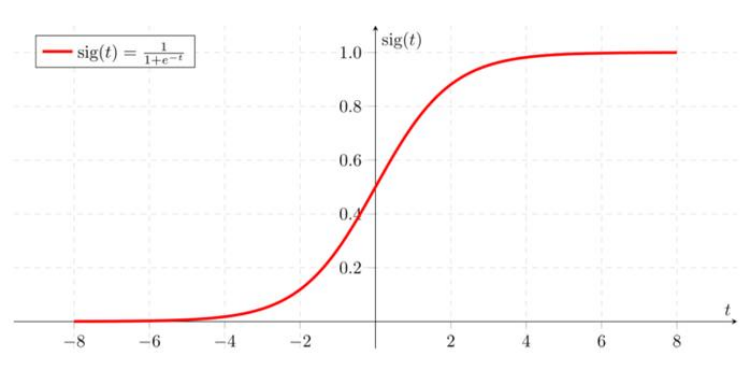
Logistička regresija (*Logistic regression*) je tip nadgledanih algoritama mašinskog učenja. Koristi se kod problema klasifikacije, kada je potrebno dati predikciju klase (kategorička vrednost) kojoj neki primerak pripada na osnovu jednog ili više njegovih atributa (kontinualnih ili kategoričkih).

Najprostiji oblik logističke regresije je **binarna logistička regresija**, kada je izlaz jedna od 2 moguće vrednosti (1 ili 0). U slučaju više od 2 moguće izlazne vrednosti radi se o multinomijalnoj logističkoj regresiji (koristi se softmax funkcija umesto sigmoid funkcije), a ukoliko je moguće uspostaviti redosled izlaznih vrednosti (npr. ocene 1-5) reč je o ordinalnoj logističkoj regresiji.

LOGISTIČKA REGRESIJA

Šta je logistička regresija i u kojim oblicima postoji?

Binarna logistička regresija koristi logističku (sigmoid) funkciju da modeluje zavisnu izlaznu promenljivu (verovatnoću pripadnosti klasi 1) na osnovu ulaznih podataka.



LOGISTIČKA REGRESIJA

Zašto se zove Logistička regresija, a ne Logistička klasifikacija?

Logistička regresija je generalizovani linearni model (utvrđuje odlike klasa, za razliku od diskriminativnog modela koji utvrđuje razlike između klasa) koji koristi formulu linearne zavisnosti izlaznog podatka od ulaznih, kao i linearna regresija

$$y = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

koju upotrebljava kao argument sigmoid funkcije za predikciju izlaza:

$$h(x) = \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-(w0 + w1x1 + w2x2 + \ldots + wnxn)}}$$

Izlaz Linearne regresije može biti bilo koja kontinualna vrednost u opsegu [-∞, +∞], dok sigmoid funkcija dovodi te vrednosti u opseg [0, 1].

LOGISTIČKA REGRESIJA

Kako odrediti parametre modela?

Kao i kod Linearne regresije, parametri modela treba da budu takvi da predviđena vrednost izlaza bude što približnija stvarnoj vrednosti. Za razliku od Linearne regresije, kod Logističke regresije u opštem slučaju ne postoji analitički pristup određivanja parametara modela (nije moguće rešiti jednačinu gde je prvi izvod cost funkcije jednak nuli). Utvrđivanje parametara moguće je korišćenjem iterativnih algoritama optimizacije, kao što je **gradijentni spust uz minimizaciju funkcije greške**.

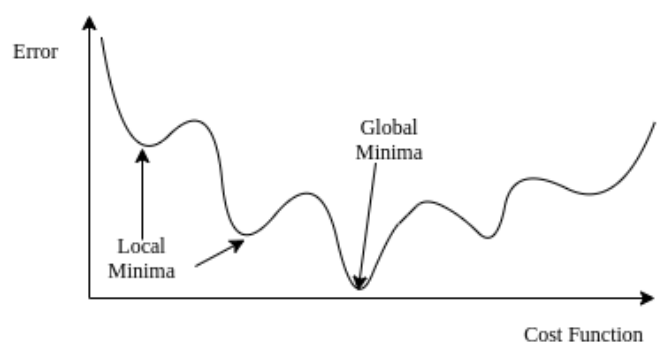
Da li se može koristiti MSE funkcija u metodi gradijentnog spusta?

Za razliku od Linearne regresije, gde je funkcija greške *Mean Squared Error* konveksna, kod Logističke regresije to nije slučaj, jer se linearna kombinacija ulaznih vrednosti i parametara modela koristi kao argument sigmoid funkcije (koja i sama nije konveksna).

LOGISTIČKA REGRESIJA

Da li se može koristiti MSE funkcija u metodi gradijentnog spusta?

Stoga, moguće je da metod gradijentnog spusta ne konvergira ka globalnom minimumu, već ka nekom od lokalnih minimuma, te ovakva funkcija greške ne odgovara modelu Logističke regresije. Zbog toga nije moguće korišćenje MSE funkcije kao cost funkcije.

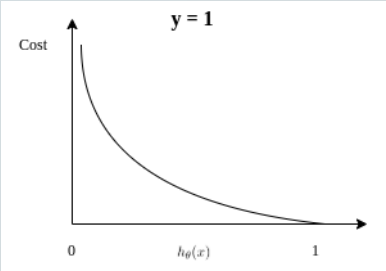


LOGISTIČKA REGRESIJA

Koju funkciju greške koristiti i zašto?

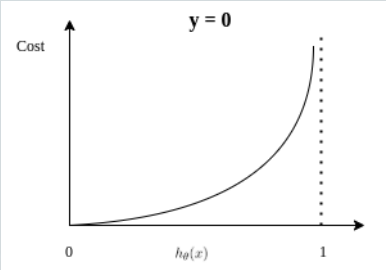
$\boldsymbol{y = 1} \mid Cost\ (h(x), y) = -\ln(h(x))$

- $h(x) = 0 \mid Cost = -\ln(0) = +\infty$
- $h(x) = 1 \mid Cost = -\ln(1) = 0$



$\boldsymbol{y = 0} \mid Cost\ (h(x), y) = -\ln\big(1 - h(x)\big)$

- $h(x) = 0 \mid Cost = -\ln(1) = 0$
- $h(x) = 1 \mid Cost = -\ln(0) = +\infty$



LOGISTIČKA REGRESIJA

Koju funkciju greške koristiti i zašto?

Objedinjena formula

$$\text{Cost}\,(\text{h}(\text{x}),\text{y}) = -\,\text{y}\,*\ln\big(\text{h}(\text{x})\big)\,-\,(1-\,\text{y})*\ln\big(1-\,\text{h}(\text{x})\big)$$

S obzirom da je verovatnoća pripadanja primerka sa atributima x klasi 1

$$P\,(y\,=\,1\mid x)\,=\,h(x)$$

onda je u slučaju dve klase

$$P\,(y\,=\,0\mid x)\,=\,1-\,h(x)$$

odnosno $P\,(\text{y}\mid\text{x}) = h(x)^{y}\,*\,(1\,-\,h(x))^{1-y}$

$$\text{Cost}(\text{h}(\text{x}),\text{y}) = -\ln\left(h(x)^{y}\,*\,\left(1\,-\,h(x)\right)^{1-y}\right) = -\ln(P\,(y\mid x))$$

Veća verovatnoća pripadnosti primerka označenoj klasi daje manji cost!

Zadatak 7 - Ispit

★★★☆☆

Na osnovu posmatranog skupa podataka o zavisnosti polaganja ispita u odnosu na broj sati učenja, odrediti model Logističke regresije

$$\boldsymbol{h(x)} = \frac{1}{1+ e^{-y}}$$

koji bolje reprezentuje podatke: y = 0.25 * x – 0.125 ili y = 0.125 * x + 0.25?

Nezavisni atribut	Atribut odluke
Sati učenja	Položio ispit
1	0
1.5	0
2	0
2.5	0
3	1
3.5	0
4	1
4.5	1
5	1
5.5	1

LOGISTIČKA REGRESIJA

Odnos ishoda i linearne kombinacije ulaznih parametara i težina

Inverzna funkcija sigmoid (logističke funkcije) je logit funkcija:

$$\text{logit}(\text{p}) = sigmoid^{-1}(p)$$

$$= \ln\left(\frac{p}{1-p}\right) = \ln\left(\frac{h(x)}{1-h(x)}\right) = \ln\left(\frac{\frac{1}{1+e^{-y}}}{\frac{e^{-y}}{1+e^{-y}}}\right) = \ln(e^{y}) = y$$

odnosno $\left(\frac{p}{1-p}\right) = e^{y}$, tj. $\text{p} = (1-p) * e^{y}$, tj. $p + p * e^{y} = e^{y}$. Odatle je:

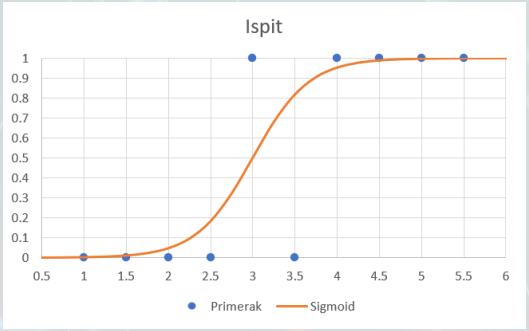
$$p = \frac{e^{y}}{e^{y}+1}$$

U slučaju da je p, odnosno h(x) = 0, onda je y = ln (0) = -∞.

Ukoliko je p, odnosno h(x) = 1, onda je y = ln (1 / 0) = +∞.

Zadatak 7 - Rešenje

Prikazom zavisnosti ishoda polaganja ispita (verovatnoća polaganja) od broja sati učenja možemo videti da podatke najbolje modeluje sigmoid funkcija. Raspoložive podatke možemo prikazati i u zavisnosti od logit (p) i pronaći hiperravan (u ovom slučaju jednog ulaznog parametra to je linija) koja je najbolje prilagođena zadatim podacima. Međutim, problem je što su vrednosti y (tj. logit (p)) za sve ulazne podatke $+\infty$ ili $-\infty$, te nije moguće metodom MSE oceniti kvalitet kandidat best-fit linije.



Zadatak 7 - Rešenje

Stoga, za svaki ulazni podatak izračunaće se best-fit vrednost y (logit (p)), a zatim će se na osnovu te vrednosti izračunati verovatnoća pripadanja klasi 1 podatka koji pripada best-fit liniji, a sve to u cilju ocene kvaliteta kandidat best-fit sigmoida.

Sada na osnovu labele svakog podatka (da li je ishod 1 ili 0) i njegove verovatnoće (pripadanja klasi 1) određujemo očekivanje da taj podatak pripada labeliranoj klasi.

likelihoods = [for i in len(data) p[i] if labels[i] == 1 else 1 – p[i]]

Nakon toga se sva očekivanja izmnože. Zbog mogućih malih vrednosti verovatnoća češće se izračunava **log-likelihood** (suma logaritama pojedinačnih očekivanja). Funkcija većeg očekivanja je bolji kandidat, odnosno ima manji cost.

Zadatak 7 - Rešenje

Za funkciju y = 0.25 * x – 0.125

Sum (Log (Likelihood)) = -6.216

Sati (x)	Ishod (y)	Logit (y)	Best-fit y	p	Likelihood	Log (Like.)
1	0	-inf	0.125	0.531	0.469	-0.758
1.5	0	-inf	0.250	0.562	0.438	-0.826
2	0	-inf	0.375	0.593	0.407	-0.898
2.5	0	-inf	0.5	0.622	0.378	-0.974
3	1	+inf	0.625	0.651	0.651	-0.429
3.5	0	-inf	0.750	0.679	0.321	-1.137
4	1	+inf	0.875	0.706	0.706	-0.348
4.5	1	+inf	1	0.731	0.731	-0.313
5	1	+inf	1.125	0.755	0.755	-0.281
5.5	1	+inf	1.250	0.777	0.777	-0.252

Zadatak 7 - Rešenje

Za funkciju y = 0.125 * x + 0.25

Sum (Log (Likelihood)) = -6.778, što znači da je prethodna funkcija bolja.

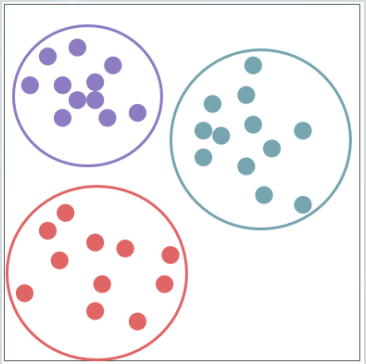
Sati (x)	Ishod (y)	Logit (y)	Best-fit y	p	Likelihood	Log (Like.)
1	0	-inf	0.375	0.593	0.407	-0.898
1.5	0	-inf	0.438	0.608	0.392	-0.936
2	0	-inf	0.500	0.622	0.378	-0.974
2.5	0	-inf	0.563	0.637	0.363	-1.013
3	1	+inf	0.625	0.651	0.651	-0.429
3.5	0	-inf	0.688	0.665	0.335	-1.095
4	1	+inf	0.750	0.679	0.679	-0.387
4.5	1	+inf	0.813	0.693	0.693	-0.367
5	1	+inf	0.875	0.706	0.706	-0.348
5.5	1	+inf	0.938	0.719	0.719	-0.330

NENADGLEDANO UČENJE

Tipovi izlazne vrednosti

Klasterovanje je tip nenadgledanog mašinskog učenja čiji je cilj da razdvoji podatke na određen broj grupa (klastera) na osnovu sličnosti i razlika.

Za razliku od klasifikacije, kod problema klasterovanja nemamo labelirane podatke.



ALGORITAM K-MEANS

Kako funkcioniše algoritam k-means?

Algoritam k srednjih vrednosti (*k-means*) je tip nenadgledanih algoritama mašinskog učenja. Koristi se kod problema klasterovanja.

Algoritam polazi od pretpostavke da će se slične instance nalaziti bliže u prostoru.

Algoritam započinje biranjem vrednosti za k (broj klastera), potom kreira k centroida (težišta klastera) na slučajan način, nakon čega iterativno dodeljuje svakom podatku jedan od uspostavljenih klastera na osnovu najbližeg centroida i ažurira sve centroide dokle god se ne uspostavi stabilno stanje (nema promena u klasterima) ili se ne dostigne maksimalan broj iteracija.

ALGORITAM K-MEANS

Kako funkcioniše algoritam k-means?

Valjanost dobijenih klastera može da se izračuna na osnovu funkcije greške:

$$J(c,t) = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - t_c{}^{(i)}\right)^2$$

c – klaster

t – težište klastera (centroid)

x – primerci

Da bi se izbeglo zaglavljivanje u lokalnom minimumu, algoritam se često pušta nekoliko stotina puta i bira rešenje za minimalnom vrednošću funkcije greške.

Vrednost *k* se često bira na osnovu domenskog znanja, ili upoređivanjem fukcije greške za različite vrednosti *k*.

Zadatak 1 – Oblici

★★★☆☆

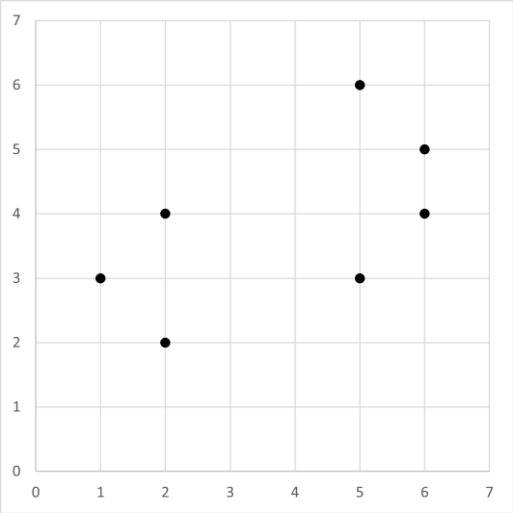
Na osnovu posmatranog skupa o dimenzijama oblika, podeliti skup na dve grupe koristeći *k-means* algoritam sa Euklidskim rastojanjem i izračunati grešku.

Nezavisni atributi		**Atribut odluke**
Širina (cm)	Visina (cm)	Grupa
1	3	?
5	6	?
5	3	?
2	2	?
6	4	?
6	5	?
2	4	?

Zadatak 1 – Rešenje

Na grafiku su prikazani posmatrani primerci.

Širina (cm)	Visina (cm)
1	3
5	6
5	3
2	2
6	4
6	5
2	4

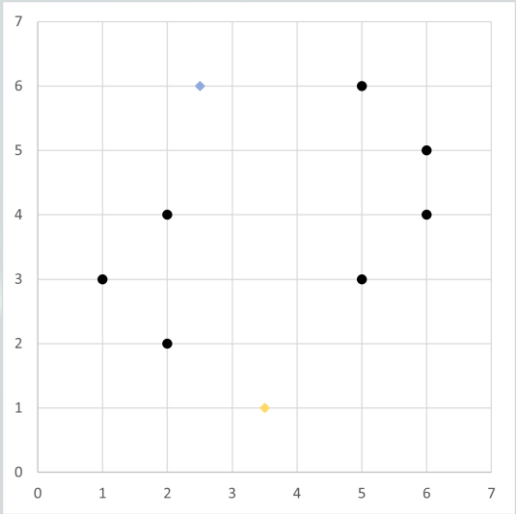


Zadatak 1 – Rešenje

Algoritam započinje nasumičnim odabirom dve tačke koje služe kao početni centroidi klastera.

Širina (cm)	Visina (cm)
1	3
5	6
5	3
2	2
6	4
6	5
2	4

Širina (cm)	Visina (cm)
2.5	6
3.5	1

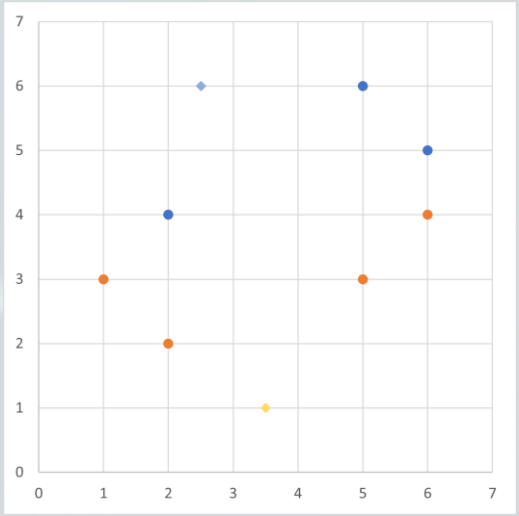


Zadatak 1 – Rešenje

Za svaki primerak se računa udaljenost od oba centroida i primerak se pridružuje klasteru koji mu je bliži.

Š (cm)	V (cm)	C1	C2
1	3	3.35	3.2
5	6	2.5	5.22
5	3	3.9	2.5
2	2	4.03	1.8
6	4	4.03	3.9
6	5	3.64	4.71
2	4	2.06	3.35

Širina (cm)	Visina (cm)
2.5	6
3.5	1

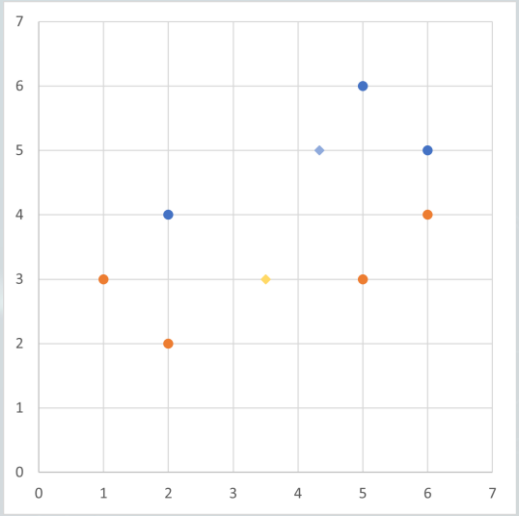


Zadatak 1 – Rešenje

Potom se ažuriraju centroidi tako što se on pomera u težište svoje grupe.

Š (cm)	V (cm)	C1	C2
1	3	3.35	3.2
5	6	2.5	5.22
5	3	3.9	2.5
2	2	4.03	1.8
6	4	4.03	3.9
6	5	3.64	4.71
2	4	2.06	3.35

Širina (cm)	Visina (cm)
4.33	5
3.5	3

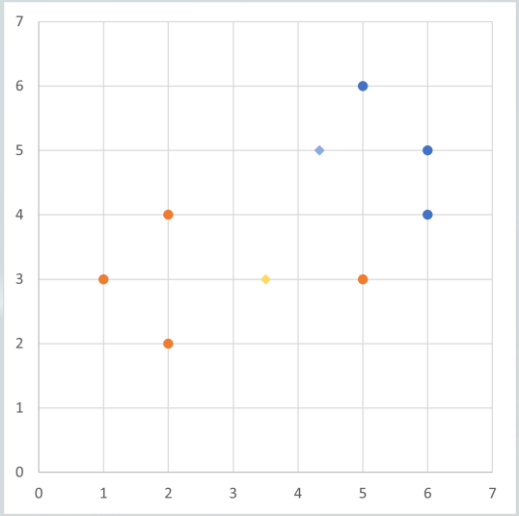


Zadatak 1 – Rešenje

Algoritam se iterativno ponavlja dokle god primerci menjaju klaster kojem pripadaju.

Š (cm)	V (cm)	C1	C2
1	3	3.88	2.5
5	6	1.2	3.35
5	3	2.1	1.5
2	2	3.79	1.8
6	4	1.94	2.69
6	5	1.67	3.2
2	4	2.53	1.8

Širina (cm)	Visina (cm)
4.33	5
3.5	3

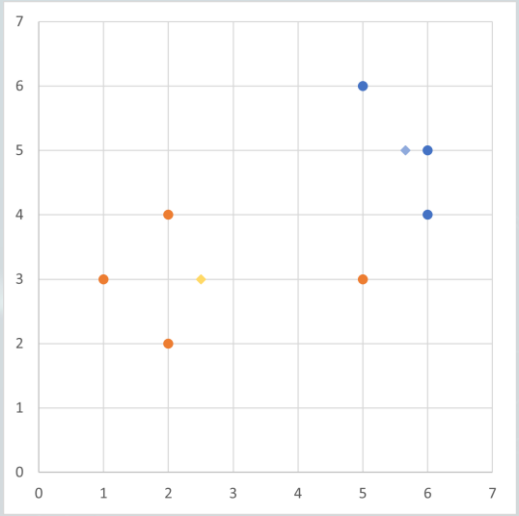


Zadatak 1 – Rešenje

Algoritam se iterativno ponavlja dokle god primerci menjaju klaster kojem pripadaju.

Š (cm)	V (cm)	C1	C2
1	3	3.88	2.5
5	6	1.2	3.35
5	3	2.1	1.5
2	2	3.79	1.8
6	4	1.94	2.69
6	5	1.67	3.2
2	4	2.53	1.8

Širina (cm)	Visina (cm)
5.67	5
2.5	3

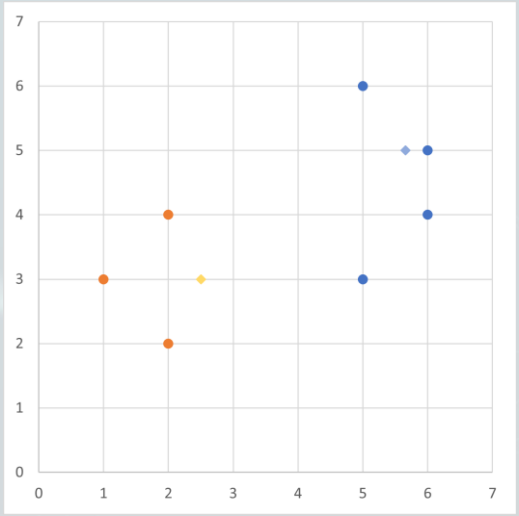


Zadatak 1 – Rešenje

Algoritam se iterativno ponavlja dokle god primerci menjaju klaster kojem pripadaju.

Š (cm)	V (cm)	C1	C2
1	3	5.08	1.5
5	6	1.2	3.9
5	3	2.11	2.5
2	2	4.74	1.12
6	4	1.05	3.64
6	5	0.33	4.03
2	4	3.8	1.12

Širina (cm)	Visina (cm)
5.67	5
2.5	3

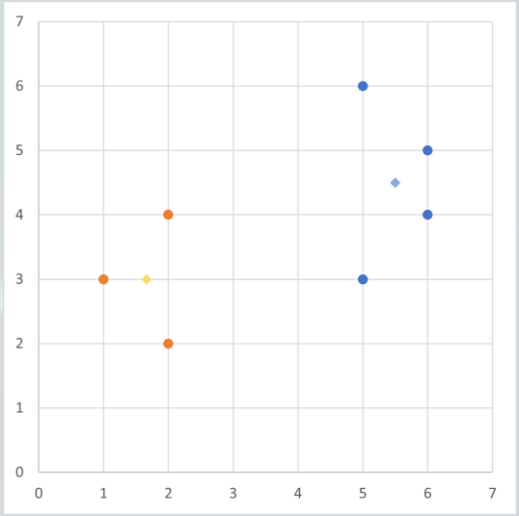


Zadatak 1 – Rešenje

Algoritam se iterativno ponavlja dokle god primerci menjaju klaster kojem pripadaju.

Š (cm)	V (cm)	C1	C2
1	3	5.08	1.5
5	6	1.2	3.9
5	3	2.11	2.5
2	2	4.74	1.12
6	4	1.05	3.64
6	5	0.33	4.03
2	4	3.8	1.12

Širina (cm)	Visina (cm)
5.5	4.5
1.67	3

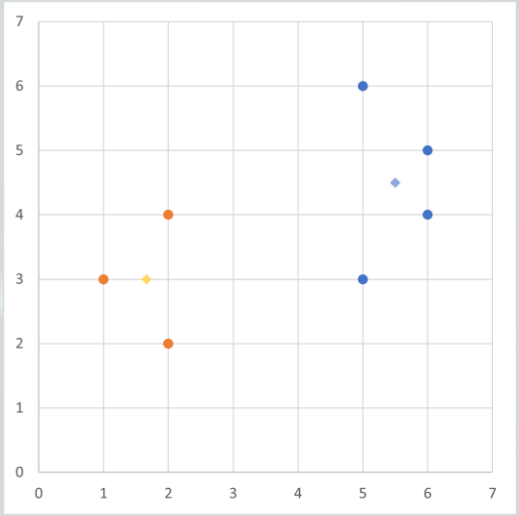


Zadatak 1 – Rešenje

Kako nema izmena u klasterima nakon izračunavanja udaljenosti od centroida, algoritam se završava i konačan klaster se dodeljuje primercima.

Š (cm)	V (cm)	C1	C2
1	3	4.74	0.67
5	6	1.58	4.48
5	3	1.58	3.33
2	2	4.3	1.05
6	4	0.71	4.44
6	5	0.71	4.77
2	4	3.53	1.05

Širina (cm)	Visina (cm)
5.5	4.5
1.67	3



Greška = $\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - t_c{}^{(i)}\right)^2$

= (1/7) * (0.67^2 + 1.58^2 + 1.58^2 + 1.05^2 + 0.71^2 + 0.71^2 + 1.05^2) = 1.23

Zadatak za samostalnu vežbu – Oblici

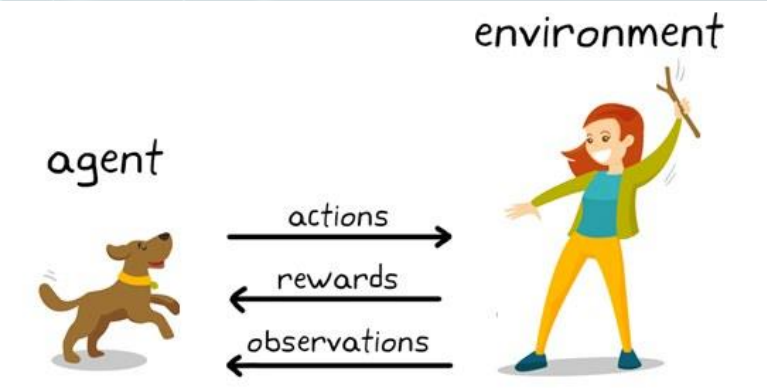
★★★☆☆

Na osnovu posmatranog skupa o dimenzijama oblika, podeliti skup na dve grupe koristeći *k-means* algoritam sa Euklidskim rastojanjem i izračunati grešku. Centroidi su se nasumično formirali na tačkama (2, 1) i (6, 6).

Nezavisni atributi		**Atribut odluke**
Širina (cm)	Visina (cm)	Grupa
6	1	?
4	4	?
2	5	?
3	6	?
5	3	?
1	6	?
5	1	?

Učenje sa podrškom

Učenje sa podrškom (eng. **Reinforcement learning**) je oblast mašinskog učenja koja se bavi razvojem pametnih agenata koji interaguju sa dinamičkim okruženjem sprovođenjem akcija čime čime vrše njegovu modifikaciju i dobijaju nagrade od okruženja.



Učenje sa podrškom - odlike

1. Dinamičnost – za razliku od statičke prirode supervizijskog i nesupervizijskog učenja, učenje sa podrškom dinamički istražuje okruženje u cilju otkrivanja strategije za izbor akcija.
2. Neoznačenost podataka – za razliku od labeliranih podataka kod supervizijskog učenja, učenje sa podrškom nema eksplicitno naveden tačan odgovor te mehaniznom probe, greške i nagrade otkriva da li je proces učenja na dobrom ili lošem putu.
3. Višestruke odluke – za razliku od jedne odluke kod supervizijskog učenja (jedan primerak – jedna odluka), učenje sa podrškom treba da formira složenu odluku u vidu lanca odluka kojima će se posao završiti kroz vreme.

Učenje sa podrškom – ključni koncepti

Agent – entitet koji sprovodi akcije u okruženju u cilju što boljeg obavljanja specificiranog posla i dobijanja bolje nagrade.

Akcija – moguće odluke za agenta u bilo kom stanju i čija primena vrši izmene u okruženju.

Okruženje – svet (realni ili virtuelni) u kojem se agent nalazi i u kojem primenjuje akcije.

Nagrada – dobit od okruženja nastala primenom poslednje akcije agenta u nekom stanju i koja može biti pozitivna, negativna ili 0.

Stanje – pojedinačan scenario iz okruženja.

Politika – strategija koju agent koristi da na osnovu trenutnog stanja odredi sledeću akciju koju će primeniti.

Učenje sa podrškom – kako se vrši?

U učenju sa podrškom postoje sledeći koraci:

1. Agent se nalazi u početnom stanju u okruženju.
2. Agent preduzima akciju na osnovu svoje specifične strategije.
3. Na osnovu izabrane akcije i trenutnog stanja modela okruženja agent dobija nagradu ili kaznu od okruženja.
4. Na osnovu trenutnog stanja okruženja, nagrade i prethodnog iskustva, agent ažurira svoju politiku u cilju maksimizacije kumulativne dobiti.
5. Koraci 2-5 se ponavljaju do pronalaska optimalne politike koju će agent koristiti u rešavanju problema.

Učenje sa podrškom

U procesu učenja sa podrškom agent može primenjivati:

- istraživanje neistraženog dela okruženja (**eksploracija**) u cilju smanjivanja nesigurnosti i sticanja novih saznanja o okruženju.
- iskorišćavanje trenutno stečenog znanja (**eksploatacija**) u cilju donošenja odluka koje vode ka poznatim visokim nagradama.

Fokus učenja sa podrškom je da se pronađe balans između procesa eksploracije i eksploatacije u cilju maksimizacije kumulativne dobiti kroz vreme. Previše eksploracije može dovesti do sporog napretka u ostvarivanju cilja, dok previše eksploatacije može da rezultuje u propuštanju prilika za istraživanje i otkrivanje novih, potencijalno boljih strategija.

Učenje sa podrškom – politika

Način na koji agent interaguje sa okruženjem u stanju **s** definisan je njegovom politikom π. U učenju sa podrškom teži se pronalasku takve politike da je primena akcije u svakom stanju uvek optimalna u pogledu dobijanja maksimalne nagrade u budućnosti. Politika može biti:

- **deterministička**, pri čemu se za svako stanje **s** uvek primenjuje akcija **a** definisana politikom π, odnosno **a = π (s)**.
- **stohastička**, pri čemu se svaka akcija primenjuje sa odgovarajućom verovatnoćom i deo je politike agenta, odnosno **a ~ π (* | s)**.

Učenje sa podrškom – model

U interakciji agenta sa okruženjem, ono može biti poznato ili ne:

- **model baziran** (eng. **model-based**), pri čemu ili postoji pristup modelu ili se pokušava pravljenje aproksimativnog modela. Određivanje optimalne politike vrši se korišćenjem verovatnoća tranzicija između stanja i funkcije nagrađivanja koje su poznate. Ovo može biti korisno jer se planiranje akcija može obaviti unapred, bez njihovog prethodnog izvršavanja.
- **bez modela** (eng. **model-free**), pri čemu nisu jasno definisane verovatnoće tranzicija i/ili funkcija nagrađivanja. Ne teži se eksplicitnom razumevanju načina na koji model okruženja funkcioniše, već se izvršavanjem akcija prikuplja iskustvo sa ciljem da se otkrije politika koja će biti optimalna.

Markovljevo svojstvo i Markovljev lanac

Markovljevo svojstvo se odnosi na svojstvo stohastičkog procesa u smislu da je budućnost procesa uslovljena samo trenutnim stanjem i ne zavisi eksplicitno od istorije procesa.

Markovljev lanac je stohastički model koji opisuje sekvencu mogućih događaja u sistemu, pri čemu verovatnoća budućih događaja i prelaska u naredno stanje zavisi samo od trenutnog stanja sistema. Markovljev lanac je primena Markovljevog svojstva u stohastičkom procesu.

Markovljev lanac se opisuje parom (S, P), pri čemu S predstavlja skup stanja, a P (s, s') predstavlja verovatnoću prelaska iz stanja s u trenutku t u stanje s' u trenutku t + 1.

Markovljev proces odlučivanja

Markovljev proces odlučivanja (eng. Markov decision process) je osnovna komponenta za omogućavanje procesa učenja sa podrškom, a koja pruža formalni matematički okvir za modeliranje okruženja koje se posmatra kao Markovljev proces.

Markovljev proces odlučivanja može da reši većinu problema učenja sa podrškom u kojima su akcije diskretizovane. Koristeći Markovljev proces odlučivanja pametni agent može da dosegne optimalnu politiku za maksimizaciju svojih ciljeva.

Nezavisnost budućnosti procesa od njegove istorije nije realan slučaj, ali predstavlja veoma dobru aproksimaciju i omogućava jednostavnije matematičke modele i algoritme za realizaciju procesa učenja sa podrškom.

Markovljev proces odlučivanja

Markovljev proces odlučivanja se od Markovljevih lanaca razlikuje po tome što uvodi koncept akcija, odnosno da naredno stanje ne zavisi samo od prethodnog stanja već i od izabrane akcije.

Markovljev proces odlučivanja se opisuje torkom **(S, A, P, R, γ)**:

- S predstavlja skup stanja.
- A predstavlja skup akcija.
- P (s, a, s') predstavlja verovatnoću da će akcija a usloviti prelazak iz stanja s u trenutku t u stanje s' u trenutku t + 1.
- R (s, a, s') predstavlja kratkoročnu nagradu dobijenu tranzicijom iz stanja s u stanje s' pod dejstvom akcije a.
- γ predstavlja faktor vrednovanja dugoročnih nagrada.

Belmanova jednačina

Korišćenjem Belmanove jednačine, očekivana kumulativna dobit R_t za stanje u trenutku t može se rekurzivno izraziti kao suma:

- kratkoročne nagrade za prelazak u stanje iz trenutka t + 1 i
- očekivane kumulativne dobiti za stanje u koje se prešlo u trenutku t + 1.

$$E[R_t|s_t = s] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \,|s_t = s\right]$$
$$= E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots |s_t = s]$$
$$= E[r_{t+1} + \gamma(r_{t+2} + \gamma^1 r_{t+3} + \gamma^2 r_{t+4} + \cdots)|s_t = s]$$
$$= E[r_{t+1} + \gamma E[R_{t+1}]|s_t = s]$$

Belmanova jednačina

Neka sekvencijalne, kratkoročne nagrade iznose -10, 30, 40 u trenucima t, t + 1 i t + 2, respektivno. Tada očekivana dugoročna nagrada u trenutku t iznosi:

- $\gamma = 0.7$, Nagrada = $-10 + 0.7^1 * 30 + 0.7^2 * 40 = 30.6$
- $\gamma = 1.0$, Nagrada = $-10 + 1^1 * 30 + 1^2 * 40 = 60$
- $\gamma = 0.0$, Nagrada = $-10 + 0^1 * 30 + 0^2 * 40 = -10$

Faktor γ određuje koliko ćemo vrednovati nagrade iz budućnosti i omogućava balansiranje između kratkoročnih i dugoročnih ciljeva.

Korišćenjem Belmanove jednačine možemo odrediti vrednosne i optimalne vrednosne funkcije stanja i akcija za ocenu kvaliteta stanja odnosno akcije izabrane u stanju.

Vrednosne funkcije

Očekivana kumulativna dobit počevši od stanja **s** (eng. State-value function) korišćenjem politike π za izbor akcija iznosi:

$$V(s) = \sum_a \pi(a|s) \sum_{s'} P(s,a,s') * (r(s,a,s') + \gamma * V(s'))$$

a očekivana kumulativna dobit izborom akcije **a** u stanju **s** (eng. Action-value function), a potom korišćenjem politike π:

$$Q(s,a) = \sum_{s'} P(s,a,s') * \left(r(s,a,s') + \gamma * \sum_{a'} \pi(a'|s') * Q(s',a')\right)$$

- P (s, a, s') – verovatnoća da će akcija a usloviti prelazak iz stanja s u trenutku t u stanje s' u trenutku t + 1.
- r (s, a, s') – kratkoročna dobit pri prelasku iz stanja s u trenutku t u stanje s' u trenutku t + 1 usled akcije a.

Optimalne vrednosne funkcije

Da bi maksimizovao kumulativnu dobit agent mora da pronađe optimalnu politiku, odnosno da za svako stanje sistema pronađe najbolju akciju koja će mu doneti najveću kumulativnu dobit.

Tada je optimalna očekivana kumulativna dobit stanja **s**:

$$V^*(s) = max_a \sum_{s'} P(s,a,s') * (r(s,a,s') + \gamma * V^*(s'))$$

dok je optimalna očekivana kumulativna dobit izbora akcije **a** u stanju **s**:

$$Q^*(s,a) = \sum_{s'} P(s,a,s') * (r(s,a,s') + \gamma * max_{a'} Q^*(s',a'))$$

Pristupi u određivanju optimalne politike

Ukoliko nam je poznat model okruženja u potpunosti, odnosno ako su poznate verovatnoće tranzicija između stanja izvršavanjem akcija kao i funkcija nagrađivanja, onda je moguće odrediti optimalne vrednosne funkcije stanja ili parova stanje-akcija korišćenjem tehnike dinamičkog programiranja i pristupa **iteracije vrednosti** (eng. Value iteration).

Vrednosna funkcija se izračunava za svako stanje ili par stanje-akcija korišćenjem estimacija sledbenika (eng. Bootstraping) sve do konvergencije. U ovom pristupu politika je implicitna (ne čuva se ili se ne ažurira eksplicitno) i izvodi se direktno na osnovu vrednosne funkcije tako što se u svakom stanju izabere akcija koja donosi najveću dobit.

Pristupi u određivanju optimalne politike

Postoji i suprotan pristup orijentisan ka politici (eng. **Policy iteration**) u kome se politika čuva na eksplicitan način. Inicijalno se na arbitraran način odredi politika i na osnovu nje i dinamike sistema odredi se vrednosna funkcija za svako stanje ili svaki par stanje-akcija. Na osnovu dobijenih vrednosti koriguje se trenutna politika tako što se za svako stanje izabere najbolja akcija i ponovo se računa vrednosna funkcija korišćenjem izračunate politike i dinamike sistema. Proces se ponavlja do konvergencije.

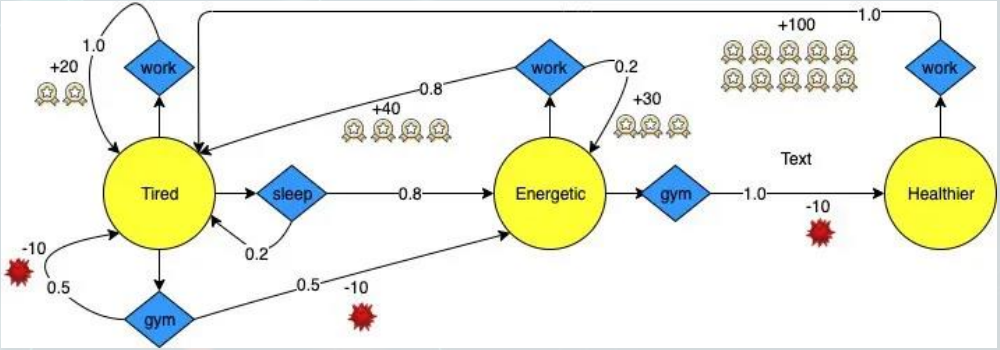
Zapravo, pristup iteracije vrednosti se poklapa sa pristupom orijentisanom ka politici, osim što se politika ne čuva i modifikuje eksplicitno nakon modifikacije vrednosne funkcije, već se to radi implicitno.

Markovljev proces odlučivanja

Stanja: Tired, Energetic, Healthier.

Akcije: work, sleep, gym.

Tranziciona funkcija je određena datim verovatnoćama prelaza, koje su zadate na granama tranzicija. Kratkoročne nagrade su navedene ispod/iznad grana tranzicija.



Value iteration (DP) pristup

```
import numpy as np

# actions: work = 0, sleep = 1, gym = 2
# states: tired = 0, energetic = 1, healthier = 2
legal_actions = [[0, 1, 2], [0, 2], [0]]

# P (s, a, s')
P = np.array([
    [[1., 0, 0], [0.2, 0.8, 0], [0.5, 0.5, 0]],
    [[0.8, 0.2, 0], [None] * 3, [0, 0, 1.0]],
    [[1.0, 0, 0], [None] * 3, [None] * 3]
])

# R (s, a, s')
R = np.array([
    [[20, 0, 0], [0, 0, 0], [-10, -10, 0]],
    [[40, 30, 0], [None] * 3, [0, 0, -10]],
    [[100, 0, 0], [None] * 3, [None] * 3]
])
```

Value iteration (DP) pristup

```
Q = np.full((len(P), len(P[0])), -np.inf)
for s, a in enumerate(legal_actions):
    Q[s, a] = 0
gamma = 0.99
iterations = 0
while True:
    deltas = []
    Q_before = Q.copy()
    for s in range(len(P)):
        for a in legal_actions[s]:
            q_value = 0
            for s_next in range(len(P[s, a])):
                q_value += P[s, a, s_next] * (R[s, a, s_next] +
                                    gamma * np.max(Q_before[s_next]))
            Q[s, a] = q_value
            deltas.append(np.abs(Q[s, a] - Q_before[s, a]))
    if not max(deltas):
        break
    iterations += 1
```

Value iteration (DP) pristup

```
print(f'Converged after {iterations} iterations.')
print(Q)

Converged after 3196 iterations.
# actions: work = 0, sleep = 1, gym = 2
# states: tired = 0, energetic = 1, healthier = 2
[[2728.88792274 2736.25042701 2715.9894879 ]
 [2753.7285488            -inf 2770.79904351]
 [2808.88792274           -inf          -inf]]

# Optimalna politika agenta je:
# - izbor akcije sleep (1) u stanju tired (0).
# - izbor akcije gym (2) u stanju energetic (1).
# - izbor akcije work (0) u stanju healthier (0).
```

Šta, ukoliko model nije poznat?

Ukoliko model okruženja nije u potpunosti poznat, nije moguće koristiti Belmanove jednačine za izračunavanje vrednosnih funkcija stanja i parova stanje-akcija. Često nisu poznate verovatnoće tranzicija, što oslikava situaciju iz realnog sveta u kojem agent nema potpune informacije o okruženju.

Međutim, to što verovatnoće tranzicija nisu poznate, ne znači da ne postoje. U tom slučaju agent mora da interaguje sa okruženjem da bi sakupio nagrade i iskustvo. Na taj način je moguće dobiti približnu procenu vrednosnih funkcija koje su bliske stvarnim vrednostima.

U ovakvim situacijama koriste se pristupi: Monte Karlo i metod vremenske razlike.

Monte Karlo pristup

Monte Karlo pristup omogućava učenje vrednosnih funkcija i (poslednično iz njih) optimalne politike na osnovu iskustva agenta prikupljenog kroz epizode. Smatra se da je iskustvo podeljeno u epizode i da će svaka epizoda da se završi u terminalnom stanju Markovljevog procesa odlučivanja (MDP) bez obzira na izbor akcija. Svaka epizoda predstavlja jedan potpuni ciklus interakcije agenta sa okruženjem, od početnog stanja do terminalnog stanja.

Monte Karlo pristup je baziran na usrednjavanju nagrada dobijenih iz epizoda. Da bi Monte Karlo metode mogle da se koriste, svaka epizoda mora da se završi i samo tada je omogućeno agentu da izračuna ukupnu nagradu koju je dobio tokom te epizode.

Monte Karlo evaluacija politike

Estimacija vrednosti V stanja s koristeći politiku π vrši se generisanjem epizoda primenom te politike. Ideja je da se usrednje vrednosti nagrada dobijenih nakon što se stanje s poseti, odnosno nakon što se naiđe na njega u epizodi. Što više puta se stanje poseti, to će usrednjena vrednost biti bliža očekivanoj vrednosti. Postoje dva pristupa određivanja vrednosti V za stanje s u zavisnosti od toga koje posete se računaju:

Prva poseta (eng. First visit) – usrednjavaju se dobiti dobijene do kraja epizode koje slede samo prvu pojavu stanja u epizodi.

Svaka poseta (eng. Every visit) – usrednjavaju se dobiti dobijene do kraja epizode koje slede svaku pojavu stanja u epizodi.

Monte Karlo evaluacija politike - primer

Neka su date epizode nastale primenom politike π koje čine stanja i kratkoročne nagrade koje ih neposredno slede:

A +3 -> B -1 -> A -2 -> C +4 -> A +2 -> END

B -1 -> A +3 -> B -3 -> END

C +4 -> A -2 -> C +1 -> END

Formule za vrednosti stanja V (s) i para stanje-akcija Q (s, a):

$$V(s) = \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i \text{ i } Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} G_i$$

- N(s) i N(s,a) broj prvih/svakih nailazaka na stanje s, odnosno par stanje-akcija (s, a).
- Gi je dobit do kraja epizode u i-tom nailasku na stanje/par.

Monte Karlo evaluacija politike - primer

Određivanje vrednosti stanja koristeći prvi nailazak (first visit):

$V(A) = (3 - 1 - 2 + 4 + 2) / 1 = 6$; 1. epizoda

$V(A) = ((3 - 1 - 2 + 4 + 2) + (3 - 3)) / 2 = 3$; 2. epizoda

$V(A) = ((3 - 1 - 2 + 4 + 2) + (3 - 3) + (-2 + 1)) / 3 = 1.67$, 3. epizoda

Određivanje vrednosti stanja koristeći svaki nailazak (every visit):

$V(A) = ((3 - 1 - 2 + 4 + 2) + (-2 + 4 + 2) + (2)) / 3 = 4$; 1. epizoda

$V(A) = ((3 - 1 - 2 + 4 + 2) + (-2 + 4 + 2) + (2) + (3 - 3)) / 4 = 3$; 2. epizoda

$V(A) = ((3 - 1 - 2 + 4 + 2) + (-2 + 4 + 2) + (2) + (3 - 3) + (-2 + 1)) / 5 = 2.2$; 3. epizoda

Monte Karlo pristup

U Monte Karlo pristupu estimacije vrednosti stanja su nezavisne i predstavljaju deo celokupne dobiti epizode (eng. Sampling), dok se u DP pristupu izgrađuju na osnovu estimacija stanja sledbenika (eng. Bootstraping).

U slučaju da model okruženja nije poznat u potpunosti, onda određivanje samo vrednosti stanja nije dovoljno (neophodne su verovatnoće tranzicija) i potrebno je određivanje vrednosti parova stanje-akcija pretragom njihovih pojavljivanja u epizodama. U suprotnom, vrednosti stanja su dovoljne za određivanje politike tako što se bira akcija koja daje najbolju kombinaciju vrednosti kratkoročne dobiti i vrednosti stanja sledbenika do koga je ta akcija dovela.

Monte Karlo pristup

Problem predstavlja činjenica da u slučaju generisanja epizoda korišćenjem determinističke politike, mnogi parovi stanje-akcija neće biti generisani, već će se birati samo po jedna (trenutno najbolja, politikom izabrana) akcija za svako stanje. Estimacije ostalih akcija neće se poboljšati sa iskustvom, a one potencijalno predstavljaju bolje akcije. Potrebna je estimacija svih akcija iz svakog stanja, a ne samo onih parova stanje-akcija koje trenutna politika favorizuje. Jedan od načina za obezbeđivanje eksploracije je da postoji nenulta verovatnoća izbora svakog para stanje-akcija na početku svake epizode. Ovaj način obično ne oslikava situaciju iz realnog sveta, te se obezbeđivanje razmatranja svih parova stanje-akcija omogućava razmatranjem stohastičkih politika sa nenultim verovatnoćama izbora svojih akcija.

Monte Karlo kontrola

Monte Karlo evaluacija vrednosti parova stanje-akcija generisanjem epizoda korišćenjem trenutne politike ima za cilj da ažurira trenutnu funkciju vrednosti koja treba da poboljša trenutnu politiku i da je aproksimira optimalnoj, a zatim se izmenjena politika koristi za generisanje novih epizoda za evaluaciju vrednosti parova stanje-akcija i proces se ponavlja dok se ne konvergira ka optimalnoj politici; MK kontrola je predstavljena E-I ciklusom (eng. Evaluation; eng. Improvement).

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{E} Q^* \xrightarrow{I} \pi^*$$

Optimalna politika svakog stanja se bira kao pohlepna (greedy):

$$\pi(s) = argmax_a Q(s,a)$$

Monte Karlo kontrola

Monte Karlo kontrola započinje izborom arbitrarne politike i vrednosti za parove stanje-akcija. Zatim se generiše epizoda primenom trenutne politike i za svaki par stanje-akcija koji se javlja u epizodi računaju se nove vrednosti parova stanje-akcija usrednjavanjem dobiti metodama prvi ili svaki nailazak. Dobijenim vrednostima parova stanje-akcija koriguje se izabrana politika. Proces se ponavlja sve dok trenutna politika ne konvergira ka optimalnoj. U stacionarnim okruženjima su distribucije nagrada stacionarne i pogodno je koristiti usrednjavanje vrednosti dobiti. U nestacionarnim okruženjima koja se menjaju, gde se politika menja i poboljšava, obično se koristi korak učenja da se „zaborave“ starije epizode, vrednuju skorašnje informacije i izvrše adaptacije na promene u okruženju.

Monte Karlo kontrola

Ažuriranje vrednosti stanja/parova može se vršiti inkrementalno uz korak učenja (ušteda memorije nasuprot usrednjavanju):

$$\mathrm{V}(S_t) = \mathrm{V}(S_t) + \alpha * [R_t - \mathrm{V}(S_t)]$$

$$Q(s_t, a) = \mathrm{Q}(s_t, a) + \alpha * [R_t - \mathrm{Q}(s_t, a)]$$

- α – predstavlja korak učenja (eng. Learning rate) i utiče na konvergenciju učenja.
- $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T$, pri čemu je T terminalni trenutak epizode – predstavlja kumulativni dugoročni dobitak.
- $\mathrm{V}(S_t)$ / $\mathrm{Q}(s_t, a)$ - trenutna procena vrednosti stanja, odnosno para stanje-akcija.

Metod vremenske razlike

Mana **M**onte **K**arlo pristupa je što se ažuriranja vrednosti (odnosno politike agenta) vrše tek na kraju epizode. Metodi vremenske razlike (eng. **T**emporal **D**ifference) vrše ažuriranja već u sledećem vremenskom trenutku. Na taj način omogućavaju učenje agenta u realnom vremenu, čime se minimizuje odlaganje učenja. Predstavljaju kombinaciju metoda MK (sampling) i DP (bootstraping). Ipak, TD metodi su osetljivi na inicijalne procenjene vrednosti stanja i parova stanje-akcija kao i parametare učenja. Za razliku od njih, MK metodi su konceptualno jednostavniji i lak za implementaciju, ali obično sporiji od TD metoda. MK metod koristi stvarnu dugoročnu kumulativnu dobit za ažuriranje vrednosti, dok TD metodi koriste Belmanovu jednačinu optimalnosti za estimaciju buduće vrednosti.

Metod vremenske razlike TD (0)

Izmenom dugoročne kumulativne dobiti u Monte Karlo metodu sumom kratkoročne nagrade i estimacije dugoročne kumulativne dobiti od sledećeg trenutka dobija se **TD (0)** metod učenja.

Ažuriranje vrednosti stanja vrši se po formuli:

$$\mathrm{V}(S_t) = \mathrm{V}(S_t) + \alpha * [R_{t+1} + \gamma * \mathrm{V}(S_{t+1}) - \mathrm{V}(S_t)]$$

gde:

$R_{t+1} + \gamma * \mathrm{V}(S_{t+1})$ predstavlja TD vrednost cilja.

$R_{t+1} + \gamma * \mathrm{V}(S_{t+1}) - \mathrm{V}(S_t)$ predstavlja TD grešku.

Analogno, ažuriranje vrednosti para stanje-akcija:

$$Q(s_t, a) = \mathrm{Q}(s_t, a) + \alpha * [R_{t+1} + \gamma * \mathrm{Q}(S_{t+1}) - \mathrm{Q}(s_t, a)]$$

Q-učenje

Q-učenje (eng. Q-learning) je algoritam učenja sa podrškom koji je direktno izveden iz TD (0) metoda, nije baziran na modelu (eng. **model-free**) i apolitičan je (eng. **off-policy**). Slovo Q potiče od engleske reči **quality**, koja predstavlja koliko je akcija značajna u maksimizaciji budućih nagrada koje nas očekuju od narednog stanja dobijenog primenom te akcije pa sve do ciljnog stanja.

Apolitičan se odnosi na način ažuriranja Q-vrednosti. Ukoliko se primenom akcije **a** u stanju **s** napravi tranzicija ka stanju **s'**, ažuriranje Q-vrednosti stanja **s** primenom akcije **a** može da se obavi na osnovu najbolje moguće akcije dostupne u stanju **s'** (eng. **off-policy**) ili na osnovu akcije koja je deo trenutne politike i koja je dovela agenta u stanje **s'** (eng. **on-policy**, npr. SARSA).

Q-funkcija

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma * max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Nova estimacija Q-vrednosti za stanje S_t i akciju A_t ažurira se Q-funkcijom koja koristi Belmanovu jednačinu u kojoj figurišu:

- α – korak učenja (eng. **Learning rate**), koji je u opsegu [0-1].
- R_{t+1} - neposredno dobijena nagrada nakon primene akcije A_t u stanju S_t (eng. **Immediate reward**).
- γ – faktor popusta (eng. **Discount factor**), koji je u opsegu [0-1] i čime se utiče na važnost nagrada iz budućnosti.
- $Q(S_t, A_t)$ – stara estimacija Q-vrednosti trenutnog stanja.
- $max_a Q(S_{t+1}, a)$ – estimacija optimalne vrednosti sledećeg stanja.

Q-učenje – ključni koncepti

Stanje (s) – pojedinačna situacija iz okruženja.

Akcija (a) – moguće odluke za agenta u bilo kom stanju.

Nagrada – dobit/kazna od okruženja nastala primenom poslednje akcije agenta u nekom stanju.

Epizoda – sekvenca stanja koja se završava stanjem koje je konačno (ciljno, u slučaju uspeha ili porazno, u slučaju neuspeha).

Q(s, a) – očekivana Q-vrednost do ciljnog stanja koja se dobija primenom akcije **a** u stanju **s**.

Q-tabela – koja čuva Q-vrednosti za kombinacije stanja i akcija. Ova tabela se ažurira i procesu učenja i nju agent koristi prilikom donošenja odluka za maksimizaciju očekivane dobiti.

Q-učenje – algoritam

1. **Inicijalizacija Q-tabele.** Vrednosti kojima će tabela biti inicijalizovana zavise i od vrednosti nagrada.
2. **Izbor akcije od strane agenta na osnovu izabrane politike.** Na početku se vrši eksploracija (biranjem slučajne akcije), a zatim, kako vreme prolazi i kako vrednosti Q-tabele dobijaju na značaju, sve više se koristi eksploatacija.
3. **Izvršavanje izabrane akcije od strane agenta.** Dobija se novo stanje u koje se prelazi, kao i nagrada/kazna.
4. **Ažuriranje Q-tabele na osnovu dobijene nagrade.** Korišćenjem Belmanove jednačine za optimalno donošenje odluka.
5. **Koraci 2-5 se ponavljaju dok Q-tabela ne bude zadovoljavajuća.**

Epsilon-greedy strategija

Predstavlja jednostavan metod za balansiranje eksploracije i eksploatacije. **Epsilon** vodi poreklo od oznake za verovatnoću kojom se određuje da li se vrši eksploracija ili eksploatacija okruženja.

Na samom početku, vrednost epsilon je velika, čime agent vrši eksploraciju okruženja. U procesu eksploracije agent sve više postaje sigurniji u estimacije Q-vrednosti i značenje izračunatih vrednosti dobija na značaju. Vremenom, vrednost epsilon opada i agent sve više vrši eksploataciju na račun eksploracije okruženja i to koristeći prethodno utvrđene Q-vrednosti za izbor svojih akcija, odnosno u svakom stanju bira onu akciju čija je estimacija Q-vrednosti najveća (**greedy** pristup).

Primer zaleđenog jezera (determinističko)

Vilenjak treba da dođe do poklona najkraćim putem preko leda.

Postoji 16 različitih stanja (4 x 4 matrica polja).

Postoje 4 različite akcije (levo = 0, dole = 1, desno = 2, gore = 3).

Dostizanje cilja daje nagradu +1.

Upadanje u vodu daje nagradu 0, kao i stajanje na praznom polju.

Potrebna nam je Q-tabela sa 64 (16 x 4) različite vrednosti.

Q-tabela se može inicijalizovati vrednostima 0,
jer će opseg Q-vrednosti sigurno biti [0, 1].

Stanje je kodirano pozicijom vilenjaka po vrstama
(0 – startno, 15 – ciljno).

Treniranje modela (Q-tabela)

```
def get_action_eps_greedy_policy(env, q_tab, st, eps):
    prob = random.uniform(0, 1)
    return np.argmax(q_tab[st]) if prob > eps else env.action_space.sample()

def train(num_episodes, max_steps, lr, gamma, eps_min, eps_max, eps_dec_rate, env):
    avg_returns = []
    avg_steps = []
    q_tab = np.zeros((env.observation_space.n, env.action_space.n))
    for episode in range(num_episodes):
        avg_returns.append(0.)
        avg_steps.append(0)
        eps = eps_min + (eps_max - eps_min) * np.exp(-eps_dec_rate * episode)
        st = env.reset()[0]
        for step in range(max_steps):
            act = get_action_eps_greedy_policy(env, q_tab, st, eps)
            new_st, rew, done, _, _ = env.step(act)
            q_tab[st][act] = q_tab[st][act] +
                        lr * (rew + gamma * np.max(q_tab[new_st]) - q_tab[st][act])
            if done:
                avg_returns[-1] += rew
                avg_steps[-1] += step + 1
                break
            st = new_st
    return q_tab, avg_returns, avg_steps
```

Treniranje modela (Q-tabela)

Primer Q-tabele dobijene treniranjem sa sledećim izborom hiperparametara:

number_of_episodes = 7000

max_steps = 100

learning_rate = 0.05

gamma = 0.95

epsilon_max = 1.0

epsilon_min = 0.005

epsilon_decay_rate = 0.001

	Levo	Dole	Desno	Gore
0	0.734988	0.773781	0.764389	0.734951
1	0.463819	0	0.811648	0.422923
2	0.477883	0.856953	0.147741	0.517226
3	0.388033	0	0.04126	0.054973
4	0.773612	0.814506	0	0.734868
5	0	0	0	0
6	0	0.902495	0	0.585386
7	0	0	0	0
8	0.814122	0	0.857375	0.773148
9	0.812783	0.9025	0.901957	0
10	0.8561	0.9495	0	0.854985
11	0	0	0	0
12	0	0	0	0
13	0	0.494726	0.949999	0.624876
14	0.901569	0.949803	1	0.900139
15	0	0	0	0

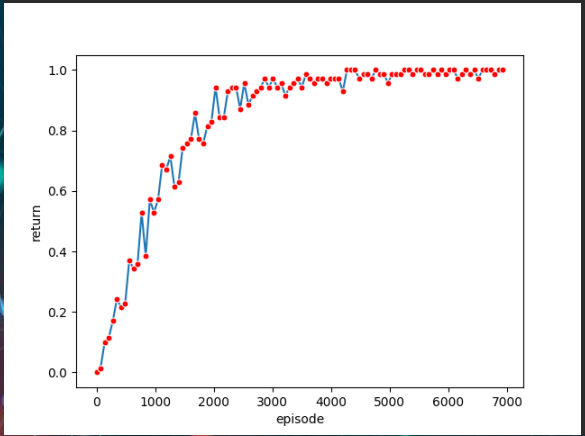
Evaluacija modela (100 epizoda)

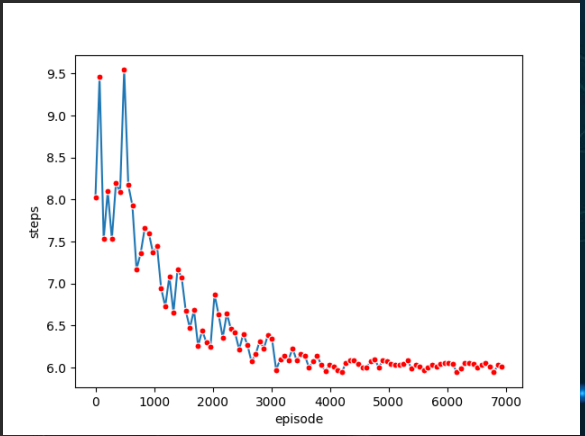
```python
def evaluate(num_episodes, max_steps, env, q_tab):
    ep_rew_lst = []
    steps_lst = []
    for episode in range(num_episodes):
        st = env.reset(seed=episode)[0]
        step_cnt = 0
        ep_rew = 0
        for step in range(max_steps):
            act = np.argmax(q_tab[st])
            new_st, rew, done, _, _ = env.step(act)
            step_cnt += 1
            ep_rew += rew
            if done:
                break
            st = new_st
        ep_rew_lst.append(ep_rew)
        steps_lst.append(step_cnt)
    print(f'TEST Mean reward: {np.mean(ep_rew_lst):.2f}')
    print(f'TEST STD reward: {np.std(ep_rew_lst):.2f}')
    print(f'TEST Mean steps: {np.mean(steps_lst):.2f}')

# TEST Mean reward: 1.00
# TEST STD reward: 0.00
# TEST Mean steps: 6.00
```

Prikaz metrika

```
def line_plot(data, name, show):
    pyplot.figure(f'Average {name} per episode: {np.mean(data):.2f}')
    df = pandas.DataFrame({
        name: [np.mean(data[i * config.chunk:(i + 1) * config.chunk])
               for i in range(config.number_of_episodes // config.chunk)],
        'episode': [config.chunk * i
                    for i in range(config.number_of_episodes // config.chunk)]})
    plot = seaborn.lineplot(data=df, x='episode', y=name, marker='o',
                            markersize=5, markerfacecolor='red')
    plot.get_figure().savefig(f'{name}.png')
    if show:
        pyplot.show()
```





Simulacija kretanja agenta

	Levo	Dole	Desno	Gore
0	0.734988	0.773781	0.764389	0.734951
1	0.463819	0	0.811648	0.422923
2	0.477883	0.856953	0.147741	0.517226
3	0.388033	0	0.04126	0.054973
4	0.773612	0.814506	0	0.734868
5	0	0	0	0
6	0	0.902495	0	0.585386
7	0	0	0	0
8	0.814122	0	0.857375	0.773148
9	0.812783	0.9025	0.901975	0
10	0.8561	0.9495	0	0.854985
11	0	0	0	0
12	0	0	0	0
13	0	0.494726	0.949999	0.624876
14	0.901569	0.949803	1	0.900139
15	0	0	0	0

Zaleđeno jezero (stohastičko)

U slučaju stohastičkog okruženja zaleđenog jezera, izdavanjem akcije agent sa verovatnoćom 1/3 uspeva u svojoj nameri, dok u suprotnom slučaju sa podjednakom verovatnoćom u iznosu 1/3 prelazi u stanje kao da je izvršio akciju koja ga vodi perpendikularno u odnosu na svoju originalnu nameru.

Npr, ukoliko je agent hteo levo, sa verovatnoćom 1/3 uspeva u svojoj nameri, a sa verovatnoćama po 1/3 prelazi u stanja kao da je izvršio akcije gore, odnosno dole.

Na isti način će se izvršiti obučavanje agenta, koristeći identične parametre modela.

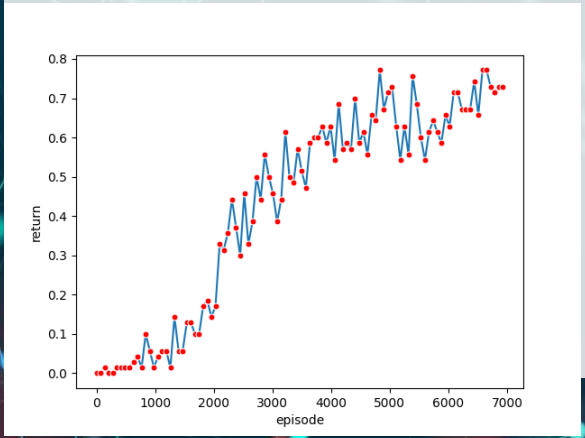
Zaleđeno jezero (stohastičko)

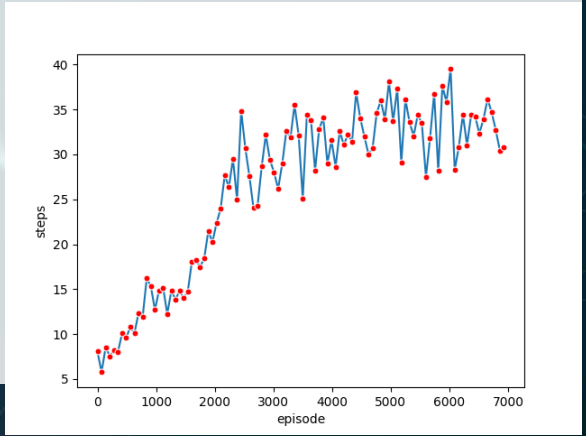
Dobijeni rezultati testiranja na 100 epizoda:

TEST Mean reward: 0.76

TEST STD reward: 0.43

TEST Mean steps: 35.19





Prednosti Q-učenja

- Pristup bez modela (eng. model-free) je osnovna odlika Q-učenja, jer algoritam ne zahteva prethodno znanje o samom okruženju, već agent uči o okruženju po principu probe i greške dok se obučava. Ovaj pristup je naročito pogodan u situacijama gde je teško modelovati unutrašnju dinamiku okruženja ili gde je ona potpuno nepoznata.
- Apolitična optimizacija algoritma, koji nije striktno vezan za određenu politiku, može dovesti do najboljih mogućih rezultata, naročito u situacijama gde striktnost u primeni politike ne može dovesti do istog nivoa optimizacije.

Fleksibilnost ovakvog pristupa omogućava Q-učenju da radi na velikom broju raznovrsnih problema i okruženja.

Nedostaci Q-učenja

- Može biti veoma teško pronaći takve hiperparametre modela koji omogućavaju dobar balans između eksploracije i eksploatacije, što je inače i poznata dilema u oblasti učenja sa podrškom (eng. Exploration vs. exploitation tradeoff).
- Q-učenje potencijalno može biti podložno fenomenu koji se zove „prokletsvo dimenzionalnosti“ (eng. Curse of dimensionality). Velika količina podataka u velikom broju dimenzija posledično zahteva velike Q-tabele, što može dovesti i do problema u izračunavanju i smanjene preciznosti. Stoga, Q-učenje je pogodno za diskretizovane akcije i prostore problema, dok je za kontinualne potrebno izvršiti diskretizaciju problema. Pored toga, kvantizacija podataka (uži tipovi podataka npr.) može doprineti boljem modelu.

Nedostaci Q-učenja

- Model Q-učenja ponekad može biti previše optimističan u pogledu kvaliteta neke akcije ili strategije (eng. Overestimation).

Duboko Q-učenje (eng. Deep Q-learning) je izvedeno iz osnovnog modela Q-učenja i može da razreši većinu njegovih problema. Umesto kreiranja Q-tabela, duboko Q-učenje podrazumeva formiranje modela duboke neuralne mreže radi rukovođenja velikim okruženjima koja mogu da uključuju kontinualne akcije i prostore problema. Na ulaze mreže postavljaju se parovi stanje-akcija i mreža se obučava.

Sa druge strane, Q-učenje je pogodno za manja okruženja koja su diskretizovana.

PITANJA?

http://ri4es.etf.rs/