Strojno učenje s jasnim tumačenjem, indukcija pravila i optimalna stabla odluke

Dubinska analiza podataka

7. predavanje

Pripremio: izv. prof. dr. sc. Alan Jović

Ak. god. 2023./2024.

Sadržaj

- Strojno učenje s jasnim tumačenjem – uvod
- Indukcija pravila
- Algoritmi indukcije pravila
  - OneRule
  - PRISM
  - RIPPER
  - FINDBESTRULE
- Optimalna stable odluke
  - OCT & OCT-H

* napomena: slajdovi 13-29 pripremljeni su na temelju prezentacije:
J. Fürnkranz, Introduction to rule learning, Institut Ruđer Bošković, 2013.

Strojno učenje s jasnim tumačenjem – uvod

Strojno učenje s jasnim tumačenjem

- Engl. *Interpretable Machine Learning*
- Uključuje modele strojnog učenja čije **tumačenje (interpretacija) je jasno razumljivo čovjeku** (engl. *white box models*)
- **Ne miješati** s objašnjavanjem *black box* modela strojnog i dubokog učenja u okviru tzv. **objašnjive umjetne inteligencije** (engl. *eXplainable AI*, kraće: ***XAI***)
  - Prednost *black box* modela je **često** (ali ne uvijek) veća točnost klasifikacije / predikcije
- Izgradnja uspješnih *white box* modela strojnog učenja je korisna i dobro istražena tema
  - Prednost *white box* modela je jednostavnost tumačenja
  - Korištenje naprednih postupaka za izgradnju *white box* modela dovodi do sličnih točnosti kao i za *black box* modele

Važnost mogućnosti tumačenja modela strojnog učenja

- Neki puta odgovor na pitanje „**Što** je model predvidio?” **nije dovoljan**, pogotovo u kontekstu primjene u područjima s visokim ulozima (engl. *high-stakes areas*) kao što su medicina i autonomna vožnja
- **Povjerenje u model**
  - Gradi se na temelju razine uspješnosti predviđanja i na temelju jasnoće tumačenja
  - Model koji ne daje dobro predviđanje ili onaj čije odluke su nejasne čovjeku **ne postiže povjerenje**
- **Prirodna radoznalost**
  - **Zašto** je model predvidio da ću imati dug oporavak nakon operacije, koji parametri su tu bili važni?
  - **Kako** je model odlučio da se tu radi o anomaliji u proizvodnji?
- **Sigurnosni aspekt**
  - Jasno tumačenje može nam otkriti razlog zašto model postupa na određeni način, što je važno za razumjevanje sigurnosnih pitanja, npr. model prepoznaje bicikl na temelju detekcije dva kotača – što ako je stražnji kotač prekriven djelomično s nosačima?

Metode s inherentno jasnim tumačenjem

- Razvijaju se već više od **60** godina!
- Linearna regresija (engl. *linear regression*) – koeficijenti uz varijable izravno određuju važnost pojedine varijable
- Logički modeli – logički povezane izjave o značajkama (if-then, and, or...)
  - Induktivna pravila (skupovi odlučivanja, liste odlučivanja) (engl. *induction rules, decision sets, decision lists*)
  - Stabla odluke (engl. *decision trees*)
- Druge metode:
  - Rasuđivanje zasnovano na slučajevima (engl. *case-based reasoning*) – najbliži susjedi, prototipovi
  - Generalizirani aditivni modeli (engl. *generalized aditive models*, GAM)

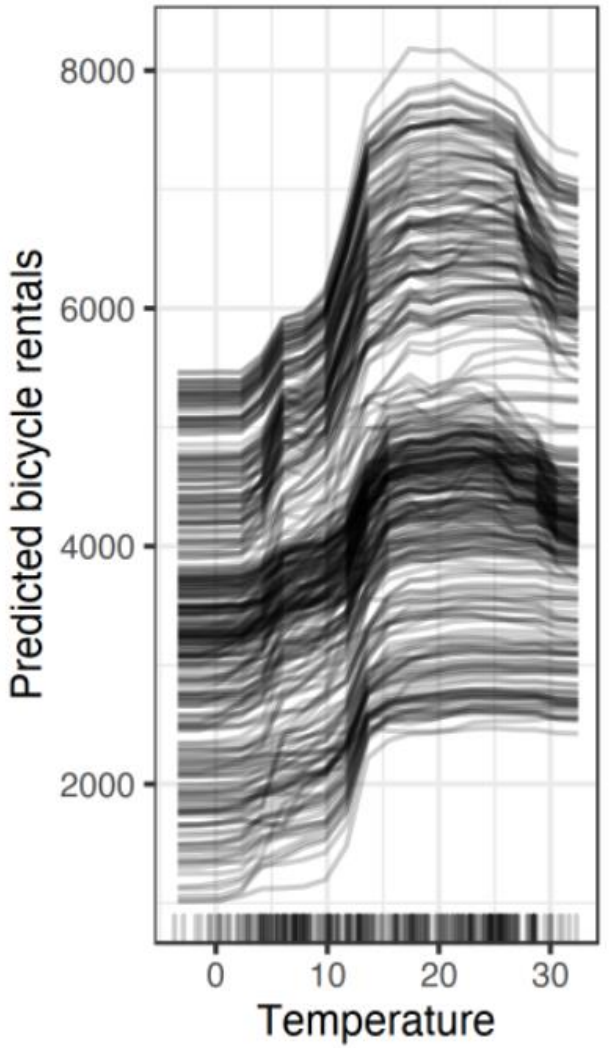
Više u: Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, Chudi Zhong "Interpretable machine learning: Fundamental principles and 10 grand challenges," Statist. Surv. 16, 1-85, (2022)

Objašnjiva umjetna inteligencija – globalne i lokalne metode

Post hoc objašnjenje modela crne kutije:

- **Permutacijska važnost značajke** (engl. *permutation feature imporatance*) – globalna metoda
- Dijagram djelomične ovisnosti (engl. *partial dependence plot*) – globalna metoda
- Pojedinačno uvjetno očekivanje (engl. *Individual Conditional Expectation,* ICE) – lokalna metoda
- **Shapleyjeva aditivna objašnjenja** (engl. *SHapley Additive exPlanations*, SHAP) – lokalna metoda
- Lokalni surogat LIME (engl. *local surrogate*, LIME) – lokalna metoda
- ...

Više informacija: https://christophm.github.io/interpretable-ml-book/global-methods.html
https://christophm.github.io/interpretable-ml-book/local-methods.html



ICE

Indukcija pravila

Indukcija pravila (induktivna pravila)

- Engl. *induction rules*, *rule induction*, *rule learning, decision lists, decision sets*
- Zadatak: naučiti korisna **if-then** pravila (jedno ili više njih) **automatski na temelju skupa podataka**
- Najčešća varijanta: na **then** strani nalazi se **ciljna značajka**
  - Alternativa: na **if** i na **then** strani nalaze se prediktivne značajke (asocijativna pravila) – 8. predavanje
- Najčešća primjena je na klasifikacijske probleme (induktivna klasifikacijska pravila), ali može i na regresijske probleme
- Neki algoritmi mogu raditi s numeričkim značajkama, a svi s kategoričkim značajkama
- **Prednost:** vrlo jasno tumačenje modela
- **Nedostatak**: točnost rezultata modela je ponekad slabija od drugih pristupa

Indukcija pravila – primjena

- Glavni razlog primjene: **inteligentna analiza podataka** i **otkrivanje znanja**
- Nisu prikladna na raspoznavanje uzoraka / računalni vid / obradu prirodnog jezika
- Prikladna za primjenu u većem broju područja:
  - Biomedicina – pronalazak pravila za dijagnozu ciljnih oboljenja
  - Bioinformatika – analiza gena u svrhu opisa zanimljivih interakcija
  - Društvene znanosti
    - Modeliranje rizika (npr. izgradnja profila ljudi koji odustaju od usluge)
    - Analiza faktora za bankovne krize
    - …

Indukcija pravila – načini primjene

- **Učenje pravila (indukcija pravila) za cijeli skup podataka**
  - Uobičajeni pristup, koristi se na isti način kao bilo koji drugi klasifikator/regresor
- **Otkrivanje podgrupa** (engl. *subgroup discovery*)
  - Pronalazak **zanimljivih podskupova primjeraka**
- **Relacijska dubinska analiza podataka** (engl. *relational data mining*)
  - Učenje pravila iz više tablica – složeni relacijski problemi: vremenski podaci, strukturirani podaci
- **Semantička dubinska analiza podataka** (engl. *semantic data mining*)
  - Integracija pravila s ontološkim bazama znanja (engl. *ontologies*) kako bi se omogućilo učinkovitije rasuđivanje na temelju prikupljenog znanja

Učenje pravila – konjunkcijsko pravilo

- **if** $(A_i \text{ op } val_{iI})$ and ... and $(A_k \text{ op } val_{kK})$ **then** +

tijelo pravila (dio **if**) **glava** pravila (dio **then**)

- Tijelo pravila sadržava **konjunkciju uvjeta**
  - Uvjet tipično sadrži **usporedbu (op) vrijednosti prediktivne značajke A_i s nekom njenom vrijednosti, op može biti:** $=, <, >, \leq, \geq$
- Glava pravila sadržava **predviđanje**
  - Za binarnu klasifikaciju: pozitivnu klasifikaciju (+) ako primjerak pripada određenom konceptu, a negativnu (-) ako mu ne pripada
  - Za višeklasnu klasifikaciju predviđanje određene (jedne) klase
  - Za regresiju predikciju vrijednosti ciljne značajke

Konjunkcijsko pravilo – terminologija

- Pokrivanje (prekrivanje) pravila (engl. *rule coverage*)
  - Pravilo pokriva primjerak ako primjerak ispunjava sve uvjete pravila (cijelu **if** stranu)
- Predviđanje pravila (engl. *rule prediction*)
  - Ako pravilo pokriva primjerak, glava pravila je predviđena za taj primjerak
- Konzistentnost skupa pravila (engl. *rule set consistency*)
  - Skup pravila je konzistentan ako sva pravila u skupu pokrivaju samo primjerke jedne određene klase (konzistentnost se definira i za jedno pravilo – ako pokriva samo primjerke jedne klase)
- Potpunost skupa pravila (engl. *rule set completion*)
  - Skup pravila je potpun ako su svi primjerci pokriveni nekim pravilom (potpunost se može definirati i za pojedinačnu ciljnu klasu)

Uvodni primjer – skup za učenje Weather

Temperature	Outlook	Humidity	Windy	Play Golf?
hot	sunny	high	false	no
hot	sunny	high	true	no
hot	overcast	high	false	yes
cool	rain	normal	false	yes
cool	overcast	normal	true	yes
mild	sunny	high	false	no
cool	sunny	normal	false	yes
mild	rain	normal	false	yes
mild	sunny	normal	true	yes
mild	overcast	high	true	yes
hot	overcast	normal	false	yes
mild	rain	high	true	no
cool	rain	normal	true	no
mild	rain	high	false	yes

Uvodni primjer – jednostavno rješenje

- **Zadatak**: Pronaći skup pravila koji ispravno predviđa ciljnu značajku iz skupa prediktivnih značajki
- **Jednostavno rješenje:**

IF	T=hot	AND	H=high	AND	O=overcast	AND	W=false	THEN=yes
IF	T=cool	AND	H=normal	AND	O=rain	AND	W=false	THEN=yes
IF	T=cool	AND	H=normal	AND	O=overcast	AND	W=true	THEN=yes
IF	T=cool	AND	H=normal	AND	O=sunny	AND	W=false	THEN=yes
IF	T=mild	AND	H=normal	AND	O=rain	AND	W=false	THEN=yes
IF	T=mild	AND	H=normal	AND	O=sunny	AND	W=true	THEN=yes
IF	T=mild	AND	H=high	AND	O=overcast	AND	W=true	THEN=yes
IF	T=hot	AND	H=normal	AND	O=overcast	AND	W=false	THEN=yes
IF	T=mild	AND	H=high	AND	O=rain	AND	W=false	THEN=yes

- Rješenje: skup pravila koji je potpun i konzistentan na skupu za učenje
- **Problem**: neće dobro generalizirati za nove primjerke

Uvodni primjer – bolje rješenje

IF	Outlook = overcast	THEN=yes
IF	Humidity = normal and Outlook = sunny	THEN=yes
IF	Outlook = rain and Windy = false	THEN=yes

- **Rješenje:** skup pravila koji je i dalje potpun i konzistentan na skupu za učenje (ali pravila su kraća nego u jednostavnom rješenju)
- Vjerojatno će bolje generalizirati za nove primjerke

Učenje pravila po principu „razdvoji pa vladaj”

- Engl. ***separate-and-conquer rule learning***, *covering rule learning*
- Pravila se grade postepeno i tako da razdvoje primjerke od primjeraka
- Velika većina algoritama učenja pravila radi po tom principu
- Osnovni algoritam:
  1. Započni s praznom teorijom (modelom) *T* i skupom za učenje *E*
  2. Nauči jedno konzistentno pravilo *R* iz *E* i dodaj ga u *T*
  3. Ako je *T* zadovoljavajuća (najčešće: potpuna) vrati *T*
  4. Inače:
     - **Razdvoji**: Ukloni primjerke koje pokriva *R* iz skupa *E*
     - **Vladaj**: vrati se na korak 2

Ovdje (korak 2) je najveća razlika između raznih algoritama – kako naučiti pravilo R?

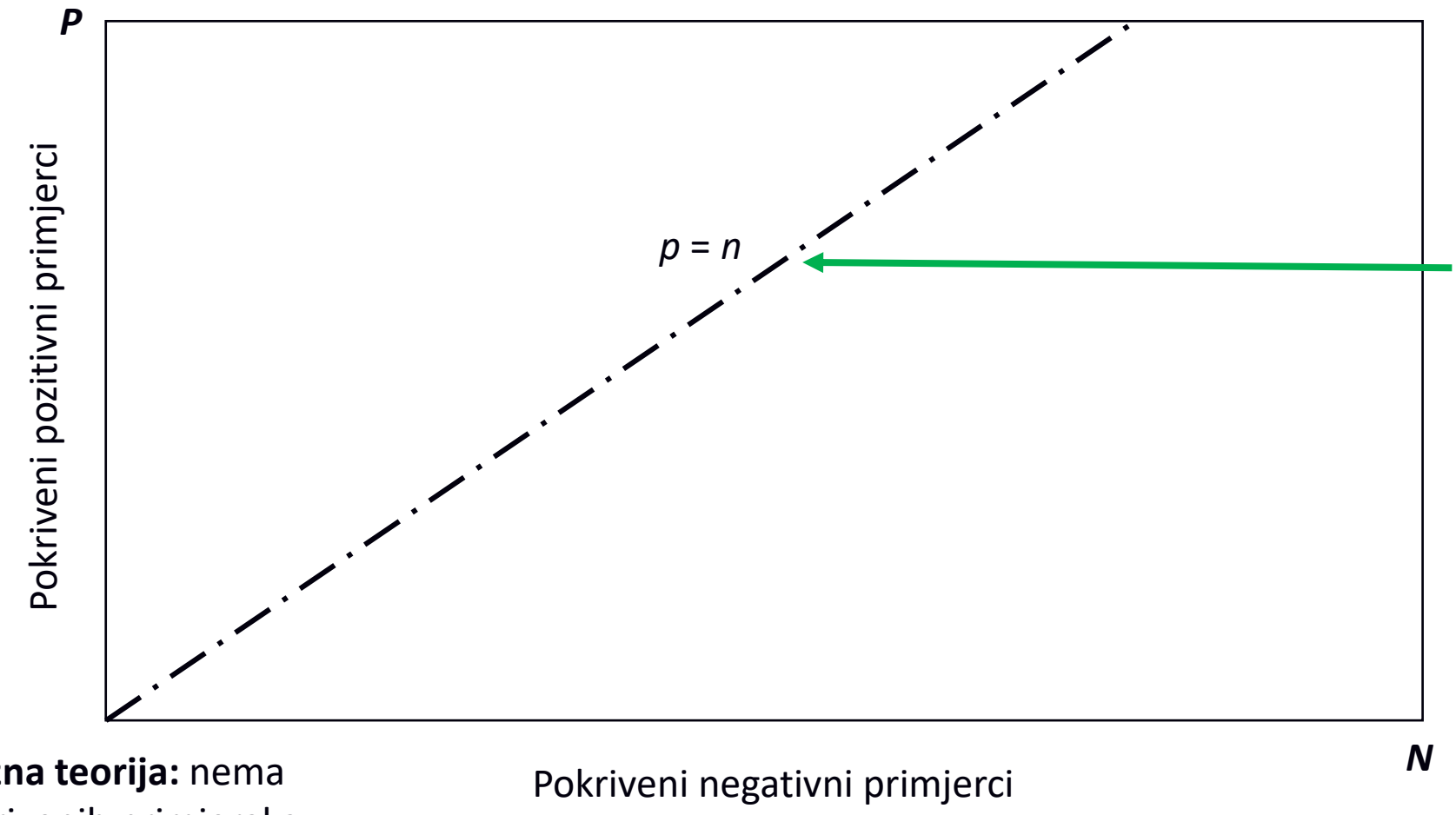
Relaksacija potpunosti i konzistentnosti

- Izgradnja skupa pravila koji je kompletan i konzistentan pokazano dovodi do **prenaučenosti**
- Daljnje razmatranje: **binarna klasifikacija** (većina ostalih slučajeva može se svesti na nju)
- Neka je s ***P*** označen ukupan broj pozitivnih primjeraka (klasa +), a s ***N*** ukupan broj negativnih (klasa -)
- Primjerci koje pokriva pravilo
  - Uistinu pozitivni: ***p*** – pozitivni primjerci koje pokriva pravilo
  - Lažno pozitivni: ***n*** – negativni primjerci koje pokriva pravilo
- Primjerci koje ne pokriva pravilo:
  - Lažno negativni: ***P - p*** – pozitivni primjerci ne pokriveni pravilom
  - Uistinu negativni: ***N - n*** – negativni primjerci ne pokriveni pravilom
- **Pokrivanje** (engl. *coverage*) ili **potpora** (engl. *support*) pravila:
  - **COV = SUP = (p + n) / (P + N)** – najčešće tim bolje što je veće

	predicted +	predicted -	
class +	*p* (TP)	P-p (FN)	P
class -	*n* (FP)	N-n (TN)	N
	p + *n*	*P* + *N* - (*p* + *n*)	P + N

Prostor pokrivanja pravilima

Savršena teorija: svi pozitivni, nema negativnih primjeraka



Univerzalna teorija: svi primjerci su pokriveni

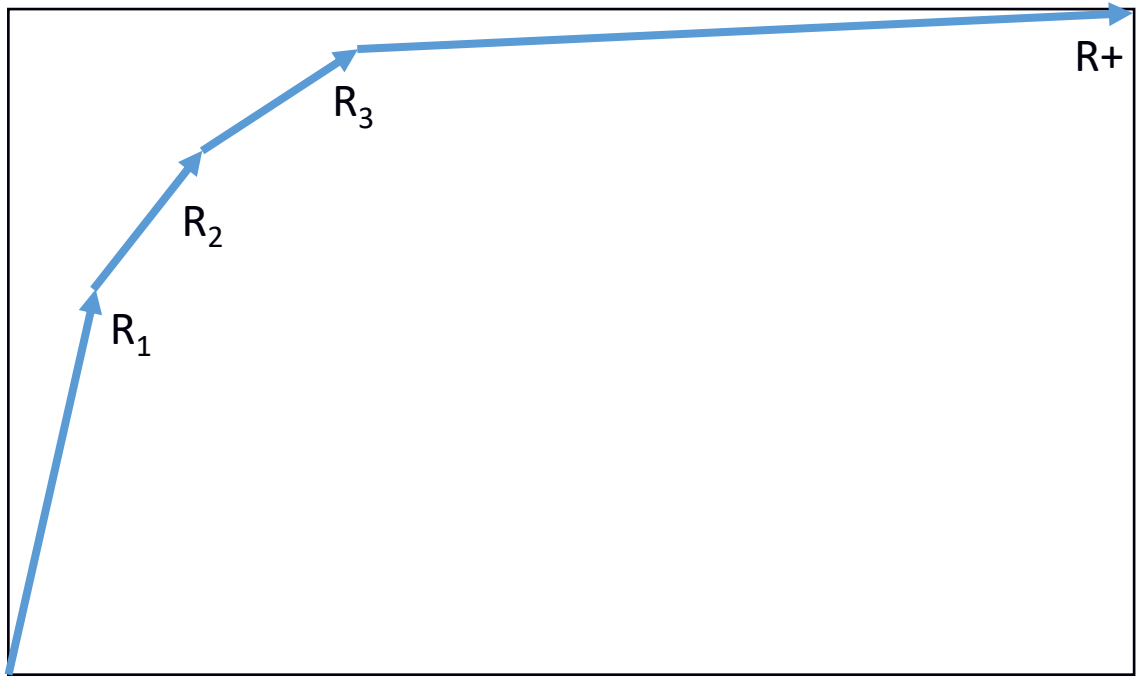
Izo-točna teorija: pokrivena je ista količina *p* i *n* primjeraka

Suprotna teorija: svi negativni su pokriveni i nijedan pozitivan nije pokriven

Prazna teorija: nema pokrivenih primjeraka

Prostor pokrivanja pravilima

- Algoritmi zasnovani na principu podijeli pa vladaj uče **jedno po jedno pravilo**
- To odgovara putu po prostoru pokrivanja
  - Prazna teorija (bez pravila) odgovara (0,0)
  - Dodavanje pravila R_i ne smanjuje *p* ni *n* jer pravilo pokriva dodatne primjerke
  - Univerzalna teorija R+ (gdje su svi primjerci u skupu pozitivni) bi odgovarala (N,P)



Heuristike pokrivanja

	predicted +	predicted -	
class +	p (TP)	$P - p$ (FN)	P
class -	n (FP)	$N - n$ (TN)	N
	$p + n$	$P + N - (p + n)$	P + N

- Dodavanje pravila treba:
  - što više povećati broj pokrivenih pozitivnih primjeraka (povećati stupanj potpunosti)
  - što manje povećati broj pokrivenih negativnih primjeraka (ne smanjivati konzistentnost)
- Heuristika za odabir pravila koje treba dodati u T treba uzeti u obzir ovo gore
- Česte heuristike u sustavima zasnovanima na pravilima (svaka ima neku teorijsku podlogu):
  - **Točnost**: ACC = $(p + N - n)/(P + N) \cong p - n$
  - **Utežana relativna točnost**: WRA = $\frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right) \cong p/P - n/N$
  - **Preciznost (pouzdanost)**: PREC = CONF = $p/(p + n)$
  - **Korelacija**: CORR = $\frac{p(N-n)-n(P-p)}{\sqrt{PN(p+n)(P-p+N-n)}}$

Heuristike pokrivanja – primjer izračuna

Uvjet		*p*	*n*	ACC	WRA	PREC	CORR
	Hot	2	2	0	-0.178	0.5	-0.189
Temperature =	Mild	3	1	2	0.133	0.75	0.141
	Cold	4	2	2	0.044	0.667	0.043
	Sunny	2	3	-1	-0.378	0.4	-0.378
Outlook =	Overcast	4	0	4	0.444	1	0.471
	Rain	3	2	1	-0.067	0.6	-0.067
Humidity =	High	4	4	-1	-0.356	0.429	-0.344
	Normal	6	1	5	0.467	0.857	0.447
Windy =	True	3	3	0	-0.267	0.5	-0.258
	False	6	2	4	0.267	0.75	0.258

***P* = 9**
***N* = 5**

- Koji uvjet će se dodati u, na početku, prazno pravilo *R* **ovisi o korištenoj heuristici**
  - Npr. ako se koristi PREC ili CORR dodat će se **Outlook = Overcast**,
    ako se koristi ACC ili WRA dodat će se **Humidity = Normal**

Heuristike pokrivanja

- Idući korak može biti da se:
  - stane s izgradnjom trenutnog pravila i nastavi s idućim (praznim) pravilom
  - nastavi dodavati sljedeći najpovoljniji uvjet u pravilo – **rafiniranje** pravila
- Što napraviti?
  - Imati jedno konzistentno pravilo (npr. za PREC i CORR), a sljedeća pravila imati sa smanjenom konzistentnosti ili dalje rafinirati pravilo?
  - Kada stati s rafiniranjem pravila ako pravilo otpočetka nije konzistentno (kao u slučaju korištenja ACC i WRA)? Možda do potpunosti pokrivanja pozitivnih primjeraka?
- Koju heuristiku koristiti za koji problem?
- **Nema jednoznačnih (ni jednostavnih) odgovora**
- Ovisno o algoritmu učenja pravila

Strategija uspona na vrh odozgo prema dolje

- Engl. *Top-Down Hill-Climbing*
- Pravila se **sukcesivno specijaliziraju (rafiniraju)**
- Izabere se najprije određena heuristika, npr. ACC

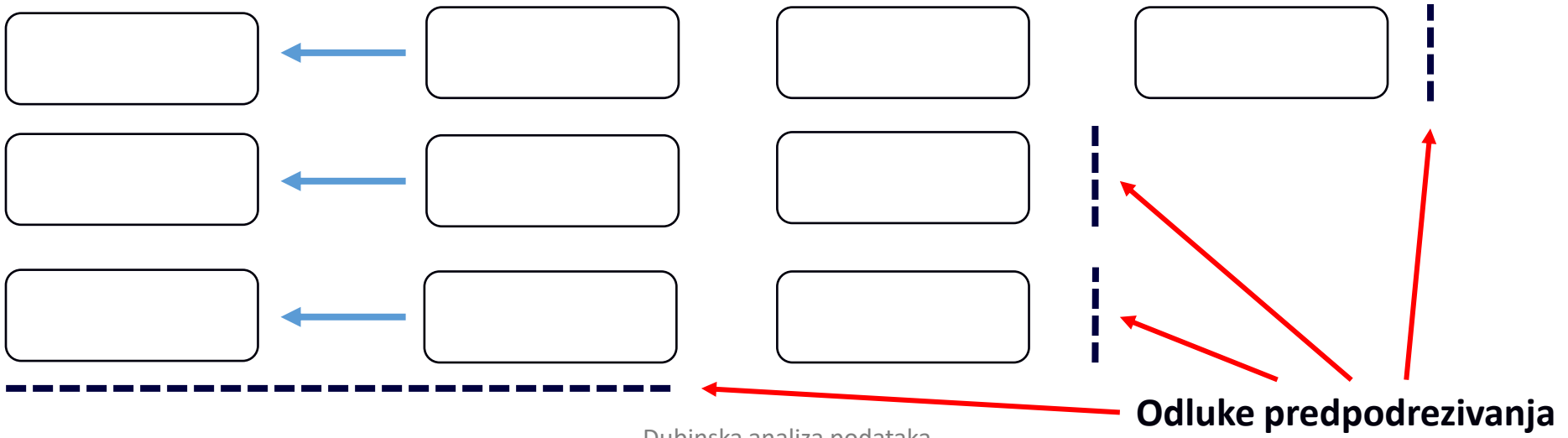
1. Počni s univerzalnim pravilom *R*: `p: -true` koje pokriva sve primjerke
2. Evaluiraj sve moguće načine za dodati jedan uvjet u *R* koristeći izabranu heuristiku
3. Odaberi najbolji način prema izabranoj heuristici
4. Ako je *R* zadovoljavajuće, vrati ga -- postavlja se kriterij na uspješnost pravila
5. Inače, vrati se na korak 2

Izbjegavanje prenaučenosti

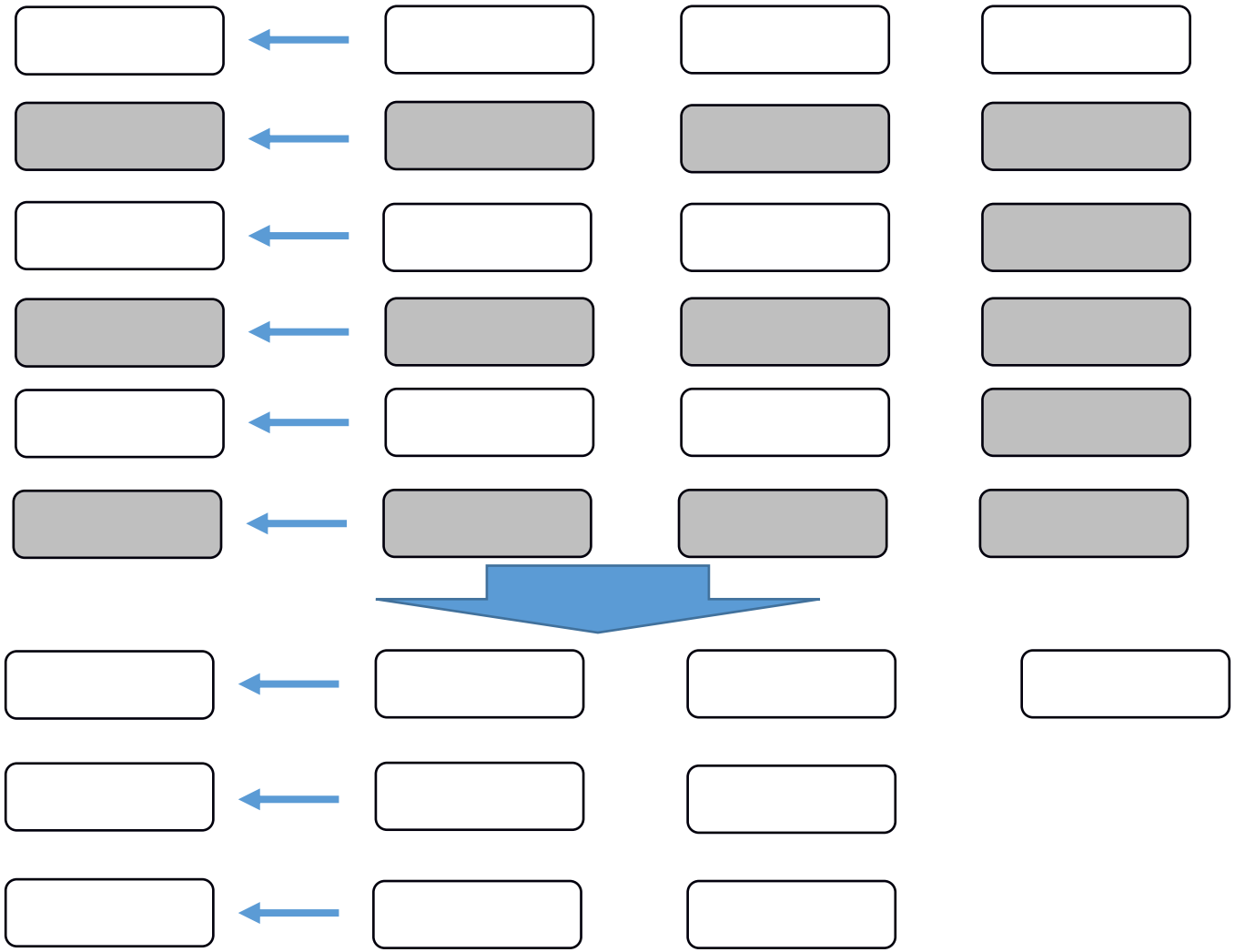
- Engl. *overfitting avoidance*
- Učenje koncepata tako da nisu svi pozitivni primjerci pokriveni s pravilom, neki negativni primjerci mogu biti pokriveni s pravilom
- **Većina algoritama fokusira se na kraća pravila** koja pokrivaju puno pozitivnih primjeraka (i neke negativne) umjesto da se fokusiraju na dulja pravila koja pokrivaju samo mali broj pozitivnih primjeraka
- Podrezivanje pravila (engl. *pruning*) – način pojednostavljenja kompleksnih pravila
  - **Predpodrezivanje** (engl. *pre-pruning*) – podrezivanje tijekom učenja pravila
  - **Postpodrezivanje** (engl. *post-pruning*) – podrezivanje nakon što je pravilo već izgrađeno
  - Kombinacija jednog i drugog načina

Predpodrezivanje

- **Teorija se nastoji zadržati jednostavnom** tijekom učenja pravila
- Potrebno je odlučiti kada treba prestati dodavati uvjete u pravilo
  - Ograničenje na opuštanje konzistentnosti (engl. *relax consistency constraint*)
- Potrebno je odlučiti kada treba prestati dodavati pravila u teoriju
  - Ograničenje na opuštanje potpunosti (engll. *relax completeness constraint*)
- Učinkovit način izbjegavanja prenaučenosti, ali nije toliko točan



Postpodrezivanje



- Na već izgrađenu teoriju primjenjuje se neki kriteriji za smanjenje veličine pojedinih pravila
- Npr. razmatra se uklanjanje zadnjeg **if** uvjeta sve dok ne dođe do značajnog smanjenja točnosti (a npr. za izgradnju se koristio kriterij preciznosti)
- Dodatni kriterij: pravilo mora pokrivati min. *X* primjeraka nakon podrezivanja, inače se miče

Višeklasna klasifikacija

- **Problem**: većina algoritama prilagođena je samo za binarne klasifikacijske probleme
- **Rješenje: binarizacija klasa** – jedan višeklasni problem pretvara se u **skup binarnih problema**
- U praksi najboljim rješenjem pokazala se binarizacija „**jedan protiv svih**” (engl. ***one-against-all, one-versus-all***)
  - Svaka klasa tretira se kao neovisni koncept, postoji *c* binarnih problema, gdje je *c* broj klasa
  - Primjerci iz jedne klase labelirani su kao pozitivni, a svi ostali kao negativni
  - Grade se pravila da pokrivaju čim više primjeraka iz pozitivne klase, nastavlja se postupak za ostale klase

Preklapanje pravila i problem nepostojanja pravila

- Što napraviti ako više pravila (njihovi **if** dijelovi) pokrivaju istog pojedinca (engl. *rule overlapping*)?
  - Problem ako su predviđanja kontradiktorna
- Dva moguća rješenja:
  - **Lista odlučivanja** (engl. *decision list*) – uvodi se **poredak po važnosti** u pravilima, prvo pravilo koje pokriva primjerak se aktivira
  - **Skup odlučivanja** (engl. *decision set*) – od pravila se zahtijeva da budu ili međusobno isključiva ili da postoji strategija za razrješavanje konflikta
    - U općem slučaju, utežano većinsko glasanje više pravila koje pokrivaju primjerak oko predikcije
- Što napraviti ako pojedinac nije pokriven ni s jednim pravilom (engl. *no-rule problem*)?
  - Češći slučaj na skupu za testiranje, ali i za primjerke koji su na skupu za učenje preostali nepokriveni izgrađenim skupom pravila
  - Rješenje: **koristiti *defaultno* pravilo**: najčešće predviđa klasu koja je u većini

Algoritmi indukcije pravila

OneRule

- Engl. *OneR algorithm, 1R, R. C. Holte, 1993.*
- Najjednostavniji algoritam učenja pravila, obično se koristi kao algoritam osnovne usporedbe (engl. *baseline*) za složenije algoritme
  - Iznenađujuće dobar u usporedbi s puno složenijim algoritmima, pogotovo za jednostavnije skupove podataka
- Uče se pravila (više njih) samo na temelju **jedne, najinformativnije značajke**
- Radi samo za kategoričku ciljnu značajku (**višeklasni** klasifikacijski problem)
- Ponaša se slično stablu odluke sa samo jednim, korijenskim čvorom
  - Za razliku od stabla odluke, grana po svim mogućim vrijednostima prediktivne značajke
- Pokriva sve primjerke u skupu podataka

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11, 6391

OneRule – algoritam

- Kontinuirane značajke se najprije diskretiziraju odabravši prikladne intervale (idući slajd)
- Za svaku prediktivnu značajku:
  - Stvara se unakrsna tablica između vrijednosti prediktivne značajke i kategoričke ciljne varijable
  - Za svaku vrijednost prediktivne značajke gradi se pravilo koje predviđa najčešću klasu za primjerke koje imaju tu određenu vrijednost prediktivne značajke
  - Računa se ukupna pogreška pravila
    - Pogreška = broj primjeraka koji ne pripadaju najčešćoj klasi / ukupan broj primjeraka
- Izabere se značajka s najmanjom ukupnom pogreškom kao rezultantna značajka (model)
  - Ako dvije značajke imaju istu najmanju ukupnu pogrešku, odabire se jedna slučajno

Npr. skup podataka Iris

petallength:	< 2.45	-> Iris-setosa
	< 4.75	-> Iris-versicolor
	>= 4.75	-> Iris-virginica

OneRule – diskretizacija

- Za kontinuirane značajke, diskretizacija na mjestima gdje se mijenja vrijednost ciljne značajke dovela bi do prenaučenosti
  - Kontrola prenaučenosti: definira se **minimalan broj primjeraka** (npr. 3) s istom vrijednosti ciljne značajke u svakom odjeljku (*binu*);
  - Optimizacija: susjedni binovi s istom pretežnom vrijednosti ciljne značajke se spajaju

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

Yes |	No |	Yes	Yes	Yes |	No	No |	Yes	Yes	Yes |	No |	Yes	Yes |	No

Yes	No	Yes	Yes	Yes |	No	No	Yes	Yes	Yes |	No	Yes	Yes	No

Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes |	No	Yes	Yes	No

Prediktivna značajka *temperature* (u °F) i pripadajuće vrijednosti ciljne značajke *play*

Podjele koje bi bile između vrijednosti ciljne značajke dovele bi do prenaučenosti

Definicija min. broja primjeraka (ovdje 3) većinske klase po podjeli, samo na rubu je 2

Optimizacija

PRISM

- *J. Cendrowska, 1987.*
- Temeljni algoritam (slijednog) pokrivanja primjeraka s više pravila (engl. *sequential covering*)
- Uklanja primjerke koji su pokriveni s određenim pravilom
- U osnovnoj varijanti radi s binarnom ciljnom značajkom, za višeklasnu klasifikaciju koristi pristup jednog protiv svih
  - Za svaku klasu gradi svoj skup pravila, što za testni primjerak pokriven s više pravila može biti problem
  - Najčešće se koristi heuristika većinskog glasanja da se odredi ispravna predikcija
- Gradi pravila koja su savršena (prema kriteriju preciznosti = 100%)
  - Sklonost prenaučenosti

Cendrowska, J. (1987). PRISM: An algorithm for inducing modular rules. International Journal of Man-Machine Studies, 27(4), 349-370

PRISM – algoritam

- Za svaku klasu ciljne značajke *C*:
  - Inicijaliziraj *E* kao skup primjeraka dotične klase
  - Dok *E* sadrži primjerke klase *C*:
    - Izgradi pravilo *R* s praznom lijevom stranom koja predviđa klasu *C*
    - Dok *R* nije 100% precizan (ili više nema značajki koje bi se mogle dodati):
      - Za sve neiskorištene parove *A=v*
        - Razmotri dodavanje uvjeta *A=v* na lijevu stranu pravila *R* i izračunaj preciznost pravila
      - Izaberi *A=v* takav da se maksimizira preciznost (pouzdanost) PREC = $p/(n+p)$, a ako ima izjednačenih koristi uvjet s najvećim brojem ispravnih (p)
      - Dodaj *A=v* u *R*
    - Ukloni primjerke pokrivene s *R* iz skupa primjeraka

- Izgradnja pravila ide dokle je potrebno (ili moguće), za preostale primjerke klase *C* grade se nova pravila

RIPPER

- Engl. *Repeated Incremental Pruning to Produce Error Reduction*, *W. W. Cohen, 1995.*
- Zasnovan na **inkrementalnom podrezivanju sa smanjenom pogreškom** (engl. *Incremental Reduced-Error Pruning*, **IREP**)
  - Kombinirana strategija podrezivanja
- Složeni algoritam učenja klasifikacijskih pravila s dobrom generalizacijom
  - Koristi nekoliko pametnih heuristika
- **Prednosti:** dobro radi s numeričkim prediktivnim značajkama, nalazi kompaktan skup korisnih pravila u većini situacija, radi brzo i na relativno velikom skupu podataka
- **Nedostatak:** Ako je problem složen (uvjeti pokrivaju mali broj primjeraka), nema garancije da će pronaći korisna pravila
- Implementacija: https://pypi.org/project/wittgenstein/

Cohen, William W. “Fast effective rule induction.” Machine Learning Proceedings (1995). 115-123

RIPPER – algoritam

- Algoritam gradi pravila za svaku pojedinu klasu ciljne značajke u četiri glavne faze
- Inicijalizira se *E* kao skup svi primjeraka za učenje
- Za svaku klasu *C*, **od najmanje do najveće** (prema broju primjeraka):
  1. **Izgradnja pravila (BUILD)**
  2. **Optimizacija pravila (OPTIMIZE)**
  3. **Pometanje preostalih primjeraka (MOP UP)**
  4. **Čišćenje loših pravila (CLEAN UP)**
     - Uklanjaju se iz *E* svi primjerci klase C_i, uči se na primjercima preostalih klasa, zadnja klasa koja ostaje je *defaultna* klasa

RIPPER – izgradnja pravila

- **1. Izgradnja pravila:**
  - *E* se podijeli u **skup za izrastanje pravila** (engl. *growing set*) i u **skup za podrezivanje** (engl. *pruning set*) u omjeru 2:1
  - Ponavljaj sljedeće faze dok (a) nema više nepokrivenih primjeraka u *C* ili (b) je opisna duljina (engl. *description length*, *DL*) skupa pravila i primjeraka 64 bita veća od najmanje dotad pronađene *DL* ili (c) stopa pogreške pravila ne pređe 50%:
    - **Faza izrastanja:** izgradi pravilo pohlepno dodajući uvjete $(A_c = v, A_n \geq v, A_n \leq v)$ dok pravilo nije 100% točno (po kriteriju točnosti *ACC*) ispitujuću svaku moguću vrijednost svake značajke i odabirajući uvjet s najvećom informacijskom dobiti *IG*
    - **Faza podrezivanja:** čim je pravilo izgrađeno, podrezuj uvjete od zadnjeg prema prvome. Nastavlja se dokle vrijednost *W* pravila raste

RIPPER – izgradnja pravila

- **Opisna duljina DL:** složena heuristička formula koja uzima u obzir (s desna na lijevo):
  - broj bitova potrebnih za poslati skup primjeraka u odnosu na skup izgrađenih pravila
  - broj bitova potrebni za poslati pravilo s *k* uvjeta
  - broj bitova potrebnih za poslati cijeli broj *k* te
  - dodatni proizvoljni faktor od 0.5 za kompenzaciju moguće redundantnosti u značajkama
  - $DL = 0.5 * \left(\|k\| + k * \log_2 \frac{1}{p} + (n-k) * \log_2 \frac{1}{1-p} \right)$
- Točnost pravila: ***ACC*** **=** (p + N - n) / (P + N)
- Informacijska dobit podjele: **IG(*X*|*Y*)** = H(*X*) – H(*X*|*Y*) (vidjeti 4. predavanje)
- Heuristička **vrijednost pravila *W*** (Cohen 1995): *W* = (*p* - *n*) / (*p* + *n*)
  - Wekina implementacija: *W* = (*p* + 1) / (*p* + *n* + 2)

RIPPER – optimizacija pravila

- **2. Optimizacija pravila:**
  - Generiranje varijanti:
    - Za svako pravilo *R* za klasu *C*:
      - Podijeli nanovo *E* u skupove za izrastanje i podrezivanje
      - Ukloni sve primjerke iz skupa za podrezivanje koji su pokriveni ostalim pravilima za klasu *C*
      - Koristi fazu izrastanja i fazu podrezivanje (iz 1. glavne faze) za generiranje i podrezivanje dvaju kompetitivnih pravila na novoj podjeli podataka:
        - *R1* je novo pravilo, izgrađeno ispočetka
        - *R2* je pravilo generirano pohlepnim dodavanjem uvjeta na *R*
      - Podreži koristeći mjeru *ACC* (umjesto *W*) na ovom reduciranom skupu podataka
  - Odabir predstavnika:
    - Zamijeni R s {R, R1 ili R2}, izaberi ono pravilo koje ima najmanji DL

RIPPER – optimizacija pravila

- Glavna ideja kod faze optimizacije: razmotriti tri varijante svakog izrađenog pravila za klasu *C* i odabrati onu najbolju
- Uzima se izvorno pravilo i dvije njegove modifikacije
  - Za modifikacije pravila se ponovno koristi postpodrezivanje
  - Razlike u odnosu na ranije:
    - primjerci koje pokrivaju druga pravila za dotičnu klasu uklanjaju se iz skupa za podrezivanje
    - umjesto ***W*** koristi se ***ACC*** kao kriterij podrezivanja
- Zašto se tako radi?
  - Iskustveno, veća varijacija kriterija i načina korištenja primjeraka za podrezivanje može dovesti do skupa pravila s boljom generalizacijom

RIPPER – pometanje preostalih primjeraka

- **3. Pometanje preostalih primjeraka:**
  - Ako postoji nepokrivenih primjeraka klase *C*:
    - Vrati se u 1. fazu (BUILD) kako bi se generirala nova pravila za preostale primjerke

  - **Ideja**: moguće je da se zbog optimizacije dogodi da neki primjerci ne budu pokriveni, a ovime im se daje (jedna) dodatna šansa u novim pravilima za klasu *C*

RIPPER – pometanje preostalih primjeraka

- **4. Čišćenje loših pravila:**
  - Izračuna se DL za cijeli skup pravila za klasu *C* te za skup podataka kada se redom pojedinačno uklanja svako pravilo
  - Uklanja se svako pravilo čijim micanjem je došlo do povećanja DL-a u odnosu na puni skup pravila

  - **Ideja**: svako pravilo u skupu bi trebalo doprinjeti **smanjenju opisne duljine čitavog skupa pravila**, ako neko pravilo to ne radi, onda se smatra suvišnim

RIPPER – hiperparametri

- RIPPER može **ponavljati optimizacijski ciklus** za svaku ciljnu klasu (2. faza) i to *k* puta (RIPPER-*k*)
- U praksi se pokazuje da algoritam radi najbolje za $k = 2$ (tzv. RIPPER2)
- Moguće je isprobati i druge vrijednosti ovog hiperparametra

- Moguće je postaviti **minimalni broj primjeraka** (ili minimalnu težinu) koje **pravilo treba pokriti** da bi ga se uvrstilo u konačni skup pravila
  - Ovo je dodatni kriterij koji se koristi prilikom izgradnje / podrezivanja

Poznatiji algoritmi indukcije pravila

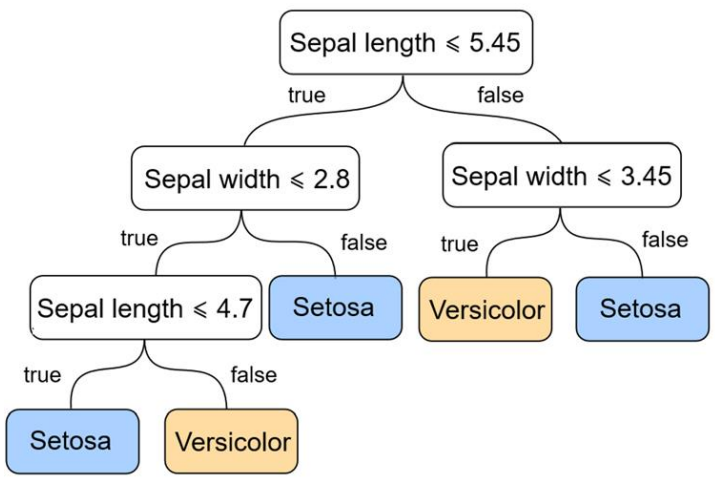
- **CN2** – jedan od prvih algoritama pokrivanja, koristi korisnički definiranu mjeru za evaluaciju najboljih pravila
- **PART** – gradi djelomično C4.5 stablo u svakoj iteraciji koristeći varijantu predpodrezivanja u kojoj se ispituje *IG* čvorova djece, odabire jedan put kao pravilo koje ima najveću pokrivenost primjeraka pozitivne klase
- **MD5Rules** (M5' trees) – generira listu odlučivanja za regresijski problem, u svakoj iteraciji generira stablo i pretvori najbolji list (linearni model) (po kriteriju MSE ili MAE) u pravilo
- **RIDOR** – *Ripple DOwn Rule learner* – najprije gradi *defaultno* pravilo i potom dodaje iznimke (Except A>v) redom na to pravilo; iznimke se grade kao u BUILD fazi RIPPER-a
- **IDS** – *Interpretable Decision Sets* – izgradnja skupa pravila kao **optimizacijski problem**, optimira se broj pravila, veličina pravila, preklapanje pravila, pokrivenost svake klase, točnost i potpora
  - Implementacija: https://github.com/jirifilip/pyIDS

Clark, P. and Niblett, T (1989) The CN2 induction algorithm. Machine Learning 3(4):261-283
Frank, E. and Witten, I.H. (1998): Generating Accurate Rule Sets Without Global Optimization. In: Fifteenth International Conference on Machine Learning, 144-151
Holmes, G., Hall M. and Frank, E. (1999): Generating Rule Sets from Model Trees. In: Twelfth Australian Joint Conference on Artificial Intelligence, 1-12
Gaines, B. R., and Compton, P. (1995) Induction of ripple-down rules applied to modeling large data bases. Journal of Intelligent Information Systems, 5(3), 211-228
Lakkaraju H. et al. (2016) Interpretable Decision Sets: A Joint Framework for Description and Prediction, Proc. SIGKDD Int. Conf. KDD, 1675-1684.

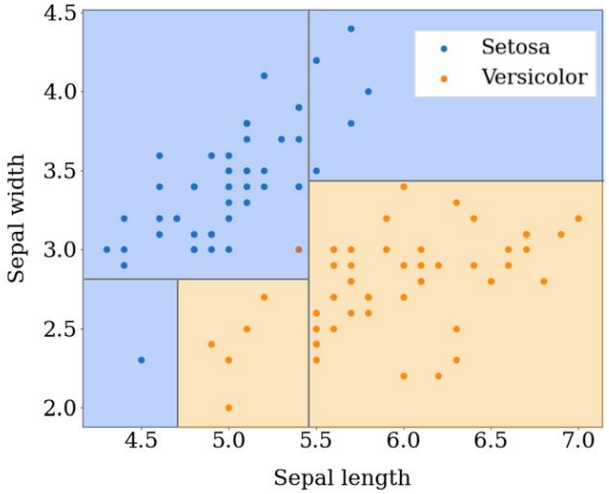
Optimalna stabla odluke

Stabla odluke

- Prediktivni modeli u obliku stabla
  - Svaki čvor sastoji se od ispitivanja uvjeta, a list daje predviđanje modela (klasa ili numeričko predviđanje)
- Potpuna optimizacija stabla odluke je **NP-težak problem**
- Teži se čim točnijim i kompaktnijim (manjim) modelima
- Suvremene metode za izgradnju stabla odluke **usporedive po točnosti s ansamblima**
- Dvije faze razvoja izgradnje stabala odluke
  - **Heuristička** (do 2016.)
  - **Optimizacijska** (od 2017. do danas)



(a) Tree visualization



(b) Partitioning visualization

Izvor: V. G. Costa and C. Pedreira, Recent advances in decision trees: an updated survey, Artificial Intelligence Review (2023) 56:4765–4800 https://doi.org/10.1007/s10462-022-10275-5

Stabla odluke

- Heuristička faza
  - Stabla se grade na pohlepan način, grananje u čvorovima prema nekom informacijskom kriteriju – heuristici
  - Stabla se mogu, ali ne moraju podrezivati: obično se koristi naknadno podrezivanje (engl. *postpruning*): zamjena podstabla (engl. *subtree replacement*) i uzdizanje podstabla (engl. *subtree raising*)
  - Predstavnici: CART (1984.), ID3 (1987.), C4.5 (1993.), C5.0 (1993. – 2022.)
- Literatura
  - CART: L. Breiman, J. Friedman, R.A. Olshen, C. J. Stone, Classification and Regression Trees, 1st ed. Chapman and Hall/CRC, 1984.
  - ID3: J.R. Quinlan, Induction of Decision Trees, Mach. Learn. 1, 81 – 106, 1986.
  - C4.5: J. R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.
  - C5.0 (komercijalno): https://www.rulequest.com/see5-info.html

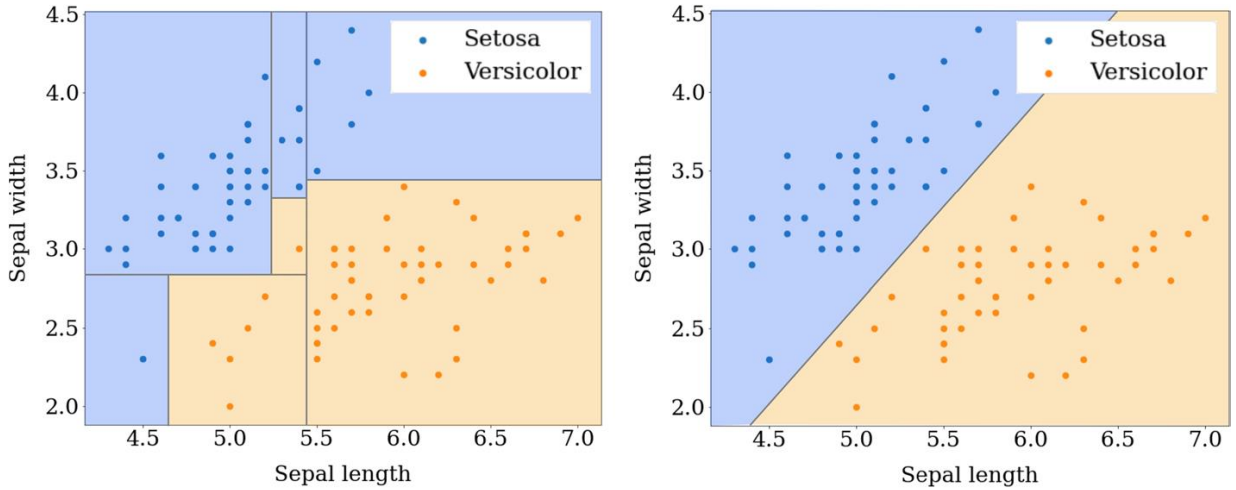
Stabla odluke

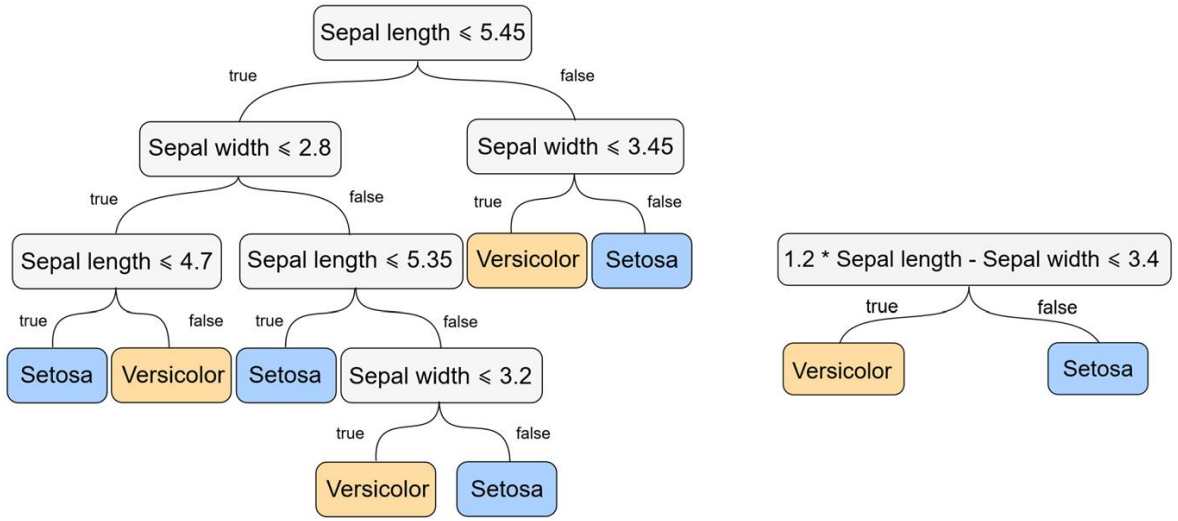
- Optimizacijska faza
  - Stabla se grade po principima optimizacije, slijedeći određene optimizacijske metode ovisno o algoritmu
  - Pri optimizaciji, optimira se neka performansa stabla (npr. točnost) uzevši u obzir ograničenja stabla (npr. *max* dubina)
  - Stabla se ne podrezuju već se odmah gradi optimalna struktura
  - Predstavnici: OCT (2017.), GOSDT (2020.), ORCT (2021.), OMDT (2023.)
- Literatura:
  - **OCT**: D. Bertsimas, J. Dunn (2017) Optimal classification trees. Mach Learn 106, 1039–1082 (2017). https://doi.org/10.1007/s10994-017-5633-9
  - GOSDT: J. Lin et al. (2020) Generalized and Scalable Optimal Sparse Decision Trees, Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119.
  - ORCT**:** R. Blanquero et al. (2021) Optimal randomized classification trees, Computers & Operations Research, 132, 105281. https://doi.org/10.1016/j.cor.2021.105281
  - OMDT: J. Boutilier, C. Michini, Z. Zhou (2023) Optimal multivariate decision trees, Constraints 28, 549–577. https://doi.org/10.1007/s10601-023-09367-y
  - V. G. Costa and C. Pedreira (2022) Recent advances in decision trees: an updated survey, Artificial Intelligence Review 56:4765–4800 https://doi.org/10.1007/s10462-022-10275-5

Stabla odluke

- Više pristupa optimizaciji stabla
  - **Matematičko programiranje** – izgradnja stabla se formulira kao matematički problem s nekoliko diskretnih izbora koji se trebaju ispitati, koristi se *mixed-integer* optimizacija ili neki drugi pristupi – **OCT**
  - **Izgradnja stabla kao SAT problem** – stablo je optimalno ako je konzistentno (svi primjerci su savršeno predviđeni) i minimalne dubine – problem ako su podaci nekonzistentni ili ako je puno podataka – nije naročito zaživjelo u praksi
  - **Specijalizirani pristupi optimizaciji** – zasnovano na specifičnim ograničenjima pri izgradnji stabla i iscrpnoj pretragi po nekom optimizacijskom kriteriju (točnost ili nešto drugo) – GOSDT
- Metode mogu koristiti **univarijatno** (uobičajeno, npr. age >= 18) ili **multivarijatno** grananje (npr. linearno: 1.2 * SepalLength – SepalWidth <= 3.4) za podjelu vrijednosti u čvorovima
- Većina algoritama razmatra značajke samo kao numeričke vrijednosti (i koristi *one-hot encoding* za pretvorbe), dok neki algoritmi znaju raditi samo s kategoričkim vrijednostima (čak samo binarnima)
- Metode imaju problema sa skalabilnosti, jer većina radi dobro samo na manjim skupovima podataka i manjom dubinom stabla (4 – 5)

Stabla odluke – univarijatno i multivarijatno grananje





(a) Univariate tree

(b) Linear multivariate tree

Izvor: V. G. Costa and C. Pedreira, Recent advances in decision trees: an updated survey, Artificial Intelligence Review (2023) 56:4765–4800 https://doi.org/10.1007/s10462-022-10275-5

OCT (OCT-H)

- Predvodnik optimizacijskih metoda za izgradnju stabala odluka iz 2017. godine
- Značajke:
  - ***Mixed-integer* optimizacija (MIO)** – optimizacija s nizom diskretnih odluka i odgovarajućih ograničenja
  - **Multivarijatno grananje** – relaksacija ograničenja na MIO dovodi do *OCT with hyperplanes* (OCT-H) – linearnim podjelama na čvorovima koje zadržavaju dobru interpretabilnost a pospješuju točnost i smanjuju veličinu stabla
  - Koristi sve numeričke značajke, radi višeklasnu klasifikacija
  - Koristi generalni rješavač ograničenja za rješenje optimizacijskog problema (GUROBI, CPLEX)
  - Gradi kompaktna stabla s točnosti usporedivom sa slučajnom šumom (ili malo lošije od nje)
- Nedostatci: ne radi dobro na velikim skupovima podataka (više od 10.000 primjeraka) zbog sporosti postavljenih ograničenja
- Implementacije:
  - https://www.interpretable.ai/products/optimal-trees/
  - https://github.com/zachzhou777/S-OCT

OCT (OCT-H)

- MIO ograničenja za OCT:
  - Na svakom novom čvoru mora se izabrati grananje ili zaustavljanje grananja
  - Nakon što se odabere zaustavljanje grananja na čvoru, mora se odabrati labela koja će se dodijeliti ovom novom listu
  - Nakon odabira grananja, mora se odabrati po kojoj značajki će se granati
  - Pri klasifikaciji primjeraka za učenje prema stablu u izgradnji mora se odabrati kojem će listu biti dodijeljen primjerak da se poštuje struktura stabla
- Hiperparametri prilikom izgradnje OCT-H stabala su: maksimalna dubina stabla D_{max}, minimalni broj primjeraka u listovima N_{min}, najveći broj mogućih grananja $C_{max} = 2^D - 1$ i najveći broj značajki za grananje u jednom čvoru p (multivarijatno grananje)
- Istražuje se prostor mogućih stabala s danim hiperparametrima i zadanim ograničenjima
- Sva ograničenja se formuliraju u matematički opis i daju se na rješavanje nekom rješavaču

OCT (OCT-H) – sva ograničenja OCT-H modela

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \cdot \sum_{t \in \mathcal{T}_B} \sum_{j=1}^{p} s_{jt}$$

$$\text{s.t.} \quad L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad k = 1, \ldots, K, \quad \forall t \in \mathcal{T}_L,$$

$$L_t \leq N_t - N_{kt} + n c_{kt}, \quad k = 1, \ldots, K, \quad \forall t \in \mathcal{T}_L,$$

$$L_t \geq 0, \quad \forall t \in \mathcal{T}_L,$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \quad k = 1, \ldots, K, \quad \forall t \in \mathcal{T}_L,$$

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in \mathcal{T}_L,$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in \mathcal{T}_L,$$

$$\boldsymbol{a}_m^T \boldsymbol{x}_i + \mu \leq b_m + (2 + \mu)(1 - z_{it}), \quad i = 1, \ldots, n, \quad \forall t \in \mathcal{T}_B, \quad \forall m \in A_L(t),$$

$$\boldsymbol{a}_m^T \boldsymbol{x}_i \geq b_m - 2(1 - z_{it}), \quad i = 1, \ldots, n, \quad \forall t \in \mathcal{T}_B, \quad \forall m \in A_R(t),$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \quad i = 1, \ldots, n,$$

$$z_{it} \leq l_t, \quad \forall t \in \mathcal{T}_L,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t, \quad \forall t \in \mathcal{T}_L,$$

$$\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, \quad \forall t \in \mathcal{T}_B,$$

$$\hat{a}_{jt} \geq a_{jt}, \quad j = 1, \ldots, p, \quad \forall t \in \mathcal{T}_B,$$

$$\hat{a}_{jt} \geq -a_{jt}, \quad j = 1, \ldots, p, \quad \forall t \in \mathcal{T}_B,$$

$$-s_{jt} \leq a_{jt} \leq s_{jt}, \quad j = 1, \ldots, p, \quad \forall t \in \mathcal{T}_B$$

$$s_{jt} \leq d_t, \quad j = 1, \ldots, p, \quad \forall t \in \mathcal{T}_B,$$

$$\sum_{j=1}^{p} s_{jt} \geq d_t, \quad \forall t \in \mathcal{T}_B,$$

$$-d_t \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_B,$$

$$d_t \leq d_{p(t)}, \quad \forall t \in \mathcal{T}_B \backslash \{1\},$$

$$z_{it}, l_t \in \{0, 1\}, \quad i = 1, \ldots, n, \quad \forall t \in \mathcal{T}_L,$$

$$s_{jt}, d_t \in \{0, 1\}, \quad j = 1, \ldots, p, \quad \forall t \in \mathcal{T}_B.$$

Izvor: D. Bertsimas, J. Dunn (2017) Optimal classification trees. Mach Learn 106, 1039–1082 (2017). https://doi.org/10.1007/s10994-017-5633-9

Zaključak

- Indukcija pravila iz podataka je jedan od najkorisnijih načina za razumjeti podatke a istovremeno izgraditi model s mogućnosti primjene na novim podacima
- Za širok raspon primjene, dostupni su različiti algoritmi za učenje pravila, svaki sa svojim prednostima i nedostacima*
- U prvoj aproksimaciji, za bilo koji tablični skup podataka uobičajenih dimenzija, preporuka je isprobati algoritam pravila RIPPER s *defaultnim* postavkama
- Optimalna stabla odluke novi su pristup starom problemu izgradnje kvalitetnih i interpretabilnih stabala odluke – stalno se razvijaju novi algoritmi

*Za detaljnije informacije o ovoj temi preporuka je knjiga:
J. Fürnkranz, N. Lavrač, D. Gamberger, Foundations of Rule Learning, Springer, 2012