Stabla odlučivanja i ansambl metode

Robić, Pavao

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, School of Applied Mathematics and Informatics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet primijenjene matematike i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:389612

Sveučilište J.J. Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni diplomski studij matematike
Smjer: Financijska matematika i statistika

Pavao Robić

Stabla odlučivanja i ansambl metode

Diplomski rad

Osijek, 2023.

Sveučilište J.J. Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni diplomski studij matematike
Smjer: Financijska matematika i statistika

Pavao Robić

Stabla odlučivanja i ansambl metode

Diplomski rad

Mentor: izv.prof.dr.sc. Nenad Šuvak

Komentor: prof.dr.sc Nataša Šarlija

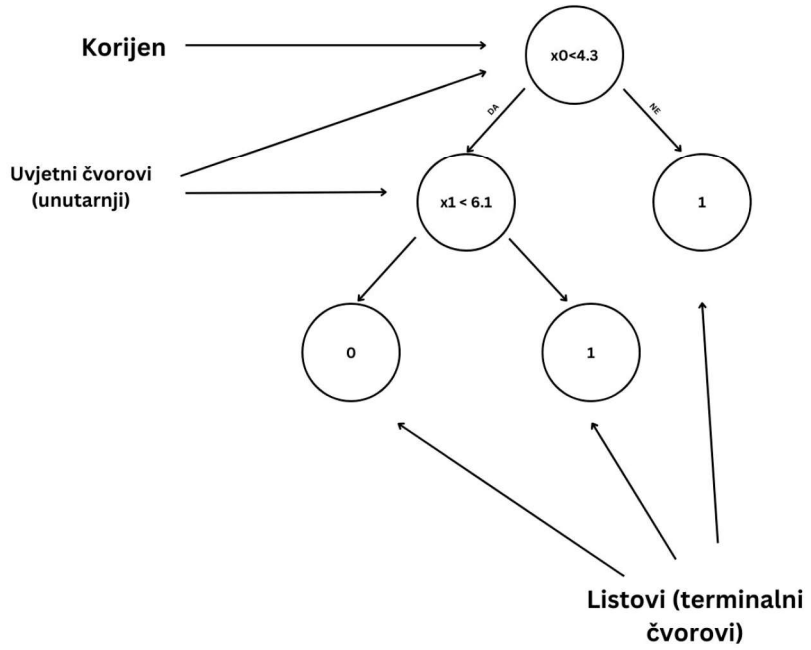
Osijek, 2023.

Sadržaj

Uvod

U današnjem svijetu, sve češće slušamo o umjetnoj inteligneciji, da ista predstavlja budućnost te da će nam ista oduzeti poslove, a koliko je to točno vrijeme će pokazati. U ovom vremenu prediktivne analize, zadatak je izvući bitne informacije od danih podataka i to je dovelo do novih tehnika strojnog učenja. Među njima, regresijska i klasifikacijska stabla ističu se kao moćna oruđa koja su dobila na popularnosti radi jednostavnosti, lake interpretabilnosti te značajnim sposobnostima predviđanja. Ovaj rad udubljuje se u fascinantan svijet regresijskih i klasifikacijskih stabala, posebno ističe njihove teorijske temelje i koliko su dobra u praksi.

Esencija odlučivanja temeljenog na podacima leži u razumijevanju obrazaca, veza i zavisnosti skrivenih među podacima. Regresijska i klasifikacijska stabla nude uvjerljiv pristup toj esenciji, dajući strukturirani način podjele podataka u homogene podskupove. Ta stabla, čija struktura grananja liči na prava stabla, pomažu nam navigirati kroz kompleksnije strukture podataka, pružajući nam uvid u neke stvari koje bi možda ostale neprimjećene. Bio nam cilj predvidjeti kretanje cijena dionica, medicinskih dijagnoza pacijenata ili klaficirati želje potrošača, stabla nude svestrana riješenja nebrojivim problemima u svijetu.



Slika 1: Primjer stabla s terminologijom

Stabla odlučivanja su tip algoritma strojnog učenja koji se može koristiti i za klasifikaciju i za regresiju. Temelje se na konceptu podjele unesenih podataka na manje podskupove, koji su zadani nekim prethodno definiranim uvjetom. Ovaj proces podjele se rekurzivno ponavlja sve dok podaci nisu do kraja podijeljeni u podskupove sa sličnim ciljanim vrijednostima. Struktura koja nastaje tim načinom podjele zove se stablo odlučivanja. Stabla odlučivanja su oblik nadziranog učenja, što znači da uče na temelju trening podataka kako bi napravila predikciju na

novim, neviđenim podacima. Kad kažemo da dijelimo podatke na *trening* i *test* skup to znači da danu bazu podataka podijelimo na dva djela; dio za trening - na njemu "treniramo" model i testni dio - na njemu testiramo kvalitetu modela. Ta dva skupa se najčešće dijele omjerom 80/20 no ne nužno i ne uvijek. Na slici 1 može se vidjeti primjer stabla s danom analogijom stabla. U ovom diplomskom radu usredotočujemo se posebno na regresijska te posebno klasifikacijska stabla te prolazimo kroz način na koji funkcioniraju, a obradit ćemo i metode poboljšavanja stabala kao što su bagging, generiranje slučajnih šuma i boosting.

1 Motivacija i osnovni koncepti

Regresijski modeli igraju veliku ulogu u analizi podataka, daju nam predviđanja numeričkih ili kvalitativnih vrijednosti zavisne varijable (*response*) koju modeliramo preko unaprijed zadanih prediktora. Nažalost, u stvarnom životu, tradicionalni linearni modeli nisu uvijek najbolji za opisivanje efekata prediktora na response. Regresijski model najčešće je oblika:

$$E(Y|X_1, ..., X_p) = \alpha + f_1(X_1) + ... + f_p(X_p) + \epsilon. \tag{1.1}$$

Po običaju, $X_1, ..., X_p$ predstavljaju prediktore, dok je Y varijabla koju modeliramo preko danih prediktora, a f_j su neparametarske funkcije. U slučaju kad imamo binomnu klasifikaciju, koristimo model logističke regresije s link funkcijom

$$\log\left(\frac{\mu(X)}{1-\mu(X)}\right) = \alpha + \beta_1 X_1 + ... + \beta_p X_p. \tag{1.2}$$

Model logističke regresije zamijenjuje svaki linearni parametar β_j s općenitijim zapisom:

$$\log\left(\frac{\mu(X)}{1-\mu(X)}\right) = \alpha + f_1(X_1) + ... + f_p(X_p),$$

gdje su f_j glatke funkcije. Aditivni model logističke regresije je primjer generaliziranog aditivnog modela. Očekivanje od X, uvjetno na Y je povezano s aditivnom funkcijom prediktora preko link funkcije g:

$$g(\mu(X)) = \alpha + f_1(X_1) + ... + f_p(X_p). \tag{1.3}$$

Primjeri najčešćih link funkcija:

- $g(\mu) = \mu$ je identiteta, koristi se za linearne i aditivne modele gdje zavisna varijabla ima normalnu distribuciju
- $g(\mu) = logit(\mu)$ za modeliranje binomnih vjerojatnosti ($logit(p) = \log(\frac{p}{1-p})$)
- $g(\mu) = \log(\mu)$ za log-linearne ili log-aditivne modele u slučaju uzorka iz Poissonove distribucije.

Tree-based metode dijele prediktorski prostor na skup pravokutnika koji su za dimenziju veću od 2 višedimenzionalni, te dodijeljuju jednostavan model (npr. konstantu) svakom od njih. Konceptualno, stabla su jednostavna i vrlo moćna. U nastavku ćemo opisati popularnu metodu za tree-based regresiju i klasifikaciju zvanu CART.

Valja još kratko uvesti krosvalidaciju i bootstrap jer iste koristimo kasnije. K kros-validacija je pristup kojim na slučajan način dijelimo podatke na K podskupova, otprilike jednake veličine. Jedan od tih K poduzoraka, k-ti dio, koristimo kao test skup, dok ostalih $K-1$ poduzoraka koristimo kao trening skup. Ovaj postupak je ponovljen za svaki $k \in \{1, ..., K\}$, što rezultira sa K srednje-kvadratnih grešaka $MSE_1, \ldots, MSE_K$ nakon procjene modela na testnim podacima. Tada je kros-validacijska greška (CV) aritmetička sredina srednje kvadratnih grešaka $MSE_1, \ldots, MSE_K$, tj.

$$CV_{(K)} = \frac{1}{K}\sum_{i=1}^{K} MSE_i, \quad MSE_i = (\widehat{y}_i - y_i)^2, \tag{1.4}$$

gdje je $\widehat{y_i}$ predikcija, a y_i stvarna vrijednost.

Bootstrap je statistička metoda koju koristimo kada nas zanima varijabilnost ili pouzdanost parametra nekog statističkog modela, a populacija nam je nepoznata ili nam je teško doći do novih uzoraka iz populacije. Bootstrap metoda koristi jedan uzorak iz kojeg na slučajan način s vraćanjem odabire nove uzorke podataka (bootstrap uzorke). Zatim za svaki novi uzorak određuje procjenu traženog parametra.

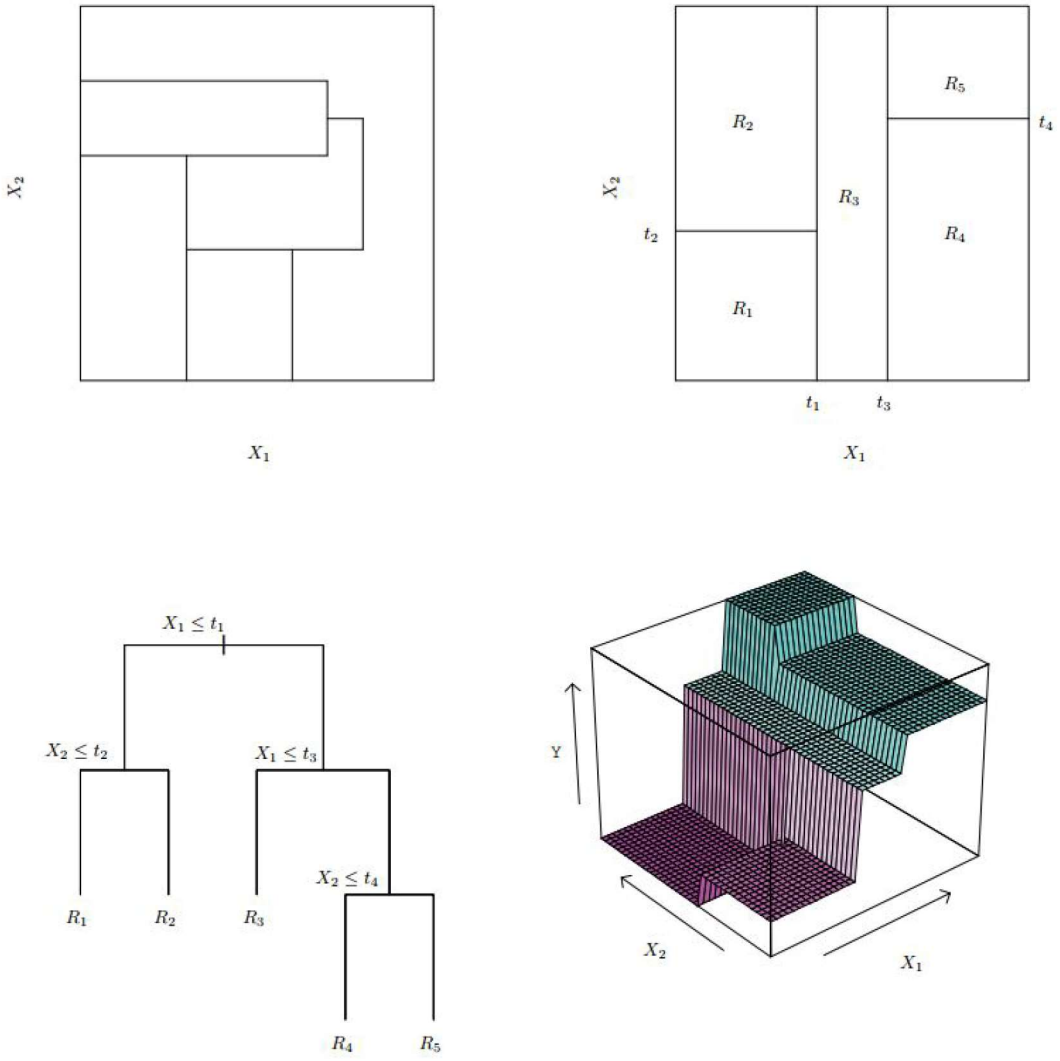
1.1 CART metoda

Pretpostavimo da imamo regresijski model s neprekidnom zavisnom varijablom Y te prediktorima X_1 i X_2 koji poprimaju vrijednosti u nekom intervalu. Slika 2 pokazuje podjelu prediktorskog prostora pravcima paralelnima s koordinatnim osima. U svakoj regiji prediktorskog prostora modeliramo Y sa različitom konstantom.

Kako bi pojednostavili stvari, ograničili smo se na rekurzivnu binarnu podjelu prediktorskog prostora, kakva je u gornjem desnom kutu na Slici 2. Prvo dijelimo prostor prediktora na dvije regije, te očekivanu vrijednost varijable Y modeliramo prosječnom vrijednosti prediktora u svakoj regiji. Biramo varijablu i točku podjele kako bismo dobili najbolju aproksimaciju. Nakon toga, dijelimo jednu ili obje regije na još dvije nove te to ponavljamo dok ne dođemo do nekog pravila zaustavljanja. Na desnom djelu Slike 2 prva podjela napravljena je u $X_1 = t_1$. Nakon tog regija $\{X_1 \leq t_1\}$ je podijeljena u $X_2 = t_2$, dok je regija $\{X_1 > t_1\}$ podijeljena u $X_1 = t_3$. Na kraju, regija $\{X_1 > t_3\}$ je podijeljenja u $X_2 = t_4$. Rezultat ovog procesa je podjela prostora u pet regija $R_1, R_2, ..., R_5$, kao što se vidi na slici. Odgovarajući regresijski model predviđa Y konstatom c_m koja je prosjek vrijednosti prediktora iz regije R_m, tj:

$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}. \tag{1.5}$$

Ovakav model može biti prezentiran binarnim stablom u donjem lijevom dijelu slike 2. Na vrhu stabla cijeli se skup podataka dijeli prvim uvjetom, gdje se podaci koji ga zadovoljavaju pridružuju lijevoj grani, a ostali desnoj. Listovi stabla odgovaraju regijama $R_1, ..., R_5$. Donji desni dio slike 2 predstavlja 3D graf plohe ovog regresijskog modela. Glavna prednost ovog stabla je lagana interpretabilnost. S više od dva prediktora, skiciranje prediktorskog prostora kakav je na gornjem desnom dijelu slike 2 postaje teško izvedivo, ali reprezentacija stabla ostaje ista. Ovakva reprezentacija jako je popularna u medicini, vjerojatno jer dobro opisuje način razmišljanja doktora koji donosi dijagnozu, obzirom na karakteristike pacijenta.



Slika 2: Podjela prediktorskog prostora.

2 Regresijska stabla

Dolazimo do pitanja, kako napraviti regresijsko stablo. Pretpostavimo da imamo bazu podataka od p prediktora i response varijable Y, te N jedinki: (x_i, y_i) za $i = 1, ..., N$ gdje je $x = (x_1, ..., x_p)$. Algoritam treba automatski odlučiti gdje napraviti podjelu, na temelju kojih prediktora i kakav oblik će stablo imati. Pretpostavimo da imamo prostor prediktora podijeljen u M regija $R_1, ..., R_M$ i modeliramo response Y kao konstantu c_m u svakoj regiji:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \tag{2.1}$$

Ukoliko kao kriterij uzmemo metodu najmanjih kvadrata $\sum_{i=1}^{M}(y_i - f(x_i))^2$, lako je vidljivo da je najbolji procjenitelj $\hat{c}_m$ prosječna vrijednost od y_i u regiji R_m:

$$\hat{c}_m = \overline{y}_i I(x_i \in R_m). \tag{2.2}$$

Pronalaženje najbolje binarne podjele metodom najmanjih kvadrata je općenito računski neizvedivo, jer prva najbolja podjela neće nužno voditi krajnjoj najboljoj podjeli (kasnije ćemo to detaljnije obraditi i pojasniti). Stoga krećemo s *greedy* algortmom. Počevši sa svim danim podacima, pretpostavimo da je X_j prediktor na temelju kojeg radimo podjelu, s točka podjele, te definiramo dvije poluravnine: $R_1(j, s) = \{X|X_j < s\}$ i $R_2(j, s) = \{X|X_j \geq s\}$.

Tada biramo prediktor podjele X_j i točku podjele s koji minimiziraju sljedeći kriterij:

$$\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_1(j,s)} (y_i - c_2)^2 \tag{2.3}$$

Rješenje ovog minimizacijskog problema je oblika je $\hat{c}_1 = \overline{y}_i I(x_i \in R_1(j, s))$ i $\hat{c}_2 = \overline{y}_i I(x_i \in R_2(j, s))$. Nakon što smo pronašli traženi par, podatke dijelimo u dvije regije i postupak ponavljamo za ostale regije.

Koliko veliko bi stablo trebalo biti? Očito, preveliko stablo je sklono *overfittingu* podataka, a s premalim nećemo dobiti dovoljno informacija kako bismo kvalitetno reprezentirali response Y preko danih prediktora. Veličina stabla je parametar koji je povezan s kompleksnosti stabla. Jedan od načina pronalaska najoptimalnijeg stabla je vršenje podjele za određeni čvor jedino ako time smanjanje sume kvadrata u izrazu (2.3) prelazi neki, unaprijed zadani prag. Najbolja strategija jest izgraditi veliko stablo T_0, zaustavljajući proces samo kad se dostigne neki minimalni broj čvorova (npr. 5). Nakon tog, stablo obrezujemo koristeći metodu *cost-complexity pruning*, koju ćemo opisati u nastavku: definiramo podstablo $T \subset T_0$ kao bilo koje stablo koje možemo dobiti obrezivanjem stabla T_0 (tj. uklanjanjem unutarnjih, neterminalnih čvorova od istog). S m indeksiramo unutarnje čvorove, gdje čvor m predstavlja regiju R_m. Neka je $|T|$ broj terminalnih čvorova (listova) u T. Uz oznake:

- $N_m = \#\{x_i \in R_m\}$
- $\hat{c}_m = \dfrac{1}{N_m} \displaystyle\sum_{x_i \in R_m} y_i$

- $Q_m(T) = \dfrac{1}{N_m} \displaystyle\sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$

definiramo *cost-complexity* kriterij

$$c_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha T. \tag{2.4}$$

Ideja je, za svaki α, pronaći podstablo $T_\alpha \subseteq T_0$ tako da se minimizira $C_\alpha(T)$. Povećavanjem parametra $\alpha \geq 0$ utječemo na veličinu stabla (koliko je kompleksno i koliki je broj listova) i koliko detaljno razdvaja prostor prediktora u manje podskupove. Velike vrijednosti α vode manjim stablima T_α i obratno. Za $\alpha = 0$ dobije se početno stablo T_0. Može se pokazati da za svaki α postoji jedinstveno najmanje podstablo T_α koje minimizira $C_\alpha(T)$.

Kako bismo pronašli traženo T_α koristimo *weakest link pruning:* mičemo unutarnji čvor koji vodi najmanjem rastu $\sum_m N_m Q_m(T)$ te nastavljamo dok ne dođemo do korijena (prvog čvora) stabla. To daje konačan niz podstabala, te se može pokazati da niz sadrži stablo T_α. Procjena α vrši se krosvalidacijom, tako da se odabire $\hat{\alpha}$ koji minimalizira krosvalidiranu sumu kvadrata. Rezultat danog postupka daje nam stablo $T_{\hat{\alpha}}$.Motivirajmo regresijska stabla jednim laganim primjerom.

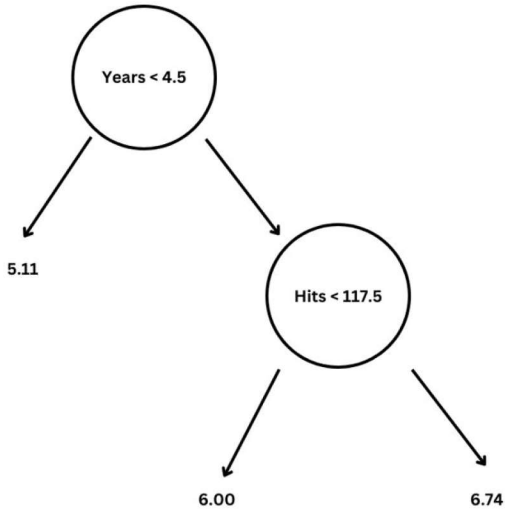
Primjer 1. *Predviđanje godišnje plaće igrača bejzbola pomoću regresijskog stabla. Koristimo bazu podataka Hitters kako bismo predvidjeli varijablu Salary (godišnju plaću u tisućama dolara) igrača bejzbola, prediktori koje koristimo su Years (broj godina koje je igrač proveo igrajući za velike lige) i Hits (broj pogodaka koje je igrač imao prošle godine). Varijable koje nemaju podatak o godišnjoj plaći odbacujemo te radimo logaritamsku transformaciju varijable Salary.*

Slika 3 pokazuje kakvo regresijsko stablo nastaje temeljem danih podataka. Sastoji se od serija pravila za podjelu koje počinju pri vrhu stabla. U vrhu stabla nalazi se prvi čvor, s uvjetom $\{Years < 4.5\}$, gdje sve varijable za koje je taj uvjet ispunjen idu u lijevu granu. Procijenjena vrijednost godišnje plaće za igrače koji imaju manje od 4.5 godina iskustva u većim ligama računa se kao prosječa vrijednost plaće igrača iz ove baze podataka s tim uvjetom ispunjenim. Za takve igrače prosjek logaritmirane varijable Salary iznosi 5.107, stoga je naša predikcija $e^{5.107}$ tisuća dolara, točnije \$165174. Igrači koji nisu zadovoljili taj uvijet (imaju više od barem 4.5 godina igranja u većoj ligi) idu u desnu granu, te se nakon tog dijele prema varijabli Hits. Na kraju svega, stablo dijeli prostor prediktora u tri regije: igrači koji su igrali četiri ili manje godina, igrači koji su igrali pet ili više godina i koji su napravili manje od 117.5 pogodaka. Te tri regije možemo pisati kao $R_1 = \{X|Years < 4.5\}$, $R_2 = \{X|Years \geq 4.5, Hits < 117.5\}$ i $R_3 = \{X|Years \geq 4.5, Hits \geq 117.5\}$. Slika 3 ilustrira regije kao funkcije varijabli Hits i Years. Predviđene plaće tih tri grupe su \$165174, \$402834 i \$845346, redom.

U analogiji stabla, regije R_1, R_2 i R_3 zovu se terminalni čvorovi ili listovi stabla. Kao što je slučaj na slici 3, stabla se često crtaju okrenuta, u smislu da su listovi na dnu stabla. Čvorovi koji sadrže uvijet koji dijeli prediktorski prostor zovu se unutarnji čvorovi. Na slici 3, dva unutarnja čvora nalaze se ispod teksta $\{Years < 4.5\}$ i $\{Hits < 117.5\}$. Djelovi stabla koji povezuju čvorove zovu se grane.

Interpretiramo dano stablo na sljedeći način: Years je najbitniji prediktor pri određivanju kategorije plaće, igrači s manje iskustva zarađuju manje nego oni iskusniji. Kod igrača koji su

pet ili više godina igrali u većoj ligi, broj pogodaka koji su imali u prošloj godini utječe na plaću - ista raste s većim brojem pogodaka.



Slika 3: Regresijsko stablo baze podataka Hitters.

2.1 Predviđanje prediktorskog prostora stratifikacijom

Algoritam građenja stabla sastoji se od dva koraka:

1. Podijelimo prostor prediktora, tj. skup svih mogućih vrijednosti za $X_1, X_2, ..., X_p$, na J disjunktnih i nepreklapajućih područja, $R_1, R_2, ..., R_J$.

2. Za svaki podatak koji upadne u regiju R_j, napravimo istu predikciju (isto predviđanje), što je prosjek vrijednosti responsea Y od varijabli koje pripadaju R_j u trening podacima.

Npr. da smo u prvom koraku dobili dva područja R_1 i R_2 te da je odgovarajući prosjek na trening podacima za prvu regiju 10, dok je za drugu 20, tada za dani $X = x$, dobivamo predikciju 10 ako je $x \in R_1$, odnosno 20, ako je $x \in R_2$.

U slučaju $R_1, R_2, ..., R_J$, prediktorski prostor dijelimo na višedimenzionalne pravokutnike, radi jednostavnosti i lakše interpretacije predikcijskog modela. Želimo pronaći pravokutnike $R_1, R_2, ..., R_J$ tako da minimiziramo sumu kvadrata reziduala (RSS), danu formulom:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y}_{R_j})^2, \tag{2.5}$$

gdje je $\widehat{y}_{R_j}$ prosječna vrijednost za skup trening podataka u j-tom pravokutniku. Nažalost, nemoguće je razmotriti svaku moguću podjelu prostora na J pravokutnika, zato se koristi *top-down, greedy* pristup koji zovemo rekurzivno binarno dijeljenje. Pristup je top-down jer počinje na vrhu stabla (u tom trenutku svi podaci pripadaju jednoj regiji), te se nakon tog dijeli prostor

prediktora - za svaku podjelu stablu niču dvije nove grane "prema dolje". Također, greedy, jer nakon svakog koraka u procesu gradnje stabla, uzima se ona podjela koja je u tom trenutku najbolja, umjesto da se gleda unaprijed i odabere podjela koja bi mogla voditi boljem stablu u budućem koraku.

Kako bismo napravili rekurzivnu binarnu podjelu, prvo odabiremo prediktor X_j i točku podjele s tako da podjela prediktorskog prostora u regije $\{X|X_j < s\}$ i $\{X|X_j \geq s\}$ vode najvećem mogućem smanjenju RSS (notacija $\{X|X_j < s\}$ označava regiju prediktorskog prostora gdje X_j ima manje vrijednosti od s). Uzimamo sve prediktore $X_1, ...X_p$ i sve moguće vrijednosti točke podjele s za svaki od prediktora, te odabiremo prediktor i točku podjele da trenutno stablo ima najmanju RSS. Detaljnije, za svaki j i s biramo parove poluravnina

$$R_1(j, s) = \{X|X_j < s\},$$

$$R_2(j, s) = \{X|X_j \geq s\},$$

te tražimo vrijednosti j i s koje minimiziraju

$$\sum_{\{i:x_i \in R_1(j,s)\}} (y_i - \widehat{y}_{R_1})^2 + \sum_{\{i:x_i \in R_2(j,s)\}} (y_i - \widehat{y}_{R_2})^2, \tag{2.6}$$

gdje je $\widehat{y}_{R_1}$ prosjek trening podataka u $R_1(j, s)$, dok je $\widehat{y}_{R_2}$ prosjek trening podataka u $R_2(j, s)$. Taj proces se onda ponavlja, ponovno se traže j i s, samo što se sad umjesto podjele cijelog prostora prediktora, dijeli samo jedna od dvije prethodno definirane regije. Sada imamo ukupno tri regije, te biramo jednu (onu s najmanjim RSS) i ponavljamo proces. Taj postupak se nastavlja sve dok se ne dohvati kriterij zaustavljanja (npr. podjela se nastavlja sve dok ni jedna regija nema više od pet podataka).

Nakon što su definirane regije $R_1, ..., R_J$, predviđamo vrijednost za test podatak koristeći prosjeke od trening podataka u regiji gdje taj test podatak pripada. Primjer prediktorskog prostora od pet regija dan je na slici 2.

2.2 Tree pruning (obrezivanje)

Prethodno objašnjeni proces može rezultirati dobrim predikcijama za trening podatke, ali je sklon overfittingu (stablo koje dobijemo često bude prekompleksno). Manje stablo s manje podjela (tj. s manjim brojem regija $R_1, ..., R_J$) može voditi manjoj varijanci i boljoj interpretaciji, ali većoj pristranosti predikcija. Jedna moguća alternativa postupku opisanom gore je izgraditi stablo samo dok smanjenje RSS zbog svake podjele ne premašuje neki (visoki) prag. Ova strategija rezultirat će manjim stablima, ali je previše nepredvidiva jer se čini da naizgled bezvrijednu podjelu u ranoj fazi stabla može pratiti vrlo dobra podjela - to jest, podjela koja kasnije dovodi do velikog smanjenja RSS.

Dakle, bolja strategija je napraviti jako veliko stablo T_0 te ga obrezati da dobijemo podstablo. Kako odrediti najbolji način obrezivanja? Intuitivno, naš cilj je odabrati podstablo koje vodi najmanjoj greški na testnim podacima. Za dano podstablo, možemo procjeniti testnu grešku pomoću krosvalidacije. Procjenjivanje greške krosvalidacijom za svako moguće podstablo bi također bilo prekompleksno, jer je velik broj mogućih podstabala. Dakle, treba nam način da odaberemo manji skup podstabala za razmatranje.

Metoda *Cost complexity pruning* nam pruža način kako da to napravimo. Umjesto razamtranja svakog mogućeg podstabla, razmatramo niz stabala indeksiran s nenegativnim parametrom α. Za svaku vrijednost α postoji podstablo $T \subset T_0$ takvo da je izraz

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \widehat{y}_{R_m})^2 + \alpha |T| \tag{2.7}$$

najmanji mogući, gdje je $|T|$ je broj listova stabla T, R_m je pravokutnik koji odgovara m-tom listnom čvoru, a $\widehat{y}_{R_m}$ je predviđena vrijednost u paru s R_m, tj. prosječna vrijednost trening podataka u R_m. Parametar α kontrolira odnos između kompleksnosti podstabla. Za $\alpha = 0$ imamo $T = T_0$ pa jednadžba (2.7) pokazuje samo trening grešku. Što je parametar α veći od nula, grane su više obrezane u predvidljivijoj metodi. Vrijednost parametra α odabiremo metodom krosvalidacije. Nakon tog generiramo podstablo za odgovarajući α. Slijedi algoritam za biranje vrijednosti parametra α:

1. Koristimo rekurzivnu binarnu podjelu da dobijemo veliko stablo za trening podatke, stanemo samo kad svaki terminalni čvor ima manje od nekog minimuma broja podataka

2. Napravimo *cost complexity* obrezivanje na velikom stablu da dobijemo niz podstabala kao funkciju parametra α

3. Koristimo K-krosvalidaciju za odabir α, tj. dijelimo trening podatke u K poduzoraka. Za svaki $k = 1, \ldots, K$:

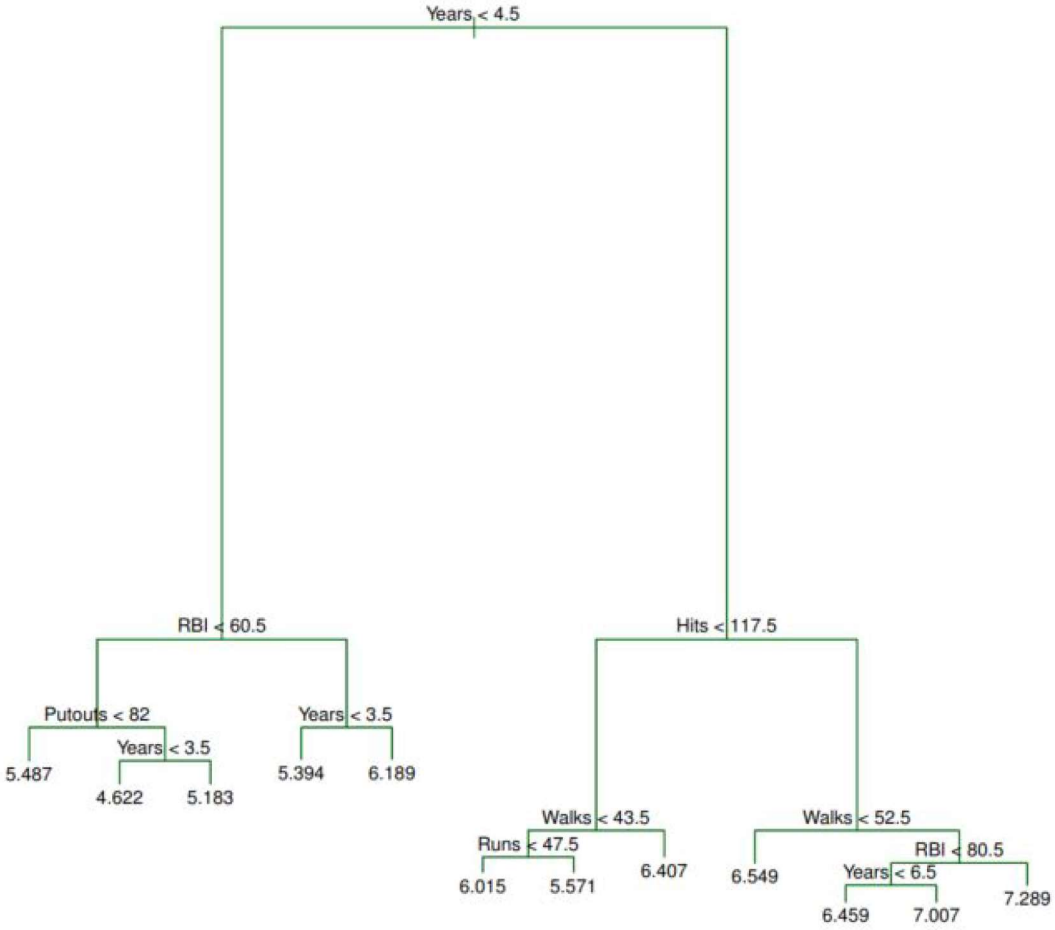
   - ponavljamo korake 1 i 2 za sve osim za k-ti poduzorak trening podataka
   - procjenimo srednje kvadratnu grešku za podatke izostavljene iz k-tog poduzorka, kao funkciju od α
   - izračunamo prosjek rezultata za svaku vrijednost parametra α i biramo α koji minimizirama srednje-kvadratnu grešku tih prosjeka.

4. Biramo podstablo iz koraka 2 koje odgovara odabranoj vrijednosti parametra α.

Primjer 2. *Na slici 4 vidimo stablo dobiveno korištenjem baze podataka hitters i devet prediktora. Prvo smo nasumično podijelili skup podataka na pola, s 132 jedinke za trening skup i 131 jedinkom za test skup. Nakon tog smo izgradili regresijsko stablo prikazano na slici 4.*

Jedna od prednosti regresijskih stabala odlučivanja jest njihova mogućnost da nađu nelinearnu vezu između prediktora i zavisne varijable koju predviđamo. To je izrazito korisno kad je odnos između varijabli kompleksan i ne može lako biti reprezentiran jednostavnim linearnim modelom.

S druge stane, regresijska stabla imaju neka ograničenja. Jedan od problema je *overfitting*, konkretno kad dopustimo stablu da bude preveliko. Overfitting se dogodi kad je model prekompleksan pa bilježi svaki šum u podacima umjesto same funkcijske veze, pa se umjesto samoj vezi između zavisne i nezavisnih varijabli počne prilagođavati i greški. Taj problem su pokušali riješiti razvojem različitih metoda kako bi obrezali stablo i spriječili overfitting. Drugo ograničenje jest što nekad znaju biti preosjetljiva na specifične podjele podataka, što može dovesti do nestabilnosti modela koji nastaje. Bez obzira na navedena ograničenja, regresijska stabla su i dalje popularna i efikasna za korištenje u raznim područjima.



Slika 4: Regresijsko stablo nastalo preko baze podataka Hitters.

3 Klasifikacijska stabla

Klasifikacijska stabla su tip stabala odlučivanja koja se koriste za predviđanje kvalitativnih varijabli. Konstrukcija je slična kao kod regresijskih, ali umjesto predviđanja neprekidne varijable, ona predviđaju kvalitativnu. Kao i kod rekurzivnih stabala, koristimo rekurzivnu binarnu podjelu da napravimo klasifikacijsko stablo. No, kod klasifikacijskih stabala ne možemo koristiti *RSS* kao kriterij podjele podataka, nego ćemo koristiti *classification error rate* (*CER*). Želimo određenom podataku u određenoj regiji pridružiti klasu koja je najzastupljenija u toj regiji. *CER* je udio trening podataka u toj regiji koji ne pripadaju najzastupnijoj klasi:

$$CER = 1 - max_k(\widehat{p}_{mk}), \tag{3.1}$$

gdje $\widehat{p}_{mk}$ predstavlja proporciju trening podataka iz k-te klase u m-toj regiji. No mjera *CER* nije naročito osjetljiva na rast stabla, pa se u praksi koriste dvije druge mjere: Gini indeks i entropija.

Gini indeks je definiran kao

$$G = \sum_{k=1}^{K} \widehat{p}_{mk}(1 - \widehat{p}_{mk}) \tag{3.2}$$

i predstavlja mjeru ukupne varijance u svih K klasa, gdje je K broj klasa na koji smo podjelili prostor prediktora. Gini indeks poprima malu vrijednost ako su svi $\widehat{p}_{mk}$ blizu nule ili jedinice. Zbog tog se Gini indeks može interpretirati kao mjera čistoće čvora - mala vrijednost ukazuje da se u čvoru dominiraju podaci iz iste klase.

Alternativa Gini indeksu jest *entropija*, mjera definirana s

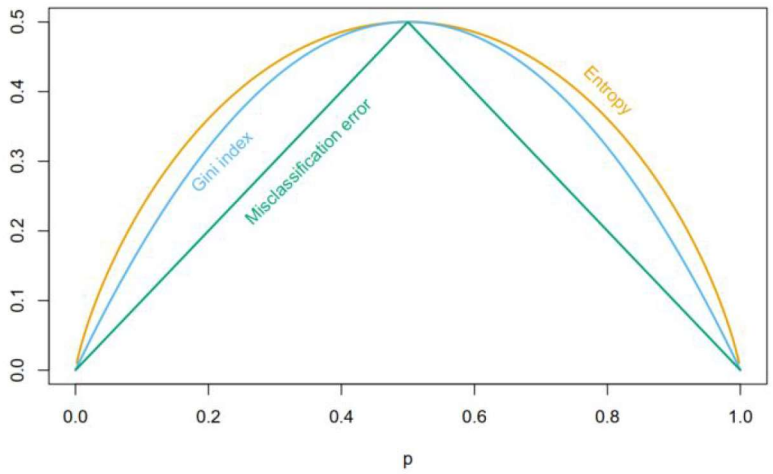
$$E = -\sum_{k=1}^{K} \widehat{p}_{mk} \log(\widehat{p}_{mk}). \tag{3.3}$$

Kako je $0 \leq \widehat{p}_{mk} \leq 1$, slijedi da je $0 \leq -\widehat{p}_{mk} \log(\widehat{p}_{mk})$. Kao i kod Gini indeksa, entropija će poprimiti malu vrijednost ako je m-ti čvor čist.

Dok gradimo klasifikacijskog stablo, Gini index ili entropija su mjere koje koristimo da ocijenimo kvalitetu određene podijele, budući da su te dvije mjere puno osjetljivije na čistoću čvora od *CER* mjere. Bilo koju od te tri mjere možemo koristiti kad obrezujemo stablo, ali *CER* je najbolja ako nam je cilj točnost predikcije završnog obrezanog stabla. Na slici 5 vidimo razliku te tri mjere.

Algoritam za klasifikacijsko stablo počinje odabirom prediktora koji je najkorisniji za predviđanje ciljane varijable. To se radi računajući mjeru nečistoće, preko Gini indeksa ili entropije, za svaki prediktor. Varijabla s najmanjom mjerom nečistoće (*node impurity*) se bira za prvu podjelu. Podaci se tada dijele na dva podskupa, s obzirom na vrijednosti te varijable. Taj proces se ponavlja za svaki podskup, gdje algoritam traži drugu najbolju varijablu za sljedeću podjelu. Postupak prestaje kad je zadovoljen kriterij zaustavljanja, npr. kad imamo minimalni broj slučajeva po čvoru ili kad sljedeće podjele ne poboljšavaju aproksimaciju ciljane varijable.

Rezultirajuće klasifikacijsko stablo se koristi za predikciju zavisne varijable za nove podatke na temelju njihovih prediktora. Stablo pruža vizualnu reprezentaciju u postupku odlučivanja, sa svakim unutarnjim čvorom koji predstavlja odluku temeljenu na vrijednosti prediktora, dok svaki list predstavlja predivđenu kategoriju za zavisnu varijablu. Klasifikacijska stabla imaju široku uporabu u više područja, kao što su marketing, medicina, financije i još puno drugih.



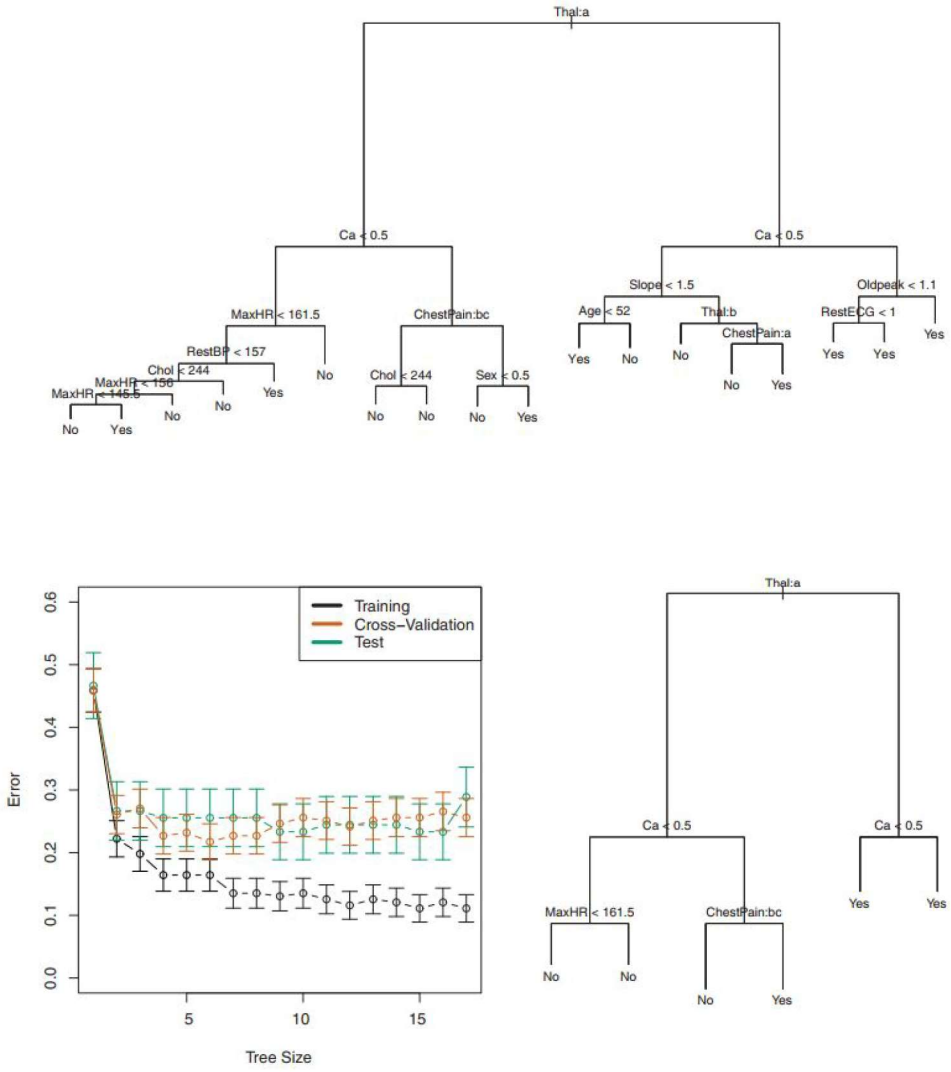
Slika 5: Mjere čistoće čvora za binomnu klasifikaciju, kao funkcije parametra proporcije

Primjer 3. *Slika 6 pokazuje primjer na bazi podataka Heart. Baza se sastoji od 303 jedinke (pacijenata) s boli u prsima za koje predviđamo imaju li srčanih bolesti ili ne (varijabla HD). U bazi imamo 13 prediktora uključujući prediktore Age, Sex, Chol (mjera kolesterola u krvi) te ostalih varijabli koje mjere rad srca i pluća. Krosvalidacija daje stablo sa šest listova.*

Do sad smo pretpostavili da su prediktori neprekidne varijable, no stabla možemo konstrurati i s prediktorima koji su kvalitativne varijable. U bazi podataka HEART, neki prediktori (Sex, Thal i ChestPain) su kvalitativne.

Podjela na osnovu nekog kvalitativnog prediktora vrši se dodjeljivanjem nekih kvalitativnih vrijednosti jednoj grani te dodjeljivanjem ostalih drugoj. Na slici 6 neki unutarnji čvorovi odgovaraju podjeli kvalitativnih varijabli; korijen velikog stabla odgovara podjeli jedinki na osnovu kvalitativne varijable Thal. Tekst Thal:a inicira da se lijeva grana koja izlazi iz tog čvora sastoji od jedinki koje imaju vrijednost prediktora Thal normalnu - unutar nekih referentnih vrijednosti (a je oznaka za normalnost u ovoj bazi podataka), dok desna predstavlja se ostale jedinke, koje imaju različitu vrijednost tog prediktora od normalne.

Slika 6 ima interesantnu karakteristiku: neke od podjela rezultiraju jednakim rezultatima, kao što je uvjet RestECG < 1 koji se nalazi dolje desno na obrezanom stablu. Bez obzira na vrijednost RestECG prediktora, rezultat je "Yes" kod svih tih jedinki. *Zašto se onda ta podjela uopće vrši*? Vrši se jer vodi većoj čistoći čvora: svih preostalih 9 jedinki koje odgovaraju desnom listu imaju vrijednost zavisne varijable "Yes", dok 7/11 jedinki koje su u lijevom listu imaju vrijednost zavisne varijable "Yes". Čistoća čvora bitan je faktor kad se bavimo predviđanjem. Ukoliko imamo novi podatak, koji provedemo kroz dano obrezano stablo, te isti završi u regiji danoj desnim listom, možemo biti vrlo sigurni da je vrijednost zavisne varijable novog podatka također "Yes". Ako je slučaj da isti podatak završi u lijevom listu, onda je njegova predviđena vrijednost opet Yes, ali smo u to manje sigurni. Iako podjela po uvjetu RestECG < 1 ne smanjuje klasifikacijsku grešku, ista poboljšava Gini indeks i entropiju, koje su osjetljive na čistoću čvora.



Slika 6: Baza podatka Heart. Gore: neobrezano stablo. Dolje lijevo: krosvalidacijska greška, trening i testna greška, za različite veličine obrezanog stabla. Dolje desno: obrezano stablo koje odgovara najmanjoj krosvalidacijskoj greški.

4 Usporedba stabala i linearnih modela

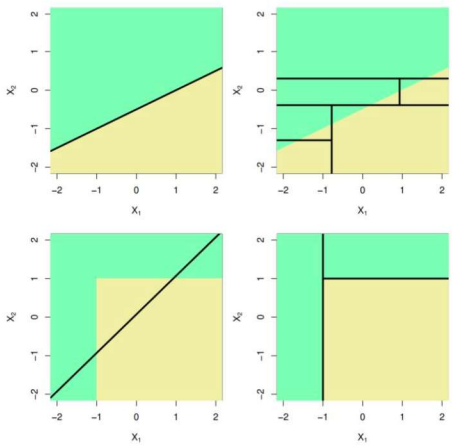
Regresijska i klasifikacijska stabla razlikuju se od linearne i logističke regresije. Linearna regresija pretpostavlja linearnu vezu između responsea i prediktora danu formulom (4.1):

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j, \tag{4.1}$$

dok regresijska stabla pretpostavljaju model oblika

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{\{X \in R_m\}}, \tag{4.2}$$

gdje $R_1, ..., R_M$ predstavljaju dijelove prediktorskog prostora kao na slici 7. Glavno pitanje jest koji model je bolji? To ovisi o situaciji u kojoj se nalazimo. Ukoliko je odnos zavisne varijable i prediktora dobro aproksimiran linearnim modelom kao u 4.1, onda će model linearne regresije najvjerojatnije raditi bolje nego model regresijskih stabala. No ukoliko je veza nelinearna i kompleksna između zavisne varijable i prediktora kao u 4.2, onda će regresijska stabla dati bolje aproksimacije nego linearna regresija. Slika 7 ilustrira primjer kad je bolje koristiti linearnu regresiju, a kad stablo.



Slika 7: Gornji red: dvodimenzionalni klasifikacijski primjer u kojem su podaci linearno odvojivi i to je prikazano obojanim područjem. Klasična linearna regresija u teoriji će dati bolje rezultate, tj. manju testnu grešku nego da se koriste metode stabala. Donji red: Ovdje podaci nisu linearno odvojivi. Ovdje linearni model ne može dobro odvojti podatke (lijevo) dok stabla to uspješno rade (desno)

5 Metode poboljšavanja stabala

Boosting je jedno od najmoćnijih oruđa, osmišljeno tek u zadnjih dvadesetakak godina. Originalno je služlo za probleme klasifikacije, no lako se može primjeniti i na probleme regresije. Počet ćemo s najpopularnijim boosting algoritmom "AdaBoost.M1". Pretpostavimo da imamo dvije klase, gdje je Y varijabla koju predviđamo, $Y \in \{-1, 1\}$, X vektor prediktora, a $G(X)$ klasifikator iz kojeg proizlaze vrijednosti $\{-1, 1\}$. Greška na trening podacima dana je izrazom

$$err = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i)). \tag{5.1}$$

Slab klasifikator je onaj kojem *err* nije značajno bolji od nasumičnog pogađanja. Svrha boostinga je primjena slabog klasifikacijskog algoritma na modificirane verzije originalnog skupa podataka, proizvodeći time niz slabih klasifikatora $G_m(x),\ m = 1, 2, ..., M$, gdje je M broj iteracija (empirijski je utvrđeno da za $M \geq 400$ dobijemo poželjne rezultate, koje vode poboljšanoj predikciji modela u odnosu na početno stablo). Predikcije svih njih se onda spajaju kroz većinski glas (*Majority vote*) kako bi se dobila konačna predikcija:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right). \tag{5.2}$$

Koeficijenti $\alpha_1, ..., \alpha_m$ se izračunaju preko boosting algorima te se izmjeri njihov doprinos klasifikatoru $G_m(x)$. Njihov smisao je dati veću važnost boljim klasifikatorima u nizu. Slika 8 pokazuje shematski prikaz AdaBoost procedure. AdaBoost je otporan na overfittanje ali izrazito osjetljiv na stršeće i vrlo varijabilne skupove podataka.

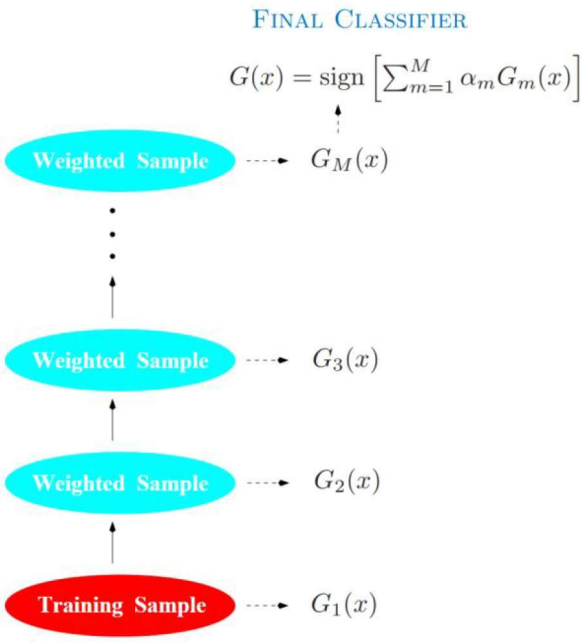
5.1 Bagging

Bagging (Bootstrap Aggregating) je tehnika strojnog učenja kojom se poboljšava preciznost i stabilnost u modelima stabala odlučivanja. Ideja je trenirati više modela stabla odlučivanja na različitim trening skupovima podataka i onda spojiti predikcije tih modela da bi dobili konačnu predikciju. Ovakav pristup je poznat kao *ensemble* metoda, kombinira predikcije više modela da se dobije točniji rezultat.

Proces bagginga uključuje slučajan odabir podskupa trening podataka (odabir s vraćanjem) i treniranjem stabla odlučivanja na tom podskupu. Taj proces se onda ponovi više puta s različitim podskupovima trening podataka, što rezultira sa skupom modela stabala odlučivanja koji su trenirani na međusobno različitima podskupovima podataka (slučajnom šumom).

Kad radimo predikciju koristeći bagged model stabla odlučivanja, predikcije svih zasebnih, novonastalih stabala odlučivanja spojimo, te onda dobijemo konačnu predikciju. Postoji više načina kako to napraviti, najčešće se uzme prosjek predikcija od svih stabala ili se koristi sustav glasanja (izabire se predikcija koja je rezultat većine stabala).

Nedostatak stabala odlučivanja jest *velika varijanca* tj., ako podijelimo trening podatke u dva slučajno izabrana djela, te fittamo stablo za oba, rezultati koje dobijemo mogu biti prilično različiti, tj. mogu jako varirati. Zato koristimo bagging, kao proceduru smanjenja varijance metode statističkog učenja.



Slika 8: Shematski prikaz AdaBoosta. Klasifikatori se treniraju po različitim verzijama skupa podataka te se na kraju spajaju da naprave finalnu predikciju

Sjetimo se da za jednostavan slučajni uzorak $Z_1, Z_2, ..., Z_n$ varijanca aritmetičke sredine $\overline{Z}_n$ iznosi σ^2/n . Drugim riječima, uprosječivanje podataka rezultirat će manjom varijancom. Najlakši način da smanjimo varijancu i povećamo preciznost metode strojnog učenja je postupak uzimanja više skupova trening podataka iz populacije, izgradnja zasebnih predikcijskih modela koristeći svaki skup i uzimanje prosjeka dobivenih rezultata.

Drugim riječima, mogli bi izračunati $\widehat{f}_1(x), \widehat{f}_2(x), ..., \widehat{f}_B(x)$ koristeći B podijeljenih podskupova treninig podataka, te ih uprosječiti ih da dobijemo model strojnog učenja male varijance:

$$\widehat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_b(x). \tag{5.3}$$

To često nije moguće, jer nemamo dovoljno trening podataka. Umjesto toga, možemo bootstrapirati, tj. uzimati na slučajan način odabrane podatke iz jednog skupa trening podataka. U tom postupku iz jednog skupa trening podataka generiramo B različitih bootstrap trening skupova podataka. Tada treniramo našu metodu na b-tom bootstrap trening skupu, $b = 1, \ldots, B$, da dobijemo $\widehat{f}^{*b}(x)$, te na kraju uprosječimo sve predikcije, da bi dobili:

$$\widehat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}^{*b}(x). \tag{5.4}$$

Taj postupak zove se bagging.

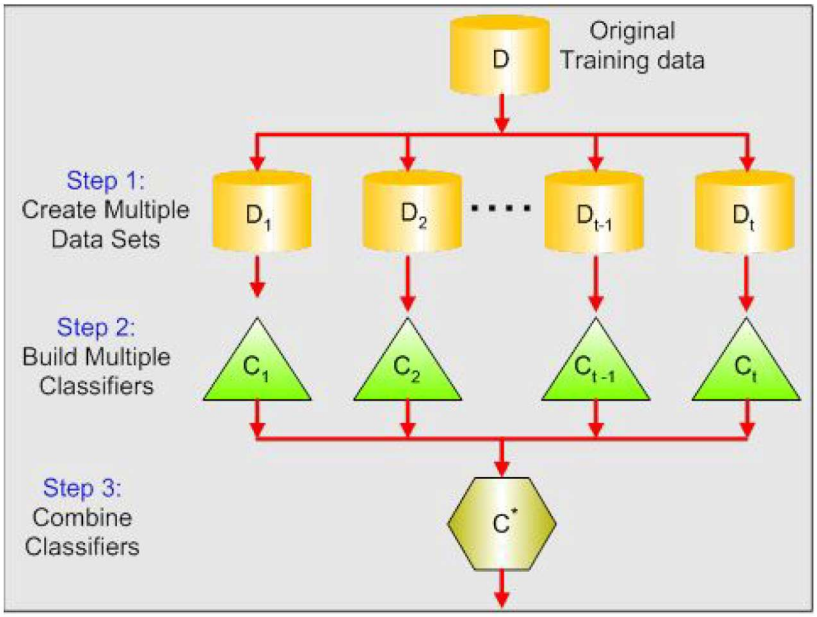
Kod primjene bagginga na regresijsko stablo, konstruiramo B regresijskih stabala, koristeći B bootstrap skupova trening podataka, te uprosječimo dobivene predikcije. Ta stabla nisu obrezana, pa svako individualno stablo ima visoku varijancu, ali nisku pristranost. Uprosječivanje

tih B stabala smanjuje varijancu predikcije. Bagging daje impresivna poboljšanja u preciznosti kad kombinirao stotine, pa čak i tisuće stabala u jednu proceduru.

Za klasifikacijska stabla, za dani test podatak, vidimo klasu koju predviđa svako od B stabala te uzmemo onu predikciju koja je rezultat većine stabala.

Određivanje testne greške u bagganom modelu je zapravo jednostavan proces. Može se pokazati da svako baggano stablo uzme oko 2/3 podataka. Preostala 1/3 podataka, koju nismo koristili u bagging modelu, spada u tzv *out-of-bag* (OOB) podatke. Možemo predvidjeti response vrijednost i-tog podatka koristeći svako stablo gdje je taj podatak bio OOB. Dobit ćemo oko $B/3$ predikcija za i-ti podatak. Uzmemo prosjek (ako govorimo o regresijskom stablu) ili većinu (klasifikacijsko stablo). To vodi jednoj OOB predikciji za i-ti podatak. Punu OOB predikciju možemo dobiti za svaki od n podataka, gdje računamo OOB MSE (za regresijsko stablo) ili CER (za klasifikacijsko stablo). Rezultirajuća vrijednost OOB greške je dobra procjena testne greške za baggani model, jer vrijednost svakog podatka je predviđena koristeći ona stabla koja nisu korištena za baš taj podatak. OOB pristup za određivanje testne greške je dobar u slučaju kad baggamo puno skupova podataka kod kojih bi krosvalidacija bila računski naporna.

Sve u svemu, bagging je moćna tehnika za poboljšanje preciznosti i stabilnosti stabala odlučivanja i ima široku uporabu u strojnom učenju. Shema bagginga dana na slici 9



Slika 9: Shema bagginga. U prvom koraku zamijenimo početne podatke novima, te na temelju njih izgradimo nova stabla u drugom koraku. Na kraju nova stabla spojimo u jedno kako bi dobili novo, unaprijeđeno stablo.

5.2 Slučajne šume

Random forests (slučajne šume) su poboljšanje "bagganih" stabala, dodavanjem novih grana tako da dekoreliramo stabla. Kao i u baggingu, izgradimo stabla odlučivanja na bootstrap trening uzorcima. No, pri gradnji stabala svaki put kad odlučujemo treba li se dogoditi neka podjela, od punog skupa p prediktora izabire se slučajni uzorak od m prediktora koji su kandidati za podjelu. Podjela se vrši tako da se bira jedan od tih m prediktora. Svaki put se bira novi uzorak od m prediktora, a najčešće uzimamo $m \approx \sqrt{p}$.

Dakle, kad gradimo slučajnu šumu, algoritam bira podjelu po najboljem od m prediktora (onom koji vodi najvećem smanjenju MSE kod numeričkih prediktora ili većoj čistoći čvora kod kvalitativnih), a to ima smisla za sljedeći slučaj: ukoliko imamo skup od p prediktora $p_1, p_2, ..., p_p$, gdje je p_j prediktor koji vodi najboljoj podjeli, u slučaju bagginga većina stabala bi uzela tog prediktora za prvu podjelu prostora prediktora. Posljedično, sva baggana stabla bi bila međusobno slična, a prediktori bagganih stabala jako korelirani. Nažalost uprosječivanje mnogo koreliranih veličina ne vodi velikom smanjenju varijance, u odnosu na slučaj kad one nisu korelirane. U ovom slučaju, bagging neće voditi značajnom smanjenju varijance u odnosu na model običnog stabla odlučivanja.

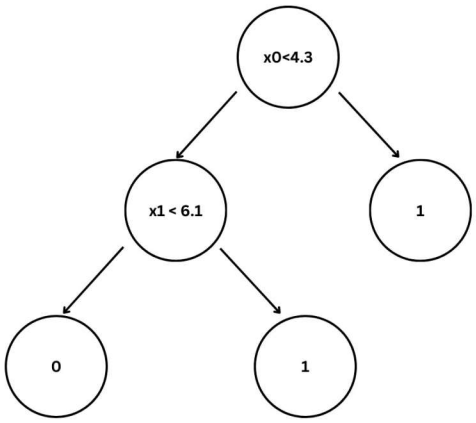
Slučajne šume taj problem rješavaju tako da svaka podjela bude sačinjena na osnovi manjeg broja prediktora. Dakle, oko $(p - m)/p$ podjela će biti sačinjeno bez uzimanja prediktora p_j kao kandidata za podjelu, te će ostali prediktori imati veću šansu biti izabrani u kriterij za podjelu. Na taj način dajemo mogućnost izgradnje stabala sa drugačijim podjelama, i ublažavamo "pohlepan" pristup koji stabla imaju ("greedy aproach" - algoritam uzima najbolju trenutnu podjelu, ne brine hoće li ona voditi sveukupnom najoptimajnijem rezultatu).

Glavna razlika između bagginga i slučajnih šuma je odabir podskupa prediktora veličine m. U slučaju da je slučajna šuma izgrađena za $m = p$, onda to vodi baggingu. Odabir malog broja m pri gradnji slučajne šume će biti korisno kad imamo velik broj koreliranih prediktora.

Ovaj cijeli postupak ilustriran je sljedećim primjerom: Pretpostavimo da imamo bazu podataka B od šest jedinki i pet prediktora $x_0, x_1, ..., x_4$ i zavisnu varijablu y, a podaci su prikazani na slici 10. Klasifikacijsko stablo koje se dobije na danim podacima je na slici 11.

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

Slika 10: Baza podataka od šest jedinki, pet prediktora i jedne zavisne varijable



Slika 11: početno stablo nastalo podacima iz baze podataka B

Pretpostavimo sad da smo nasumičnim odabirom uzeli skupove prediktora

$$\{x_0, x_1\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_1, x_3\},$$

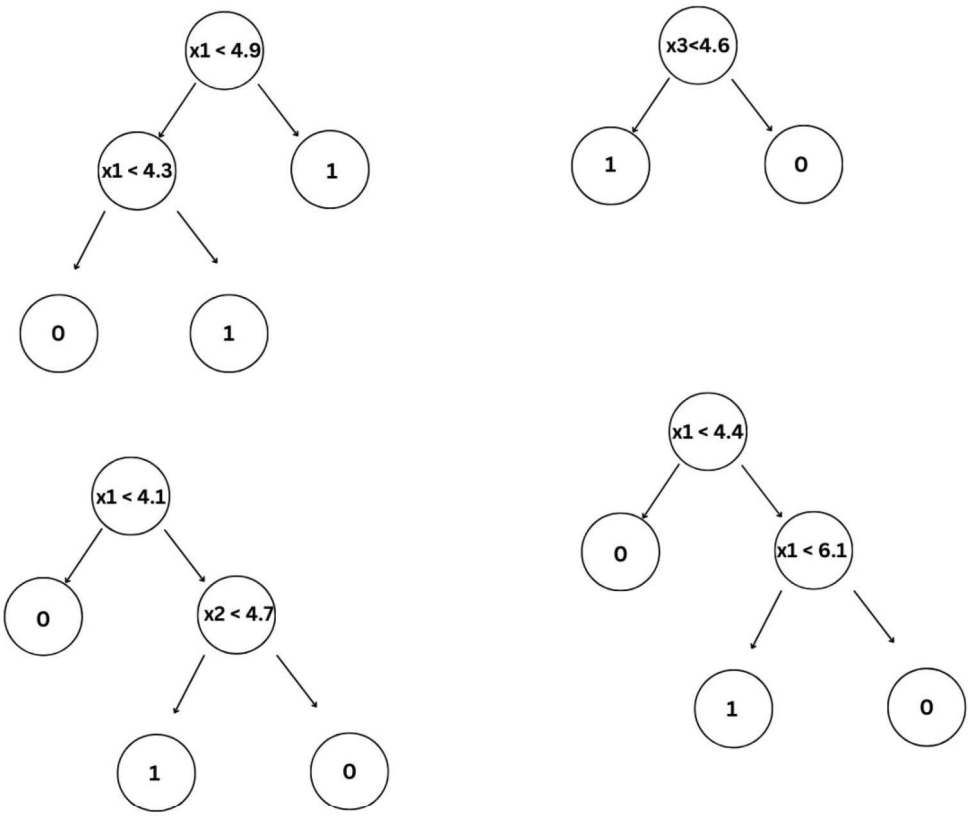
redom. Pretpostavimo da smo slučajnim odabirom s vraćanjem izvukli sljedeće jedinke, kako je prikazano na slici 12 (ovdje zbog jednostavnosti imamo četiri nova skupa podataka no kad se ovaj postupak radi, uzima se minimalno oko 100 novih skupova).

Slika 12: Odabir jedinki s vraćanjem s ciljem izgradnje slučajnih šuma. Prvo stablo bit će izgrađeno u odnosu na prediktore $\{x_0, x_1\}$, sljedeće u odnosu na $\{x_2, x_3\}$, te ostala kako smo naveli gore.

Nova stabla, tj. slučajna šuma nastala ovim postupkom je prikazana na slici 13. Označimo ta stabla s_1, s_2, s_3, s_4 redom. Neka je z novi podatak, $z = (2.8,\ 6.2,\ 4.3,\ 5.3,\ 5.5)$. Na osnovu šume prikazane na slici 13 vrijedi

$$s_1(z) = 1 \quad s_2(z) = 0 \quad s_3(z) = 1 \quad s_4(z) = 1.$$

Uzimamo većinski glas (majority vote) novonastale šume i zaključujemo da je $y_z = 1$.



Slika 13: Slučajna šuma izgrađena temeljem slučajnog odabira četiriju skupa prediktora

5.3 Boosting

Boosting je još jedna metoda poboljšavanja predikcija stabala odlučivanja. Kao kod bagginga, boosting se može primjeniti na regresijske i klasifikacijske probleme. Prisjetimo se da bagging koristi više podskupova originalnog skupa podataka koristeći bootstrap tehniku, te stvarajući zasebno stablo svakoj kopiji, te na kraju spaja sva stabla kako bi se dobilo jedno glavno stablo. Očito su stabla građena na bootstrap uzorcima međusobno nezavisna. Boosting metoda je slična ovoj, no kod boostinga gradimo stabla sekvencionalno: svako buduće stablo gradimo na temelju dobivenih prethodno izgrađenih stabala. Boosting ne koristi bootstrap uzorkovanje, već se svako stablo gradi na modificiranoj verziji originalnog skupa podataka, gdje se modifikacija vrši s obzirom na dobivene greške iz prethodnog modela. Pretpostavimo da imamo regresijski problem. Kao i bagging, boosting uključuje kombiniranje velikog broja stabala, $\hat{f}_1, ..., \hat{f}_B$. Boosting je objašnjeno algoritmom:

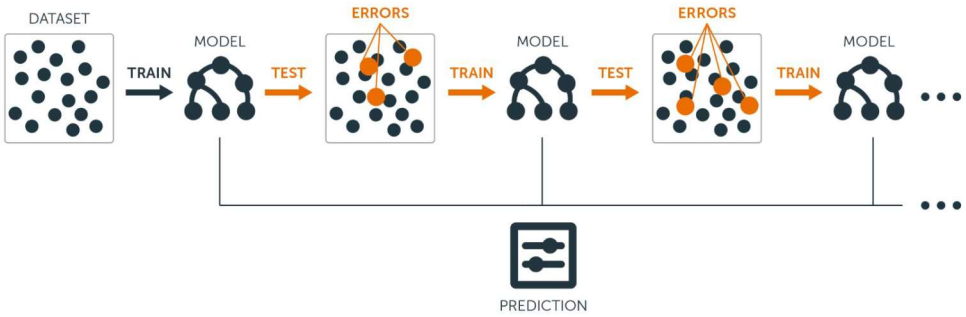
1. $\hat{f}(x) = 0$ te reziduali $r_i = y_i$ za sve i u trening skupu podataka
2. za $b = 1, 2, ..., B$ ponavljamo:
   - gradimo stablo $\hat{f}^b$ sa d podjela ($d+1$ terminalnih čvorova) za trening skup podataka (X, r).
   - Ažuriramo $\hat{f}$ dodavajući smanjenu verziju novog stabla:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   - Ažuriramo reziduale
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x)$$
3. Boostani model daje sljedeću procjenu za f:
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Skica boosting algoritma dana je na slici 14

Koja je ideja iza ovog postupka? Umjesto gradnje velikog stabla i potencijalnog overfittinga, pristup boostinga *sporo uči*. Za dani model gradimo stablo prema rezidualima modela, tj. koristimo reziduale za izgradnju umjesto zavisne varijable Y. Nakon tog dodajemo novo stablo u procijenjenu funkciju kako bismo ažurirali reziduale. Svako od novih stabala može biti proizvoljno malo s tek par terminalnih čvorova, čiji broj ovisi o parametru d u gornjem algoritmu. Postupkom dodavanja novih malih stabala utječemo na preciznost funkcije $\hat{f}$ na mjestima gdje ne aproksimira točno. Smanjivanjem parametra λ usporava proces još više, dopuštajući više stabala različitih oblika s ciljem promjene vrijednosti reziduala. Također treba primjetiti da u boostingu, za razliku od bagginga, konstrukcija svakog stabla strogo ovisi o stablima koja su izgrađena prije. Ovaj postupak je opisani proces boostanja regresijskih stabla.

Boostanje klasifikacijskih stabala je slično, no način je malo kompleksniji. Boostanje ima tri parametra na koja utječemo:

1. Broj stabala B: za razliku od bagginga i slučajnih šuma, boosting može dovesti do overfittinga ako je B prevelik. Za biranje B koristimo krosvalidaciju.



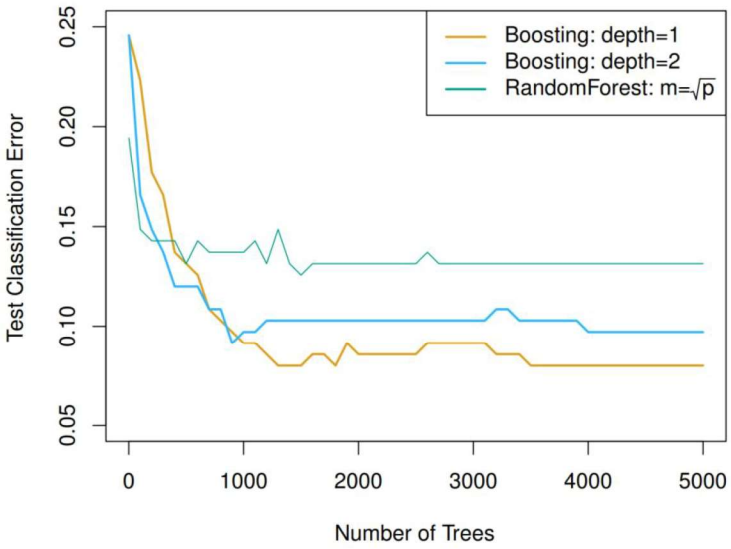
Slika 14: Skica boosting algoritma. Koristeći podijeljeni početni skup podataka napravimo stablo, te isto testiramo i dobijemo greške. Nakon tog, uz pomoć grešaka modificiramo originalni skup podataka i radimo novo stablo te postupak nastavljamo dok ne dobijemo željene rezultate (npr. točnost predikcije iznad 80%) ili dok ne dođemo do unaprijed zadanog broja iteracija.

2. Smanjenje parametra $\lambda > 0$. To kontrolira brzinu pri kojoj boosting uči, a tipične vrijednosti za λ su 0.01 ili 0.001. Izrazito male vrijednosti λ mogu uzrokovati veliki broj B kako bi se dobili dobri rezultati.

3. Broj d podjela u svakom stablu, koji utječe na kompleksnost boostane cjeline. Često $d = 1$ daje dobre rezultate i tad se stablo zove *stump* tj. panj, sastoji se od jedne podjele. U ovom slučaju boostana cjelina stvara aditivni model, a d je inače *interakcija dubine*, te kontrolira interakcijski redoslijed boostanom modelu, budući da d podjela mogu uključiti najviše d varijabli.

Na slici 15, aplicirali smo boosting na bazu podataka od 15 klasi, koja sadrži informacije o genu za rak, kako bi dobili klasifikacijsko stablo (ključ) za razlikovanje zdravog pacijenta od 14 pacijenata s dijagnozom raka.

Testnu grešku prikazujemo kao funkciju ukupnog broja stabala u ovisnosti s parametrom dubine d. Vidimo da najjednostavniji panjevi s $d = 1$ dovoljno dobro predviđaju da bi bili uključeni. Ovaj model je bolji od modela s $d = 2$, a oba su bolja od slučajnih šuma. Ovdje vidimo jednu razliku između boostinga i slučajnih šuma: kod boostinga, zbog toga što je gradnja određenog stabla pod utjecajem stabala koja su izgrađena do tad, manja stabla su često dovoljna. Korištenje manjih stabala može pomoći u interpretabilnosti, npr. korištenje panjeva vodi aditivnom modelu.



Slika 15: Rezultati apliciranja boostinga i slučajnih šuma na bazu podataka od 15 klasi, koja sadrži informacije o genu za rak. Testna greška je prikazana kao funkcija broja stabala. Za dva boostana modela, $\lambda = 0.01$. Stabla dubine 1 su bolja od stabla dubine 2, a oba su bolja od slučajne šume. U sva tri slučaja standardne greške iznose oko 0.02, pa te razlike nisu značajne.

6 Primijena stabala odlučivanja u problemu klasifikacije

U ovom poglavlju gradimo stablo odlučivanja za bazu podataka *retail* iz kolegija Upravljanje kreditnim rizicima. Cilj je klasificirati retail klijente na "dobre" i "loše", odnosno one za koje se procjenjuje da će uredno vraćati kredite i koji neće uredno vraćati kredite. Baza sadrži podatke o 374 klijenta. Za svakog klijenta znamo njegovu dob, radni staž, bračno stanje, cijenu robe koju kupuje, gotovinu koju klijent daje pri kupovini robe, stambeno stanje, ratu kredita, mjesečnu plaća klijenta, zanimanje, način plaćanja, robu koju kupuje, prethodno iskustvo u poslovanju s bankom te je li dobar ili loš, u smislu vraćanja kredita. Klijent koji kasni s vraćanjem barem jedne rate kredita više od 60 dana odlazi u stanje defaulta – stanje neispunjavanja obveza, tj. postaje loš u kreditnom smislu. U tablici 1 je kratak opis svake od varijabli koje koristimo.

Varijabla	**Opis varijable**	**Tip varijable**
Dob	Dob potrošača tražitelja kredita	Numerička diskretna
Staz	Radni staž potrošača u mjesecima	Kvalitativna ordinalna
Bracno	Bračno stanje Bračno stanje (1-neudata/neoženjen, živi u zajednici; 2-udana/oženjen; 3-udovica/udovac; 4-razvedna/razveden)	Kvalitativna nominalna
Cijena	Cijena robe koju klijent kupuje	Numerička neprekidna
Gotovina	Gotovina koju klijent daje pri kupovini robe	Numerička neprekidna
Stan	Stambeno stanje (1-unajmljen; 2-stanarsko pravo; 3-društveni stan; 4-vlastita kuća/stan; 5-s roditeljima; 6-ostalo)	Kvalitativna nominalna
Rata	Rata kredita	Numerička neprekidna
Plaća_bez0	Mjesečna plaća potrošača	Numerička neprekidna
Iskus1	Prethodno iskustvo (1-novi klijent, 2-stari klijent)	Kvalitativna ordinalna
Kval	Kvaliteta dućana (1-najboljih 10%; 2-iznad prosjeka; 3-ispod prosjeka; 4-najlošijih 10%; 5-novi dućani s manje od 6 mjeseci poslovanja; 6-mali dućani koji sklapaju do 6 ugovora mjesečno)	Kvalitativna ordinalna
Nacin	Način plaćanja (1-uplatnice; 2-trajni nalog)	Kvalitativna nominalna
Zan	Zanimanje potrošača (1-poduzetnik; 2-umirovljenik; 3-zaposlenik)	Kvalitativna nominalna

Varijabla	Opis varijable	Tip varijable
Roba 1	Roba koju potrošač kupuje (1-bijela tehnika; 2-crna tehnika, foto, kino, glazbala; 3-kućne potrebe, kućni tekstil; 4-mobiteli, telefonska tehnika; 5-namještaj, kuhinje, sanitarije; 6-računala, uredska tehnika; 7-ostalo)	Kvalitativna nominalna
LOSI	1-loši; 0-dobri	Kvalitativna nominalna

Tablica 1: Opis i tip varijabli iz baze podataka *retail*

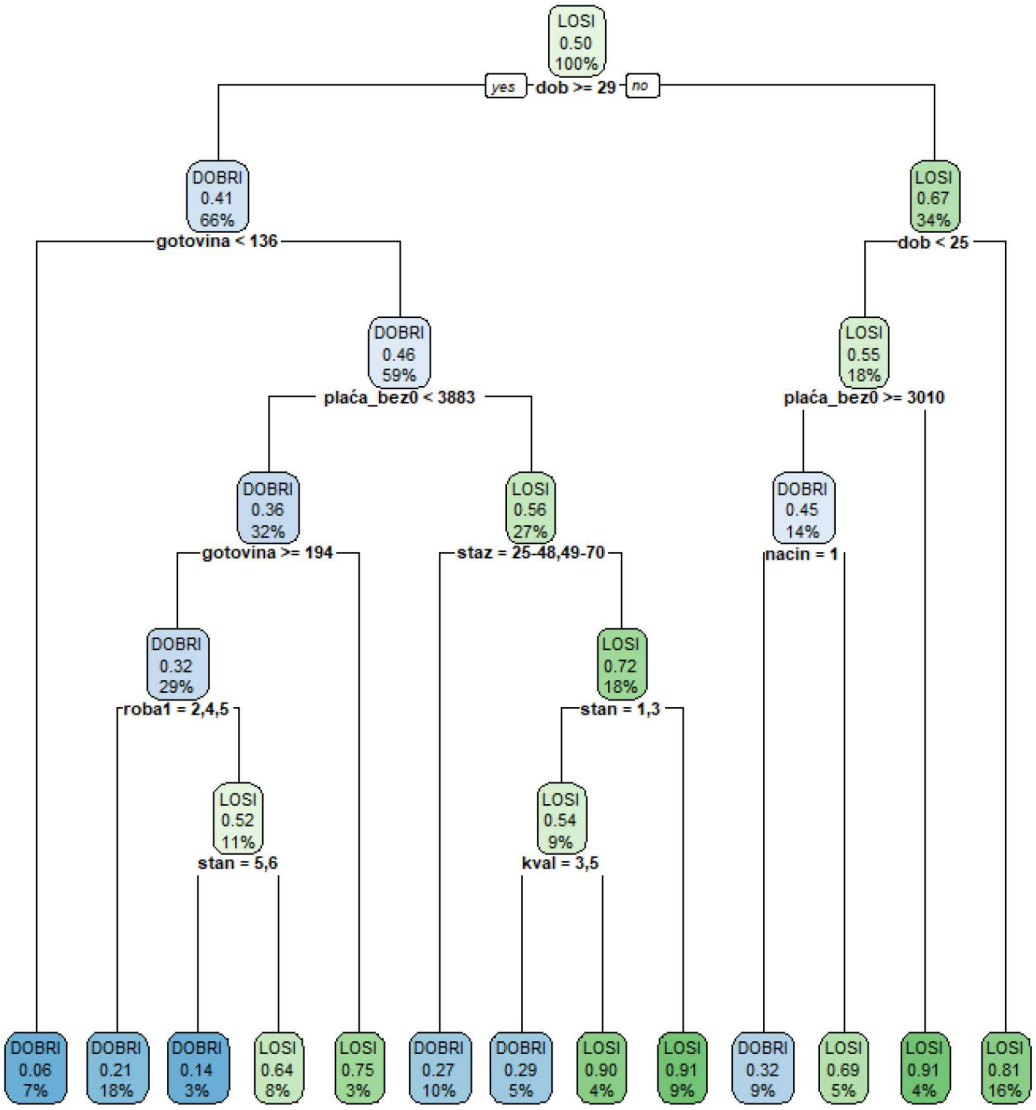
Koristeći dane podatke, želimo izgraditi klasifikacijsko stablo koje dijeli dobre od loših klijenata. Prvo nasumično biramo podatke te ih dijelimo u dva skupa, trening skup (80% podataka) i test skup (20% podataka). Klasifikacijsko stablo napravljeno od trening podataka dano je na slici 16.

Na stablu na slici 16 vidimo da su klijenti, koji su dobi 29 godina ili stariji i koji daju manje od 136 gotovine pri kupovini robe, klasificirani kao dobri. Također vidimo da čim je klijent mlađi od 29 godina i stariji od 25 biva klasificiran kao loš. Za dobivanje više informacija o izvedbi modela koristimo konfuzijsku matricu, čiji osnovni zapis je dan tablicom 2. Ona je dimenzije 2×2. U redovima su informacije o stvarno dobrim i lošim klijentima, dok su u stupcima informacije o dobrim i lošim klijentima na temelju klasifikacije dobivene modelom. Četiri elementa matrice predstavljaju četiri metrike koje broje točna i pogrešna predviđanja modela. Svaki element ima oznaku od dvije riječi: istinito ili lažno, pozitivan ili negativan. Cilj je cilj dobiti što veći hit rate, tj. dobiti što veće frekvencije na glavnoj dijagonali konfuzijske matrice.

Kod konfuzijske matrice definiramo:

- total hit rate - omjer ukupnog broja klijenata ispravno klasificiranih (prema modelu) i ukupnog broja klijenata
- good hit rate - omjer dobrih klijenata ispravno klasificiranih (prema modelu) i stvarno dobrih klijenata
- bad hit rate - omjer loših klijenata ispravno klasificiranih (prema modelu) i stvarno loših klijenata
- greška tipa I (α greška) - odobravanje kredita lošem klijentu, čitamo iz tablice: $\alpha = \frac{FP}{TN+FP}$
- greška tipa II (β greška) - neodobravanje kredita dobrom klijentu, čitamo iz tablice: $\beta = \frac{FN}{FN+TP}$.

Konfuzijska matrica za stablo na slici 16 dana je tablicom 3. Nažalost, preciznost stabla (total hit rate) je samo 51%, što znači da model dobro odvaja dobre od loših klijenata u 51% slučajeva. Greška tipa I iznosi 53%, dok tipa II 46%. Good hit rate iznosi 55%, a bad hit rate 47%.



Slika 16: Klasifikacijsko stablo izgrađeno na temelju podataka iz baze *retail*

Tablica 2: Konfuzijska matrica

Istinita klasa	Predviđena klasa	
	Loš	Dobar
Loš	TN	FP
Dobar	FN	TP

Tablica 3: Konfuzijska matrica za stablo odlučivanja nastalo iz baze podataka *retail*

Stvarni	Procijenjeni	
	Procijenjeni loši	Procijenjeni dobri
Stvarno loši	16	18
Stvarno dobri	15	18

Nasuprot tome, koristeći logističku regresiju za model oblika

$$LOSI \sim bracno + dob + \frac{gotovina}{cijena} + staz + roba1 + gotovina + stan, \tag{6.1}$$

gdje varijablu *LOSI* modeliramo preko varijabli u gornjoj jednadžbi (6.1), dobivamo preciznost od 64%. Greška tipa I iznosi 35%, a tipa II 31%. Good hit rate iznosi 65%, a bad hit rate 69%. Konfuzijska matrica dana je tablicom 4

Tablica 4: Konfuzijska matrica dobivena koristeći model logističke regresije.

Stvarni	Procijenjeni	
	Procijenjeni loši	Procijenjeni dobri
Stvarno loši	27	12
Stvarno dobri	12	22

Pomoću slučajnih šuma i boosting procedure, pokušajmo poboljšati početno stablo kako bismo dobili bolje rezultate. Za slučajne šume uzgojili smo 800 podstabala te povećalli preciznost na 57%, a konfuzijska matrica dana je tablicom 6. Greška tipa I iznosi 40%, a tipa II 47%. Good hit rate iznosi 53%, a bad hit rate 60%.

Tablica 5: Konfuzijska matrica dobivena koristeći slučajne šume.

Stvarni	Procijenjeni	
	Procijenjeni loši	Procijenjeni dobri
Stvarno loši	21	14
Stvarno dobri	15	17

Koristeći boosting s 500 iteracija i dobili smo preciznost od 61%, gdje greška tipa I iznosi 34%, a tipa II 44%. Good hit rate iznosi 56%, a bad hit rate 66%. Konfuzijska matrica za boosting dana je tablicom 6. Zaključujemo da obje metode vode uspješnijoj klasifikaciji, no model logističke regresije dan s jednadžbom (6.1) s preciznošću od 65% se pokazao kao najbolji.

Tablica 6: Konfuzijska matrica dobivena koristeći boosting model.

Stvarni	Procijenjeni	
	Procijenjeni loši	Procijenjeni dobri
Stvarno loši	23	12
Stvarno dobri	14	18

Literatura

[1] J. FRIEDMAN, T. HASTIE, R. TIBSHIRANI, *The Elements of Statistical Learning, Data Mining, Inference, and Prediction, 2nd ed.*, Springer Inc., Stanford, California, 2001.

[2] G. JAMES, T. HASTIE, R. TIBSHIRANI, D. WITTEN, *An Introduction to Statistical Learning with Applications in R, 2nd ed.*, Springer Science+Business Media, LLC, part of Springer Nature, 2021

[3] Datacamp stranica, web izvor dostupan na `https://www.datacamp.com/tutorial/decision-trees-R`

[4] Stranica kolegija Upravljanje kreditnim rizicima, web izvor dostupan na `https://www.mathos.unios.hr/images/homepages/nsarlija/UKR/3_upotreba_i_primjena_kredit_scoring_modela_validacija.pdf`

Sažetak

Stabla odlučivanja su jako oruđe u strojnom učenju i analizi podataka. Mogu se koristit i za klasifikaciju i regresiju i korisna su kad se bavimo s kompleknijim skupovima podataka. Prednosti stabala su jako lagano objašnjiva. Moguće ih je prikazati grafički i lagana su za interpretaciju. Lako mogu raditi s kvalitativnim prediktorima bez potrebe kreiranja dummy varijable. Nažalost, nemaju isti nivo prediktivne točnosti kao neki drugi klasifikacijski/regresijski modeli. Dodatno, nisu robustna - mala promjena u podacima može voditi generiranju potpuno drugačijeg stabla. Metodom bagginga možemo poboljšati preciznost i stabilnost u modelima stabla odlučivanja. Kod bagginga, stabla nezavisno rastu na slučajnom uzorku promatranih podataka, te imaju tendenciju međusobno sličiti. Za slučajne šume, stabla rastu kao i kod bagginga, no svaka podjela podataka se radi koristeći slučajno izabran podskup prediktora, te tako dekoreliramo stabla, te dolazimo do temeljitije podjele prediktorskog prostora u odnosu na bagging. Boosting je još jedna metoda kojoj možemo poboljšati stabla, boosting kombinira mnoge slabe modele u jedan bolji tako da se fokusira na ispravljanje grešaka prethodnih modela - poput tima stručnjaka koji uče iz svojih grešaka kako bi donijeli bolje odluke.

Ključne riječi

Regresijska stabla, klasifikacijska stabla, bagging, slučajne šume, boosting

Decision trees and ensemble methods

Summary

Decision trees are a powerful tool in machine learning and data analysis. They can be used for both classification and regression and are useful when dealing with more complex data sets. The benefits of trees are very easily explained. They can be displayed graphically and are easy to interpret. They can easily work with qualitative predictors without the need to create a dummy variable. Unfortunately, they do not have the same level of predictive accuracy as some other classification/regression models. Additionally, they aren't very robust - a small change in the data can lead to the generation of a completely different tree. Using the bagging method, we can improve precision and stability in stable decision-making models. In bagging, trees grow independently on a random sample of observed data, and tend to resemble each other. For random forests, the trees grow as in bagging, but each division of the data is done using a randomly selected subset of predictors, and thus we decorrelate the trees, and arrive at a more thorough division of the predictor space compared to bagging. Boosting is another method through which we can improve trees. Boosting combines many weak models into a strong one by focusing on fixing the errors of the previous models - like a team of experts who learn from their mistakes to make better decisions.

Keywords

Regression trees, classification trees, bagging, random forests, boosting

Životopis

Rođen sam 22. kolovoza 1997. godine u Zagrebu. Pohađao sam osnovnu školu Izidora Kršnjavog u Zagrebu. Po završetku osnovne škole 2012. godine, upisao sam Klasičnu gimnaziju u Zagrebu koju završavam 2016. Te godine upisujem preddiplomski studij matematika (nastavnički) u Zagrebu na Prirodoslovno-matematičkom fakultetu. Sveučilišni prvostupnik edukacije matematike postajem 2020. godine. Nakon toga, 2021. upisujem diplomski studij matematike, smjer Financijska matematika i statistika, u Osijeku, na Fakultetu primijenjene matematike i informatike. Tijekom diplomskog studija odradio sam stručnu praksu u firmi Naturalis d.o.o. u Civljanima.