Stabla odlučivanja i primjene u strojnom učenju

Vondraček, Vladimir

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Vladimir Vondraček

STABLA ODLUČIVANJA I PRIMJENE U STROJNOM UČENJU

Diplomski rad

Voditelj rada:
doc. dr. sc. Hrvoje Planinić

Zagreb, 2023

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

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

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem mentoru doc. dr. sc. Hrvoju Planiniću na uloženom vremenu i pomoći pri izradi ovog diplomskog rada. Veliko hvala mojim roditeljima, sestri i Dori koji su uvijek bili tu za mene. Posebno hvala Sonji i Zoranu na brojnim korisnim savjetima tijekom studiranja.

Sadržaj

Uvod

Strojno učenje, dinamično područje koje objedinjuje računalnu znanost i statistiku, opskrbljuje računala sa sposobnosti učenja i donošenja odluka bez eksplicitnog programiranja. Ovo područje tehnologije omogućuje računalima da izvlače znanje iz podataka i primjenjuju ga na razne zadatke, čime postaju sve bolji u predviđanju i donošenju odluka. Jedan od ključnih alata u strojnom učenju koji nam pomaže u ovom procesu su stabla odlučivanja. Stabla odlučivanja su strukturirani modeli koji oponašaju ljudsko odlučivanje tako da složene odluke razbijaju na manje i razumljive korake. Svaka grana stabla predstavlja moguću odluku ili ishod, dok čvorovi označavaju kriterije za donošenje tih odluka na temelju odabranih značajki.

U ovom radu, posebno ćemo se usredotočiti na metodu poznatu kao CART (Classification and Regression Trees) za izradu stabala odlučivanja. Kroz dublju analizu, razumjet ćemo principe konstrukcije stabala odlučivanja koristeći CART metodu i kako ih prilagoditi kako bismo postigli najbolje performanse.

Ovo univerzalno primjenjivo oruđe, nalazi svoje primjene u različitim područjima, od medicinske dijagnostike i financijske analize do sustava za preporuke i klasifikacije e-pošte kao spam ili ne. Jedna od posebnih primjena stabala odlučivanja koju ćemo istražiti je u svijetu igre pokera.

U pokeru, stabla odlučivanja pomažu igračima analizirati trenutačno stanje igre, procijeniti vjerojatne buduće poteze i strategije te donijeti najbolju odluku temeljenu na dostupnim podacima. Ovo je samo jedan primjer kako strojno učenje i stabla odlučivanja pronalaze primjenu u izazovnim i dinamičkim okruženjima.

Poglavlje 1

Pozadina

1.1 Uvod

U ovom poglavlju opisujemo nadzirano učenje i definiramo temeljne pojmove korištene u ostatku rada. Prvo ćemo opisati zadatke učenja klasifikacije i regresije te formalno definiramo pojam modela. Zatim nastavljamo s raspravom o procjeni učinka te za kraj opisujemo postupak za odabir najboljeg mogućeg modela.

1.2 Učenje iz podataka

Cilj kojemu težimo je pronaći sustavan način predviđanja fenomena s obzirom na skup mjerenja. U terminima strojnog učenja, ovaj cilj je formuliran kao zadatak nadziranog učenja, koji se sastoji od zaključivanja iz prikupljenih podataka o modelu. Taj model predviđa vrijednost izlazne varijable temeljem promatranih vrijednosti ulaznih varijabli. Pronalaženje odgovarajućeg modela temelji se na pretpostavci da izlazna varijabla nije nasumična i da postoji odnos između ulaza i izlaza.
Kako bismo preciznije formulirali problem, neka je $(\Omega, \mathcal{F}, \mathbb{P})$ vjerojatnosti prostor te (x_i, y_i), $i = 1, ..., p$, nezavisne, jednako distribuirane realizacije uzorka iz distribucije $\mathbb{P}(X, Y)$ gdje $x_j \in \mathcal{X}_j$ (za $j = 1, ..., p$) odgovara vrijednosti ulazne varijable X_j. Ulazne vrijednosti $(x_1, x_2, ..., x_p)$ zajedno čine p-dimenzionalni ulazni vektor $\mathbf{x}$, koji poprima vrijednosti u $\mathcal{X}_1 \times \mathcal{X}_2 \times ... \times \mathcal{X}_p = \mathcal{X}$, gdje smo s $\mathcal{X}$ označili ulazni prostor. Slično, označimo s $y \in \mathcal{Y}$ vrijednost izlazne varijable Y, gdje je $\mathcal{Y}$ izlazni prostor.
Među varijablama koje promatramo, razlikujemo dva opća tipa. Prvi tip odgovara kvantitativnim varijablama, čije vrijednosti su cijeli ili realni brojevi. Drugi tip su kvalitativne varijable, čije vrijednosti su simbolične, poput spola ili bračnog statusa. Formalno ih definiramo na sljedeći način:

Definicija 1.2.1. *Varijabla X_j je uređena ako je $\mathcal{X}_j$ potpuno uređen skup. Posebno, kažemo da je X_j numerička ako je $\mathcal{X}_j = \mathbb{R}$.*

Definicija 1.2.2. *Varijabla X_j je kategorička ako je $\mathcal{X}_j$ konačan skup vrijednosti, bez prirodnog poretka.*

U tipičnom zadatku nadziranog učenja, prethodna opažanja sažeta su u skupu podataka poznatom kao skup za učenje. Taj skup sastoji se od promatranih ulaznih vektora zajedno s njihovom odgovarajućom izlaznom vrijednosti i formalno je definiran na sljedeći način:

Definicija 1.2.3. *Skup za učenje $\mathcal{L}$ je skup N parova ulaznih vektora i izlaznih vrijednosti $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$, gdje su $\mathbf{x}_i \in \mathcal{X}$ i $y_i \in \mathcal{Y}$.*

Koristeći gornju notaciju, problem nadziranog učenja može se shvatiti kao učenje funkcije $\varphi : \mathcal{X} \to \mathcal{Y}$ iz skupa učenja $\mathcal{L} = (\mathbf{X}, \mathbf{y})$. Želimo pronaći model takav da su njegova predviđanja $\varphi(\mathbf{x})$, koja ćemo označavati s $\hat{Y}$, što je bolje moguća. Ako je Y kategorička varijabla, tada je zadatak učenja problem klasifikacije. Ako je Y numerička varijabla, tada je zadatak učenja problem regresije. Bez smanjenja općenitosti, rezultirajuće modele definiramo kao:

Definicija 1.2.4. *Klasifikator ili pravilo klasifikacije je funkcija $\varphi : \mathcal{X} \to \mathcal{Y}$, gdje je $\mathcal{Y}$ konačan skup klasa (ili oznaka) označenih s $\{c_1, ..., c_J\}$.*

Definicija 1.2.5. *Regresor je funkcija $\varphi : \mathcal{X} \to \mathcal{Y}$, gdje je $\mathcal{Y} = \mathbb{R}$.*

1.3 Procjena učinka

U statističkom smislu, ulazne varijable $X_1, ..., X_p$ i izlazna varijabla Y su slučajne varijable koje poprimaju vrijednosti iz skupa $\mathcal{X} \times \mathcal{Y}$ prema zajedničkoj vjerojatnosnoj distribuciji $\mathbb{P}(X, Y)$. Sukladno tome, koristeći algoritam $\mathcal{A}$ za model učenja $\varphi_{\mathcal{L}}$ čija su predviđanja najbolja moguća možemo shvatiti kao traženje modela koji minimizira očekivanu pogrešku predviđanja, definiranu na sljedeći način:

Definicija 1.3.1. *Očekivana pogreška predviđanja također poznata kao pogreška generalizacije ili pogreška testiranja modela $\varphi_{\mathcal{L}}$ je*

$$\mathrm{Err}(\varphi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\{L(Y, \varphi_{\mathcal{L}}(X))\}, \tag{1.1}$$

gdje je $\mathcal{L}$ skup za učenje korišten za izgradnju $\varphi_{\mathcal{L}}$, $\mathbb{E}_{X,Y}$ očekivanje, pri čemu slučajnost dolazi od vektora (X, Y), a L je funkcija gubitka koja mjeri odstupanje između svoja dva argumenta.

Primijetimo, uzimanjem novih nezavisnih jednako distribuiranih parova (X_i, Y_i), $i = 1, ..., m$, za velike m, prosječna greška $\frac{1}{m}\sum_{i=1}^{m} L(Y_i, \varphi_{\mathcal{L}}(X_i))$, po zakonu velikih brojeva konvergira upravo $\mathrm{Err}(\varphi_L)$. Cilj nije samo stvaranje najtočnijih predviđanja nad podskupom $\mathcal{L}$ poznatih podataka, već učenje modela koji će biti precizan i pouzdan na svim mogućim podacima.
Za klasifikaciju, najčešće korištena funkcija gubitka je indikatorska funkcija gubitka $L(Y, \varphi_{\mathcal{L}}(X)) = 1_{\{Y \neq \varphi_{\mathcal{L}}(X)\}}$, gdje se sve pogrešne klasifikacije jednako kažnjavaju. U ovom je slučaju očekivana pogreška predviđanja $\varphi_{\mathcal{L}}$ vjerojatnost pogrešne klasifikacije modela:

$$\mathrm{Err}(\varphi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\{1_{\{Y \neq \varphi_{\mathcal{L}}(X)\}}\} = \mathbb{P}(Y \neq \varphi_{\mathcal{L}}(X)). \tag{1.2}$$

Slično, za regresiju, najčešće korištena funkcija gubitka je kvadratna funkcija gubitka $L(Y, \varphi_{\mathcal{L}}(X)) = (Y - \varphi_{\mathcal{L}}(X))^2$. U ovom su slučaju velike razlike između stvarnih vrijednosti kažnjavane više nego male razlike. Uz ovakvu funkciju gubitka, očekivana pogreška predviđanja postaje:

$$\mathrm{Err}(\varphi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\{(Y - \varphi_{\mathcal{L}}(X))^2\}. \tag{1.3}$$

U praksi, vjerojatnosna distribucija $\mathbb{P}(X, Y)$ često ostaje nepoznata, što čini direktno procjenjivanje pogreške $\mathrm{Err}(\varphi_{\mathcal{L}})$ nemogućim. Također, često nije moguće doći do dodatnih podataka, što sprječava empirijsku procjenu $\mathrm{Err}(\varphi_{\mathcal{L}})$ na skupu $\mathcal{L}'$ putem neovisnog izvlačenja iz $\mathcal{L}$. U većini problema, skup podataka $\mathcal{L}$ predstavlja jedini dostupan izvor informacija, na kojem model treba učiti i procijeniti njegovu očekivanu pogrešku predviđanja. U nastavku ćemo istaknuti dva načina procjene greške (1.1).
Radi lakše notacije, definirajmo $\overline{E}(\varphi_{\mathcal{L}}, \mathcal{L}')$ kao prosječnu pogrešku predviđanja modela $\varphi_{\mathcal{L}}$ na skupu $\mathcal{L}'$, to jest

$$\overline{E}(\varphi_{\mathcal{L}}, \mathcal{L}') = \frac{1}{N'} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{L}'} L(y_i, \varphi_{\mathcal{L}}(\mathbf{x}_i)), \tag{1.4}$$

gdje je N' veličina skupa $\mathcal{L}'$.
Prva i najjednostavnija procjena očekivane pogreške predviđanja je *procjena na uzorku za učenje*. Ova procjena uključuje empirijsku procjenu $\mathrm{Err}(\varphi_{\mathcal{L}})$ na istom skupu podataka za učenje $\mathcal{L}$, koji se koristi za izgradnju modela $\varphi_{\mathcal{L}}$, to jest

$$\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi_{\mathcal{L}}) = \overline{E}(\varphi_{\mathcal{L}}, \mathcal{L}). \tag{1.5}$$

Općenito, procjena na uzorku za učenje je loša procjena $\mathrm{Err}(\varphi_{\mathcal{L}})$. To je zato što većina algoritama strojnog učenja cilja na precizno minimiziranje jednadžbe (1.5), što često rezultira pretjerano optimističnom procjenom očekivane pogreške predviđanja. Takva procjena uzima u obzir sve parove $(\mathbf{x}, y)$, a ne samo one iz skupa za učenje $\mathcal{L}$, što može biti problematično.

Drugi pristup je *procjena pomoću testnog skupa*. Ovaj pristup uključuje podjelu skupa za učenje $\mathcal{L}$ na dva disjunktna skupa, nazvana $\mathcal{L}_{\text{train}}$ i $\mathcal{L}_{\text{test}}$, poznata kao *skup za učenje* i *skup za testiranje*. Svaki od ovih skupova koristi se redom za učenje modela i procjenu očekivane pogreške predviđanja. Procjena pogreške generalizacije modela $\varphi_{\mathcal{L}}$, dobivena iz skupa $\mathcal{L}$, računa se kao prosječna pogreška predviđanja preko $\mathcal{L}_{\text{test}}$ modela $\varphi_{\mathcal{L}_{\text{train}}}$ koji je izgrađen na skupu $\mathcal{L}_{\text{train}}$:

$$\widehat{\text{Err}}^{\text{test}}(\varphi_{\mathcal{L}}) = \overline{E}(\varphi_{\mathcal{L}_{\text{train}}}, \mathcal{L}_{\text{test}}). \tag{1.6}$$

Obično, $\mathcal{L}_{\text{train}}$ se uzima kao 70% uzoraka iz skupa $\mathcal{L}$, dok preostalih 30% uzoraka čini skup $\mathcal{L}_{\text{test}}$. Pri podjeli skupa $\mathcal{L}$ na ova dva podskupa, važno je osigurati da uzorci iz $\mathcal{L}_{\text{train}}$-a budu neovisni o onima u $\mathcal{L}_{\text{test}}$-u i da budu izvučeni iz iste distribucije.
Međutim, ova metoda ima nedostatak jer smanjuje efektivnu veličinu uzorka na kojem uči model $\varphi_{\mathcal{L}_{\text{train}}}$. Iako dobivamo nepristranu procjenu za $\text{Err}(\varphi_{\mathcal{L}})$, to može utjecati na preciznost modela, jer manji skup za učenje može rezultirati manje pouzdanim predviđanjima na novim podacima.
U teoriji, kada je vjerojatnosna distribucija $\mathbb{P}(X, Y)$ poznata, najbolji mogući model, označen kao φ_B, koji minimizira očekivanu pogrešku predviđanja (1.1), može se izvesti analitički i neovisno o skupu za učenje $\mathcal{L}$. Uvjetovanjem na X, očekivana greška predviđanja postaje:

$$\mathbb{E}_{X,Y}\{L(Y, \varphi_B(X))\} = \mathbb{E}_X\{\mathbb{E}_{Y|X}\{L(Y, \varphi_B(X))\}\}. \tag{1.7}$$

U ovom obliku, model koji minimizira gornju jednakost je onaj model koji minimizira unutarnje očekivanje, to jest

$$\varphi_B(\mathbf{x}) = \arg\min_{y \in \mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\{L(Y, y)\}. \tag{1.8}$$

U literaturi, φ_B je poznat kao *Bayesov model*, a njegova očekivana pogreška predviđanja, $\text{Err}(\varphi_B)$, naziva se *ireducibilna greška* ili *Bayesova greška*. Ona predstavlja minimalnu pogrešku koju algoritam nadziranog učenja može postići, što znači da je to neizbježna pogreška koja nastaje isključivo zbog slučajnih odstupanja u podacima.

Definicija 1.3.2. *Model φ_B je Bayesov model, ako za svaki model φ izgrađen na podacima bilo kojeg skupa za učenje $\mathcal{L}$, vrijedi* $\text{Err}(\varphi_B) \leq \text{Err}(\varphi_{\mathcal{L}})$.

Ako se sada vratimo problemu klasifikacije s indikatorskom funkcijom gubitka L, vidimo da je Bayesov model:

$$\begin{aligned}\varphi_B(\mathbf{x}) &= \arg\min_{y\in\mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\{1_{\{Y\neq y\}}\} \\ &= \arg\min_{y\in\mathcal{Y}} \mathbb{P}(Y \neq y|X=\mathbf{x}) \\ &= \arg\max_{y\in\mathcal{Y}} \mathbb{P}(Y = y|X=\mathbf{x}).\end{aligned} \tag{1.9}$$

Jednakost (1.9) možemo interpretirati na način da je najbolji mogući klasifikator onaj koji sistematski predviđa najizgledniju klasu $y \in \{c_1, ..., c_J\}$ uz dano $X = \mathbf{x}$.
Slično, za regresiju s kvadratnom funkcijom gubitka imamo:

$$\begin{aligned}\varphi_B(\mathbf{x}) &= \arg\min_{y\in\mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\{(Y-y)^2\} \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\{Y\}.\end{aligned} \tag{1.10}$$

Najbolji mogući regresor stoga sustavno predviđa prosječne vrijednosti Y uz $X = \mathbf{x}$.
Za praktične probleme, vjerojatnosna distribucija $\mathbb{P}(X, Y)$ obično ostaje nepoznata, stoga Bayesov model, φ_B, ne može biti izveden analitički. U ovom kontekstu, procjena djelotvornosti modela $\varphi_{\mathcal{L}}$ može biti teška jer procjena pogreške $\mathrm{Err}(\varphi_{\mathcal{L}})$ možda nije jako indikativna za kvalitetu $\varphi_{\mathcal{L}}$, s obzirom na to da najmanja moguća pogreška, $\mathrm{Err}(\varphi_B)$, također ostaje nepoznata. Međutim, na simuliranim podacima, gdje je distribucija poznata, moguće je izvesti Bayesov model φ_B i izračunati njegovu pogrešku, $\mathrm{Err}(\varphi_B)$. Ovo postaje korisno jer se $\mathrm{Err}(\varphi_B)$ može koristiti kao referenca za usporedbu pogreške pomoću testnog skupa $\varphi_{\mathcal{L}}$ te tako procijeniti stvarnu učinkovitost modela $\varphi_{\mathcal{L}}$ u odnosu na najbolji mogući model koji je izvediv.
S teorijske perspektive, koncepti Bayesovog modela i Bayesove greške su korisni za istraživanje sposobnosti učenja algoritma. Konkretno, kada skup podataka $\mathcal{L}$ postaje proizvoljno velik, ključno pitanje je može li se proizvesti model $\varphi_{\mathcal{L}}$ koji je konzistentan, to jest, model čija očekivana pogreška predviđanja može proizvoljno težiti prema najmanjoj mogućoj očekivanoj pogrešci predviđanja $\mathrm{Err}(\varphi_B)$. Pretpostavimo da imamo niz nezavisnih jednako distribuiranih realizacija našeg uzorka. Tada je konzistentnost formalno definirana na sljedeći način:

Definicija 1.3.3. *Kažemo da je algoritam učenja $\mathcal{A}$ slabo konzistentan za distribuciju $\mathbb{P}(X, Y)$ ako $\mathbb{E}_{\mathcal{L}}\{\mathrm{Err}(\varphi_{\mathcal{L}})\} \to \mathrm{Err}(\varphi_B)$ kako veličina N skupa za učenje $\mathcal{L}$ korištenog za izgradnju modela $\varphi_{\mathcal{L}}$ pomoću algoritma $\mathcal{A}$ teži u beskonačnost.*

Definicija 1.3.4. *Algoritam učenja $\mathcal{A}$ je jako konzistentan za distribuciju $\mathbb{P}(X, Y)$ ako $\mathrm{Err}(\varphi_{\mathcal{L}}) \to \mathrm{Err}(\varphi_B)$ gotovo sigurno, kako veličina N skupa za učenje $\mathcal{L}$ korištenog za izgradnju modela $\varphi_{\mathcal{L}}$ pomoću algoritma $\mathcal{A}$ teži u beskonačnost.*

Uočimo da definicija konzistentnosti ovisi o distribuciji $\mathbb{P}(X, Y)$. Općenito, može se pokazati da je algoritam učenja $\mathcal{A}$ konzistentan za neke klase vjerojatnosnih distribucija, ali ne i za druge. Ako se konzistentost može dokazati za bilo koju distribuciju $\mathbb{P}(X, Y)$, tada se kaže da je $\mathcal{A}$ univerzalno konzistentan.

1.4 Odabir modela

Iz prethodne rasprave, čini se da bi za rješavanje problema nadziranog učenja bilo dovoljno procijeniti uvjetnu razdiobu od Y uz dano X, $\mathbb{P}(Y|X)$, iz uzorka za učenje $\mathcal{L}$ i zatim definirati model u skladu s tim pomoću jednadžbe (1.9) ili (1.10). Međutim, ovaj pristup nije izvediv u praksi jer zahtijeva eksponencijalno povećanje skupa za učenje $\mathcal{L}$ s povećanjem broja p ulaznih varijabli, kako bi se dobile točne procjene $\mathbb{P}(Y|X)$.
Da bi nadzirano učenje bilo uspješno u visokodimenzionalnim ulaznim prostorima s umjerenim veličinama skupa za učenje, potrebno je napraviti određene pretpostavke o strukturi najboljeg modela φ_B. Konkretno, algoritam nadziranog učenja pretpostavlja da φ_B - ili barem dovoljno dobra aproksimacija njega - pripada obitelji $\mathcal{H}$ modela kandidata, poznatih i kao *hipoteze* u teoriji statističkog učenja, koje imaju određene ograničene strukture. Prema ovoj pretpostavci, problem odabira modela definiran je kao pronalaženje najboljeg modela iz skupa $\mathcal{H}$, temeljenog na skupu za učenje $\mathcal{L}$.

Definicija 1.4.1. *Greška aproksimacije mjeri koliko blizu modeli u $\mathcal{H}$ mogu aproksimirati optimalni model φ_B :*

$$\mathrm{Err}(\mathcal{H}) = \min_{\varphi \in \mathcal{H}} \{\mathrm{Err}(\varphi)\} - \mathrm{Err}(\varphi_B). \tag{1.11}$$

Neka je θ vektor vrijednosti parametara koji kontroliraju izvođenje algoritma učenja $\mathcal{A}$. Naš je cilj pronaći θ koji daje najbolji mogući model koji pripada familiji $\mathcal{H}$ i uči na skupu $\mathcal{L}$:

$$\theta^* = \arg\min_{\theta} \mathrm{Err}(\mathcal{A}(\theta, \mathcal{L})). \tag{1.12}$$

Ovaj problem se ponovo u praksi ne može točno riješiti budući da zahtijeva da se stvarna očekivana pogreška predviđanja modela može izračunati. Međutim, aproksimacije $\hat{\theta}^*$ se mogu dobiti na razne načine.
Kada je $\mathcal{L}$ velik, najjednostavniji način da dobijemo $\hat{\theta}^*$ je koristeći uzorke definirane u (1.6) za usmjeravanje pretraživanja vrijednosti hiper-parametara, to jest

$$\hat{\theta}^* = \arg\min_{\theta} \widehat{\mathrm{Err}}^{\mathrm{test}}(\mathcal{A}(\theta, \mathcal{L})) = \arg\min_{\theta} \overline{E}(\mathcal{A}(\theta, \mathcal{L}_{\mathrm{train}}), \mathcal{L}_{\mathrm{test}}). \tag{1.13}$$

Primjer 1.4.2. *Linearna regresija i metoda najmanjih kvadrata.*
Pretpostavimo da nam je poznat skup podataka

$$\mathcal{L} = \{(x_1, y_1), ..., (x_n, y_n)\}, \quad (x_i, y_i) \in (\mathcal{X}, \mathcal{Y}),$$

zvan skup za učenje na temelju kojeg želimo izgraditi model. Dodatno pretpostavljamo da su $X = (x_1, ..., x_n)$ *i* $Y = (y_1, ..., y_n)$ *zavisni, preciznije, da vrijedi* $Y = f(X) + U$*, gdje je* U *slučajna varijabla nezavisna od* X *s očekivanjem* 0 *i varijancom* σ^2*. U problemu linearne regresije, pretpostavljamo da za neki* $\beta = (\beta_0, \beta_1)$*, imamo*

$$f(x) = \mathbb{E}(Y|X = x) = \beta_0 + \beta_1 x.$$

Označimo s $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$ *kvadratnu funkciju gubitka definiranu s* $L(y, y') = |y - y'|^2$ *koja mjeri veličinu pogreške. Uz danu familiju hipoteza* $\mathcal{H}$ *koje se sastoje od funkcija gornjeg oblika, suština problema regresije je pronaći* $\hat{f} \in \mathcal{H}$ *sa što manjom testnom greškom* $\mathbb{E}[L(\hat{f}(X), Y)]$*. Dakle, u slučaju linearne regresije s kvadratnom funkcijom gubitka, algoritam* $\mathcal{A}$ *na skupu za učenje* $\mathcal{L}$ *vraća linearnu funkciju* $\hat{f}$ *koja minimizira sumu kvadrata reziduala.*
Kod Ridge regresije uvodimo parametar složenosti $\lambda \geq 0$ *koji kontrolira količinu smanjenja: što je veća vrijednost* λ*, to je veće smanjenje. U ovom modelu, parametar* θ *odgovara upravo parametru složenosti* λ*. Umjesto minimiziranja greške po* β_0 *i* β_1 *kao to je slučaj kod linearne regresije, ovdje za fiksni* $\lambda \geq 0$ *minimiziramo* $L(y_1, \hat{f}(x_1)) + ... + L(y_n, \hat{f}(x_n)) + \lambda\beta_1^2$*. Dijeljenjem početnog skupa za učenje* $\mathcal{L}$ *na* $\mathcal{L}_{\text{train}}$ *i* $\mathcal{L}_{\text{test}}$*, koristeći testni skup procjenjujemo grešku od* $\mathcal{A}(\lambda, \mathcal{L}_{\text{train}})$*. Cilj nam je pronaći* λ *koji minimizira grešku generalizacije 1.13, to jest*

$$\hat{\lambda}^* = \arg\min_{\lambda} \overline{E}(\mathcal{A}(\lambda, \mathcal{L}_{\text{train}}), \mathcal{L}_{\text{test}}).$$

Prilikom optimizacije parametra θ, važno je biti oprezan kako rezultirajući model ne bi bio ni previše jednostavan ni previše složen. Prvi slučaj naziva se "underfitting" jer model premalo odgovara podacima. Drugi slučaj naziva se "overfitting" jer previše odgovara podacima. Prekomjerno prilagođavanje se također može objasniti dekompozicijom očekivane pogreške predviđanja u smislu pristranosti i varijance. Model koji je previše jednostavan obično ima visoku pristranost, ali nisku varijancu, dok model koji je previše složen obično ima nisku pristranost, ali visoku varijancu. U tom smislu, pronalaženje najboljeg modela znači postizanje odgovarajućeg kompromisa pristranosti i varijance ("bias-variance tradeoff").
U stvarnim primjenama, a posebno prilikom objavljivanja rezultata, obično želimo napraviti i odabir modela i procjenu modela. Drugim riječima, odabirom konačnog modela $\mathcal{A}(\hat{\theta}^*)$, želimo također procijeniti očekivanu grešku predviđanja. Naivna procjena očekivane pogreške predviđanja mogla bi biti jednostavno korištenje procjene testnog uzorka

$\widehat{\text{Err}}^{\text{test}}(\mathcal{A}(\theta, \mathcal{L}))$ koja je minimizirana tijekom odabira modela. Problem s ovom procjenom je taj da naučeni model nije neovisan o $\mathcal{L}_{\text{test}}$-u jer je njegova ponavljana konstrukcija bila vođena minimiziranjem pogreške predviđanja na $\mathcal{L}_{\text{test}}$-u. Kao rezultat, minimizirana pogreška testnog uzorka je zapravo pristrana, optimistična procjena stvarne očekivane pogreške predviđanja, koja ponekad dovodi do znatno podcijenjenih procjena.
Kako bi se zajamčila nepristrana procjena, testni skup na kojem se procjenjuje očekivana pogreška predviđanja, idealno bi trebao biti izvan cijelog postupka odabira modela i koristiti se isključivo nakon odabira konačnog modela. Algoritam u nastavku detaljno opisuje takav protokol.

Algoritam 1.4.3. *Protokol skupa Train-Valid-Test za odabir i procjenu modela.*

1. *Podijelite skup za učenje $\mathcal{L}$ na tri dijela: $\mathcal{L}_{\text{train}}$, $\mathcal{L}_{valid}$, $\mathcal{L}_{\text{test}}$;*

2. *Izvršite odabir modela na $\mathcal{L}_{\text{train}} \cup \mathcal{L}_{valid}$ koristeći procjene testnog uzorka, to jest, odredite:*

$$\begin{aligned} \hat{\theta}^* &= \arg\min_{\theta} \widehat{\text{Err}}^{\text{test}}(\mathcal{A}(\theta, \mathcal{L}_{\text{train}} \cup \mathcal{L}_{valid})) \\ &= \arg\min_{\theta} \overline{E}(\mathcal{A}(\theta, \mathcal{L}_{\text{train}}), \mathcal{L}_{valid}); \end{aligned} \tag{1.14}$$

3. *Ocijenite (nepristranu) očekivanu pogrešku predviđanja konačnog modela kao*

$$\overline{E}(\mathcal{A}(\hat{\theta}^*, \mathcal{L}_{\text{train}} \cup \mathcal{L}_{valid}), \mathcal{L}_{\text{test}});$$

4. *Naučite konačni model $\mathcal{A}(\hat{\theta}^*, \mathcal{L})$ na cijelom skupu za učenje.*

Poglavlje 2

Klasifikacijska i regresijska stabla

2.1 Uvod

Od samog otkrića, umjetna inteligencija služi kao alat za obradu podataka i otkrivanje veza među njima. Cilj je razviti modele koje ćemo lako interpretirati te koji rezultiraju s preciznim predviđanjima. Kao takvu metodu u ovom poglavlju proučavamo metode bazirane na stablima. One prirodno uklapaju u model oba tipa varijabli (numeričke i kategoričke). Također se koriste kao temelj mnogih modernih i složenih algoritama. Imune su na efekte outlier-a te uključivanje mnogih nebitnih prediktorskih varijabli. Kao takve, metode bazirane na stablima su jedna od popularnijih metoda za rudarenje podataka.

2.2 Modeli bazirani na stablima

Kod problema klasifikacije gdje je $\mathcal{Y}$ kategorijska varijabla, možemo primijetiti da $\mathcal{Y}$ definira particiju vjerojatnosnog prostora Ω definiranu kao:

$$\Omega = \Omega_{c_1} \cup \Omega_{c_2} \cup ... \cup \Omega_{c_J}, \tag{2.1}$$

gdje je Ω_{c_k} skup na objekata na kojim Y poprima vrijednost c_k. Slično, klasifikator φ također može particionirati vjerojatnosni prostor Ω obzirom da definira aproksimaciju $\hat{Y}$ od Y. Doduše, ova je particija definirana ja ulaznom prostoru $\mathcal{X}$:

$$\mathcal{X} = \mathcal{X}^{\varphi}_{c_1} \cup X^{\varphi}_{c_2} \cup ... \cup X^{\varphi}_{c_J}, \tag{2.2}$$

gdje je $X^{\varphi}_{c_k}$ skup vektora $\mathbf{x} \in \mathcal{X}$ takvih da je $\varphi(\mathbf{x}) = c_k$. Učenje klasifikatora možemo slično preoblikovati kao učenje particije $\mathcal{X}$ koja odgovara što je bliže moguće najboljoj particiji (onoj koju je proizveo Bayesov model φ_B) na $\mathcal{X}$:

$$\mathcal{X} = \mathcal{X}_{c_1}^{\varphi_B} \cup X_{c_2}^{\varphi_B} \cup ... \cup X_{c_J}^{\varphi_B}. \tag{2.3}$$

Iz gornje diskusije dobivamo jednostavnu motivaciju na kojoj se bazira princip metoda baziranih na stablima. Naime, ovaj princip uključuje aproksimacije particije Bayesovog modela rekurzivnim dijeljenjem ulaznog prostora $\mathcal{X}$ na potprostore, a zatim dodjeljivanje konstantnih vrijednosti predviđanja $\hat{y} \in \mathcal{Y}$ svakom objektu $\mathbf{x}$ unutar svakog završnog potprostora. U nastavku ćemo koristiti osnovne pojmove teorije grafova pa ih stoga u nastavku definiramo.

Stablo je povezan graf koji nema ciklus. *List* je vrh stupnja 1. *Ukorijenjeno stablo* je stablo u kojem je jedan od čvorova označen kao korijen. U našem slučaju dodatno pretpostavljamo da je ukorijenjeno stablo usmjereni graf, gdje su svi bridovi usmjereni od korijena. Ako postoji brid od t_1 do t_2, tada kažemo da je t_1 *roditelj* čvora t_2, a t_2 *dijete* čvora t_1. Ako postoji put od t_1 do t_2 kažemo da je t_2 *potomak* od t_1. U ukorijenjenom stablu, za čvor kažemo da je *unutarnji* ako ima jedno ili više djece i *završni* ili *terminalni* ako nema djece. Završni čvorovi su također poznati kao listovi. *Binarno stablo* je ukorijenjeno stablo gdje svi unutarnji čvorovi imaju točno dva djeteta.

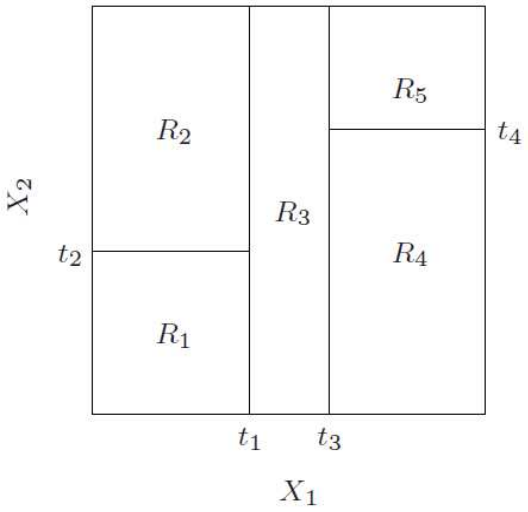
Stablo odlučivanja definiramo kao model $\varphi : \mathcal{X} \mapsto \mathcal{Y}$ predstavljen ukorijenjenim stablom, gdje čvor t predstavlja podskup $\mathcal{X}_t \subset \mathcal{X}$ ulaznog prostora. Korijen t_0 odgovara samom ulaznom prostoru $\mathcal{X}$. Unutarnje čvorove t označavamo s podjelom s_t određenom skupom pitanja Q. Ona dijeli skup $\mathcal{X}_t$ na disjunktne podskupe koji odgovaraju svakom od njegovih potomaka. Završne čvorove označavamo najboljom pretpostavljenom vrijednošću $\hat{y}_t \in \mathcal{Y}$. Predviđena izlazna vrijednost $\varphi(\mathbf{x})$ je oznaka lista do kojeg je došla instanca $\mathbf{x}$ prateći podjele s_t. Gornji postupak možemo opisati donjim algoritmom.

Algoritam 2.2.1. *Predviđanje izlazne vrijednosti $y = \varphi(\mathbf{x})$ u stablu odlučivanja.*

- ***function*** *predvidi($\varphi, \mathbf{x}$)*
  - $t = t_0$
  - ***while*** *t nije završni čvor*
    - *t=potomak t′ čvora t takav da je* $\mathbf{x} \in \mathcal{X}_{t'}$
  - ***end while***
  - ***return*** $\hat{y}_t$
- ***end function***

Na sljedećem primjeru ilustriramo algoritam stabla odlučivanja.

Primjer 2.2.2. *Proučavamo problem regresije s neprekidnim odazivom $Y \in \mathbb{R}$ te ulaznim varijablama $\mathcal{X}_1$ i $\mathcal{X}_2$ iz jediničnog intervala, to jest $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 = [0,1]^2$. Radi jednostavnosti ograničavamo se na rekurzivne binarne particije. Prvo podijelimo prostor na dvije regije i modeliramo odgovor srednjom vrijednosti Y u svakoj regiji. Odabiremo varijablu i točku podjele kako bismo postigli najbolje prilagođavanje. Zatim jednu ili obje regije dalje dijelimo na još dvije regije, i taj se proces nastavlja dok se ne primijeni određeno pravilo zaustavljanja. Na primjer, prvo dijelimo na $X_1 = t_1$. Zatim se regija $X_1 \leq t_1$ dijeli na $X_2 = t_2$, a regija $X_1 > t_1$ dijeli na $X_1 = t_3$. Konačno, regija $X_1 > t_3$ dijeli se na $X_2 = t_4$. Rezultat ovog procesa je podjela na pet regija $R_1, R_2, ..., R_5$ prikazanih na slici.*

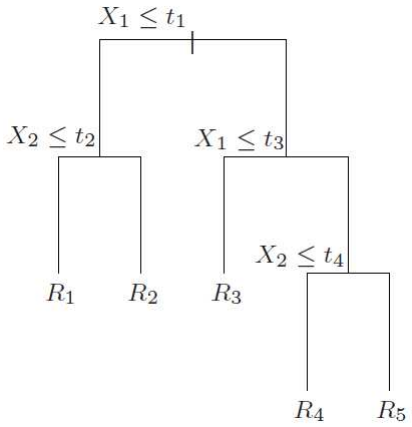


Slika 2.1: The Elements of Statistical Learning: Data mining, Inference, and Prediction - particija skupa $\mathcal{X}$

Odgovarajući regresijski model predviđa Y s konstantom c_m u regiji R_m, odnosno:

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{5} c_m 1_{\{(\mathbf{x}) \in R_m\}}. \tag{2.4}$$

Isti model može biti predstavljen binarnim stablom - Slika 2.2. Cijeli skup podataka nalazi se na vrhu stabla. Opservacije koje zadovoljavaju uvjet na svakom razdjeljku dodjeljuju se lijevom granu, a druge desnoj grani. Završni čvorovi ili listovi stabla odgovaraju regijama R_1, $R_2, ..., R_5$.



Slika 2.2: The Elements of Statistical Learning: Data mining, Inference, and Prediction - prikaz na binarnom stablu

2.3 Indukcija kod stabla odlučivanja

U idealnom slučaju, učenje stabla odlučivanja značilo bi određivanje strukture stabla koja proizvodi particiju najbližu particiji koju stvara Y na $\mathcal{X}$. Međutim, pošto ta particija nije poznata, konstrukcija stabla odlučivanja se obično provodi s ciljem pronalaženja modela koji što bolje podijeli skup za učenje $\mathcal{L}$. Među svim stablima odlučivanja $\varphi \in \mathcal{H}$, može postojati nekoliko njih koji jednako dobro objašnjavaju skup za učenje $\mathcal{L}$. Stoga navodimo mjeru *nečistoće* $i(t)$ (koju ćemo kasnije formalno definirati) kao funkciju koja procjenjuje prilagodbu svakog čvora. Manji $i(t)$ predstavlja *čišći* čvor i bolje predikcije $\hat{y}_t(\mathbf{x})$ za sve $\mathbf{x} \in \mathcal{L}_t$, gdje je $\mathcal{L}_t$ skup svih $(\mathbf{x}, y) \in \mathcal{L}$ takvih da $\mathbf{x} \in \mathcal{X}_t$, to jest $\mathcal{L}_t = \{(\mathbf{x}, y) \in \mathcal{L} : \mathbf{x} \in \mathcal{X}_t\}$. Krenuvši od jednog čvora koji obuhvaća cijeli skup za učenje $\mathcal{L}$, gotovo-optimalna stabla odlučivanja mogu se stvarati pohlepno. To znači da se iterativno dijele čvorovi u manje nečiste čvorove. Postupak se sastoji od iterativnog dijeljenja podskupova od $\mathcal{L}$ u manje podskupove, do nekog kriterija zaustavljanja (koji sami odabiremo), što osigurava čišće i čišće potomke te gotovo optimalna predviđanja za $\mathcal{L}$. Kako bismo dobili što manje rezultirajuće stablo, u svakom koraku dijeljenja čvora t tražimo ono koje lokalno maksimizira smanjenje nečistoće čvora potomka. Mi ćemo se radi jednostavnosti fokusirati na binarna dijeljenja. Formalno, smanjenje nečistoće definiramo na sljedeći način:

Definicija 2.3.1. *Smanjenje nečistoće binarne podjele $s \in Q$ dijeljenja čvora t u lijevi čvor t_L i desni čvor t_R je*

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R), \tag{2.5}$$

gdje p_L (p_R) predstavlja proporciju $\frac{N_{t_L}}{N_t}$ ($\frac{N_{t_R}}{N_t}$) uzoraka za učenje iz $\mathcal{L}_t$ koji idu u t_L (t_R) te

gdje je N_t veličina podskupa $\mathcal{L}_t$.

Na temelju ovog koncepta, opću pohlepnu proceduru za indukciju stabala odlučivanja klasifikacije formalno opisujemo sljedećim algoritmom.

Algoritam 2.3.2. *Pseudokod izgradnje stabla odlučivanja koristeći CART algoritam.*

1. *Krećemo od korijena;*
2. *Za svaki čvor t, nađimo dijeljenje $s^* \in Q$ skupa X koje maksimizira smanjenje nečistoće te podijelimo skup na taj način;*
3. *Ako smo postigli kriterij zaustavljanja, algoritam je gotov. Inače primijeni korak 2 na svoj djeci nastaloj u trenutnoj iteraciji.*

Glavna razlika između CART algoritma za klasifikaciju i CART algoritma za regresiju je u mjeri nečistoće koja se koristi za određivanje najbolje točke podjele s^* i kovarijate po kojoj dijelimo. Također valja napomenuti da je ovo bazični CART algoritam te da postoje razne varijacije i poboljšanja poput slučajnih šuma, obrezivanje uz kriterij trošak-složenost, stabla uz poticanje gradijenta i drugi.
U nastavku ovog poglavlja detaljno objašnjavamo dijelove algoritma 2.3.2. Idući odjeljak govori o pravilima dodjeljivanja i dijeljenju čvorova kod problema klasifikacije, dok odjeljak 2.5 opisuje to isto za problem regresije. U oba ćemo potpoglavlja formalno definirati pojam nečistoće. Poglavlje završavamo odjeljkom 2.6 koji daje pravila zaustavljanja izgradnje stabla.

2.4 Klasifikacija

Pretpostavimo sada da smo čvor t odredili kao terminalni, koristeći neki kriterij zaustavljanja. Sljedeći korak u postupku indukcije je dodjela čvoru t konstantne vrijednosti $\hat{y}_t$, koja će se koristiti kao predviđanje izlazne varijable Y. Čvor t može se promatrati kao pojednostavljeni model koji je lokalno definiran na prostoru $\mathcal{X}_t \times \mathcal{Y}$ i proizvodi istu predviđenu vrijednost $\hat{y}_t$ za sve moguće ulazne vektore koji pripadaju čvoru t. Primijetimo da je za stablo φ fiksne strukture, minimiziranje globalne pogreške generalizacije ekvivalentno minimiziranju lokalne pogreške generalizacije svakog pojednostavljenog modela u terminalnim čvorovima. Doista,

$$\mathrm{Err}(\varphi) = \mathbb{E}_{X,Y}\{L(Y, \varphi(X))\} = \sum_{t \in \widetilde{\varphi}} \mathbb{P}(X \in \mathcal{X}_t)\mathbb{E}_{X,Y|t}\{L(Y, \hat{y}_t)\}, \tag{2.6}$$

gdje $\widetilde{\varphi}$ označava skup terminalnih čvorova u φ, a promatrano očekivanje $\mathbb{E}_{X,Y|t}$ je zapravo očekivanje uz uvjet $X \in \mathcal{X}_t$.

S obzirom na to da nam je cilj pronaći model koji minimizira grešku generalizacije $\mathrm{Err}(\varphi)$, prema gornjem raspisu vidimo da u slučaju stabla odlučivanja to odgovara pronalasku najboljih konstanti $\hat{y}_t$ u svakom terminalnom čvoru. U nastavku ovog pododjeljka nam je cilj odrediti čemu su jednaki $\hat{y}_t$ u slučaju klasifikacije i regresije.
Pretpostavimo prvo da je L indikatorska funkcija gubitka. Tada je $\hat{y}_t^*$ koji minimizira očekivanje u jednadžbi (2.6) određen s:

$$\begin{aligned}\hat{y}_t^* &= \arg\min_{c\in\mathcal{Y}} \mathbb{E}_{X,Y|t}\{1_{\{(Y,c)\}}\} \\ &= \arg\min_{c\in\mathcal{Y}} \mathbb{P}(Y \neq c|X \in \mathcal{X}_t) \\ &= \arg\max_{c\in\mathcal{Y}} \mathbb{P}(Y = c|X \in \mathcal{X}_t).\end{aligned} \tag{2.7}$$

Riječima, pogreška generalizacije u čvoru t je minimizirana predviđanjem klase koja je najvjerojatnija za uzorke iz potprostora od t. S obzirom na to da nam funkcija distribucije $\mathbb{P}(X, Y)$ nije poznata, za njenu procjenu koristimo empirijsku funkciju distribucije $\hat{\mathbb{P}}(X, Y)$. Neka je N_t broj objekata u $\mathcal{L}_t$ te N_{ct} broj objekata koji pripadaju klasi c u $\mathcal{L}_t$. Tada vrijedi $\hat{\mathbb{P}}(Y = c|X \in \mathcal{X}_t) = \frac{N_{ct}}{N_t}$. Također, primijetimo da vrijedi $\hat{\mathbb{P}}(X \in \mathcal{X}_t) = \frac{N_t}{N}$. Korištenjem gornje notacije dobivamo da je aproksimacija rješenja jednadžbe (2.7) dana s:

$$\hat{y}_t = \arg\min_{c\in\mathcal{Y}}\{1 - \hat{\mathbb{P}}(Y = c|X \in \mathcal{X}_t)\} = \arg\max_{c\in\mathcal{Y}} \hat{\mathbb{P}}(Y = c|X \in \mathcal{X}_t). \tag{2.8}$$

Radi lakše notacije, koristimo $\mathbb{P}(c|X \in \mathcal{X}_t)$ kao oznaku za $\mathbb{P}(Y = c|X \in \mathcal{X}_t)$. Uvrštavanjem empirijske funkcije distribucije te gornjeg rješenja u jednadžbu (2.6) dobivamo:

$$\begin{aligned}\widehat{\mathrm{Err}}(\varphi) &= \sum_{t\in\widetilde{\varphi}} \hat{\mathbb{P}}(X \in \mathcal{X}_t)(1 - \hat{\mathbb{P}}(\hat{y}_t|X \in \mathcal{X}_t)) \\ &= \sum_{t\in\widetilde{\varphi}} \frac{N_t}{N}(1 - \frac{N_{\hat{y}_t t}}{N_t}) \\ &= \frac{1}{N}\sum_{t\in\widetilde{\varphi}} N_t - N_{\hat{y}_t t} \\ &= \frac{1}{N}\sum_{t\in\widetilde{\varphi}}\sum_{\mathbf{x},y\in\mathcal{L}_t} 1_{\{y\neq\hat{y}_t\}} \\ &= \frac{1}{N}\sum_{\mathbf{x},y\in\mathcal{L}} 1_{\{y\neq\varphi(\mathbf{x})\}} \\ &\overset{(1.5)}{=} \widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi).\end{aligned}$$

Prethodnim raspisom dobili smo da se aproksimacija $\widehat{\mathrm{Err}}(\varphi)$ svodi na procjenu na uzorku za učenje. $\hat{y}_t$ definiran kao u (2.8) zapravo minimizira grešku *treniranja*, a ne stvarnu pogrešku generalizacije.
Iduća propozicija dokazuje važno svojstvo pravila pridruživanja (2.8), a to je da što više netko dijeli terminalni čvor, to manja postaje procjena na uzorku za učenje $\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi)$.

Propozicija 2.4.1. *Za bilo koju nepraznu podjelu terminalnog čvora $t \in \widetilde{\varphi}$ na t_L i t_R, što rezultira novim stablom φ' gdje su pritom $\hat{y}_{t_L}$ i $\hat{y}_{t_R}$ definirani kao u 2.8, vrijedi:*

$$\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi) \geq \widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi').$$

Jednakost vrijedi za $\hat{y}_t = \hat{y}_{t_L} = \hat{y}_{t_R}$.

Dokaz.

$$\begin{aligned}
\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi) &\geq \widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi') \\
\sum_{t\in\widetilde{\varphi}} \hat{\mathbb{P}}(X \in \mathcal{X}_t)(1 - \hat{\mathbb{P}}(\hat{y}_t | X \in \mathcal{X}_t)) &\geq \sum_{t\in\widetilde{\varphi'}} \hat{\mathbb{P}}(X \in \mathcal{X}_t)(1 - \hat{\mathbb{P}}(\hat{y}_t | X \in \mathcal{X}_t)) \\
\hat{\mathbb{P}}(X \in \mathcal{X}_t)(1 - \hat{\mathbb{P}}(\hat{y}_t | X \in \mathcal{X}_t)) &\geq \hat{\mathbb{P}}(X \in \mathcal{X}_{t_L})(1 - \hat{\mathbb{P}}(\hat{y}_{t_L} | X \in \mathcal{X}_{t_L})) \\
&\quad + \hat{\mathbb{P}}(X \in \mathcal{X}_{t_R})(1 - \hat{\mathbb{P}}(\hat{y}_{t_R} | X \in \mathcal{X}_{t_R})) \\
\frac{N_t}{N}(1 - \max_{c\in\mathcal{Y}} \frac{N_{ct}}{N_t}) &\geq \frac{N_{t_L}}{N}(1 - \max_{c\in\mathcal{Y}} \frac{N_{ct_L}}{N_{t_L}}) + \frac{N_{t_R}}{N}(1 - \max_{c\in\mathcal{Y}} \frac{N_{ct_R}}{N_{t_R}}) \\
N_t - \max_{c\in\mathcal{Y}} N_{ct} &\geq N_{t_L} - \max_{c\in\mathcal{Y}} N_{ct_L} + N_{t_R} - \max_{c\in\mathcal{Y}} N_{ct_R} \\
\max_{c\in\mathcal{Y}} N_{ct} &\leq \max_{c\in\mathcal{Y}} N_{ct_L} + \max_{c\in\mathcal{Y}} N_{ct_R} \\
\max_{c\in\mathcal{Y}}(N_{ct_L} + N_{ct_R}) &\leq \max_{c\in\mathcal{Y}} N_{ct_L} + \max_{c\in\mathcal{Y}} N_{ct_R},
\end{aligned}$$

što vrijedi, obzirom da je $\max_{c\in\mathcal{Y}} N_{ct_L}$ ($\max_{c\in\mathcal{Y}} N_{ct_R}$) uvijek veće ili jednako N_{ct_L} (N_{ct_R}). □

Iz prethodne propozicije možemo zaključiti da je procjena na uzorku za učenje minimalna kada se terminalni čvorovi ne mogu više dijeliti, to jest ako se terminalni čvorovi mogu podijeliti sve dok sve ne sadrže točno jedan element iz $\mathcal{L}$.
Kao što smo prethodno najavili, sada ćemo formalno definirati dijeljenje čvora t. Suštinski, ne radi se o ničemu drugom nego particioniranju skupa $\mathcal{X}_t$ na disjunktne potprostore koji odgovaraju potomcima čvora t.

Definicija 2.4.2. *Dijeljenje s čvora t je particija skupa $\mathcal{X}_t$, to jest skup nepraznih podskupova od $\mathcal{X}_t$ takvih da je svaki element $\mathbf{x} \in \mathcal{X}_t$ u točno jednom od tih podskupova ($\mathcal{X}_t$ je unija disjunktnih podskupova).*

Ako s podijeli $\mathcal{X}_t$ u dva podskupa, kažemo da je s binarno dijeljenje te lijevi i desni potomak od t označavamo redom s t_L i t_R. Općenito, to ne mora biti slučaj, s može dijeliti $\mathcal{X}_t$ u više od dva podskupa, što rezultira u više potomaka $t_{i_1}, ..., t_{i_N}$. Kao što smo ranije napomenuli, mi ćemo se fokusirati na binarna dijeljenja.
Cilj nam je particionirati t koristeći dijeljenje s^* koje maksimizira smanjenje nečistoće

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R),$$

gdje funkcija nečistoće $i(t)$ mjeri prilagodbu čvora t. To znači da za sve kovarijate $j = 1, ..., p$ moramo razmotriti sva moguća binarna dijeljenja po toj kovarijati i odabrati ono dijeljenje koje maksimizira smanjenje nečistoće. Mjera nečistoće je u problemu klasifikacije definirana tako da minimizira udio pogrešnih klasifikacija.

Definicija 2.4.3. *U klasifikaciji, funkcija nečistoće $i_R(t)$ na temelju lokalne procjene na uzorku za učenje definirana na indikatorskoj funkciji gubitka je:*

$$i_R(t) = 1 - \hat{\mathbb{P}}(\hat{y}_t | X \in \mathcal{X}_t) = 1 - \max_{c \in \mathcal{Y}} \hat{\mathbb{P}}(c | X \in \mathcal{X}_t).$$

Ova definicija ima sljedeća dva nedostatka:

- $\Delta i(s,t)$ je jednak nuli za sva dijeljenja ako većinska klasa ostane ista u oba čvora djeteta, to jest ako $\hat{y}_t = \hat{y}_{t_L} = \hat{y}_{t_R}$.
- Ne uzima u obzir aposteriorne promjene u distribuciji klasa $\hat{\mathbb{P}}(\hat{y} | X \in \mathcal{X}_{t_L})$ i $\hat{\mathbb{P}}(\hat{y} | X \in \mathcal{X}_{t_R})$.

Idući jednostavan primjer pobliže ilustrira gornja dva nedostatka.

Primjer 2.4.4. *Promotrimo problem binarne klasifikacije definiranog na tri binarne kovarijate X_1, X_2 te X_3. Pretpostavimo da nam je dan uzorak za učenje $\mathcal{L}$, sadržan u čvoru t_0 od 10 podataka prikazanih u sljedećoj tablici:*

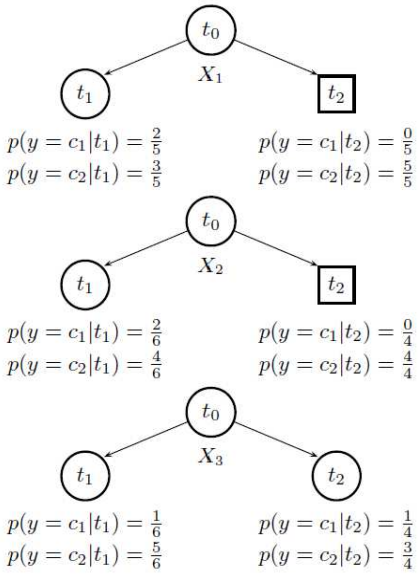
i	X_1	X_2	X_3	Y
1	0	0	0	c_1
2	0	0	1	c_1
3	0	1	0	c_2
4	0	1	1	c_2
5	0	1	1	c_2
6	1	0	0	c_2
7	1	0	0	c_2
8	1	0	0	c_2
9	1	0	0	c_2
10	1	1	1	c_2

Želimo pronaći najbolje binarno dijeljenje definirano na jednoj od kovarijata. S obzirom na to da su sve kovarijate binarne, za svaku od njih imat ćemo po jedno binarno dijeljenje koje particionira skup za učenje $\mathcal{L}$ na dva podskupa određena s:

$$\mathcal{L}_{t_L} = \{(\mathbf{x}, y) : (\mathbf{x}, y) \in \mathcal{L}, x_j = 0\};$$

$$\mathcal{L}_{t_R} = \{(\mathbf{x}, y) : (\mathbf{x}, y) \in \mathcal{L}, x_j = 1\}.$$

Označimo s t_1 lijevo dijete čvora t_0, a t_2 desno. Dijeljenjem s obzirom na kovarijatu X_1 dobivamo da je $\hat{\mathbb{P}}(y = c_1|X \in t_1) = \frac{2}{5}$ i $\hat{\mathbb{P}}(y = c_2|X \in t_1) = \frac{3}{5}$, dok je $\hat{\mathbb{P}}(y = c_1|X \in t_2) = \frac{0}{5}$ i $\hat{\mathbb{P}}(y = c_2|X \in t_2) = \frac{5}{5}$. Analogni račun provedemo za kovarijate X_2 i X_3 te dobijemo grafički prikaz dijeljenja 2.3.



Slika 2.3: Understanding Random Forests: From Theory to Practice - dijeljenja obzirom na različite kovarijate

Uvrštavanjem dobivenih podataka dobijemo $i_R(t_0) = 1 - \max\{\frac{2}{10}, \frac{8}{10}\} = \frac{2}{10} = \frac{1}{5}$. Dijeljenjem po kovarijati X_1, dobivamo $i_R(t_1) = \frac{2}{5}$ te $i_R(t_2) = 0$. Stoga je smanjenje nečistoće jednako

$$\Delta i(s, t_0) = i_R(s, t_0) - p_L i_R(s, t_1) - p_R i_R(s, t_2) = \frac{1}{5} - \frac{5}{10}\frac{2}{5} - \frac{5}{10}0 = 0.$$

Analognim računom dobije se da je $\Delta i(s, t_0)$ također 0 za dijeljenja s obzirom na kovarijate X_2 te X_3, što ilustrira prvi nedostatak i značilo bi da su sva gornja dijeljenja jednako dobra. Dakle, u ovom primjeru ne postoji dijeljenje koje smanjuje nečistoću. Usprkos

tome, dijeljenjem s obzirom na prvu kovarijatu nam daje stablo koje je jednostavnije od druga dva. Na primjer, uspoređujući prva dva dijeljenja, u oba slučaja dobivamo desno dijete koje je terminalno, samo što u drugom slučaju lijevo dijete sadrži više podataka što sugerira da bi mogli imati još dijeljenja. Dijeljenjem s obzirom na X_3 dobivamo stablo u kojem je potrebno dijeliti oba djeteta. Ovo ilustrira drugi nedostatak, jer bismo trebali dijeliti s obzirom na X_1, premda to nismo uspjeli pokazati samim računom.

Problem algoritma pohlepne indukcije 2.3.2 je da se temelji na proceduri optimizacije u jednom koraku. Dobar kriterij nečistoće trebao bi također uzeti u obzir mogućnost daljnjeg poboljšanja dublje u stablu. Klasa funkcija nečistoće $i(t)$ koja ispunjava ovaj zahtjev dana je u narednom teoremu:

Teorem 2.4.5. *Neka je $\Phi(p_1, ..., p_J)$ strogo konkavna J-dimenzionalna, nenegativna funckija definirana na $0 \leq p_k \leq 1$, za $k = 1, ..., J$, $\sum_{k=1}^{J} p_k = 1$ takva da:*

- $\Phi(1, 0, ..., 0) = \Phi(0, 1, ..., 0) = \Phi(0, 0, ..., 1)$ *je minimalan;*
- $\Phi(\frac{1}{J}, \frac{1}{J}, ..., \frac{1}{J})$ *je maksimalan.*

Tada za $i(t) = \Phi(\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_t), ..., \hat{\mathbb{P}}(c_j|X \in \mathcal{X}_t))$ i bilo koje dijeljenje s vrijedi

$$\Delta i(s, t) \geq 0.$$

Jednakost vrijedi ako i samo ako $\hat{\mathbb{P}}(c_k|X \in \mathcal{X}_{t_L}) = \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_{t_R}) = \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_t)$ za sve $k = 1, ..., J$.

Dokaz. Prvo primijetimo da je

$$\begin{aligned}
\hat{\mathbb{P}}(c_k|X \in \mathcal{X}_t) =& \frac{N_{c_k t}}{N_t} \\
&= \frac{N_{c_k t_L}}{N_{c_k t_R}} \\
&= \frac{N_{t_L}}{N_t}\frac{N_{c_k t_L}}{N_{t_L}} + \frac{N_{t_R}}{N_t}\frac{N_{c_k t_R}}{N_{t_R}} \\
&= p_L \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_{t_L}) + p_R \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_{t_R}).
\end{aligned}$$

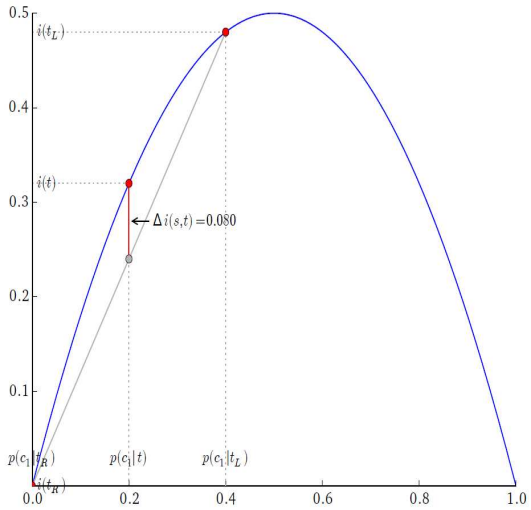
Zbog stroge konkavnosti imamo

$$\begin{aligned}
i(t) &= \Phi(\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_t), ..., \hat{\mathbb{P}}(c_J|X \in \mathcal{X}_t)) \\
&= \Phi(p_L \hat{\mathbb{P}}(c_1|X \in \mathcal{X}_{t_L}) + p_R \hat{\mathbb{P}}(c_1|X \in \mathcal{X}_{t_R}), ..., p_L \hat{\mathbb{P}}(c_J|X \in \mathcal{X}_{t_L}) + p_R \hat{\mathbb{P}}(c_J|X \in \mathcal{X}_{t_R})) \\
&\geq p_L \Phi(\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_{t_L}), ..., \hat{\mathbb{P}}(c_J|X \in \mathcal{X}_{t_L})) + p_R \Phi(\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_{t_R}), ..., \hat{\mathbb{P}}(c_J|X \in \mathcal{X}_{t_R})) \\
&= p_L i(t_L) + p_R i(t_R).
\end{aligned}$$

Jednakost vrijedi ako i samo ako je $\hat{\mathbb{P}}(c_k|X \in \mathcal{X}_{t_L}) = \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_{t_R})$ za $k = 1, ..., J$. □

Sljedeći primjer pojašnjava što smo postigli prethodnim teoremom.

Primjer 2.4.6. *Pretpostavimo da imamo problem klasifikacije s dvije klase - 0 ili 1 te dolazi do binarnog dijeljenja u čvoru t. Za strogo konkavnu nenegativnu funkciju jedne varijable Φ, promatramo graf funkcije nečistoće i obzirom na vjerojatnost prve klase $\hat{\mathbb{P}}(c_1)$ definiranu kao $i(t) = \Phi(\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_t))$. Kao što je prikazano na dijagramu, $i(t)$ je maksimalan kada je nesigurnost na izlaznoj varijabli najveća ($\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_t) = \hat{\mathbb{P}}(c_2|X \in \mathcal{X}_t) = \frac{1}{2}$) te postaje manji kako sigurnost raste prema jednoj od klasa (npr. kako se $\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_t)$ približava 0 ili 1). Kao što smo pokazali u teoremu, $i(t)$ je također nužno veći od srednje vrijednosti nečistoće u potomcima. Vizualno, što su $\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_{t_L})$ i $\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_{t_R})$ udaljeniji od $\hat{\mathbb{P}}(c_1|X \in \mathcal{X}_t)$ veći je $\Delta i(s,t)$ (prikazano crvenom linijom) i bolje je dijeljenje s.*



Slika 2.4: Understanding Random Forests: From Theory to Practice - dijeljenje čvora t vrednovano strogo konkavnom funkcijom nečistoće. Prikazan je graf funkcije nečistoće i obzirom na lokalnu vjerojatnost prve klase.

Najčešći kriteriji nečistoća koji se koriste za klasifikacijska stabla su Shannonova entropija i Ginijev indeks:

Definicija 2.4.7. *Funkcija nečistoće $i_H(t)$ temeljena na Shannonovoj entropiji je:*

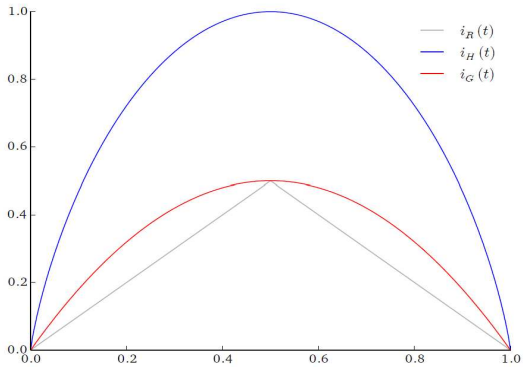
$$i_H(t) = -\sum_{k=1}^{J} \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_t) \log_2(\hat{\mathbb{P}}(c_k|X \in \mathcal{X}_t)) \tag{2.9}$$

Definicija 2.4.8. *Funkcija nečistoće $i_G(t)$ temeljena na Ginijevom indeksu je:*

$$i_G(t) = \sum_{k=1}^{J} \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_t)(1 - \hat{\mathbb{P}}(c_k|X \in \mathcal{X}_t)) \tag{2.10}$$

Primjer 2.4.9. *Izračunajmo sada smanjenje nečistoća temeljenih na Shanonovoj entropiji i Ginijevom indeksu za podatke iz primjera 2.4.4. Prvo, imamo* $i_H(t_0) = -\frac{2}{10}\log_2(\frac{2}{10}) - \frac{8}{10}\log_2(\frac{8}{10}) = 0,72$ *te* $i_G(t_0) = \frac{2}{10}(1 - \frac{2}{10}) + \frac{8}{10}(1 - \frac{8}{10}) = 0,32$. *Za dijeljenje obzirom na* X_1, *dobivamo* $i_H(t_1) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) = 0,97$ *i* $i_H(t_2) = 0$ *te* $i_G(t_1) = \frac{2}{5}(1 - \frac{2}{5}) + \frac{3}{5}(1 - \frac{3}{5}) = 0,48$, $i_G(t_2) = 0$, *stoga dobivamo da smanjenja nečistoća iznose* $\Delta i_H(t_0) = i_H(t_0) - \frac{1}{2}i_H(t_1) - \frac{1}{2}i_H(t_2) = 0,72 - \frac{1}{2}0,97 - \frac{1}{2}0 = 0,235$ *i* $\Delta i_G(t_0) = i_G(t_0) - \frac{1}{2}i_G(t_1) - \frac{1}{2}i_G(t_2) = 0,32 - \frac{1}{2}0,48 - \frac{1}{2}0 = 0,08$. *Analognim računom, dobijemo da smanjenja nečistoća za druga dva dijeljenja iznose* $\Delta i_H(t_0) = 0,17$ *i* $\Delta i_G(t_0) = 0,05$ *za dijeljenje s obzirom na* X_2 *te* $\Delta i_H(t_0) = 0,005$ *i* $\Delta i_H(t_0) = 0,003$, *za dijeljenje s obzirom na* X_3. *Ovim računom smo pokazali da su iznosi smanjenja nečistoća na temelju Shanonove entropije i Ginijevog indeksa najveći pri dijeljenju s obzirom na* X_1, *čime možemo naslutiti da su imune na nedostatke koje ima funkcija nečistoće na uzorku za učenje.*

Na donjoj slici možemo vidjeti da $i_H(t)$ i $i_G(t)$ zadovoljavaju uvjete teorema 2.4.5, dok $i_R(t)$ nije strogo konkavna, što ju ne čini dobrom mjerom nečistoće. Nečistoća bazirana na entropiji $i_H(t)$ kvantificira nesigurnost Y unutar čvora t. Uz ovaj kriterij, smanjenje nečistoće $\Delta i_H(s,t)$ također je poznato kao informacijski dobitak te predstavlja informaciju stečenu o Y dijeljenjem t na t_L i t_R. Nečistoća na bazi Ginija $i_G(t)$ mjeri koliko bi često nasumično odabrani objekt $\mathbf{x} \in \mathcal{L}_t$ bio netočno klasificiran da je nasumično označen klasom $c \in \mathcal{Y}$ obzirom na distribuciju $\hat{\mathbb{P}}(y|X \in \mathcal{X}_t)$.



Slika 2.5: Understanding Random Forests: From Theory to Practice - usporedba mjera nečistoće.

Idući primjer ilustrira razliku među ovim mjerama nečistoće te pokazuje zašto se u praksi koriste funkcije nečistoće temeljene na Shannonovoj entropiji i Ginijevom indeksu, umjesto funkcije nečistoće na temelju procjene na uzorku za učenje.

Primjer 2.4.10. *Nečistoće bazirane na entropiji i Gini indeksu su osjetljivije na promjene u vjerojatnosti unutar čvora, nego nečistoća bazirana na testnom skupu. Naime, razmotrimo problem s dvije klase i 400 opservacija u svakoj klasi (oznaka (400,400)). Iznosi nečistoća u korijenu su:* $i_R(t) = 1 - \max\{\frac{1}{2}, \frac{1}{2}\} = \frac{1}{2}$, $i_H(t) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1$, $i_G(t) = \frac{1}{2}(1 - \frac{1}{2}) + \frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{2}$. *Usporedimo sada dva različita binarna dijeljenja, takva da kod prvog dijeljenja dobijemo čvorove s (300,100) i (100,300) podataka, a u drugom (200,400) i (200,0). Kod oba dijeljenja, nečistoća na uzorku za učenje se smanjuje za isti iznos* ($\frac{1}{2}\frac{1}{4} + \frac{1}{2}\frac{1}{4} = \frac{3}{4}\frac{1}{3} + \frac{1}{4}\frac{1}{2} = 0.25$)*, ali u drugom slučaju imamo čisti čvor, gdje su svi uzorci jedne klase sadržani u lijevom djetetu i intuitivno bi takvo dijeljenje preferirali. Pogledajmo sada koji su iznosi smanjenja nečistoća baziranih na entropiji i Gini indeksu. Kod prvog dijeljenja iznos nečistoće baziran na entropiji za lijevo dijete iznosi* $i_H(t) = -\frac{3}{4}\log_2(\frac{3}{4}) - \frac{1}{4}\log_2(\frac{1}{4}) = 0,81$*, dok je iznos nečistoće za lijevo dijete temeljen na Gini indeksu* $i_G(t) = \frac{3}{4}(1 - \frac{3}{4}) + \frac{1}{4}(1 - \frac{1}{4}) = \frac{3}{8} = 0,375$. *Analognim računom za desno dijete, dobivamo da je* $\Delta i_H(s,t) = 1 - \frac{1}{2}0,81 - \frac{1}{2}0,81 = 0.19$*, a* $\Delta i_G(s,t) = \frac{1}{2} - \frac{1}{2}\frac{3}{8} - \frac{1}{2}\frac{3}{8} = 0.125$. *Pri drugom dijeljenju, imamo* $\Delta i_H(s,t) = 1 - \frac{3}{4}0,92 - \frac{1}{4}0 = 0.31$ *te* $\Delta i_G(s,t) = \frac{1}{2} - \frac{3}{4}\frac{4}{9} - \frac{1}{4}0 = \frac{1}{6} = 0,16$. *Vidimo da su nečistoće bazirane na entropiji i Gini indeksu prepoznale da je drugo dijeljenje bolje i rezultirale većim smanjenjem nečistoće u drugom slučaju.*

Za kraj ovog pododjeljka pogledajmo na detaljnom primjeru kako funkcionira algoritam u problemu klasifikacije.

Primjer 2.4.11. *Razmotrimo problem klasifikacije rezultata na ispitu na temelju broja sati učenja. Klase rezultata su: "Nizak/Srednji/Visok". Kao kriterij dijeljenja koristit ćemo funkciju nečistoće temeljenu na Gini indeksu te zadajemo maksimalnu dubinu stabla 2 kao kriterij zaustavljanja. Dani su nam sljedeći podaci:*

Podatak	*Broj sati*	*Rezultat*
1	*6*	*Visok*
2	*3*	*Srednji*
3	*5*	*Srednji*
4	*2*	*Nizak*
5	*7*	*Visok*
6	*4*	*Srednji*
7	*3*	*Nizak*
8	*8*	*Visok*
9	*5*	*Srednji*
10	*2*	*Nizak*

Sada je potrebno odabrati optimalno dijeljenje. Kako je traženje maksimalnog smanjenja nečistoće ekvivalentno minimizaciji težinske sume nečistoće djece, izračunajmo potonje za različite točke dijeljenja:

- ***Dijeljenje za Broj sati*** ≤ 2 : *Dobivamo skupove (4,10) kada se dva puta javlja klasa Nizak te (1,2,3,5,6,7,8,9) kada se jednom javlja klasa Nizak, 4 puta Srednji te 3 puta Visok. Iznosi nečistoća su* $i_{G_L} = \frac{2}{2}0 = 0$ *te* $i_{G_R} = \frac{1}{8}\frac{7}{8} + \frac{4}{8}\frac{4}{8} + \frac{3}{8}\frac{5}{8} = 0,59$. *Težinska suma nečistoće djece je* $\frac{1}{5}i_{G_L} + \frac{4}{5}i_{G_R} = 0 + \frac{4}{5}0,59 = 0,47$.

- ***Dijeljenje za Broj sati*** ≤ 3 : *Dobivamo podjelu na skupove (2,4,7,10) te (1,3,5,6,8,9) s težinskom sumom nečistoće djece* $\frac{4}{10}0,38 + \frac{6}{10}0,5 = 0,45$.

- ***Dijeljenje za Broj sati*** ≤ 4 : *Dobivamo podjelu na skupove (2,4,6,7,10) te (1,3,5,8,9) s težinskom sumom nečistoće djece* $\frac{1}{2}0,48 + \frac{1}{2}0,36 = 0,42$.

- ***Dijeljenje za Broj sati*** ≤ 5 : *Dobivamo podjelu na skupove (2,3,4,6,7,9,10) te (1,5,8) s težinskom sumom nečistoće djece* $\frac{7}{10}0,49 + \frac{3}{10}0 = 0,34$.

- ***Dijeljenje za Broj sati*** ≤ 6 : *Dobivamo podjelu na skupove (1,2,3,4,6,7,9,10) te (5,8) s težinskom sumom nečistoće djece* $\frac{4}{5}0,59 + 0 = 0,47$.

- ***Dijeljenje za Broj sati*** ≤ 7 : *Dobivamo podjelu na skupove (1,2,3,4,5,6,7,9,10) te (8) s težinskom sumom nečistoće djece* $\frac{9}{10}0,64 + 0 = 0,57$.

Stoga, možemo zaključiti da je najniži iznos težinske sume nečistoće djece, a samim time i maksimalan iznos smanjenja nečistoće upravo za Broj sati ≤ 5. *S obzirom na to da smo zadali maksimalnu dubinu stabla 2, ovdje algoritam staje. Primijetimo kako je ovaj rezultat i očekivan jer se u desnom djetetu nalazi samo klasa Visokih rezultata.*

2.5 Regresija

Sada prelazimo na problem regresije, kada je funkcija gubitka L kvadratna pogreška. Želimo odrediti:

$$\hat{y}_t^* = \arg\min_{\hat{y}\in\mathcal{Y}} \mathbb{E}_{X,Y|t}\{(Y-\hat{y})^2\} = \mathbb{E}_{X,Y|t}Y. \tag{2.11}$$

Kako nam nije poznata vjerojatnosna distribucija $\mathbb{P}(X, Y)$, koristimo aproksimaciju:

$$\hat{y}_t = \arg\min_{c\in\mathcal{Y}} \frac{1}{N_t}\sum_{(\mathbf{x},y)\in\mathcal{L}_t}(y-c)^2 = \frac{1}{N_t}\sum_{(\mathbf{x},y)\in\mathcal{L}_t} y. \tag{2.12}$$

Analognim raspisom kao u problemu klasifikacije, može se dobiti da $\hat{y}_t$ definiran kao u (2.12) minimizira grešku treniranja, a ne stvarnu pogrešku generalizacije. Također vrijedi da dodatnim dijeljenjem terminalnog čvora smanjujemo procjenu na uzorku za učenje $\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi)$, kao što je pokazano u narednoj propoziciji.

Propozicija 2.5.1. *Za bilo koju nepraznu podjelu terminalnog čvora $t \in \widetilde{\varphi}$ na t_L i t_R, što rezultira novim stablom φ' gdje su pritom $\hat{y}_{t_L}$ i $\hat{y}_{t_R}$ definirani kao u 2.12, vrijedi:*

$$\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi) \geq \widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi').$$

Jednakost vrijedi za $\hat{y}_t = \hat{y}_{t_L} = \hat{y}_{t_R}$.

Dokaz.

$$\begin{aligned}
\widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi) &\geq \widehat{\mathrm{Err}}^{\mathrm{train}}(\varphi') \\
\sum_{t\in\widetilde{\varphi}} \hat{\mathbb{P}}(X\in\mathcal{X}_t)\Big(\frac{1}{N_t}\sum_{(\mathbf{x},y)\in\mathcal{L}_t}(y-\hat{y}_t)^2\Big) &\geq \sum_{t\in\widetilde{\varphi'}} \hat{\mathbb{P}}(X\in\mathcal{X}_t)\Big(\frac{1}{N_t}\sum_{(\mathbf{x},y)\in\mathcal{L}_t}(y-\hat{y}_t)^2\Big) \\
\sum_{(\mathbf{x},y)\in\mathcal{L}_t}(y-\hat{y}_t)^2 &\geq \sum_{(\mathbf{x},y)\in\mathcal{L}_{t_L}}(y-\hat{y}_{t_L})^2 + \sum_{(\mathbf{x},y)\in\mathcal{L}_{t_R}}(y-\hat{y}_{t_R})^2 \\
N_t\hat{y}_t^2 &\leq N_{t_L}\hat{y}_{t_L}^2 + N_{t_R}\hat{y}_{t_R}^2 \\
\frac{1}{N_t}\Big(\sum_{(\mathbf{x},y)\in\mathcal{L}_t} y\Big)^2 &\leq \frac{1}{N_{t_L}}\Big(\sum_{(\mathbf{x},y)\in\mathcal{L}_{t_L}} y\Big)^2 + \frac{1}{N_{t_R}}\Big(\sum_{(\mathbf{x},y)\in\mathcal{L}_{t_R}} y\Big)^2
\end{aligned}$$

Radi jednostavnosti, označimo $s(t) = \sum_{(\mathbf{x},y)\in\mathcal{L}_t} y$. Primijetimo da je tada $s(t) = s(t_L) + s(t_R)$, stoga imamo:

$$\begin{aligned}
\frac{s(t)^2}{N_t} &\leq \frac{s(t_L)^2}{N_{t_L}} + \frac{s(t_R)^2}{N_{t_R}} \\
\frac{(s(t_L)+s(t_R))^2}{N_{t_L}+N_{t_R}} &\leq \frac{s(t_L)^2}{N_{t_L}} + \frac{s(t_R)^2}{N_{t_R}} \\
\frac{s(t_L)^2 N_{t_R}(N_{t_L}+N_{t_R}) + s(t_R)^2 N_{t_L}(N_{t_L}+N_{t_R}) - (s(t_L)+s(t_R))^2 N_{t_L} N_{t_R}}{N_{t_L} N_{t_R}(N_{t_L}+N_{t_R})} &\geq 0 \\
\frac{s(t_L)^2 N_{t_R}^2 + s(t_R)^2 N_{t_L}^2 - 2s(t_L)s(t_R)N_{t_L}N_{t_R}}{N_{t_L} N_{t_R}(N_{t_L}+N_{t_R})} &\geq 0 \\
\frac{(s(t_L)N_{t_R} - s(t_R)N_{t_L})^2}{N_{t_L} N_{t_R}(N_{t_L}+N_{t_R})} &\geq 0,
\end{aligned}$$

što vrijedi, obzirom da su i brojnik i nazivnik nenegativni. Za $\hat{y}_t = \hat{y}_{t_L} + \hat{y}_{t_R}$, po 2.12 imamo

$$\begin{aligned}
\frac{1}{N_t} \sum_{(\mathbf{x},y)\in\mathcal{L}_t} y &= \frac{1}{N_{t_L}} \sum_{(\mathbf{x},y)\in\mathcal{L}_{t_L}} y + \frac{1}{N_{t_R}} \sum_{(\mathbf{x},y)\in\mathcal{L}_{t_R}} y \\
\frac{s(t)^2}{N_t} &= \frac{s(t_L)^2}{N_{t_L}} + \frac{s(t_R)^2}{N_{t_R}}.
\end{aligned}$$

□

Što se tiče pravila dijeljenja, kada je izlazna varijabla Y kvantitativna, smanjenje kvadratne greške na skupu za treniranje je pozitivno dokle god su vrijednosti pridružene potomcima različite od vrijednosti u čvoru t. Naspram klasifikacije, to se rijetko događa (kada se srednje izlazne vrijednosti na potomcima podudaraju sa srednjom izlaznom vrijednosti u t). Zato su procjene na uzorku za učenje kod regresije osjetljive na promjene u (srednjim vrijednostima) a posteriori distribucija, čak i ako su one samo male. Kao rezultat, lokalna procjena na uzorku za učenje ne pokazuje nepoželjna svojstva koja smo imali kod klasifikacije i zapravo predstavlja dobar kriterij za regresiju.

Definicija 2.5.2. *U regresiji, funkcija nečistoće $i_R(t)$ na temelju procjene na uzorku za učenje definirna na kvadratu gubitka pogreške:*

$$i_R(t) = \frac{1}{N_t} \sum_{(\mathbf{x},y)\in\mathcal{L}_t} (y - \hat{y}_t)^2. \tag{2.13}$$

Za kraj ovog potpoglavlja dajemo primjer regresijskog stabla na problemu predviđanja cijene nekretnine s obzirom na njenu površinu.

Primjer 2.5.3. *Kao kriterij zaustavljanja zadajmo da čvor postaje terminalan ako ima 3 ili manje elemenata. Dani su nam sljedeći podaci o površini stanova u kvadratnim metrima te cijeni u tisućama eurima:*

Podatak	*Površina*	*Cijena*
1	*140*	*200*
2	*157*	*250*
3	*176*	*220*
4	*185*	*300*
5	*195*	*280*
6	*213*	*330*

Kao čvorove dijeljenja uzmimo aritmetičke sredine susjednih površina $148, 166, 180, 190$ *i* 204 *metara kvadratnih te pronađimo optimalno dijeljenje.*

- ***Dijeljenje za Površina stana*** ≤ 148 : *Dobivamo skupove (1) te (2,3,4,5,6). Iznosi nečistoća su* $i_{R_L} = \frac{1}{1}(200-200)^2 = 0$ *te* $i_{R_R} = \frac{1}{5}((250-276)^2 + (220-276)^2 + (300-276)^2 + (280-276)^2 + (330-276)^2) = 1464$*, jer je u lijevom djetetu* $\hat{y}_{t_L} = \frac{1}{1}200$*, a u desnom djetetu* $\hat{y}_{t_R} = \frac{1}{5}(250+220+300+280+330)$*. Težinska suma nečistoće djece je* $\frac{1}{6}i_{R_L} + \frac{5}{6}i_{R_R} = \frac{1}{6}0 + \frac{5}{6}1464 = 1220$.
- ***Dijeljenje za Površina stana*** ≤ 166 : *Dobivamo podjelu na skupove (1,2) te (3,4,5,6) s težinskom sumom nečistoće djece* $\frac{1}{3}625 + \frac{2}{3}1618,5 = 1287,3$.
- ***Dijeljenje za Površina stana*** ≤ 180 : *Dobivamo podjelu na skupove (1,2,3) te (4,5,6) s težinskom sumom nečistoće djece* $\frac{1}{2}422,2 + \frac{1}{2}422,2 = 422,2$.
- ***Dijeljenje za Površina stanja*** ≤ 190 : *Dobivamo podjelu na skupove (1,2,3,4) te (5,6) s težinskom sumom nečistoće djece* $\frac{2}{3}1417,45 + \frac{1}{3}625 = 1153,3$.
- ***Dijeljenje za Površina stanja*** ≤ 204 : *Dobivamo podjelu na skupove (1,2,3,4,5) te (6) s težinskom sumom nečistoće djece* $\frac{5}{6}1360 + \frac{1}{6}0 = 1133,3$.

Možemo vidjeti da je najmanji težinski zbroj nečistoća djece, a samim time maksimalno smanjenje nečistoće upravo za dijeljenje pri površini stana od 180 *metara kvadratnih. Kako se u lijevom i desnom djetetu nalazi po tri podatka, algoritam se zaustavlja.*

2.6 Pravila zaustavljanja

Kao što smo dokazali u propozicijama 2.4.1 i 2.5.1, što je stablo odlučivanja dublje, manja je procjena na uzorku za učenje. Međutim, kao što smo ranije spomenuli, povećanje složenosti modela dovodi do overfittinga. Iz tog razloga potrebno je napraviti kompromis te

pronaći stablo koje neće biti ni preplitko ni preduboko. Najprije navodimo kriterije zaustavljanja kada je čvor t neizbježno postavljen kao terminalni čvor, kada se $\mathcal{L}_t$ ne može više podijeliti:

1. Kada su izlazne vrijednosti uzoraka u $\mathcal{L}_t$ homogene. Preciznije, ako je $y = y'$ za sve $(\mathbf{x}, y), (\mathbf{x}', y') \in \mathcal{L}_t$. Ovaj uvjet je trivijalno zadovoljen kada je $N_t = 1$.

2. Kada su ulazne varijable X_j lokalno konstantne u $\mathcal{L}_t$, to jest ako je $x_j = x'_j$ za sve $(\mathbf{x}, y), (\mathbf{x}', y') \in \mathcal{L}_t$, za svaku ulaznu varijablu X_j. U ovom slučaju nije moguće podijeliti $\mathcal{L}_t$ u dva ili više neprazna podskupa.

Kako bismo spriječili overfitting, koriste se razne heuristike koje zaustavljaju rekurzivnu particiju poput:

- Zadavanja minimalnog broja uzoraka po čvoru $N_{\min}$;
- Zadavanja maksimalne dubine stabla $d_{\max}$;
- Ako je ukupno smanjenje nečistoće manje od zadanog praga β, to jest $\mathbb{P}(X \in \mathcal{X}_t)\Delta i(s^*, t) \leq \beta$;
- Ako ne postoji dijeljenje takvo da čvorovi t_L i t_R zajedno imaju barem N_{leaf} uzoraka.

Poglavlje 3

Primjena

3.1 Uvod

U ovom poglavlju proučavamo primjenu CART algoritma u igri *Heads-up No-limit* pokera. Prvo opisujemo pravila igre i definiramo osnovne pojmove teorije igara potrebne za razumijevanje minimizacije protučinjeničnog žaljenja (counterfactual regret minimization).

3.2 Pravila igre

Heads-Up No-Limit Texas Hold'em (HUNL) je izuzetno popularna varijanta pokera koja uključuje dva igrača i standardan špil od 52 karte. U ovom formatu, oba igrača započinju s jednakim iznosom od 20.000 dolara. Na početku svake ruke, svaki igrač dobiva po dvije privatne karte iz špila. Osim toga, prvi igrač mora uložiti 50 dolara u pot, dok drugi igrač ulaže 100 dolara.
Igra započinje prvim krugom klađenja, poznatim kao *Pre-flop*, pri čemu prvi igrač ima pravo prvi donijeti odluku. Tijekom svakog kruga klađenja, igrači imaju nekoliko opcija. Mogu odlučiti *pozvati* (call-izjednačiti ulog prethodnog igrača), *odustati* (fold-napustiti ruku i izgubiti ulog), provjeriti (check-nastaviti igru bez ulaganja dodatnog novca) ili podići ulog (dodati više novca u pot). Važno je napomenuti da pri podizanju uloga, igrač mora izjednačiti ulog prethodnog igrača i dodati dodatni iznos, pri čemu je minimalno povećanje 100 dolara. Također, igrač ne može uložiti više novca nego što trenutno posjeduje na svom računu.
Nakon završetka prvog kruga klađenja, slijedi druga faza poznata kao *Flop*, tijekom koje se na stol izlažu tri javne karte. To se ponavlja u još dvije faze - *Turn* i *River* - pri čemu se u svakoj fazi dodaje po jedna javna karta na stol.
Ako se dosegne završna faza igre - *River*, pobjednik se određuje na temelju najbolje kombinacije od pet karata. Ova kombinacija uključuje dvije privatne karte svakog igrača i tri

javne karte na stolu. U rijetkom slučaju kada imaju jednako dobre ruke, novac se ravnomjerno dijeli između oba igrača.
Za igrače koji žele odigrati više ruku, uloge se izmjenjuju između prvog i drugog igrača, te se saldo svakog igrača vraća na početni iznos od 20.000 dolara pri svakom početku nove ruke.

3.3 Struktura podataka

Kao što je običaj u strojnom učenju, ključno je kako algoritmu prikazujemo podatke. Da bismo transformirali trenutno stanje igre u oblik pogodan za obradu, koristimo činjenicu da se kompleksno stanje igre pokera može razložiti na dvije glavne komponente: karte u rukama svakog igrača (koje su podijeljene na privatne karte igrača i javno vidljive karte), te niz dosadašnjih klađenja svakog igrača.
Za uspješno prikazivanje stanja igre, koristimo kombinaciju dvaju vektora – jedan predstavlja karte, a drugi povijest klađenja. Na kraju, te dvije reprezentacije jednostavno sjedinjujemo kako bismo dobili konačni vektor koji dočarava trenutno stanje igre. Donja slika prikazuje kako je ovakav prikaz izveden za početak igre prije flopa.

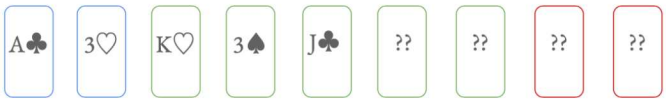
Feature 1	Features 2-11	Features 12-21	Features 22-31	Features 32-35
Current Hand's Win-Rate	Hand Win-Rate 1-Round in the Future	Hand Win-Rate 2-Rounds in the Future	Hand Win-Rate 3-Rounds in the Future	Opponent Bets, per Round

Slika 3.1: World-class Interpretable Poker - Vizualizacija značajki za ruku prije flopa. To uključuje ukupno 35 značajki: 4 se odnose na povijest klađenja i 31 koja se odnosi na privatne i javne karte.

Sada uvodimo pristup predstavljanja javnih i privatnih karata vidljivih igraču kao skup interpretabilnih značajki. Za početak, stopu pobjede pokeraške ruke definiramo kao vjerojatnost da će pobijediti u odnosu na uniformno izvlačenje svih preostalih javnih karata i protivnikovih privatnih karata. S obzirom na ovu definiciju vjerojatnosti pobjede, koristimo ovaj broj kao prvu značajku u našem predstavljanju karata u bilo kojem stanju igre.
Prirodno proširenje ove definicije je izračunavanje stope dobitka u budućim rundama igre pokera. Pri tome bismo uzeli u obzir informacije o potencijalnom poboljšanju ili pogoršanju ruke igrača kako se runde odvijaju. Kako bismo to postigli za buduće runde ($n \geq 1$), sistematično bismo istražili svako moguće otkrivanje javnih karata tijekom sljedećih n rundi. Nakon toga, utvrdili bismo odgovarajuće stope uspjeha za svaki od ovih scenarija koristeći prethodnu definiciju stope uspjeha. Valja napomenuti da, u ovom kontekstu, stopa uspjeha n rundi unaprijed više nije samostalna brojka. Umjesto toga, poprima oblik distribucije stopa pobjeda - gdje svaka točka podataka unutar distribucije odražava stopu

uspjeha mogućeg razvoja karata tijekom n rundi u budućnosti. Sljedeći primjer ilustrira ovakvo razmatranje.

Primjer 3.3.1. *Pretpostavimo da se igra nalazi u stanju Flop, tako da su poznate 3 javne karte, kao na donjoj slici. Naše privatne karte su* A♣ *i* 3♡*, dok su javne karte* K♡*,* 3♠ *i* J♣*.*



Slika 3.2: World-Class Interpretble Poker - Vizualizacija izračunavanja vjerojatnosti pobjede naše ruke, u trenutku Flopa. Plavo obrubljene karte predstavljaju našu privatnu ruku, teleno obrubljene karte predstavljaju javne karte, a crvene predstavljaju protivnikovu privatnu ruku.

Cilj je simulirati sva moguća otkrivanja 4 *nepoznate karte* $\binom{47}{2}\binom{45}{2}$ *računajući broj događaja u kojima završavamo s boljom rukom od našeg protivnika i dijeleći s brojem mogućih raspleta. Da bismo izračunali stopu pobjede u sljedećoj rundi, nabrojali bismo svih mogućih 47 otkrivanja četvrte javne karte, izračunali stopu pobjede za svaki od tih 47 scenarija i pohranili te stope pobjeda u distribuciju stopa pobjeda. Konačno, da bismo dobili stopu pobjeda nakon 2 runde, nabrojali bismo svih mogućih* $\binom{47}{2} = 1081$ *otkrivanja četvrte i pete javne karte, izračunali stopu pobjeda za svaki od tih 1081 događaja te ih spremili u distribuciju stopa pobjeda.*

Prirodno rješenje za integriranje svih ovih informacija u reprezentaciju određene ruke podrazumijeva uključivanje njezine trenutne stope pobjeda, kao i stopa pobjeda u svim budućim rundama. S obzirom na to da distribucije stopa pobjeda za buduće runde obuhvaćaju brojne vrijednosti, uzet ćemo decile iz svake distribucije i koristimo te vrijednosti kao značajke. Stoga naša reprezentacija karata za ruku prije pojave Flopa sadržava 31 vrijednost - jednu za trenutnu stopu pobjeda i 10 decilnih vrijednosti za svaku od 3 distribucije stopa pobjeda u svakoj od 3 buduće runde klađenja. Analogno tome, reprezentacija karata za ruku prije pojave Turna sastoji se od 11 vrijednosti - jednu za trenutnu stopu pobjeda i 10 decilnih vrijednosti za distribuciju stopa pobjeda posljednje runde klađenja.
Drugi dio oblikovanja značajki uključuje analizu podataka o klađenju tijekom partije pokera. Očito je važno sačuvati ovu vrstu informacija kako bi igrač bolje mogao shvatiti dinamiku igre te posebno naučiti prepoznavati obrasce blefiranja temeljene na načinima na koje protivnik ulaže uloge. Kako bismo to postigli, zabilježavamo iznose uloga protivnika pomoću četiri broja, svaki od njih predstavlja količinu novca koju je protivnik uložio u jednom od četiri kruga klađenja. Također, valja napomenuti da ako se određeni krug igre još nije odigrao (primjerice, River ako se trenutno nalazimo u Pre-Flop fazi), tada je zabilježena uloga protivnika za taj krug postavljena na 0.

3.4 Minimizacija protučinjeničnog žaljenja

Za početak, definiramo osnovne pojmove teorije igara korištene u nastavku ovog potpoglavlja.
Strategija je unaprijed određeni "program igre" koji govori koje radnje treba poduzeti kao odgovor na svaku moguću strategiju koju drugi igrači mogu koristiti. Neka je n prirodan broj. Skup $N = \{1, ..., n\}$ nazivamo *skupom igrača* dok je *strateška igra s n igrača* uređeni par (A,u), koji se sastoji od:

- Skupa strateških profila: Za svaki $i \in N$, neka je $A_i \neq \emptyset$ skup strategija igrača i. Skup $A = A_1 \times ... \times A_n$ nazivamo skupom strateških profila.
- Funkcije korisnosti: Za svaki $i \in N$, neka je $u_i : A \mapsto \mathbb{R}$ funkcija korisnosti igrača i. Funkciju $u = (u_1, ..., u_n) : A \mapsto \mathbb{R}^n$ nazivamo funkcijom korisnosti.

Nashov ekvilibrij strateške igre (A, u) s n igrača je strateški profil $a^* \in A$ za koji vrijedi da je za svaki $i \in N$, $u_i(a^*) \geq u_i(a^*_{-i}, a_i)$ za svaki $a_i \in A_i$. *Ekstenzivna igra s n igrača* je uređena šestorka $\Gamma = (T, P, \mathcal{H}, C, \tau, u)$ takva da

- $T = (X, E)$ je usmjereno stablo s korijenom x_0 i skupom završnih vrhova Z.
- $P : X \backslash Z \mapsto N \cup \{0\}$ je funkcija koja svakom nezavršnom vrhu pridružuje ili igrača ili Prirodu (koju označavamo kao igrača 0). U prvom slučaju taj ćemo vrh nazivati vrhom odluke, a u drugom slučajnim vrhom.
- $\mathcal{H} = (H_i)_{i \in N}$ svakom igraču i pridružuje particiju skupa $P^{-1}(i)$, odnosna vrhova odluke u kojima je taj igrač na potezu. Skupove $h \in H_i$ nazivamo skupovima informacija igrača i te za svaki takav skup h pretpostavljamo da sa svakim putom u T dijeli najviše jedan vrh, te da svi vrhovi u h imaju jednak broj izlaznih bridova.
- $C = (C_h)_{h \in H}$, pri čemu je $H = \bigcup_{i \in N} H_i$, svakom $h \in H$ pridružuje particiju skupa bridova koji izlaze iz vrhova u h. Particija C_h je takva da za svaki vrh $x \in h$ i svaki $c \in C_h$, c sadrži točno jedan brid koji izlazi iz x. Skupove $c \in C_h$ nazivamo potezima u h. Pretpostavljamo da je $|C_h| \geq 2$ za sve h.
- $\tau = (\tau_x)_{x \in P^{-1}(0)}$ svakom slučajnom vrhu pridružuje vjerojatnosnu distribuciju na skupu bridova koji iz njega izlaze. Pretpostavljamo da su svim bridovima dane pozitivne vjerojatnosti.
- $u = (u_i)_{i \in N}$ svakom igraču i pridružuje funkciju korisnosti $u_i : Z \mapsto \mathbb{R}$.

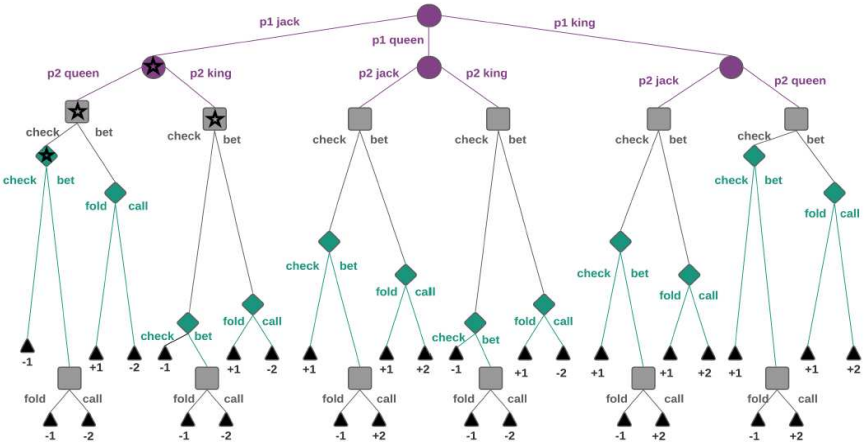
Za ekstenzivnu igru reći ćemo da je s *potpunim informacijama* ako je za sve igrače i particija H_i diskretna, odnosno ako za sve $h \in H_i$ vrijedi $|h| = 1$. U suprotnom ćemo reći da se radi o igri s *nepotpunim informacijama*. *Strateška igra pridružena ekstenzivnoj igri* Γ je strateška igra $(A_1 \times ... \times A_n, (u_1, ..., u_n))$, pri čemu je, za svaki i, A_i skup čistih strategija igrača i u ekstenzivnoj igri Γ, a u_i funkcija korisnosti igrača i definirana na sljedeći način. Za proizvoljni strateški profil $a \in A_1 \times ... \times A_n$, neka je p_a vjerojatnosna distribucija na skupu završnih vrhova Z koju taj strateški profil inducira – drugim riječima, $p_a(z)$ je vjerojatnost da će igra, prateći strateški profil a, završiti u završnom vrhu z. Tada je

$$u_i(a) = \sum_{z \in Z} p_a(z) u_i(z).$$

(Čisti) Nashov ekvilibrij ekstenzivne igre je Nashov ekvilibrij njoj pridružene strateške igre. Koristeći prethodne definicije, vidimo da je HUNL poker primjer ekstenzivne igre s nepotpunim informacijama. Kao takav možemo ga prikazati u obliku stabla, gdje čvorovi stabla odgovaraju raznim stanjima igre, a bridovi koji izlaze iz tih čvorova predstavljaju sve moguće poteze koje igrač može napraviti. Formalno, s H označimo skup svih mogućih stanja igre te $h \in H$ predstavlja neko konkretno stanje. Nadalje, sa $Z \subset H$ označimo skup završnih čvorova tako da $z \in Z$ predstavlja konkretan završni čvor. Svaki završni čvor povezan je s isplatom za svakog igrača i, koja je određena funkcijom $u_i : Z \mapsto \mathbb{R}$. Drugim riječima, $u_i(z)$ označava isplatu za igrača i u terminalnom stanju z.
Nadalje, definiramo strategiju σ_i za svakog pojedinog igrača kao vjerojatnosnu distribuciju za svaki mogući korak koji igrač može napraviti u svakom čvoru igre. To podrazumijeva dodjelu vjerojatnosti svakoj grani u svakom čvoru gdje je igrač i na redu. Primjerice, ako određena strategija dodjeljuje vjerojatnost $0,5$ za odustajanje (fold) i $0,5$ za provjeru (check) u čvoru h, igrač će odustati 50% vremena i provjeriti 50% vremena te nikada neće poduzeti bilo koju drugu potencijalnu akciju. Na ovaj način, precizno definiranje strategije σ_i potpuno utječe na način na koji će igrač i postupati u svakoj situaciji u igri. Profil strategija σ predstavlja skup strategija, pri čemu svaki igrač ima svoju strategiju u igri.
Zbog prirode igre s nepotpunim informacijama u HUNL pokeru, određeni čvorovi igre mogu se percipirati kao istovjetni iz perspektive određenog igrača. Na primjer, zamislimo da je na početku igre igraču 1 podijeljen $A\spadesuit$ i $K\spadesuit$, dok igraču 2 ide $2\spadesuit$ i $3\spadesuit$. Da je igraču 2 umjesto toga podijeljeno $5\clubsuit$ i $7\spadesuit$, obje situacije bile bi iste iz perspektive igrača 1, budući da igrač 1 ne vidi karte igrača 2. Stoga, iako bi oba scenarija odgovarala dvama različitim čvorovima igre, igrač 1 ne bi mogao razlikovati ta dva scenarija. Zbog toga definiramo informacijski skup (skraćeno: infoset) I_i kao skup koji obuhvaća sve čvorove igre koji su nediferencirani međusobno za igrača i. Kroz upravo navedeni primjer, infoset bi sadržavao svih $\binom{50}{2}$ čvorova u kojima je igrač 1 upravo dobio $A\spadesuit$ i $K\spadesuit$ (tako da se svaki čvor unutar tog infoseta razlikuje po kartama koje su podijeljene igraču 2).
Zatim, označimo s $v_i^\sigma(h)$ očekivanu vrijednost budućih nagrada za igrača i u čvoru h kada svi igrači igraju prema σ. Dakle, što je $v_i^\sigma(h)$ veći, očekuje se da će čvor h biti nagrađujući

za igrača i. Slično tome, ovu ideju možemo proširiti na informacijski skup definiranjem očekivane vrijednosti budućih nagrada za igrača i za infoset I_i kao $v_i^\sigma(I_i)$.
Konačno, označimo $\pi^\sigma(h)$ kao vjerojatnost dolaska do čvora h u skladu sa strategijom σ. Za određeni čvor h, ovo se računa kao produkt vjerojatnosti svakog koraka koji vodi od korijena do čvora h. Također, korisno je raščlaniti $\pi^\sigma(h)$ na doprinose svakog sudionika (oba igrača i djelitelja) toj vjerojatnosti. Dakle, označavamo $\pi_i^\sigma(h)$ kao vjerojatnosti koje proizlaze iz odluka igrača i (tj. grana), a $\pi_{-i}^\sigma(h)$ kao vjerojatnosti od svih ostalih sudionika, tako da vrijedi $\pi^\sigma(h) = \pi_i^\sigma(h)\pi_{-i}^\sigma(h)$. Primjerice, za određeni čvor h, $\pi_i^\sigma(h)$ se računa kao produkt vjerojatnosti svake grane koja vodi od korijena do čvora h, u kojoj je igrač i bio na redu; $\pi_i^\sigma(h)$ je jednostavno produkt svih preostalih grana koje vode od korijena do čvora h. Za kraj, za skup informacija I_i, definiramo $\pi^\sigma(I_i) = \sum_{h \in I_i} \pi^\sigma(h)$.
Idući primjer ilustrira uvedene oznake u posebnoj varijanti pokera zvanoj Kuhn poker.

Primjer 3.4.1. *U Kuhn pokeru postoje samo 3 karte - dečko (jack), kraljica (queen) i kralj (king). Također, postoji samo 1 krug klađenja, i svaka akcija igrača je ograničena na provjeru (check), pozivanje (call), odustajanje (fold) ili klađenje 1 dolar. Zbog jednostavnosti ovog oblika pokera, cijelo stablo igre prikazano je idućoj slici.*



Slika 3.3: World-class Interpretable Poker - stablo igre Kuhn pokera. Ljubičasti, sivi i zeleni čvorovi odgovaraju stanjima u kojima su na redu igrač 1, igrač 2 i dijelitelj, redom.

U ovom stablu, svaki čvor predstavlja stanje igre u kojem sudionik (igrač 1, igrač 2 ili djelitelj) treba poduzeti akciju, a grane koje izlaze iz tog čvora predstavljaju akcije. Na primjer, na korijenu vidimo kako djelitelj prvo dijeli kartu igraču jedan - tako se iz korijena protežu 3 grane, po jedna za svaku kartu. Na listovima ovog stabla nalazimo krajnja stanja igre; brojevi koji odgovaraju svakom od tih stanja predstavljaju nagrade funkcije isplate u_1 za igrača 1.

Definiranje strategije σ_i za igrača i zahtijevalo bi dodjeljivanje vjerojatnosti svakoj radnji u svakom stanju igre u kojem je igrač i na redu. Na primjer, za definiranje strategije za igrača 1 dodijelili bismo vjerojatnosti svakoj od sivih grana u stablu igre.
U okviru prethodnog stabla igre, na primjer, $\pi^\sigma(h_1)$ u kojem je h_1 označeni ljubičasti čvor na slici 3.3 bi jednostavno bio jednak vjerojatnosti da je igraču 1 podijeljen dečko (1/3). *Alternativno, $\pi^\sigma(h_2)$ u kojem je h_2 označeni zeleni čvor na slici 3.3 je produkt vjerojatnosti da je igraču 1 podijeljen dečko, igraču 2 podijeljena kraljica, i da je igrač 1 provjerio na lijevom sivom označenom čvoru. Mogli bismo razložiti $\pi^\sigma(h_2)$ na $\pi_1^\sigma(h_2)$ i $\pi_{-1}^\sigma(h_2)$. Prvi bi se izračunavao kao doprinos igrača 1 za $\pi^\sigma(h_2)$, što je vjerojatnost da igrač 1 provjeri na lijevom sivom označenom čvoru; $\pi_{-1}^\sigma(h_2)$ bi se izračunavao kao doprinos svih ostalih sudionika za $\pi^\sigma(h_2)$, što bi u ovom slučaju bila vjerojatnost da je igraču 1 podijeljen dečko i igraču 2 podijeljena kraljica.*

Nakon uvođenja potrebnih definicija i oznaka, spremni smo za opis algoritma minimizacije protučinjeničnog žaljenja. Radi se o iterativnom algoritmu za pronalaženje Nashovog ekvilibrija strateških igara. U suštini, algoritam radi tako da iterativno prolazi kroz stablo igre (u našem slučaju igra igre HUNL pokera) i akumulira žaljenja. Konceptualno, ta su žaljenja numeričke vrijednosti koje predstavljaju koliko žalimo zato što smo poduzeli određenu radnju. Svaka iteracija stabla igre akumulira nova žaljenja, dopuštajući algoritmu da poboljša strateški profil. Na kraju dobivamo da ova strategija konvergira strategiji Nashovog ekvilibrija kroz shemu težinskog uprosječivanja koju detaljno opisujemo u nastavku.
Formalno, neka je t trenutna iteracija algoritma. Prvo definirajmo *trenutno* žaljenje poteza a_i u I_i kao $r^t(I_i, a_i) = \pi_{-i}^{\sigma^t}(I_i)(v_i^{\sigma^t}(I_i, a_i) - v_i^{\sigma^t}(I_i))$, što je zapravo razlika u nagradi između odabiranja poteza a_i umjesto igranja prema strategiji σ^t u I_i, uprosječeno s $\pi_{-i}^{\sigma^t}(I_i)$. Zatim, protučinjenično žaljenje infoseta I_i za akciju a_i u T-toj iteraciji je $R_i^T(I_i, a_i) = T^{-1}\sum_{t=1}^{T} r_i^t(I_i, a_i)$, to jest suma trenutnih žaljenja svih prethodnih iteracija algoritma za taj konkretan infoset i akciju.
Uz ove definicije, CFR algoritam generira strategiju za svaku iteraciju baziranu na akumuliranim žaljenjima. Formalno, strategija igrača i u iteraciji $t + 1$ je

$$\sigma_i^{t+1}(I_i, a_i) = \begin{cases} \frac{\max\{0, R_i^t(I_i, a_i)\}}{\sum_{\alpha_i \in I_i} \max\{0, R_i^t(I_i, \alpha_i)\}}, & \text{ako} \quad \sum_{\alpha_i \in I_i} \max\{0, R_i^t(I_i, \alpha_i)\} > 0 \\ \frac{1}{|\{a_i\}_{a_i \in I_i}|}, & \text{inače} \end{cases}$$

Stoga možemo vidjeti da algoritam odabire akcije za koje žalimo da ih nismo odabrali (one koje bi nam dale veće nagrade).
Sljedeći ovu definiciju, CFR algoritam prelazi cijelo stablo igre za HUNL poker pri svakoj iteraciji, a zatim ažurira i bilježi strategije za oba igrača. Treba napomenuti da ne konvergira niz iterativnih strategija Nashovom ekvilibriju, već njihov prosjek. Formalno, prosjek

strategija $\overline{\sigma_i}^T$ definiramo kao:

$$\overline{\sigma_i}^T(I_i, a_i) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}}{\sum_{t=1}^{T} \pi_i^{\sigma_t}(I_i)}.$$

Primjer 3.4.2. *Razmotrimo igru "kamen, škare, papir" kako bismo ilustrirali kako CFR funkcionira u praksi. Pretpostavimo da su vjerojatnosti akcija prvog igrača (1, 0, 0) (prva, druga i treća vrijednost predstavljaju vjerojatnost igranja kamena, škara i papira), a vjerojatnosti akcija drugog igrača su (0, 0, 1). Korisnosti igrača su dane na slici 3.4. Na temelju ovih postavki vjerojatnosti, očekivani povrat za prvog igrača kada igra kamen bit će:*

$$\mathbb{E}(r_1) = \mathbb{P}(r_2)u_1(r_1 - r_2) + \mathbb{P}(s_2)u_1(r_1 - s_2) + \mathbb{P}(p_2)u_1(r_1 - p_2) = 0 \cdot 50 + 0 \cdot 100 + 1 \cdot 0 = 0,$$

gdje smo s u_1 označili korisnost prvog igrača, r, s, p slučaj kada igrač odigra kamen, škare ili papir redom. Na primjer, $\mathbb{P}(s_2)u_1(r_1 - s_2)$ predstavlja vjerojatnost da drugi igrač odigra škare pomnoženo s korisnosti prvog igrača kada prvi igrač odigra kamen, a drugi škare. Analognim računom kao gore dobijemo da za igranje škara i papira očekivani povrat prvog igrača iznosi 100 i 50, redom. Dakle, trenutni očekivani povrat za prvog igrača bit će:

$$\mathbb{E}(P1) = \mathbb{P}(r_1)\mathbb{E}(r_1) + \mathbb{P}(s_1)\mathbb{E}(s_1) + \mathbb{P}(p_1)\mathbb{E}(p_1) = 1 \cdot 0 + 0 \cdot 100 + 0 \cdot 50 = 0.$$

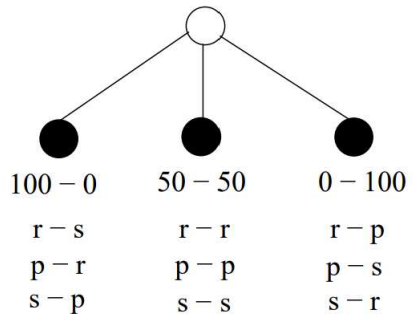
Protučinjenično žaljenje za neigranje škara od strane prvog igrača bit će:

$$R(s_1) = E(s_1) - E(P1) = 100 - 0 = 100.$$

Žaljenje za papir i kamen je analogno $R(p_1) = 50$ $R(r_1) = 50$.
Ažurirane vjerojatnosti akcija za prvog igrača za sljedeću iteraciju bit će

$$(\frac{0}{100 + 50}, \frac{100}{100 + 50}, \frac{50}{100 + 150}) = (0, \frac{2}{3}, \frac{1}{3}).$$

Slični izračuni će biti napravljeni i za drugog igrača, a njegove vjerojatnosti akcija će također biti ažurirane prije sljedeće iteracije.



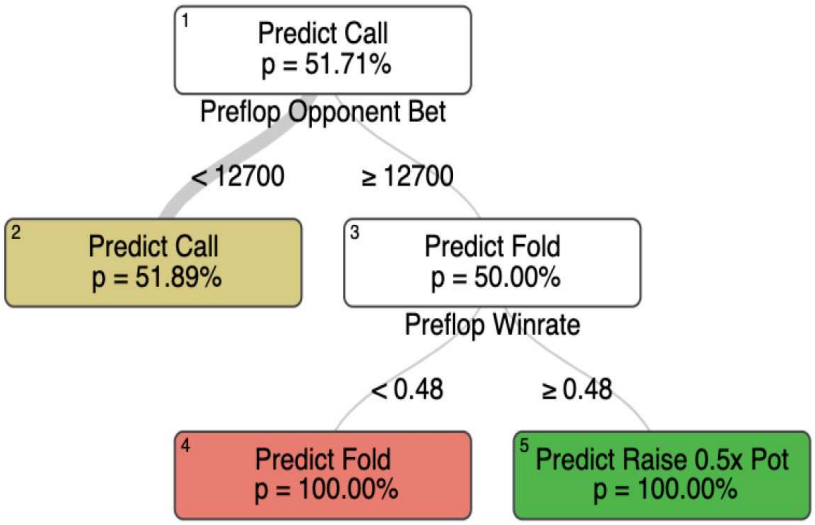
Slika 3.4: Comparing UCT versus CFR in Simultaneous Games - iznosi korisnosti u ovisnosti o akcijama igrača. Prvi terminalni čvor, 100 - 0 znači da prvi igrač ima korisnost 100, a drugi 0 za akcije koje pripadaju tom čvoru.

3.5 Algoritmi za učenje

U nastavku ćemo usporediti tri algoritma za učenje kako bismo naučili prosječnu strategiju koju je pronašao samo-igrajući CFR algoritam. Za sve algoritme učenja, model treniramo na podacima koji odgovaraju mnogim različitim stanjima igre HUNL Pokera predstavljenim putem navedenih interpretabilnih reprezentacija značajki i povezanih oznaka koje određuju radnju koju bi igrač trebao poduzeti u svakom scenariju.
Prvi od tri algoritma je CART algoritam, koji koristimo zbog visoke interpretabilnosti stabala odlučivanja, što je ključno za učenje dobrih strategija od samo-igrajućeg agenta. Glavna mana CART algoritma su vidljiva u dubokim stablima - pohlepna priroda samog algoritma počinje se mučiti s učenjem, zbog velikog broja čvorova koje dijelimo. Zbog toga uvodimo specijaliziranu alternativu, Optimalna stabla klasifikacije (OCT) koja općenito rade znatno bolje od CART-a, a istovremeno zadržavaju svoju interpretabilnost. Napomenimo još da su CART stabla odlučivanja trenirana s maksimalnom dubinom 15 za koju je utvrđeno da algoritam postiže dobru ravnotežu između interpretabilnosti i izvedbe. Iako bi dublja stabla vjerojatno imala bolju izvedbu, interpretabilnost značajno trpi zbog eksponencijalnog povećanja veličine stabla.
Optimalna stabla klasifikacije su algoritam učenja koji koristi tehnike optimizacije cjelobrojnog programiranja za generiranje cijelog stabla odlučivanja u jednom koraku - tako generirajući stablo na nepohlepan način i dopuštajući da se svaka podjela odredi uz puno znanje o drugim dijeljenjima. U ovom radu nećemo ulaziti u detalje problema cjelobrojnog programiranja, ali u suštini radi se o posebnom obliku zadaće linearnog programiranja u kojoj neke (ili sve) varijable poprimaju cjelobrojne vrijednosti. Odlična izvedba i visoka interpretabilnost bit će ključne za OCT-ove interpretacije pokera, jer će jasno prikazivati smislene i dobre poker strategije. Glavni nedostatak ovog algoritma naspram klasičnog CART-a je dulje vrijeme treniranja zbog složenosti optimizacije cjelobrojnog programira-

nja. Sljedeći primjer ilustrira jednostavnost i interpretabilnost OCT stabla za HUNL Poker Pre-Flop igru.

Primjer 3.5.1. *U prvom koraku igre, OCT algoritam donosi odluku o klađenju na temelju protivnikovog prvog klađenja te je li ono veće od 12700$ ili ne. Ako nije, algoritam nalaže da moramo pozvati; inače nastavljamo na desno dijete. Tamo algoritam donosi odluku na temelju stope pobjede naše ruke. Ako je vjerojatnost pobjede manja od 48%, algoritam nam kaže da odustanemo; inače se kladimo za pola iznosa ukupnog pota. Vidimo dakle da ovaj vrlo jednostavan model sugerira da je veličina protivničkog uloga najrelevantniji prediktor naše akcije u Pre-Flop igri. Dodatno možemo profiniti svoj izbor radnje temeljen na stopi pobjede naše ruke. U ovom primjeru, na temelju dodijeljene ruke bez ikakvih dodatnih informacija, algoritam potezu Call dodjeljuje vjerojatnost* $51,71\%$*, dok npr uz informaciji o nama dodijeljenim kartama te protivnikovom okladom manjom od* 12700$ *dodjeljuje potezu Call vjerojatnost* $51,89\%$ *itd. Slika 3.5 prikazuje gore opisano stablo odlučivanja.*



Slika 3.5: World-class Interpretable Poker - grafički prikaz optimalnog stabla klasifikacije

Zadnji algoritam koji proučavamo su stabla ekstremnog pojačavanja gradijenta (XGBoost) koja imaju bolju preciznost predviđanja od oba prethodna algoritma. Radi se o algoritmu koji iterativno prilagođava niz stabala odlučivanja, gdje model u trenutku t nije prilagođavan samo na originalnim podacima, nego dodatno i na rezidualima modela iz trenutka $t-1$. Zbog toga, model usmjerava pozornost na stabla koja su dalje u nizu zbog grešaka i nedostataka stabala prije njih, što rezultira visokoučinkovitim modelima. Međutim, budući

da je rezultirajući model linearna kombinacija velikog broja stabala odlučivanja, kritična zamjerka ovog pristupa je da model nije interpretabilan.

3.6 Rezultati

Obzirom da naša interpretabilna reprezentacija značajki (koje predstavljaju različita stanja igre) proizvodi različit broj značajki za različite krugove (npr. 35 značajki za Pre-flop i 15 značajki za Turn), moramo trenirati zaseban model za svaki kurg. Zatim iz tih stanja igre kojima treniramo model, CFR algoritam odabire najbolju distribuciju strategija za tu rundu igre. Drugim riječima, za svaku rundu igre, stanja igre predstavljaju kovarijate x_i, a varijabla odaziva y predstavlja distribuciju optimalne strategije za tu rundu igre dobivenu pomoću CFR algoritma. Za početak treniramo algoritam s gore navedenim podacima. To znači da za fiksni algoritam, dobijemo po jednu funkciju za svaku rundu, koja za svako stanje igre daje njegovo predviđanje najbolje odluke koju igrač treba napraviti. Nakon toga se simuliraju igre pri čemu je prvi igrač jedan od prethodno navedenih algoritama (tj. igrač koji odluke donosi na temelje funkcija dobivenih iz algoritma), a drugi igrač je Slumbot, popularni HUNL Poker bot i nedavni pobjednik (2018) godišnjeg računalnog natjecanja u HUNL-u. Osim toga, Slumbot je također proizveo CFR self-play i koristi pohranjenu tablicu za svoju prosječnu strategiju - za razliku od našeg pristupa učenja prosječne strategije putem algoritma za učenje.
U tablici 3.1 prikazani su prosječni dobitci u dolarima sa svakim parom poker agenata. Prosjeci su izvedeni iz 150000 igara HUNL pokera te uključuju standardne devijacije. Svaki podatak u tablici prikazan je iz perspektive metode u prvom stupcu - na primjer $4,3 \pm 0,7$ označava da je XGBoost osvojio u prosjeku 4, 3 dolara po igri protiv Slumbota. Naglasimo da rezultati u drugom stupcu pokazuju usporedbu performansi između naših poker agenata i Slumbota, dok 3., 4. i 5. stupac prikazuju usporedbe performansi izravno među našim poker agentima.
Iz tablice možemo vidjeti da su XGBoost i OCT algoritmi značajno pobijedili Slumbot, dok je CART izgubio. Također primijetimo kako je OCT impresivno učinkovit algoritam. Prvo, OCT agent može igrati značajno bolje od CART agenta, unatoč tome što su oba uvježbana do iste dubine i imaju istu konačnu strukturu - stablo odlučivanja. Stoga možemo zaključiti da je stvaranje stabla za optimizaciju svih mogućih podjela čvorova (naspram optimizacije u svakom pojedinom čvoru) vrijedna osobina.

	SLUMBOT	XGBOOST	OCT	CART
XGBOOST	$4,3 \pm 0,7$		$1,2 \pm 0,5$	$4,9 \pm 0,6$
OCT	$2,6 \pm 0,8$			$3,6 \pm 0,8$
CART	$-2,1 \pm 0,9$			

Tablica 3.1: World-class Interpretable Poker - usporedba algoritama za učenje

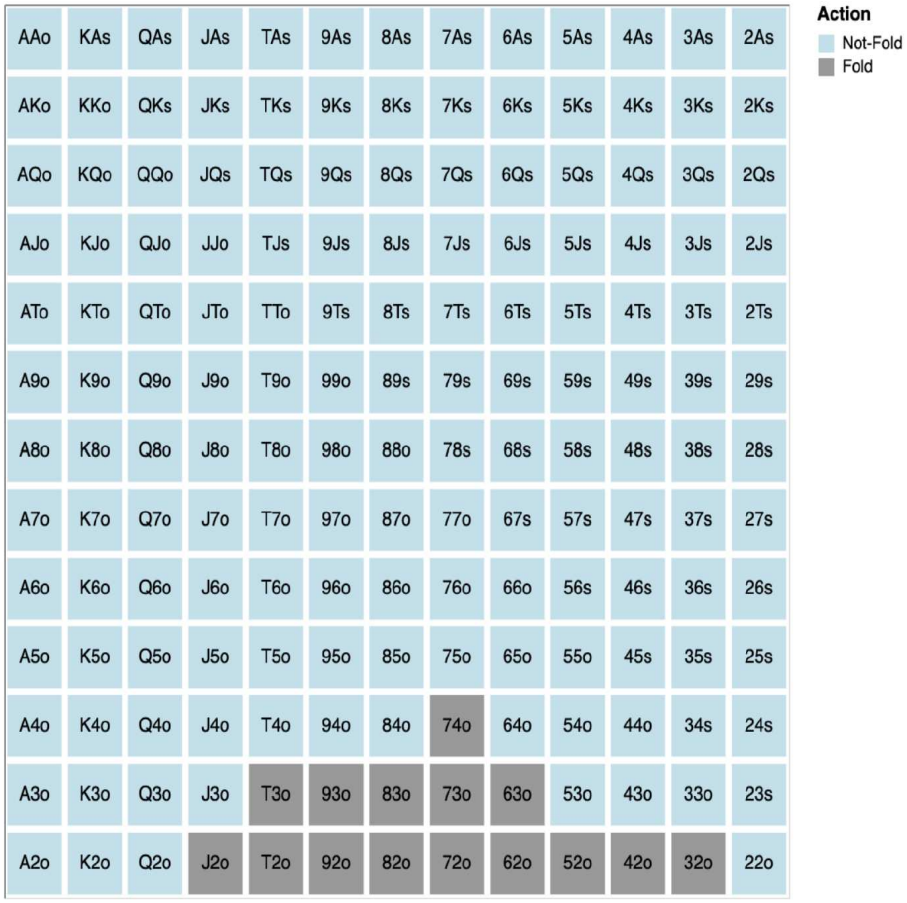
3.7 Interpretabilnost

U ovom odjeljku bavimo se bitnim pojmom koji smo mnogo puta do sad spomenuli, a to je interpretabilnost našeg modela. Započinjemo s jednim od najpopularnijih alata za vizualizaciju strategije koji se koristi u pokeru: dijagramom početnih poteza iz našeg modela, prikazanim na slici 3.6. Zatim na slikama 3.7 i 3.8 predstavljamo novi način prikaza poker strategije korištenjem optimalnog stabla odlučivanja.

Na slici 3.6, prikazujemo početne poteze OCT agenta koja ilustrira hoće li agent odmah odustati (sive ćelije) ili neće (plave ćelije). Na tom grafikonu prva dva slova označavaju koje dvije karte su nam dodijeljene, dok treće slovo po redu označava jesu li iste boje ili ne. Slovo "o" označava da su dodijeljene karte različite boje, dok "s" znači da su karte identične boje. Na primjer, 4Ks znači da smo dobili 4 i kralja u istoj boji (npr 4♣ i K♣), dok TAo predstavlja desetku i asa u različitim bojama (npr. 10♠ i A♡). Primijetimo da naš poker agent u velikoj većini slučajeva neće odustati odmah u početku te nikada neće odustati ako su mu dodijeljene karte iste boje. Dok su ovakvi grafikoni početnih poteza vrlo popularni u literaturi o pokeru i zapravo mogu biti zanimljivi za istraživanje, u odnosu na stablo odlučivanja ograničeni su u nekoliko važnih dimenzija. Kao prvo, ne ilustrira kako će agent igrati do kraja Pre-flopa ili do kraja HUNL Poker igre. Osim toga, ne daje intuiciju ili obrazloženje o tome zašto agent odustaje u nekim situacijama, a ne u drugim, što je ključno za educiranje o dobroj igri pokera i usavršavanje vlastite intuicije. Nasuprot tome, slike 3.7 i 3.8 prikazuju dijelove stabla odlučivanja agenta OCT-a, koji ne samo da prikazuju strategiju za cijeli Pre-Flop, već daju objašnjenje odluka praćenjem stabla odlučivanja prema dolje.

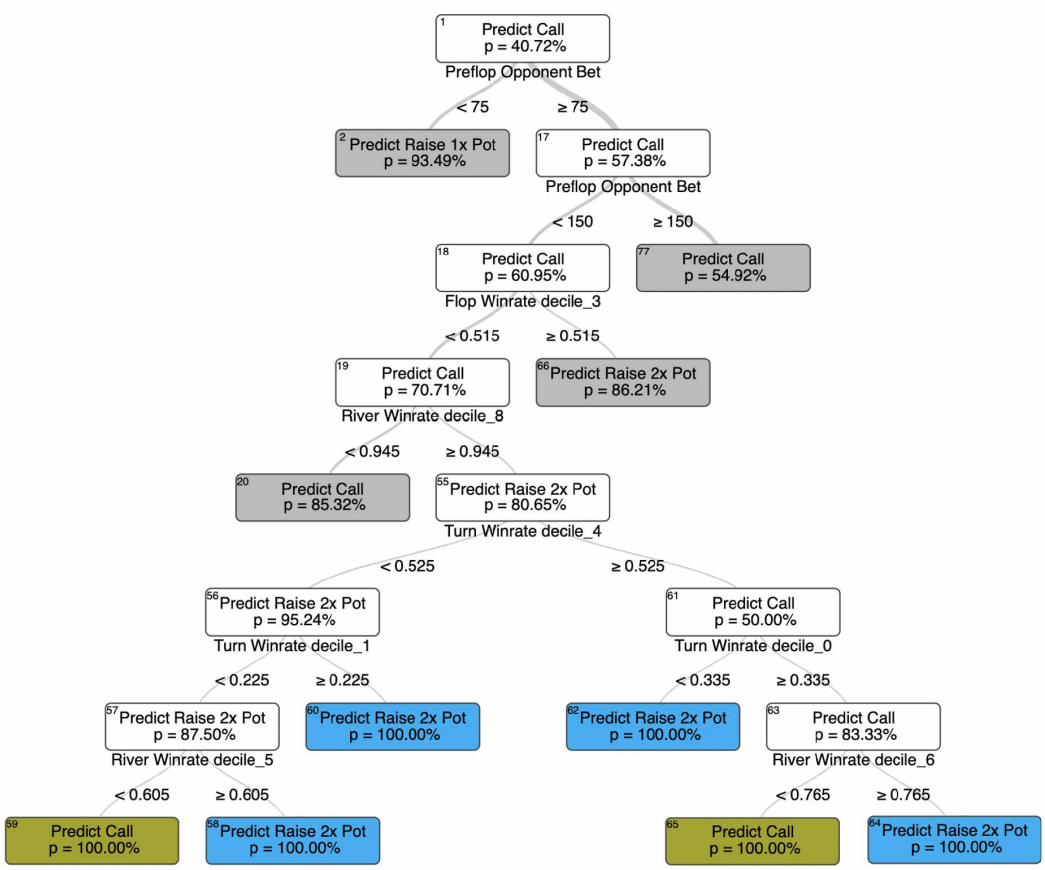
Na slici 3.7 vizualiziramo dio OCT-a koji naglašava njegovo logično donošenje odluka. Konkretno, vidimo kako se važu značajke stope pobjede. U čvoru 63, na primjer, ispituje imaju li agentove karte jaku stopu pobjede na kraju igre, ispitivanje 60-og percentila histograma stope pobjede trenutne ruke + River karta. Ako je ova dobitna stopa visoka (točnije $76,5\%$), agent podiže dvostruko veći iznos u potu, inače odustaje. Identične odluke i logična vaganja pojavljuju se u ovoj grani, ponovo naglašavajući racionalni proces donošenja odluka OCT-a.

Na slici 3.8 vizualiziramo dio OCT-a koji odgovara blefiranju, to jest kada igrač povisuje

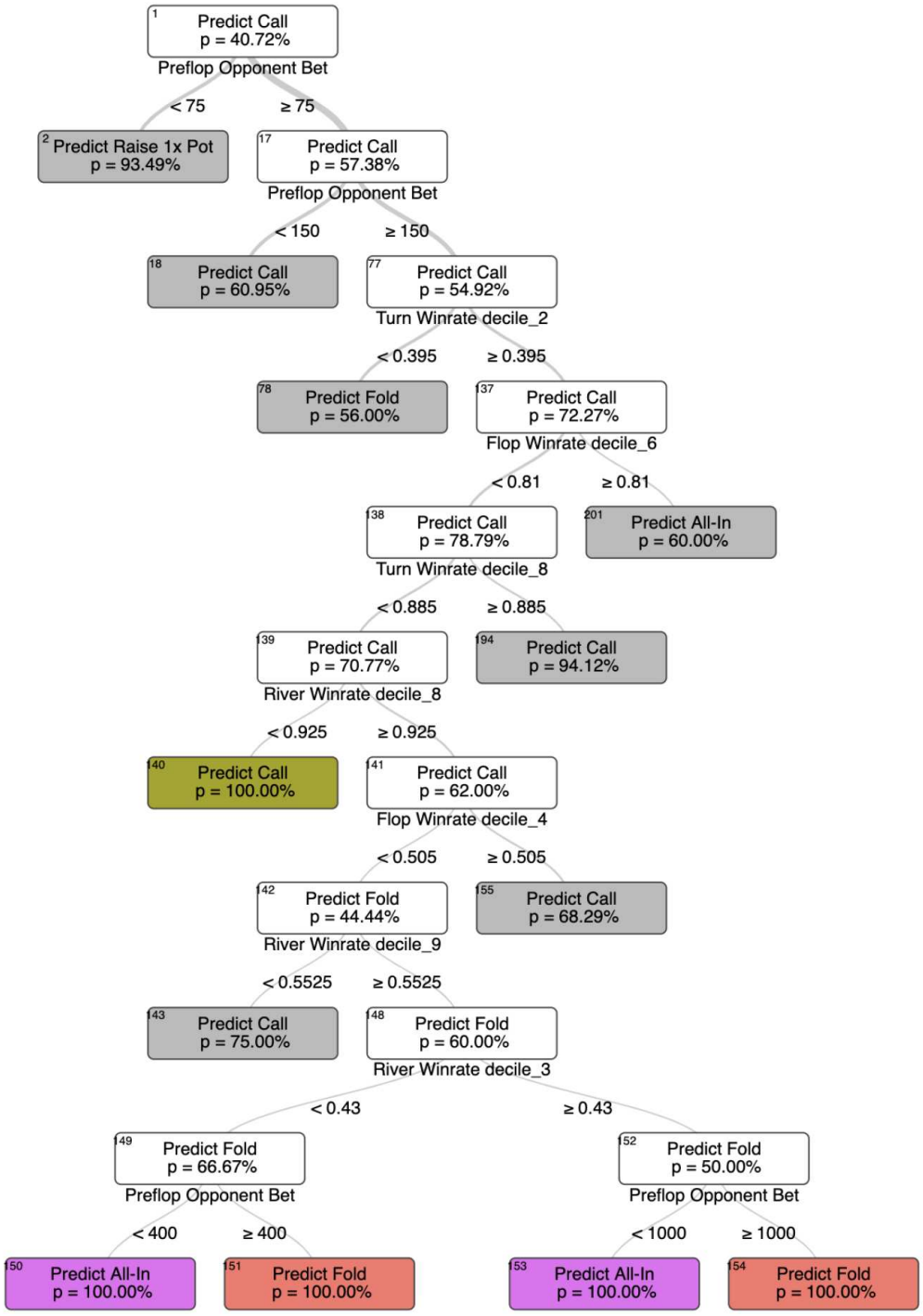


Slika 3.6: World-class Interpretable Poker - grafički prikaz početnih poteza OCT agenta

ulog u situaciji kada to nije logično jer ima loše karte. Kako bismo to razumjeli, treba uzeti u obzir da put odluke do ljubičastih i crvenih čvorova otkriva situaciju u kojoj agent nema izrazito jake karte. Konkretno, čvor 139 na tom putu pobijediti u pokazivanju karata s vjerojatnosti većom od $92,5\%$ s dobrim kartama na kraju igre (80. percentil), ali gubi više od 57% vremena s blago lošim kartama na stolu na kraju igre (30. percentil). S obzirom na to da agent nema izrazito jake karte, čvorovi 149 i 152 određuju treba li agent blefirati ili ne na temelju toga koliko pouzdano protivnik kladi. Ako je protivnik uložio previše ($\geq$ 400\$ u čvoru 149 ili $\geq$ 1000\$ u čvoru 152), tada agent odustaje; inače, agent ulaže sav novac (all-in) jer smatra da protivnik nije uložio previše samouvjereno. Blefira se ako protivnik nije igrao previše samopouzdano ili ako još uvijek ima velike šanse za oporavak ako protivnik prihvati blef.



Slika 3.7: World-class Interpretable Poker - vizualizacija interpretabilnosti OCT algoritma za Pre-Flop igru. Sivi čvorovi su zbijeni dijelovi drva, radi jednostavnosti prikaza.



Slika 3.8: World-class Interpretable Poker - vizualizacija interpretabilnosti OCT algoritma za Pre-Flop igru kod blefiranja. Sivi čvorovi su zbijeni dijelovi drva, radi jednostavnosti prikaza.

Bibliografija

[1] R. A. Berk, *Statistical Learning from a Regression Perspective*, Springer Cham, 2018.

[2] D. Bertsimas i A. Paskov, *World-class Interpretable Poker*, Machine Learning **111** (2022), br. 8, 3063–3083, ISSN 1573-0565, https://doi.org/10.1007/s10994-022-06179-8.

[3] T. Hastie, R. Tibshirani i J. Friedman, *The Elements of Statistical Learning: Data mining, Inference, and Prediction*, Springer New York, NY, 2009.

[4] Gilles Louppe, *Understanding Random Forests: From Theory to Practice*, 2015, https://arxiv.org/abs/1407.7502.

[5] M. Marohnić i M. Bašić, *Diskretna matematika: Vježbe*, 2015, https://web.math.pmf.unizg.hr/nastava/komb/SKRIPTA.pdf.

[6] R. Mrazović, *Teorija igara*, 2022, https://web.math.pmf.unizg.hr/nastava/tigara/files/tigara-predavanja.pdf.

[7] M. Shafiei, N. Sturtevant i J. Schaeffer, *Comparing UCT versus CFR in Simultaneous Games*, 2009, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=61afeccbde868f4f49b150239d44eaf5e7f49ec2.

Sažetak

Ovaj rad započeli smo matematičkom formulacijom problema učenja iz podataka i nadziranog učenja. Zatim smo definirali modele bazirane na stablima te posebno analizirali klasifikacijska i regresijska stabla. Za kraj smo istražili kako odrediti optimalnu strategiju u igri pokera koristeći CART (Classification and Regression Trees) algoritm pomoću minimizacije protučinjeničnog žaljenja.

Summary

We started this paper with a mathematical formulation of the problem of learning from data and supervised learning. Then, we defined tree-based models and specifically analyzed classification and regression trees. Finally, we explored how to determine the optimal strategy in the game of poker using the CART (Classification and Regression Trees) algorithm by minimizing counterfactual regret.

Životopis

Roden sam 8. veljače 2000. u Koprivnici, gdje nakon osnovne škole upisujem opću gimnaziju koju završavam 2018. godine. Nakon završetka gimnazije upisujem preddiplomski sveučilišni studij Matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu, a 2021. godine upisujem diplomski studij Financijska i poslovna matematika na istom fakultetu. Ljetni semestar akademske godine 2021./2022. provodim na studentskoj razmjeni na Sveučilištu u Gentu u Belgiji. Za vrijeme diplomskog studija stječem primjenjena znanja iz područja investiranja tijekom stručne prakse u PBZ Croatia osiguranju te aktuarstva u Allianz osiguranju kao aktuar pripravnik.