Ansambli, alati i baze podataka

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

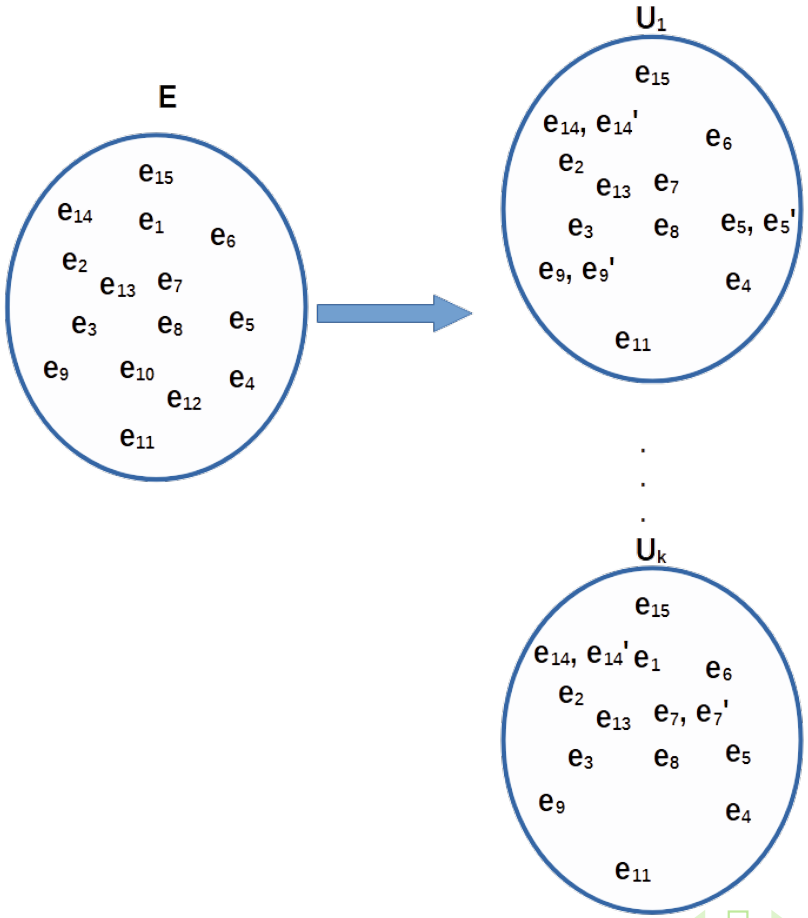
20. ožujka, 2026.

Ansambli stabala odlučivanja

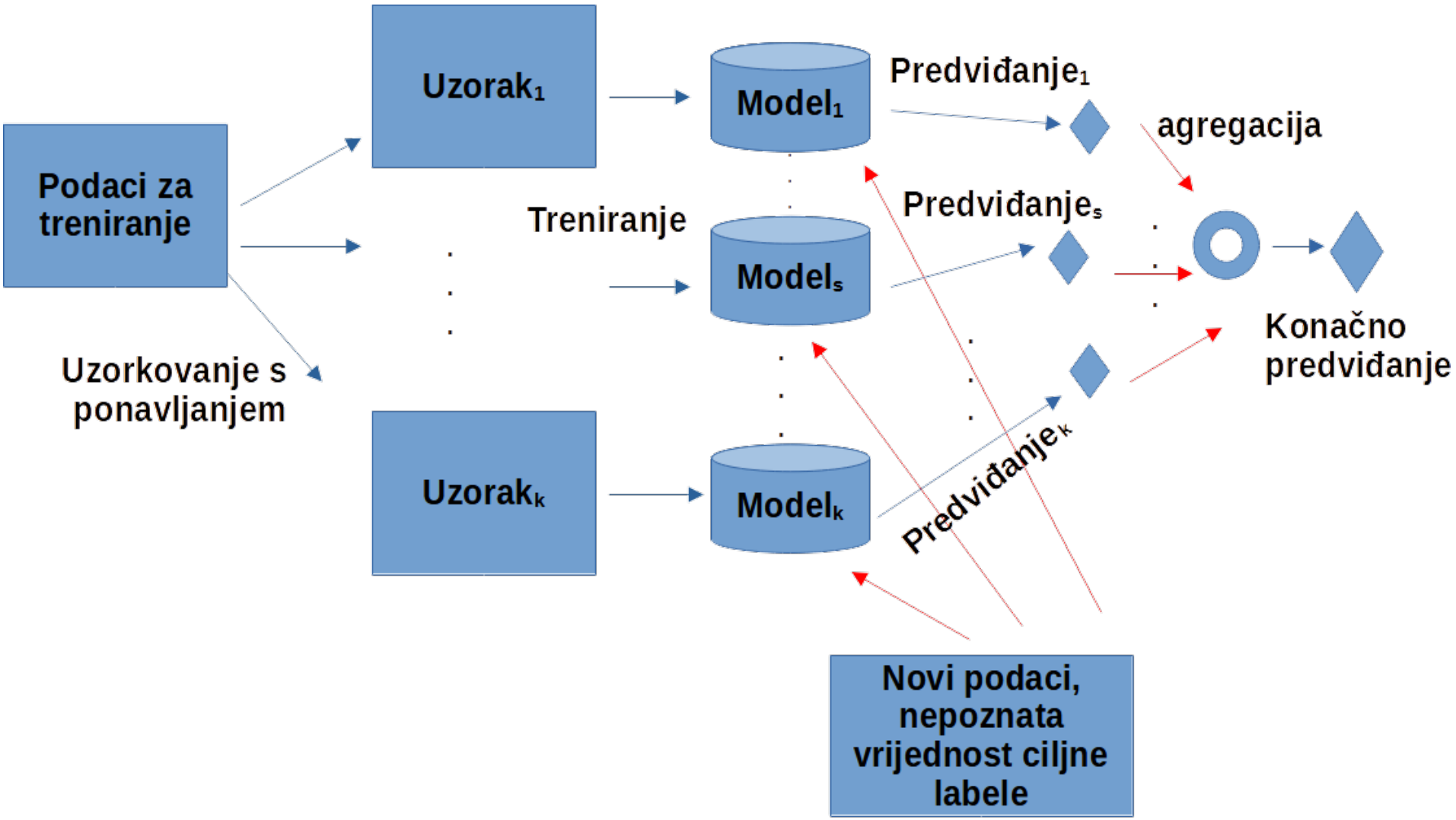
- Ansambli su modeli koji donose odluke kombiniranjem odluka više osnovnih (potencijalno različitih) modela. Intuitivno, odluku donosimo konsenzusom nakon konzultacije više eksperata.
- Prednosti ansambala su što često daju točnija i pouzdanija predviđanja u odnosu na jedan osnovni model.
- Stabla odlučivanja u svakom koraku biraju najbolju točku podjele, što je pohlepan postupak. Zbog toga mogu zaglaviti u lokalnim ekstremima. Također, osjetljivi su na promjene u podacima.
- Korištenjem ansambala se smanjuje varijanca (manja ovisnost o malim promjenama u skupu za učenje).
- Određeni postupci konstrukcije ansambala rješavaju i problem pohlepnosti stabala odlučivanja.
- Korištenje ansambala reducira pristranost, možemo reprezentirati složenije koncepte od modela stabala odlučivanja.

Bagging

- Bagging (eng. Bootstrap aggregating) - iz skupa za treniranje veličine $|E|$, stvaramo $k \in \mathbb{N}$ uzoraka s ponavljanjem veličine $|E|$. Neki entiteti u generiranim uzorcima se javljaju više puta, dok neki entiteti nisu prisutni u generiranim uzorcima.



Bagging


Podaci za treniranje
Uzorkovanje s ponavljanjem
Uzorak$_1$
Uzorak$_k$
Treniranje
Model$_1$
Model$_s$
Model$_k$
Predviđanje$_1$
Predviđanje$_s$
Predviđanje$_k$
agregacija
Konačno predviđanje
Novi podaci, nepoznata vrijednost ciljne labele

Bagging

Pretpostavimo:

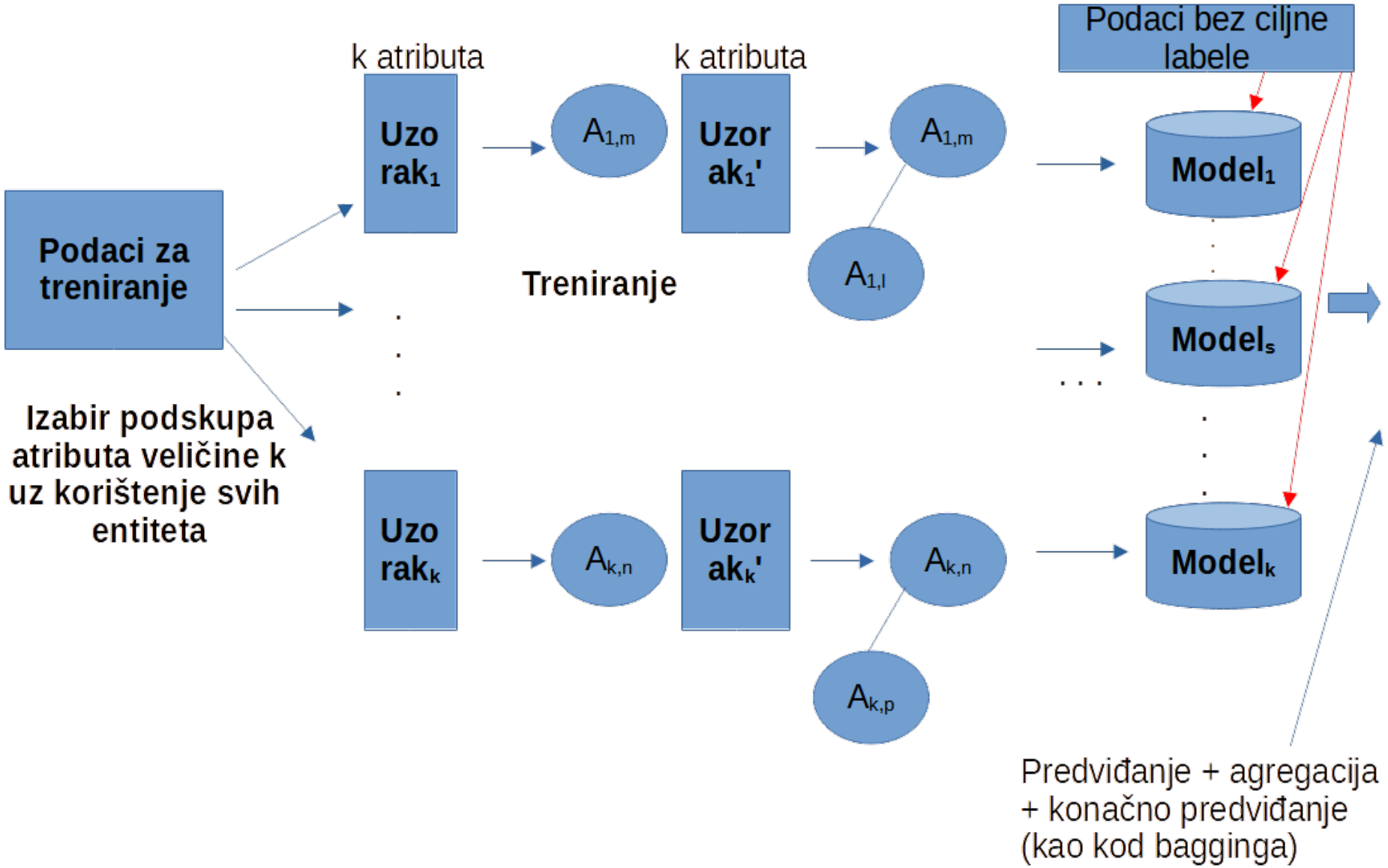
- Da imamo regresijski zadatak.
- Očekivana kvadratna pogreška modela je: $h_i(\vec{x}) = y(\vec{x}) + \varepsilon_i(\vec{x})$, gdje $y(\vec{x})$ predstavlja točnu vrijednost cilja, a $\varepsilon_i(\vec{x})$ označava pogrešku modela.
- $E[(h_i(\vec{x}) - y(\vec{x}))^2] = E[\varepsilon_i(\vec{x})^2]$.
- Neka za svaki model:
  - $E[\varepsilon_i(\vec{x})] = 0$ (očekivana greška svakog modela je 0).
  - $E[\varepsilon_i(\vec{x})\varepsilon_j(\vec{x})] = 0$ (greške između različitih modela nisu korelirane).
- Prosječna kvadratna pogreška m modela je: $\bar{E} = \frac{1}{m}\sum_{i=1}^{m} E[(\varepsilon_i(\vec{x}))^2]$.
- Prosječna kvadratna pogreška bagging ansambla uz korištenje prosječne vrijednosti predviđanja je:

  $E_b = E[(y(\vec{x}) - \frac{1}{m}\sum_{i=1}^{m} h_i(\vec{x}))^2] = \frac{1}{m}\bar{E}$, dakle $E_b \leq \bar{E}$.

Slučajni podskupovi atributa

- Kod bagginga imamo često slučaj da bazni modeli koriste identične (najprediktivnije) atribute za dijeljenje podataka. To onemogućuje izlaske iz lokalnih ekstrema.
- Postupak slučajnih podskupova atributa (eng. random subspaces) svaki bazni model trenira tako da u svakom čvoru točku dijeljenja bira samo iz podskupa originalnih atributa iz skupa za treniranje. Time prisiljava bazne modele da za treniranje koriste razne atribute (koji bi inače bili ignorirani).
- Standardno slučajni podskupovi vrše treniranje baznih modela koristeći sve entitete iz skupa za treniranje, ali koristeći slučajno izabran podskup atributa pri svakom dijeljenju (podskupovi između baznih modela se mogu preklapati).

Slučajni podskupovi atributa



Slučajna šuma stabala

- Kombinira bagging sa slučajnim odabirom podskupa atributa.
  - Gradi stabla iz bootstrap uzorka skupa za učenje.
  - Točku dijeljenja bira iz skupa od k slučajno odabranih atributa. Za svaki čvor se ponovo bira slučajno odabrani podskup atributa.
- Često uzimamo $k = \sqrt{m}$, gdje je m broj atributa kod klasifikacijskih zadataka, a $\lfloor \frac{m}{3} \rfloor$ kod regresijskih zadataka (isto je i kod slučajnih podskupova atributa).
- Kod svih navedenih ansambala bazni modeli su individualno manje točni i jednostavniji od stabla odlučivanja treniranog koristeći sve entitete i sve atribute, stoga se bazni modeli još nazivaju i **slabi učenici** (eng. weak learner).
- Snaga ansambala leži u korištenju većeg broja slabih učenika koji zajedničkim naporom (konsenzusom) daju točnije i pouzdanije rezultate od onih dobivenih jednim jakim modelom.

Konsenzus kod donošenja odluke

Načini agregiranja predviđanja baznih modela:

- Regresija:
  - Prosječna vrijednost predviđanja baznih modela.
  - Težinska agregacija - svakom baznom klasifikatoru se daje težina (važnost), najčešće ovisna o točnosti klasifikatora. Predviđanje je težinska suma predviđanja baznih klasifikatora.
- Klasifikacija:
  - Najčešće predviđena klasa (eng. majority vote).
  - Težinska agregacija - svakom baznom modelu se dodijeli težina vezana uz točnost baznog modela. Dobijemo težinska predviđanja iz kojih računamo konačnu klasu koju predviđa ansambl.

Boosting

- Postupak stvaranja ansambla kombiniranjem slabih učenika (modela).
- Slabi modeli obično imaju visoku pristranost (kod klasifikacije malo bolji od slučajnog klasifikatora).
- Modeli se rade sekvencijalno (jedan iza drugoga), na uzorcima podataka.
- Predviđanje boosting modela se dobije kao težinska kombinacija predviđanja individualnih (slabih) modela u kreiranom nizu.

XGBoost

- Koristi boosting pristup (iterativno stvara stabla odlučivanja koja popravljaju točnost predviđanja ciljne varijable).
- Prostor CART stabala označavamo s $\mathcal{F} = \{f(x) = w_{q(x)}\}$, gdje $q(x) : \mathbb{R}^m \mapsto |N_l|$, $\vec{w} \in \mathbb{R}^{|N_l|}$. $f(x)$ označava funkciju koju modelira stablo odlučivanja, q definira strukturu stabla, $\vec{w}$ sadrži težine listova stabla. $|N_l|$ označava broj listova stabla.
- Minimiziramo funkciju: $\mathcal{L}(\phi) = \sum_{i=1}^{|E|} l(\hat{y}, y) + \sum_k \Omega(f_k)$, gdje l označava diferencijabilnu, konveksnu funkciju gubitka. Regularizaciju Ω definiramo kao: $\Omega(f) = \gamma|N_l| + \frac{1}{2}\lambda\|w\|^2$.
- Parametar γ penalizira stabla s puno listova (veća vrijednost γ zahtjeva veliko povećanje točnosti modela da bi grananje bilo prihvatljivo).
- Parametar λ regularizira težine listova (distribuira težine kroz više listova).

XGBoost

- Ansambl se trenira tako da iterativno dodajemo nova stabla koja poboljšavaju funkciju cilja. U koraku t treniranja optimiziramo:
$$\mathcal{L}^{(t)} = \sum_{i=1}^{|E|} l(y_i, \hat{y_i}^{(t-1)} + f_t(\vec{x_t})) + \Omega(f_t).$$
- f_t dodajemo na pohlepan način (takvu da se $\mathcal{L}^{(t)}$ maksimalno smanji.
- Funkciju možemo aproksimirati kao:
$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{|E|} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(\vec{x_t}) + \frac{1}{2} h_i f_t(\vec{x_t})^2] + \Omega(f_t)$$ (Taylorov razvoj do trećeg člana).
- $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}))$ (prva i druga derivacija funkcije gubitka).
- Uz izbacivanje konstantnih članova dobijemo:
$$\tilde{\mathcal{L}^{(t)}} = \sum_{i=1}^{|E|} [g_i f_t(\vec{x_t}) + \frac{1}{2} h_i f_t(\vec{x_t})^2] + \Omega(f_t)$$

XGBoost

- Označimo s N_l^j skup svih entiteta koji pripadaju listu j.
- $\tilde{\mathcal{L}^{(t)}} = \sum_{i=1}^{|E|}[g_i f_t(\vec{x_t}) + \frac{1}{2}h_i f_t(\vec{x_t})^2] + \gamma|N_l| + \frac{1}{2}\lambda\sum_{j=1}^{|N_l|} w_j^2.$
- To možemo zapisati kao:

$$\tilde{\mathcal{L}^{(t)}} = \sum_{j=1}^{|N_l|}[(\sum_{i\in N_l^j} g_i)w_j + \frac{1}{2}(\sum_{i\in N_l^j} h_i + \lambda)w_j^2] + \gamma|N_l|.$$

- Optimalnu vrijednost težine w_j^* dobijemo iz: $w_j^* = -\frac{\sum_{i\in N_l^j} g_i}{\sum_{i\in N_l^j} h_i + \lambda}$
- Optimalna vrijednost $\tilde{\mathcal{L}^{(t)}}(q) = -\frac{1}{2}\sum_{j=1}^{|N_l|}\frac{(\sum_{i\in N_l^j} g_i)^2)}{\sum_{i\in N_i^j} h_i + \lambda} + \gamma|N_l|.$

XGBoost

- Pošto nije moguće istražiti sve podstrukture stabla, algoritam kreće od lista i pohlepno dodaje djecu s obzirom na mjeru dobrote.
- Označimo s N skup entiteta koji pripadaju nekom čvoru stabla, a s N_L i N_R skupove entiteta koji pripadaju lijevom i desnom djetetu.
- Kvalitetu dijeljenja računamo kao:

$$\mathcal{L}_{split} = \frac{1}{2}[\frac{(\sum_{i\in N_L} g_i)^2}{\sum_{i\in N_L} h_i + \lambda} + \frac{(\sum_{i\in N_R} g_i)^2}{\sum_{i\in N_R} h_i + \lambda} - \frac{(\sum_{i\in N} g_i)^2}{\sum_{i\in N} h_i + \lambda}] - \gamma.$$

- Uz regularizaciju, algoritam koristi postupak sužavanja (eng. shrinkage), kod kojega se važnost svakog stabla u ansamblu skalira parametrom η. To omogućava budućim modelima da dodatno poprave točnost ansambla.
- Algoritam također koristi postupak slučajnih podskupova atributa (uz sve njihove prednosti).

XGBoost - pronalazak točke dijeljenja

- Egzaktni algoritam:
  - Sortiramo vrijednosti atributa po vrijednosti uzlazno.
  - Za svaku vrijednost, izračunaj kvalitetu dijeljenja (povećavanjem vrijednosti točke dijeljenja, sve više entiteta smještamo u lijevo dijete).
  - Biramo točku dijeljenja koja maksimizira mjeru dobrote.
- Aproksimativni algoritam:
  - Predloži kandidate za točku dijeljenja prema percentilima distribucije vrijednosti atributa.
  - Stvori grupe temeljene na intervalima koje dobijemo iz točaka dijeljenja.
  - Računamo vrijednosti g_i i h_i na navedenim intervalima.
  - Računamo najbolju točku dijeljenja od navedenih kandidata optimiranjem mjere kvalitete dijeljnja.
- Dijeljenje u prisutnosti nedostajućih vrijednosti:
  - Računamo točke dijeljenja na podskupu entiteta koji ne sadrži nedostajuće vrijednosti.
  - Za svaku točku dijeljenja izračunamo kvalitetu podjele u slučaju da nedostajuće vrijednosti dodamo lijevom djetetu i u slučaju da nedostajuće vrijednosti dodamo desnom djetetu.
  - Izaberemo podjelu koja maksimizira dobrotu dijeljenja.

Izabir značajki pomoću slučajne šume stabala

- Opisat ćemo postupak za selekciju svih **značajnih** atributa - **Boruta**.
- Pošto tražimo sve značajne atribute, a ne samo najbolje, moramo imati objekte za usporedbu (trebamo odrediti bazu od koje želimo mjeriti poboljšanje).
- Boruta kao bazu uzima kopije originalnih atributa (atribute sjene, eng. shadow attributes) koje dobijemo tako da vrijednosti originalnog atributa permutiramo između entiteta (primijenimo slučajnu permutaciju na vrijednosti originalnog atributa).
- Određujemo jesu li originalni atributi prediktivno značajno bolji od randomiziranih koristeći slučajnu šumu stabala.

Izabir značajki pomoću slučajne šume stabala

Koraci pristupa Boruta:

1. Proširi skup podataka kreiranjem randomizirane verzije atributa (atributa sjene) za svaki originalni atribut iz skupa podataka koji nije proglašen nevažnim. Permutiranjem vrijednosti razbijamo korelacije vrijednosti novo kreiranih atributa s vrijednostima ciljne varijable.
2. Izvrši algoritam slučajne šume stabala i izračunaj z-vrijednosti atributa. z-vrijednost atributa se dobije kao $z(a) = \frac{\mu_I}{\sigma_I}$. μ_I označava prosječnu redukciju u točnosti stabala iz slučajne šume stabala koji sadrže grananje po atributu a nakon što mu permutiramo vrijednosti. σ_I označava standardnu devijaciju navedene mjere.
3. Računamo maksimalnu vrijednost z-mjere dobivenu nad randomiziranim atributima. Označimo ju s z_{maxR}. Označimo sve originalne atribute sa z-mjerom boljom od z_{maxR}.

Izabir značajki pomoću slučajne šume stabala

4. Provedi binomni test značajnosti dobrote atributa u odnosu na z_{maxR}. $p = \binom{n}{k} p_0^k (1-p_0)^{n-k}$. n označava ukupan broj iteracija (2-3) koje smo izvršili. k je broj iteracija kod kojih je izračunata z vrijednost atributa bila veća od z_{maxR}.
5. Podijeli atribute u tri kategorije ovisno o testu značajnosti s definiranom razinom značajnosti α : a) **važni atributi**, oni s $z >> z_{maxR}$, b) **nevažni atributi**, oni s $z << z_{maxR}$, te preostale atribute u skupinu potencijalno značajnih atributa.
6. Brišemo atribute sjene, the nevažne atribute.
7. Ponavljamo postupak 1 - 5 dok sve atribute ne podijelimo u važne/nevažne ili dok ne dostignemo maksimalni broj treniranja slučajne šume stabala *maxIter* (parametar).

Važniji alati i izvori podataka

Softver koji omogućava učenje i evaluaciju raznih stabala odlučivanja i ansambala:

- WEKA - `https://ml.cms.waikato.ac.nz/weka/` - softver za strojno učenje otvorenog koda (`C4.5`, `Random Subspaces`, `Random Forest`, `Bagging`).
- Scikit learn (python) - `https://scikit-learn.org/stable/` - python biblioteka za strojno učenje (`CART`, `Random Forest`, `Bagging`, `XGBoost`, `BORUTA`)
- CLUS - `https://github.com/knowledge-technologies/clus` - softver za indukciju pravila i stabala odlučivanja, implementira Prediktivna stabla klasteriranja (PCT).
- QUEST - `https://github.com/hetianle/QuestDecisionTree` - implementacija algoritma QUEST.
- BORUTA - `https://www.jstatsoft.org/article/view/v036i11` - R paket s implementacijom BORUTA-e.

Važniji alati i izvori podataka

Baze podataka:

- UCI skupovi podataka za strojno učenje - https://archive.ics.uci.edu/datasets
- Kaggle skupovi podataka - https://www.kaggle.com/datasets
- OpenML skupovi podataka - https://www.openml.org/search?type=data&sort=runs&status=active
- Amazon skupovi podataka - https://registry.opendata.aws/
- Microsoft skupovi podataka - https://www.microsoft.com/en-us/research/tools/?facet%5Bdate%5D%5Bfixed%5D=any&pg=1&sort_by=most-recent
- Awsome public skupovi podataka - https://github.com/awesomedata/awesome-public-datasets
- Google tražilica podataka - https://datasetsearch.research.google.com/
- Skupovi podatka za strojno učenje - https://github.com/topics/machine-learning-dataset