Metode ansambala i njihovo objašnjavanje

Dubinska analiza podataka

6. predavanje

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

Ak. god. 2023./2024.

Sadržaj

- Ansambli klasifikatora i regresora
- Algoritmi ansambala
  - Slučajna šuma
  - Iznimno slučajna stabla
  - Rotacijska šuma
  - AdaBoost i MultiBoost
  - XGBoost
  - CatBoost
- Postupci objašnjavanja modela ansambala
  - Permutacijska važnost značajki
  - SHAP

Ansambli klasifikatora i regresora

Ansambli i njihovo korištenje

- **Ansambl** (engl. *ensemble*) u strojnom učenju je skup od dva ili više modela strojnog učenja koji ima cilj **poboljšati uspješnost rezultata** u odnosu na pojedinačne modele
- Ansambli se tipično razmatraju u kontekstu **nadziranog učenja** (engl. *supervised learning*), za klasifikacijske ili regresijske probleme
- Pojedinačni modeli mogu biti izgrađeni s **istim** ili **različitim** temeljnim algoritmom strojnog učenja (engl. *base algorithm*)
- Čitava teorija iza uspješnosti ansambala je dosta heuristički orijentirana i **ne garantira** uspjeh
- Ipak, temeljna pretpostavka uspješnih ansambala je da **pojedinačni modeli trebaju biti raznoliki**
  - U praksi se postiže tehnikama ponovnog uzorkovanja ili odabirom algoritama iz različitih familija

Kanta modela

- Engl. *bucket of models*
- Najjednostavniji pristup ansamblima gdje se zadatak ansambla modela svodi na **odabir pojedinačnog najboljeg modela** (engl. *model selection*)
  - Npr. između rezultata Naivnog Bayesa, C4.5, višeslojnog perceptrona i stroja s potpornim vektorima
- Najčešći algoritam za odabir modela je onaj koji odabire najbolji model na temelju **rezultata unakrsne validacije na skupu za učenje** (engl. *cross-validation model selection*)
  - Metrika po kojoj se radi izbor najboljeg modela treba biti prilagođena problemu koji se rješava, npr. za nebalansirane skupove podataka ne bi se smjela koristiti klasifikacijska točnost
- Odabrani model se potom primijenjuje na skupu za testiranje

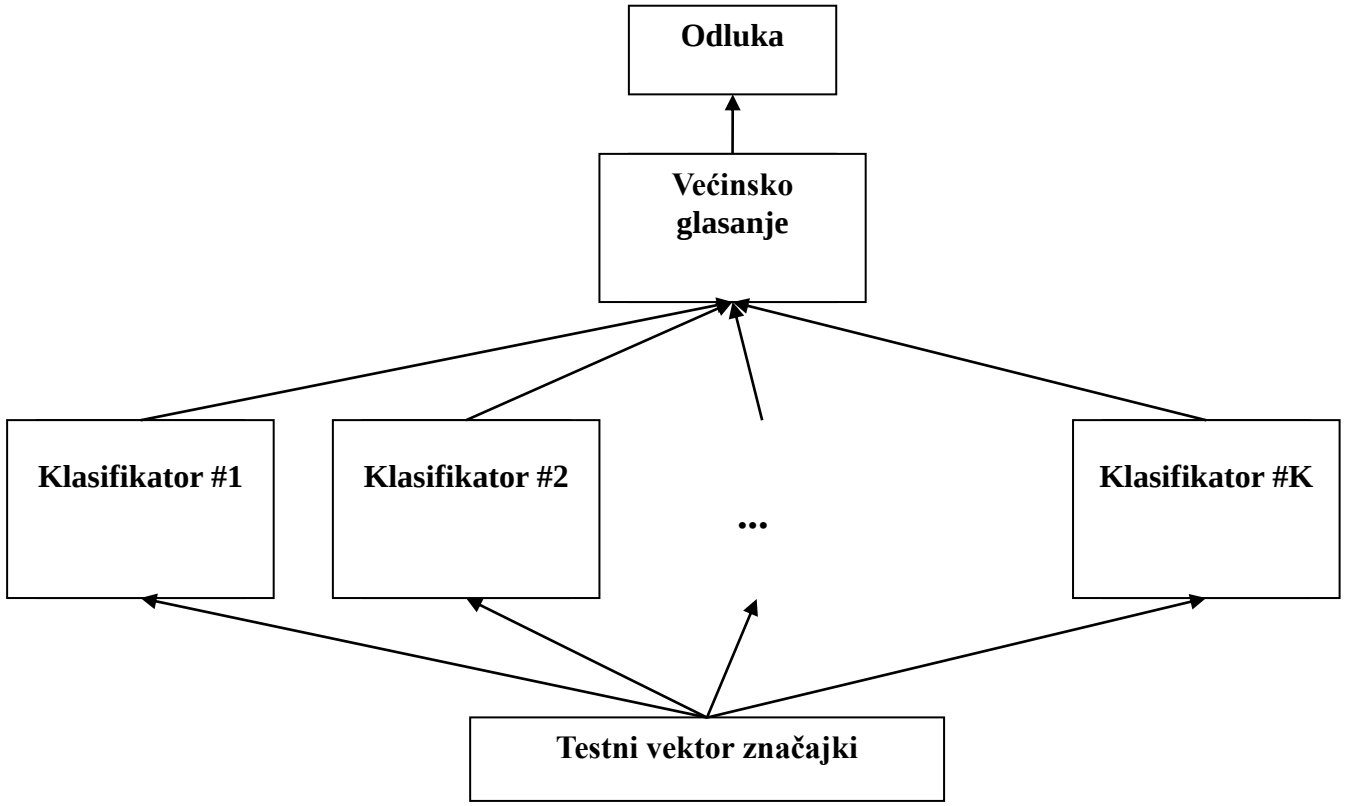
Temeljna podjela pristupa ansambala

- **Sustavi ansambala više stručnjaka** (engl. *multi-expert systems*)
  - **Heterogeni jednostavni ansambl**
  - *Stacking*
  - *Bagging*
- **Sustavi ansambala u više koraka** (iterativni sustavi ansambala) (engl. *multi-stage systems*)
  - *Boosting*
  - Kaskadirajući klasifikatori (engl. *cascading classifiers*)

Heterogeni jednostavni ansambl

- Heterogeni jednostavni ansambl sastoji se od nekoliko (najčešće tri) različita pojedinačna algoritma strojnog učenja
  - Svaki pojedinačni algoritam može, ali ne mora biti ansambl
  - Primjer: MLP, *k*-NN i slučajna šuma za detekciju „spamajućih" recenzija proizvoda
    - M. Fayaz, A. Khan, J. U. Rahman, A. Alharbi, M. I. Uddin, B. Alouffi, "Ensemble Machine Learning Model for Classification of Spam Product Reviews", Complexity, vol. 2020, Article ID 8857570, 2020. https://doi.org/10.1155/2020/8857570
- Pretpostavka dobrog korištenja u praksi: **algoritmi trebaju biti značajno različiti** (dolaziti iz različitih familija metoda)
- Odluke u fazi testiranja donose se najčešće **većinskim glasanjem** (engl. *majority voting*)
- Za posebne primjene heterogeni jednostavni ansambl daje (u prosjeku) bolje rezultate nego bilo koji od korištenih pojedinačnih modela
  - Nema garancije da ne postoji neki drugi klasifikator koji će dati bolji rezultat od heterogenog ansambla

Većinsko glasanje



Stacking

- Također i: engl. *stacked generalization*
- Smatra se nadogradnjom heterogenog jednostavnog modela
- Ansambl modela gradi se u **dvije razine**
  - Niža razina (razina 0) sastoji se od **jednog ili više modela** (najčešće tri) različitih (heterogenih) algoritama koji uče na jednom većem dijelu ulaznog skupa (skup za učenje za nižu razinu)
  - Viša razina (razina 1) sastoji se od **jednog** modela koji uči na temelju podataka – izlaza modela razine 0 za preostali dio ulaznog skupa (koji čini validacijski skup za nižu razinu)
- **Zadatak modela razine 1 je odrediti kako najbolje iskombinirati doprinose modela razine 0 (na zasebnom skupu za validaciju)**
- Vidjeti: `sklearn.ensemble.StackingClassifier`

D. H. Wolpert (1992). Stacked generalization. Neural Networks. 5:241-259

Stacking

- Model razine 1 može biti rezultat bilo kojeg algoritma strojnog učenja, ali najčešće se uzima obični, **regresijski model** (takav *stacking* zove se ***blending***)
  - **Linearna regresija u slučaju regresijskog problema ili linearna regresija s višestrukim odgovorom u slučaju klasifikacije**
  - *Stacking* u ovom slučaju zapravo predstavlja **utežano većinsko glasanje** (engl. *weighted majority voting*)
  - Rezultati pojedinih modela razine 0 predstavljaju vrijednosti **značajki** za model razine 1
- Najčešći pristup *stackingu* je da se koristi unakrsna validacija s *k* preklopa za vrednovanje modela (engl. *k-fold cross-validation*), i to čak *LOOCV*
  - Modeli razine 0 daju predikcije na validacijskim skupovima za cijeli skup podataka (postepeno, putem *k* preklopa)
  - Model razine 1 koristi te predikcije da bi naučio težine pojedinih modela razine 0 (težine uz značajke u modelu razine 1)
  - Na kraju, modeli razine 0 se nauče na cijelom skupu podataka i *stacking* ansambl je potom spreman za testiranje na novim podacima
- *Stacking* često postiže bolje rezultate od bilo kojeg pojedinačnog modela razine 0, nedostatak je povećanje složenosti klasifikatora / regresora

K. M. Ting, I. H. Witten, Issues in Stacked Generalization, Journal Of Artificial Intelligence Research, Volume 10, pages 271-289, 1999.

Bagging

- Skraćenje od engl. *bootstrap aggregation*
- Ansambl koji se sastoji od većeg broja modela istog tipa (najčešće) ili različitih (rjeđe)
- Ulazni podaci za svaki pojedinačni model dobivaju se uzorkovanjem tipa *bootstrap*
- Odluke u fazi testiranja donose se agregacijom rezultata pojedinačnih modela i to **većinskim glasanjem** (engl. *majority voting*)
- Glavna značajka: *Bagging* **smanjuje varijancu** pojedinačnih modela
  - Najčešće se koristi za algoritme s relativno visokom varijancom, kao što su stabla odluke
- Vidjeti: `sklearn.ensemble.BaggingClassifier`

Bagging

- Uzorkovanje tipa *boostrap*:
  - Izabire se na slučajan način ***N*** primjeraka iz skupa od ***N*** primjeraka, **s ponavljanjem**
  - Svaki pojedinačni primjerak iz početnog skupa ima vjerojatnost da će biti (uspješno) izabran približno jednaku **0.632**
  - U prosjeku 36,8% primjeraka iz izvornog skupa podataka neće biti izabrano u skup za učenje te će oni činiti tzv. **skup OOB** (engl. ***out-of-bag***), koji je različit za svaki model
  - Skup OOB omogućuje **nepristranu** procjenu pogreške za određeni model i čini njegov **validacijski skup** (ne skup za testiranje)
  - **Prednost**: generiranje ponešto različitog ulaznog skupa za svaki model **doprinosi različitosti modela** u ansamblu
  - **Nedostatak**: primjerci koji su u skupu OOB **ne uzimaju se u obzir** za učenje, što ovisno o veličini skupa za učenje može biti problem

Boosting

- Postupak *boosting* gradi ansambl modela **iterativno**, nastojeći poboljšati („*boostati”*) decizijsku granicu među klasama iz koraka u korak
  - Može se koristiti i za regresiju, gdje iz koraka u korak smanjuje pogrešku modela (neka mjera, npr. MSE)
- U svakoj novoj iteraciji uči se novi model na skupu za učenje
- Skup za učenje se iz **iteracije u iteraciju mijenja** u ovisnosti od rezultata modela u prethodnom koraku
- Na početku svi primjerci u skupu za učenje imaju jednaku težinu, a u idućim iteracijama oni **primjerci koji nisu točno klasificirani dobivaju veću težinu**, dok se težina točno klasificiranih smanjuje
- Na skupu za testiranje odgovor ansambla za svaki primjerak koristi neki model glasanja
- *Boosting* poglavito **smanjuje pristranost** modela, a dijelom i varijancu modela (ali ne toliko kao *bagging*)

Algoritmi ansambala

Slučajne šume (engl. *random forests*)

Slučajna šuma

- Algoritam je razvio Leo Breiman 2001. godine
- Poznati ansambl slučajnih stabala odluke, pripada u skupinu vrhunskih algoritama strojnog učenja i dubinske analize podataka
- U klasifikacijskim problemima točnost rezultata je usporediva s najboljim algoritmima strojnog učenja, a **brzina mu je često bitno veća**
- Bazira se na ranijim saznanjima o (ne)uspješnosti pojedinih pristupa korištenju ansambala klasifikatora kao i nedostatcima pri korištenju jednog stabla odluke
- Koristi **kombinaciju izvora slučajnosti**, **veliki broj stabala** i **većinsko glasanje** kako bi postigao izvrsne rezultate
- Može se koristiti i za regresijske probleme, ali češće se koristi za klasifikacijske
- Vidjeti: `sklearn.ensemble.RandomForestClassifier`

Breiman, L. Random Forests. *Machine Learning* **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324

Slučajna šuma – značajke

- **Minimizira pogrešku nastalu zbog varijance i pristranosti**
  - Pristranost se minimizira tako što se stablo gradi do kraja i ne podrezuje se (u pravilu, ili se podrezuje na nekom dubokom nivou)
  - Pogreška zbog varijance se minimizira uz pomoć ***bootstrap*** **uzorkovanja, slučajnog odabira atributa u stablu i velikog broja stabala**
- **Postupak je vrlo malo osjetljiv na prenaučenost**
  - Na temelju zakona velikih brojeva, Breiman je teorijski i empirijski pokazao da kako broj stabala raste, to će šuma biti sve točnija, ali i da će u jednom trenutku doći u zasićenje i neće naučiti šum
- Osigurava visoku točnost rezultata na većini problema zbog **velike snage pojedinog stabla** i **niske korelacije (značajne različitosti) među stablima**

Slučajna šuma – primjena

- **Jedan od prvih algoritama koje treba isprobati za klasifikaciju**
- Najbolji za skupove podataka s **velikim brojem značajki**
- Ugrađeni odabir značajki pri izgradnji stabala
- Omogućuje klasifikaciju u više razreda ciljne značajke
- **Nedostatci:** osjetljivost na nebalansiranost klasa, nije dobar algoritam za modeliranje linearnog odnosa između prediktivnih i ciljne značajke
- Značajna primjena algoritma u širokom spektru znanstvenih istraživanja i u industriji:
  - Biomedicina
  - Genetika i bioinformatika
  - Industrija (elektroenergetika, promet, proizvodni procesi)

Slučajna šuma – početne postavke

- Ansambl se sastoji od *K* slučajnih stabala odluke $h_1(\bar{X}), h_2(\bar{X}),..., h_K(\bar{X})$, svaki s **vlastitim skupom** za učenje
- Određuje se broj $m = \sqrt{M}$, gdje je *M* ukupan broj značajki u skupu. *m* je broj značajki koje će se **razmatrati prilikom podjele svakog čvora** u stablu
  - Broj značajki koji treba razmatrati u svakom čvoru Breiman je dobio iskustveno, isprobavanjem na raznim skupovima podataka
  - Kasnije je pokazano da ovaj broj nije optimalan za sve probleme, ali je za većinu skupova blizu optimalnog pa se takav najčešće uzima
- Postavlja se najmanji dozvoljeni broj primjeraka u čvoru koji nije list na $n_{min} = 2$
- Postavlja se broj stabala na *K* = 100

Slučajna šuma – algoritam

Za svako stablo:

1. Provodi se uzorkovanje tipa ***bootstrap***
2. Za svaki čvor u stablu:

   Ako čvor nije list*:

   - odaberi *m* značajki kandidata iz skupa svih značajki *M*
   - generiraj *m* podjela (za svaku značajku kandidata po jednu), takvu da je podjela najinformativnija za dotičnu značajku (prema mjeri nečistoće Gini indeks)
   - uzmi najinformativniju od *m* podjela, podijeli primjerke u čvoru na lijevi i desni čvor prema toj podjeli i pokreni 2. korak prvo za lijevi, zatim za desni čvor (stablo se gradi u dubinu)

   Inače: vrati list označen s najčešćom klasom

* Čvor je list ako ima manje primjeraka od n_{min} ili ako su sve značajke kandidati s konstantnim vrijednostima u čvoru ili ako je ciljna značajka *y* konstantna u čvoru. Svako stablo najčešće se gradi do kraja.

Gini indeks

- Jedna od nekoliko **poznatih mjera za procjenu vrijednosti podjele** podataka na značajki (engl. *feature split*) (uz informacijsku dobit, MDL, itd.)
- **Mjeri nečistoću** (engl. *impurity measure*) u razdiobi vrijednosti ciljne klase prije i poslije podjele na značajki

$$Gini(Z_i) = -\sum_{i=1}^{c} p(y_i)^2 + \sum_{j=1}^{m_i} p(v_{i,j}) \sum_{i=1}^{c} p(y_i|v_{i,j})^2$$

- pri čemu je *c* broj klasa ciljne značajke, $p(y_i)$ je vjerojatnost pojave klase *i* u ciljnoj značajki *y*, m_i je broj vrijednosti značajke Z_i , $p(v_{i,j})$ je vjerojatnost da značajka Z_i ima vrijednost v_j , a $p(y_i|v_{i,j})$ je vjerojatnost pojave klase *i* u ciljnoj značajki *y* uz uvjet da značajka Z_i ima vrijednost v_j (tj. ako je podjela na značajki izvršena)

Slučajna šuma – testiranje

- Šuma se testira kao i većina ostalih algoritama strojnog učenja
- Nakon što se izgradi na skupu za učenje, testira se na novom skupu za testiranje
- Algoritam koristi **većinsko glasanje** (engl. *majority voting*) kod klasifikacijskih problema, a **usrednjavanje** (engl. *averaging*) kod predviđanja vrijednosti numeričkog atributa.
- Ako postoji samo jedan skup podataka, tada ovisno o broju uzoraka potrebno je **prije izgradnje** šume provesti:
  - Odvajanje na skup za učenje i skup za testiranje (obično 67% – 33%, ali ovisno o problemu) – za više od par tisuća primjeraka
  - 10x unakrsna validacija – za sve između 100 i par tisuća primjeraka
  - *Leave-one-out* unakrsna validacija (LOOCV) – za manje od 100 primjeraka

Iznimno slučajna stabla

- Ansambli **iznimno slučajnih stabala** nastoje popraviti točnost slučajnih šuma tako što uvode neke **dodatne izvore slučajnosti, dok neke zanemaruju**
- Mogu se podijeliti na **iznimno slučajna stabla** (engl. *extremely randomized trees, Extra-Trees*) i **potpuno slučajna stabla** (engl. *totally randomized trees*)
- Iznimno slučajna stabla slučajno odabiru podjelu neke značajke na čvoru, ali izabiru najbolju takvu podjelu (prema mjeri Gini) između slučajno odabranih $\sqrt{M}$ značajki u čvoru
- Potpuno slučajna stabla su iznimno slučajna stabla koja u svakom čvoru **slučajno biraju značajku koju slučajno dijele**
- Vidjeti: `sklearn.ensemble.ExtraTreesClassifier`

Iznimno slučajna stabla

- Za razliku od slučajne šume, **ne izdvajaju skup OOB (ne rade *bootstrap*)**, nego su koncentrirana samo na točnost predviđanja – na svakom se stablu koristi čitav skup primjeraka za učenje
- Iznimno slučajna stabla su na klasifikacijskim problemima 2–10 puta **brža** od slučajnih šuma na umjereno velikim skupovima podataka, ali zauzimaju 2–4 puta **više prostora** jer generiraju više čvorova i listova
- Ubrzanja i potrošnja su značajno veći pri korištenju potpuno slučajnih stabala
- Potreban broj stabala se pokazuje da je sličan kao i kod slučajnih šuma, točnost je također slična
- Koriste se jednako uspješno i u predviđanju numeričkih vrijednosti ciljnog atributa

Rotacijska šuma

- Ansambl klasifikacijskih stabala odluke
- Zasniva se na ***bootstrap* uzorkovanju**, **analizi glavnih komponenti (PCA)** i **ansamblu stabala odluke C4.5**
- Prednosti: postiže vrhunske rezultate (među najboljima) na skupovima podataka s numeričkim vrijednostima, može se paralelizirati
- Nedostatak: značajno sporiji od slučajne šume na velikim skupovima podataka (postoje raspodijeljene implementacije)
- Vidjeti: https://pypi.org/project/rotation-forest/

J. J. Rodriguez, L. I. Kuncheva and C. J. Alonso, "Rotation Forest: A New Classifier Ensemble Method," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 10, pp. 1619-1630, Oct. 2006, doi: 10.1109/TPAMI.2006.211.

Anthony J. Bagnall, Aaron Bostrom, Gavin C. Cawley, Michael Flynn, James Large, Jason Lines, Is rotation forest the best classifier for problems with continuous features? https://arxiv.org/abs/1809.06705 2018, 2020.

Mario Juez-Gil, Álvar Arnaiz-González, Juan J. Rodríguez, Carlos López-Nozal, César García-Osorio, Rotation Forest for Big Data, Information Fusion, Volume 74, 2021, Pages 39-49, https://doi.org/10.1016/j.inffus.2021.03.007.

Rotacijska šuma

- **Definira *L* klasifikatora C4.5: $D_1 \dots D_L$** od kojih se svaki može neovisno (u paraleli) učiti (*L* je parametar, *default* je 10)
- Za svaki klasifikator, skup značajki *F* dijeli se na slučajan način u ***K* podskupova** (*K* je hiperparametar, *default* je 3) bez ponavljanja – sve značajke se odaberu u neki od podskupova
- Za svaki takav podskup značajki F_j izabere se na slučajan način neprazni **podskup klasa ciljne značajke** (svaka klasa ima 50% šanse za upasti, npr. za tri klase, odaberu se dvije) te se od primjeraka tih klasa odabere *p*% (npr. 75%) primjeraka koristeći *bootstrap* uzorkovanje (alternativa: može i bez ponavljanja)
  - Ovo se sve radi u cilju kako bi skupovi primjeraka za učenje između klasifikatora bili što više različiti
- Na svakom podskupu značajki F_j za njegov podskup primjeraka pokreće se PCA i pamte se **sve njezine glavne komponente** (PC) $\boldsymbol{a}_{i,j}^{(1)}, \boldsymbol{a}_{i,j}^{(2)}, \dots, \boldsymbol{a}_{i,j}^{(M_j)}$, dok je $M_j \leq M = \|F_j\|$
  - M_j nije nužno jednako *M* jer je moguće da neke vlastite vrijednosti budu 0

Rotacijska šuma

- Od svih podskupova *K* formira se rotacijska matrica R_i kojom se množe svi ulazni podaci za stablo te je konačan skup ulaznih podataka u ansambl jednak $X \times R_i^{(a)}$, gdje je *X* originalni skup podataka, a (*a*) je oznaka da se matrica R_i preuredi tako da značajke odgovaraju početnom skupu

$$R_i = \begin{bmatrix} \mathbf{a}_{i,1}^{(1)}, \mathbf{a}_{i,1}^{(2)}, \ldots, \mathbf{a}_{i,1}^{(M_1)}, & [\mathbf{0}] & \cdots & [\mathbf{0}] \\ [\mathbf{0}] & \mathbf{a}_{i,2}^{(1)}, \mathbf{a}_{i,2}^{(2)}, \ldots, \mathbf{a}_{i,2}^{(M_2)}, & \cdots & [\mathbf{0}] \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{0}] & [\mathbf{0}] & \cdots & \mathbf{a}_{i,K}^{(1)}, \mathbf{a}_{i,K}^{(2)}, \ldots, \mathbf{a}_{i,K}^{(M_K)} \end{bmatrix}$$

- *L* stabala odluke C4.5 uče se pomoću tako transformiranih ulaznih podataka te se odluka o ciljnoj klasi testnog primjerka donosi većinskim glasanjem, kao i kod slučajne šume

AdaBoost i MultiBoost

- AdaBoost i MultiBoost algoritmi su ansambala temeljeni na **iterativnom pristupu izgradnji modela korištenjem postupka *boosting* nad inicijalno slabim modelom**
- U izvornom članku, inicijalno slabi model koji koristi AdaBoost je **panj odluke** (engl. *decision stump*)
  - Sastoji se od korijena (panja) i dva lista s primjercima koji se dobivaju grananjem u panju po nekoj značajci (koristeći npr. informacijski dobitak kao mjeru)
  - Tijekom iteracija AdaBoost koristi sve veću šumu panjeva da donese ispravne odluke
  - Nakon *K* koraka iteriranja izgrađeni model **koristi se zajednički** pri donošenju odluke
- **Prednosti**: učinkovita i brza izgradnja modela, visoka točnost pogotovo za binarne probleme
- **Nedostatci**: nije uvijek točniji od *bagging* metoda, može lakše prenaučiti skup podataka
- Vidjeti: `sklearn.ensemble.AdaBoostClassifier`

Y. Freund, R. E. Schapire. "Experiments with a New Boosting Algorithm." *ICML* (1996).
J. Zhu, H. Zou, S. Rosset, T. Hastie, "Multi-class adaboost." Statistics and its Interface 2.3 (2009): 34

AdaBoost i MultiBoost

- Na početku AdaBoosta svi primjerci imaju jednaku težinu, $w_0 = 1/n$, gdje je *n* broj primjeraka u skupu za učenje te se oni predoče slabom klasifikatoru
- Izabere se ona značajka (i njezin panj) koji daje najtočniji model pri grananju
- Izračuna se ukupna pogreška (gubitak) modela te koeficijent pogreške:

$$\alpha = \frac{1}{2} ln \frac{1-error}{error}$$

- **Težine primjeraka** u skupu revidiraju se prema tome je li primjerak bio **ispravno (težina pada)** ili **neispravno (težina raste)** klasificiran u panju:

$$w_i = w_{i-1} * e^{\pm\alpha}$$

- U svakoj sljedećoj iteraciji gradi se novi panj (stari ostaju) te se nastoji poboljšati točnost klasifikacije
- Ukupna odluka nakon *K* koraka kombinira pojedinačne odluke te odabire onu klasu s **utežanom većinskom odlukom** za testni primjerak (utežanost prema točnosti pojedinih klasifikatora na skupu za učenje)

AdaBoost i MultiBoost

- **MultiBoost** je varijanta AdaBoosta u kojoj se koristi stablo odluke **C4.5** u svakoj iteraciji umjesto panja odluke
- Također, MultiBoost koristi **utežani *bagging*** (engl. *wagging*) klasifikatora
  - Primjerci koji ulaze u MultiBoost imaju **slučajno izabranu težinu** na početku prema Poissonovoj razdiobi (log Random((1..999)/1000)), a ne 1/n za sve primjerke, čime se pospješuje raznolikost pri učenju klasifikatora
  - Hiperparametri algoritma su broj iteracija *K* i broj podansambala odluke *L*
  - Provodi se algoritam AdaBoost, svakih $t = K / L$ iteracija algoritma prelazi se u **novi podansambl**, prilikom čega se ponovi *wagging* na primjercima kako bi **resetirao težine** na nove slučajno izabrane težine
  - Računanje pogreške i revidiranje težina slični su kao kod AdaBoosta, donošenje odluke također
- U praksi, MultiBoost često pobjeđuje AdaBoost po pitanju točnosti jer kombinira značajnu uspješnost *bagginga* u smanjenju varijance modela

Benbouzid, D., R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl. 2012. "MultiBoost: a Multi-Purpose Boosting Package." *Journal of Machine Learning Research* 13: 549–553.
Webb, G.I. MultiBoosting: A Technique for Combining Boosting and Wagging. Machine Learning 40, 159–196 (2000). https://doi.org/10.1023/A:1007659514849

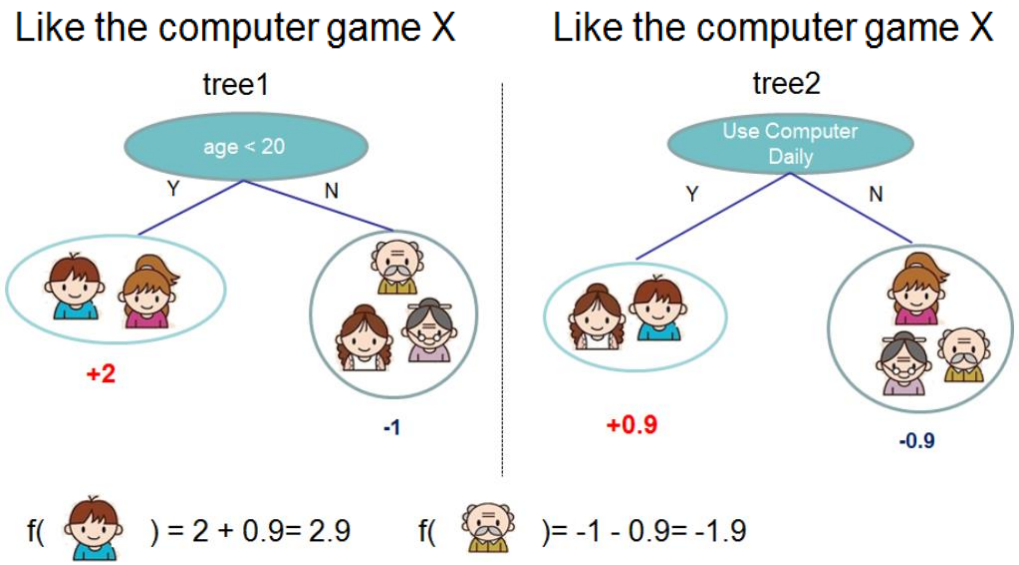
XGBoost

- *e**X**treme **G**radient **Boost**ing* (2016.) – napredan algoritam strojnog učenja temeljen na postupku ***boostinga*** i **optimizaciji gradijentnim spustom** (engl. *gradient boosting, gradient tree boosting*)
- Kod „gradijentnog boostinga”, *boosting* se razmatra kao problem numeričke optimizacije gdje je cilj **minimizirati gubitak modela** (proizvoljna funkcija gubitka) u svakom koraku tako da se u ukupni model dodaju slabi modeli iterativno, koristeći proceduru **sličnu** minimizaciji gradijentnim spustom
- Novi modeli nadodaju se postojećim starim modelima (stari se ne uklanjaju), kao i kod običnog *boostinga*
- Kao slabi model koriste se **stabla odluke CART** (engl. *classification and regression trees*), koja su asimetrična
  - Obično podrezana na neku kraću dubinu (npr. 4 – 8 nivoa), da budu malo točnija od panja, a opet brza za izgradnju
  - Razna ograničenja: broj čvorova, broj listova, dubina, min. broj primjeraka u čvoru da se radi grananje, min. poboljšanje gubitka pri grananju
  - Za grananje značajki u čvorovima koristi se indeks Gini
- Vidjeti: https://xgboost.readthedocs.io/en/stable/index.html

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

XGBoost

- Gradi se određeni, unaprijed definirani, broj CART stabala *K*
- CART stabla umjesto ciljne klase imaju u listovima realni broj koji oslikava skup primjeraka – taj broj (rezidual) dobiva se u svakom koraku izgradnje ansambla kao prosjek pogrešaka primjeraka u listovima u odnosu na stvarnu vrijednost



https://xgboost.readthedocs.io/en/stable/tutorials/model.html

XGBoost

- Dodavanje novog stabla u model (izgradnja stable) uzima u obzir **minimizaciju zbroja**:
  - **Gubitka na skupu za učenje** (MSE za regresijske probleme, logistički gubitak za klasifikacijske probleme) i
  - **Regularizacijskog gubitka zbog složenosti stabla** (funkcija koja ovisi o broju listova i vrijednostima u njima)
- Novo stablo se gradi tako da pokuša smanjiti pogrešku (reziduale) iz prethodne iteracije algoritma
- Kako XGBoost daje predikciju vrijednosti pojedinog primjerka:
  - Predikcija = izvorna predikcija + stopa_učenja * suma reziduala primjerka u pojedinačnim stablima; izvorna predikcija je minimum funkcije gubitka na skupu za učenje

XGBoost

- U odnosu na izvorni *gradient boosting*, algoritam XGBoost donosi **paralelizaciju učenja**: na distribuiranim platformama (npr. Spark), običnu paralelizaciju na CPU *threadovima* ili na GPU *threadovima*
- **Pretpostavke uspješnosti XGBoosta**: broj pojedinaca puno veći od broja značajki, stršeće vrijednosti su uklonjene iz skupa (one su loše za *boosting* postupke), podaci nisu rijetki
- XGBoost može **prenaučiti podatke**, što je generalna sklonost algoritama temeljenih na *gradient boostingu*, ali se to može kontrolirati **pravilnim izborom hiperparametara**:
  - Bolje je graditi više stabala manje dubine nego manje stabala veće dubine
  - Bolje je povećati broj iteracija učenja i smanjiti stopu učenja
  - Predlaže se koristiti inicijalno slučajno pridruživanje težina primjercima po određenoj razdiobi

CatBoost

- Kraće od: *Category Boosting*, 2017., AI tvrtka Yandex
- *Gradient boosting* algoritam usporediv s XGBoostom (ili čak i bolji)
- Glavna značajka: **ne zahtijeva korisničku konverziju kategoričkih** (i string) podataka u brojeve (za razliku od XGBoosta), već to radi sam od sebe
  - Detaljnije: https://catboost.ai/en/docs/concepts/algorithm-main-stages_cat-to-numberic
- Gradi simetrična stabla, brzo se gradi za veće skupove podataka
- Ima izvrsno podešenu *defaultnu* hiperparametrizaciju radi izbjegavanja prenaučenosti, ali omogućuje i detaljnije istraživanje vrijednosti brojnih hiperparametara
  - https://catboost.ai/en/docs/concepts/parameter-tuning

Postupci objašnjavanja modela ansambala

Postupci objašnjavanja modela ansambala

- Objašnjavanje modela ansambala može biti teško
  - Ansambl nije izgrađen u obliku jednostavno razumljivih pravila
- Ansambli stabala odluke pogodni su za objašnjavanje u kontekstu **procjene važnosti pojedinačnih značajki – sličan pristup može se koristiti i za odabir značajki**
  - Npr. slučajna šuma i iznimno slučajna stabla mogu koristiti metodu **TreeInterpreter**:
    - https://github.com/andosa/treeinterpreter
    - za detalje vidjeti: http://blog.datadive.net/random-forest-interpretation-with-scikit-learn/
- Predložen je veći broj postupaka za procjenu važnosti značajki za **općenite** modele crne kutije, a razmatrat će se:
  - Permutacijska važnost (engl. *Permutation Importance*)
  - SHAP

Permutacijska važnost

- Predložio L. Breiman, 2001., spada u objašnjive modele globalnog tipa (objašnjava se **cjelokupni model**)
- Permutacijska važnost značajke definira se kao **smanjenje rezultantne mjere modela kada se vrijednosti te značajke po primjercima nasumično promiješaju**
  - Značajka je to važnija što permutiranje njezinih vrijednosti više smanjuje rezultantnu mjeru modela
- Miješanje vrijednosti primjeraka **prekida odnos između značajke i ciljne značajke** pa pad rezultatne mjere modela pokazuje koliko model ovisi o svakoj značajci

Vidjeti: https://scikit-learn.org/stable/modules/permutation_importance.html

L. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001.

Permutacijska važnost

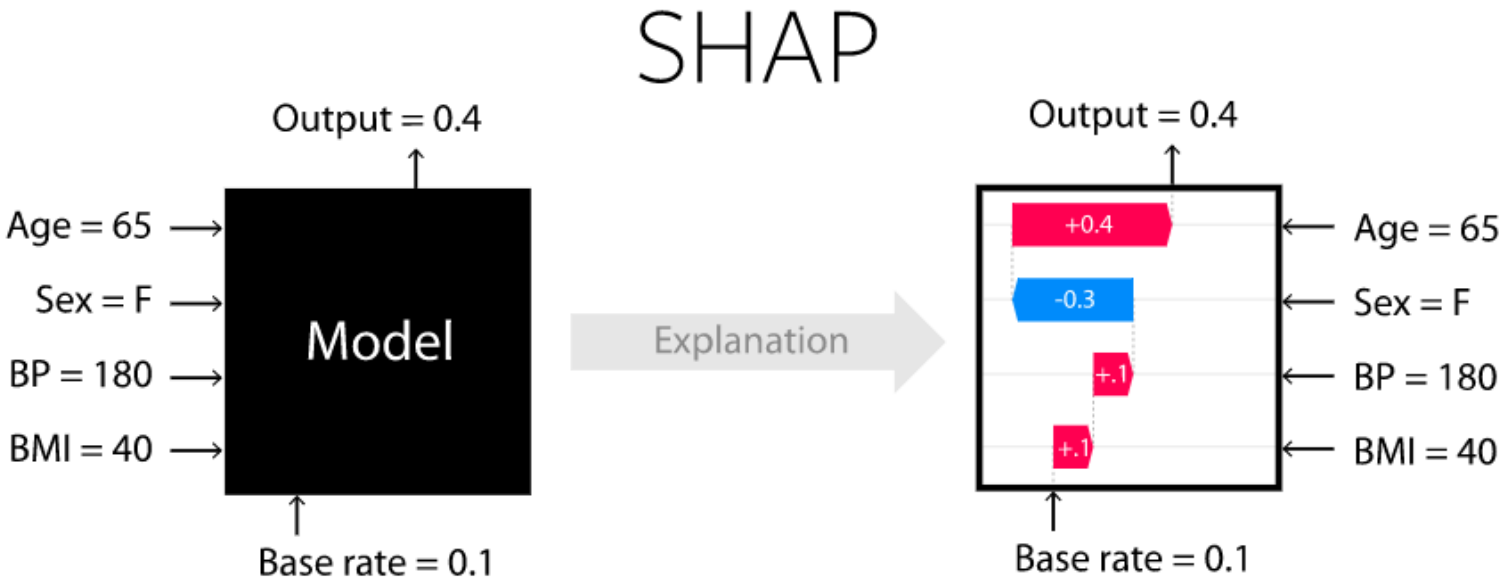
- Važnost i_j značajke F_j iznosi:

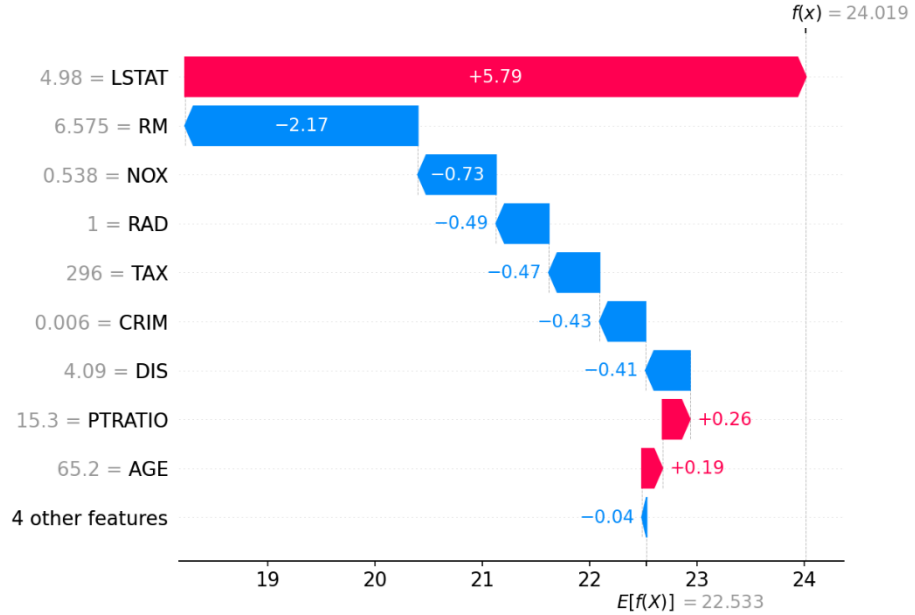
$$i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j}$$

- gdje je *s* inicijalna rezultantna mjera modela (npr. točnost), a $s_{k,j}$ je rezultantna mjera modela za skup podataka s **permutiranim vrijednostima** značajke *j*
- Hiperparametri su:
  - broj ponavljanja *K* permutiranja vrijednosti značajke za procjenu važnosti značajke
  - izbor rezultantne mjere modela (paziti na nebalansiranost klasa!)
- Prednosti korištenja: neovisna je o algoritmu strojnog učenja, radi i za klasifikacijske i regresijske algoritme
- Najbolje je da se određuje na **izdvojenom validacijskom skupu** (npr. OOB skup kod slučajne šume)

SHAP

- Kraće: ***SH**apley **A**dditive ex**P**lanations*, Lundberg, 2017., objašnjivi postupak **lokalnog tipa**
- Postupak koji se koristi za objašnjavanje predikcije primjerka bilo kojeg modela strojnog učenja
- Model koji je crna kutija objašnjava se u vidu vizualizacije doprinosa pojedinih značajki predikciji primjerka
- Trenutačno radi sa scikit-learnom, XGBoostom, CatBoostom i nekim drugim modelima
- Vidjeti: https://shap.readthedocs.io/en/latest/

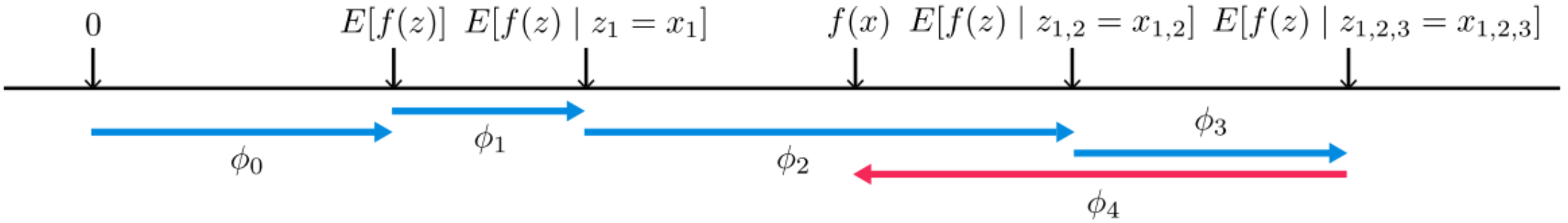




Izvor: https://github.com/slundberg/shap

SHAP

- **Značajke** doprinose **pomaku izlaza modela od bazne vrijednosti** (prosječnog izlaza iz modela na skupu za učenje bez prediktivnih značajki) prema stvarnom izlazu iz modela za **određeni primjerak**
  - Značajke koje guraju predviđanje modela na više od bazne vrijednosti označene su crveno, a one koje guraju na manje označene su plavo
  - Značajke se obično sortiraju po magnitudi utjecaja neovisno o smjeru



Izvor: S. M. Lundberg, S.-I. Lee, A Unified Approach to Interpreting Model Predictions}, NeuroIPS, vol. 30, pp. 4765-4774, 2017

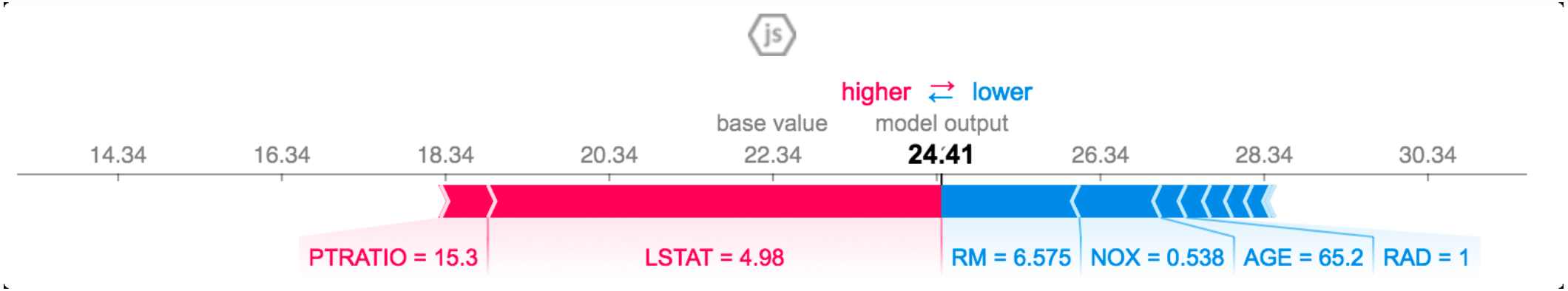
- SHAP vrijednosti φ_i pridružuju svakoj značajki **promjenu u očekivanoj predikciji** modela uz uvjet da se ta značajka koristi za predikciju
- Inicijalna bazna vrijednost modela E[$f(z)$] označava **model bez značajki čiji je izlaz samo srednja vrijednost ciljne klase**
- Svaka značajka doprinosi na svoj negativan (φ_1, φ_2, φ_3) ili pozitivan način (φ_4) na konačnu vrijednost izlaza primjerka f(x)
- **Poredak dodavanja značajki modelu ima veze** pa se točne SHAP vrijednosti dobivaju **usrednjavanjem** φ_i preko svih mogućih poredaka dodavanja značajki

Detalji: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

SHAP – *force plot*

- SHAP-ov ***force plot*** alternativni je način vizualizacije utjecaja značajki na model (u odnosu na vodopadni prikaz od dva slajda ranije), koristeći na isti način dobivene SHAP vrijednosti
- Na ovoj slici naveden je utjecaj na temelju samo jednog primjera:

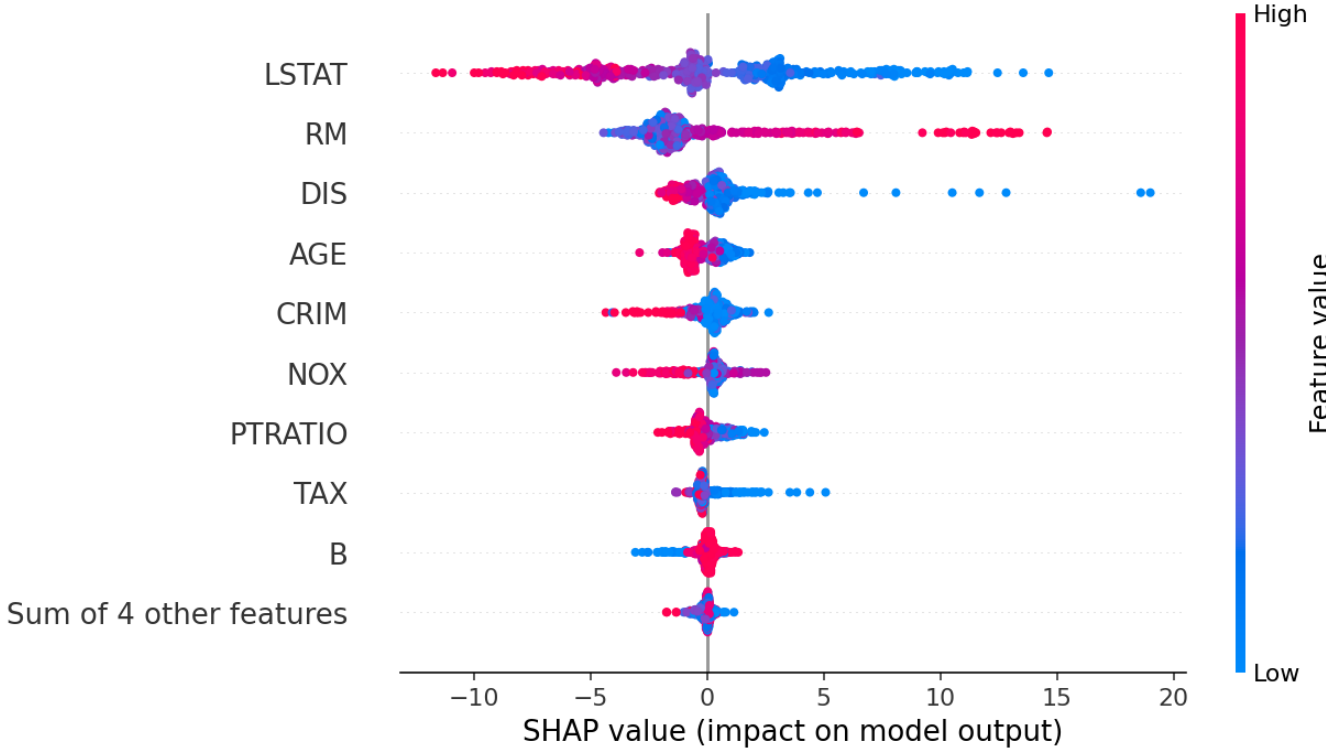
```
shap.plots.force(shap_values[0])
```



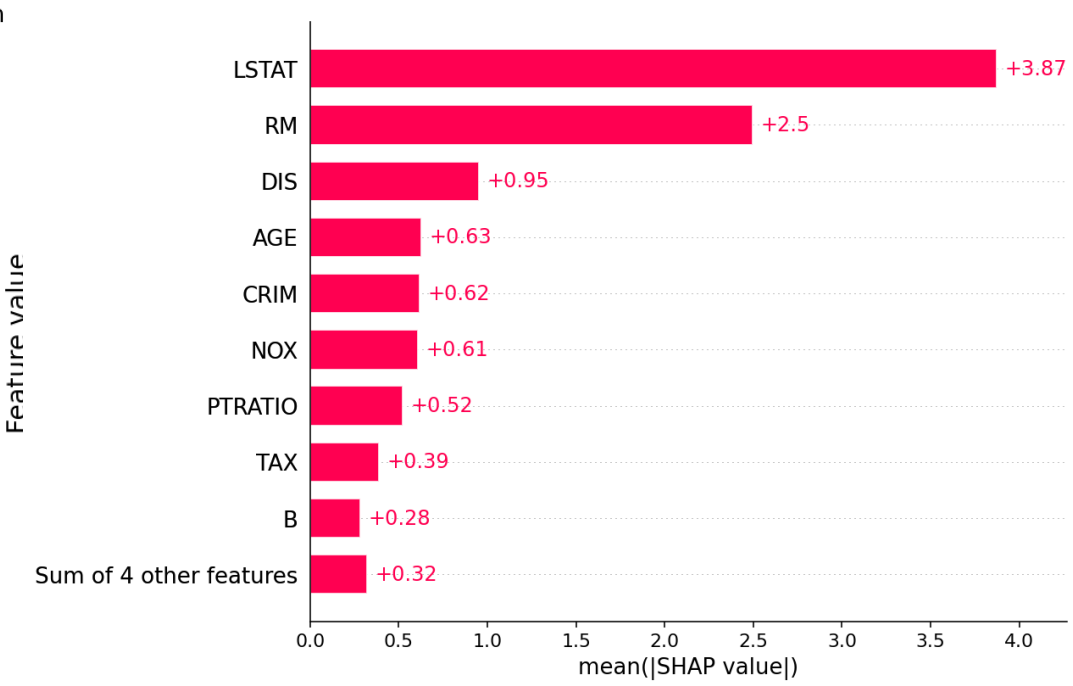
Izvor: Lundberg, S.M., Nair, B., Vavilala, M.S. *et al.* Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nat Biomed Eng* **2,** 749–760 (2018).

SHAP – *beeswarm*

- Ako želimo istovremeno prikazati utjecaj svih značajki korištenjem **svih primjeraka** na model pomoću SHAP vrijednosti (globalni postupak), možemo nacrtati **dijagram roja pčela (*beeswarm*)** – prikazuje sve primjerke ili stupčasti graf (*bar chart)* – prikazuje srednju apsolutnu SHAP vrijednost po svim primjercima



`shap.plots.beeswarm(shap_values)`



`shap.plots.bar(shap_values)`

Izvor: https://github.com/slundberg/shap

SHAP prostor metoda vizualizacije

- SHAP za objašnjenje stabala odluke
  - TreeExplainer, npr: https://slundberg.github.io/shap/notebooks/NHANES%20I%20Survival%20Model.html
- SHAP za objašnjenje dubokih modela
  - DeepExplainer, npr. https://slundberg.github.io/shap/notebooks/deep_explainer/Front%20Page%20DeepExplainer%20MNIST%20Example.html
  - GradientExplainer, npr.
  - https://slundberg.github.io/shap/notebooks/gradient_explainer/Explain%20an%20Intermediate%20Layer%20of%20VGG16%20on%20ImageNet.html

Lundberg, S.M., Erion, G., Chen, H. *et al.* From local explanations to global understanding with explainable AI for trees. *Nat Mach Intell* **2,** 56–67 (2020).

Zaključak

- Ansambli se dijele na sustave više eksperata (npr. *bagging*, *stacking*) i iterativne sustave (npr. *boosting, gradient boosting*)
- Postoji veliki broj specifičnih algoritama za pojedinačne pristupe izgradnji ansambala (npr. slučajna šuma, iznimno slučajna stabla, AdaBoost, XGBoost)
- Ansambli načelno postižu mnogo bolje rezultate od individualnih modela
  - Preporuka je najprije isprobati slučajnu šumu, potom ostale, računski zahtjevnije postupke
- Interpretacija složenih modela ansambala i drugih modela crne kutije nije jednostavna
  - Interpretacija na cijelom skupu podataka ili za individualni primjerak
  - Veći broj postupaka, ali preporuka je isprobati SHAP