

# Usporedba algoritama temeljenih na paradigmi gradient boostinga: Primjena na stvarne medicinske i fizikalne skupove podataka

---

Brkičić, Alan

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:217:145560>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-02-23**



Repository / Repozitorij:

[Repository of the Faculty of Science - University of Zagreb](#)



SVEUČILIŠTE U ZAGREBU  
PRIRODOSLOVNO-MATEMATIČKI FAKULTET  
FIZIČKI ODSJEK

Alan Brkičić

USPOREDBA ALGORITAMA TEMELJENIH NA  
PARADIGMI GRADIENT BOOSTINGA:  
PRIMJENA NA STVARNE MEDICINSKE I  
FIZIKALNE SKUPOVE PODATAKA

Diplomski rad

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU  
PRIRODOSLOVNO-MATEMATIČKI FAKULTET  
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ  
FIZIKA I INFORMATIKA; SMJER NASTAVNIČKI

**Alan Brkičić**

Diplomski rad

**Usporedba algoritama temeljenih na  
paradigmi gradient boostinga:  
Primjena na stvarne medicinske i  
fizikalne skupove podataka**

Voditelj diplomskog rada: prof. dr. sc. Davor Horvatić

Ocjena diplomskog rada: \_\_\_\_\_

Povjerenstvo: 1. \_\_\_\_\_

2. \_\_\_\_\_

3. \_\_\_\_\_

Datum polaganja: \_\_\_\_\_

Zagreb, 2024.

Zahvaljujem se mentoru prof. dr. sc. Davoru Horvatiću na provedbi teme ovog diplomskog rada.

Zahvaljujem se i doc. dr. sc. Andreju Novaku na mentorstvu i na predlaganju teme ovog diplomskog rada, kroz koji sam mnogo naučio o strojnom učenju. Međutim, najviše sam zahvalan na njegovim izvrsnim predavanjima, zahvaljujući kojima sam naučio i zavolio programiranje.

Zahvaljujem se i profesoru Šimi Manoli, doc. dr. sc. Marinu Pavlovu, dr. sc. Ivani Jurin koji su sudjelovali u prikupljanju podataka za registar iz plućnih embolija, koji sam koristio u sklopu medicinskih primjena strojnog učenja.

Zahvaljujem se od srca svojoj obitelji i Jani na godinama podrške i potpore tijekom mog studiranja. Bez vas ne bih mogao svoje studiranje privesti kraju jednako lijepo kao s vama.

Posebno pozdravljam i sve svoje prijatelje i kolege koji su učinili moje studiranje nezaboravnim i punim lijepih uspomena.

## Sažetak

U ovom radu se proučavaju razni pristupi strojnom učenju s naglaskom na algoritme bazirane na paradigmi gradient boostinga, ulazi se u njihovu matematičku pozadinu i u dubine funkcioniranja algoritama. Uspoređuje se efikasnost gradient boostinga, XGBoosta, LightGBM-a, CatBoosta i slučajne šume na dva skupa podataka. Skup podataka Higgsovog bozona nastoji identificirati Higgsov bozon unutar raznih signala. Skup podataka za emboliju nastoji predvidjeti hoće li pacijent preživjeti do svoje sljedeće kontrole. Rezultati navedenih algoritama se mjere njihovom točnošću, opozivom, preciznošću i ROC-AUC vrijednosti. Mjeri se i značajnost značajki pripadnih skupova podataka pomoću Shapleyjevih vrijednosti.

Ključne riječi: Strojno učenje, Gradient boosting, XGBoost

# Comparison of algorithms based on the gradient boosting paradigm: application to real medical and physical datasets

## Abstract

This thesis examines various approaches to machine learning, with a focus on algorithms based on the gradient boosting paradigm. It delves into their mathematical foundations and the intricacies of their inner workings. The efficiency of gradient boosting, XGBoost, LightGBM, CatBoost, and random forests is compared using two datasets: the Higgs boson dataset, which aims to identify the Higgs boson among various signals, and the embolism dataset, which seeks to predict patient outcomes based on disease severity and other comorbidities. The performance of these algorithms is evaluated using accuracy, recall, precision, and ROC-AUC values. Additionally, the significance of the features in each dataset is assessed using Shapley values.

Keywords: Machine learning, Gradient boosting, XGBoost

# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>1</b>
<b>2</b>	<b>Metode iz strojnog učenja</b>	<b>2</b>
2.1	Gradijentni spust . . . . .	5
2.2	Stabla odlučivanja . . . . .	7
2.3	Gradient boosting . . . . .	8
2.4	XGBoost . . . . .	13
2.5	Light gradient boosting machine . . . . .	19
2.6	CatBoost . . . . .	20
2.7	Slučajna šuma . . . . .	21
<b>3</b>	<b>Primjena modela</b>	<b>22</b>
3.1	Metrike . . . . .	22
3.2	Shapleyjeve vrijednosti . . . . .	23
3.3	Higgsov bozon . . . . .	24
3.4	Plućna embolija . . . . .	31
<b>4</b>	<b>Zaključak</b>	<b>39</b>
	<b>Dodaci</b>	<b>40</b>
<b>A</b>	<b>Kod za dobivanje rezultata na skupu podataka za Higgsov bozon</b>	<b>40</b>
A.1	Gradient boosting . . . . .	40
A.2	XGBoost . . . . .	42
A.3	LightGBM . . . . .	44
A.4	CatBoost . . . . .	46
A.5	Slučajna šuma . . . . .	47
<b>B</b>	<b>Kod za dobivanje rezultata na skupu podataka za plućnu emboliju</b>	<b>49</b>
B.1	Gradient boosting . . . . .	49
B.2	XGBoost . . . . .	51
B.3	LightGBM . . . . .	52
B.4	CatBoost . . . . .	53
B.5	Slučajna šuma . . . . .	54

<b>C Funkcija za dobivanje SHAP grafova</b>	<b>55</b>
<b>Literatura</b>	<b>57</b>



# 1 Uvod

Tom Michellova definicija strojnog učenja tvrdi da je računalni program naučio iz iskustva  $I$  u odnosu na zadatak  $Z$  i mjeru učinkovitosti  $U$  ako se njegova izvedba na  $Z$ , mjerena pomoću  $U$ , poboljša s iskustvom  $I$ . Takvi računalni programi se svakodnevno upotrebljavaju jer efikasno rješavaju probleme klasificiranja, regresije i rangiranja. Međutim, kako bi se ti problemi riješili potrebno je imati prikladan i uređen skup podataka u kojem računalni program može promatrati svaki primjer. Svi primjeri unutar istog skupa podataka su sastavljeni od istih značajki pomoću kojih algoritmi uče te stvaraju generalizaciju odnosa traženog izlaza o značajkama skupa podataka.

Algoritmi temeljeni na paradigmi gradient boostinga su algoritmi strojnog učenja koji stvaraju ansambl stabla odlučivanja u kojem svako novo stablo ispravlja greške prošlih. Od 1999. godine, kada je Jerome Friedman osmislio gradient boosting algoritam, izašlo je nekoliko algoritama koji poboljšavaju brzinu i preciznost gradient boostinga poput XGBoosta, CatBoosta i LightGBM-a. Algoritam slučajnih šuma također koristi ansambl stabla odlučivanja, ali ga stvara s međusobno nezavisnim stablima. Završno predviđanje daje težinskim prosjekom predviđanja svih stabla u ansamblu, dok gradient boosting algoritmi gledaju rezultat prvog stabla koji je ispravljen pomoću ostalih stabla.

Preciznost i učinkovitost navedenih modela će se uspoređivati pomoću raznih metrika na dva skupa podataka čiji je cilj riješiti klasifikacijski problem. Prvi skup podataka je preuzet s web stranice Kaggle koja je zajedno s CERN institutom održala natjecanje u kojem se tražio model koji može najbolje identificirati Higgsov bozon. Upravo u tom natjecanju se prvi put isticao XGBoost. Drugi skup podataka je u vlasništvu Kliničke bolnice Dubrava i sadrži podatke vezane za pacijente kojima je dijagnosticirana plućna embolija. Cilj je razviti model koji može predvidjeti ishode takvih pacijenata.

## 2 Metode iz strojnog učenja

Skup podataka predstavlja se kao matrica  $\mathcal{X}$  sastavljena od  $N$   $M$ -dimenzionalnih transponiranih stupac vektora  $x$ . Vektori predstavljaju primjere, a njihove dimenzije predstavljaju značajke skupa podataka. Algoritmi temeljeni na paradigmi gradient boostinga uče koristeći pristup nadziranog učenja. To jest, za svaki primjer unutar skupa podataka vektor  $y$  sadrži traženi izlaz pomoću kojeg model uspoređuje svoja predviđanja  $\hat{y}$ . U slučaju da je  $y$  neprekidna vrijednost, algoritmi rješavaju regresijski problem poput predviđanja vrijednosti kuće. Vektor  $y$  koji sadrži diskretne vrijednosti ukazuje na klasifikacijske probleme, na primjer razlikovanje znamenki u slikanom tekstu. Skupu podataka se mogu pridodati težine. Težine pomažu ispravljanju neuravnoteženih skupova podataka te sprječavanju nastanka pristranosti modela. Predviđanje  $i$ -tog primjera  $x_i$  možemo zapisati kao

$$\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^M x_{i,j} \hat{\beta}_j. \quad (2.1)$$

Uključivanjem konstante 1 u vektor  $x$ , prethodna jednadžba (2.1) može se izraziti kao skalarni umnožak

$$\hat{y} = x_i^T \hat{\beta}, \quad (2.2)$$

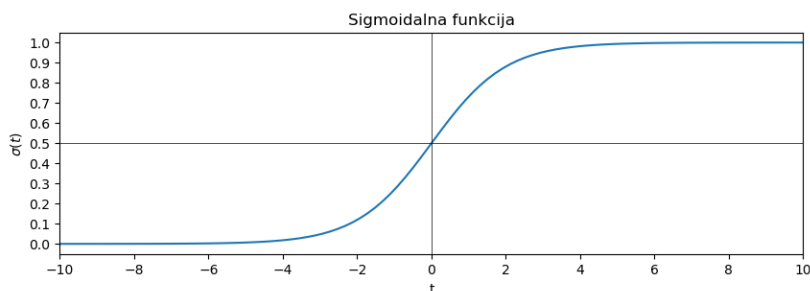
gdje  $\hat{\beta}$  predstavlja parametre stvorene pomoću modela. [1]

Model ispravlja svoje parametre minimiziranjem funkcije gubitka  $\mathcal{L}(y, \hat{y})$  koja kažnjava greške u predviđanju. Najčešća funkcija gubitka za regresijske probleme je srednja kvadratna greška

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2. \quad (2.3)$$

Za razumijevanje logaritamske regresije i pripadne jednadžbe gubitka, koje rješavaju probleme klasifikacije, prvo je potrebno razumjeti kako model predviđa klase. Kao i dosad model predviđa pomoću jednadžbe (2.2), ali sada to predviđanje dodatno obrađuje sigmoidalna funkcija

$$\sigma(x^T \hat{\beta}) = \sigma(t) = \frac{1}{1 + \exp(-t)}. \quad (2.4)$$



Slika 2.1: Grafički prikaz sigmoidalne funkcije

Rezultat sigmoidalne funkcije se može interpretirati kao vjerojatnost primjera da je član pozitivne klase. Ta vjerojatnost se označava s  $\hat{p}$ . Predviđanje modela je u tom slučaju

$$\hat{y} = \begin{cases} 1, & \hat{p} \geq 0,5 \\ 0, & \hat{p} < 0,5 \end{cases} \quad (2.5)$$

gdje 1 označava pozitivnu klasu, a 0 označava negativnu. Sada se može uvesti logaritamska funkcija gubitka

$$J(y, \hat{p}) = - \sum_{i=1}^N [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)], \quad (2.6)$$

u kojoj se  $y_i \log(\hat{p}_i)$  odnosi na predviđanje pozitivne klase, dok se  $\log(1 - \hat{p}_i)$  odnosi na predviđanje negativne klase. Ako model krivo predvidi klasu primjera, vrijednost pripadnog logaritma će rasti te kazniti pogrešno predviđanje. [2]

Jednadžbu (2.6) je korisno zapisati preko logaritma izgleda. Izgledi su omjer broja slučaja pozitivne klase ( $t$ ) i broja slučaja negativne klase ( $f$ ) u ukupno  $N$  primjera

$$\text{izg} = \frac{t}{f}. \quad (2.7)$$

Lagano je vidjeti kako se povezuju izgledi s vjerojatnošću  $\hat{p}$

$$\text{izg} = \frac{t}{f} \cdot \frac{\frac{1}{N}}{\frac{1}{N}} = \frac{\frac{t}{N}}{\frac{f}{N}} = \frac{\hat{p}}{1 - \hat{p}}. \quad (2.8)$$

Logaritam izgleda se koristi zbog simetričnosti svojih rezultata. Na primjer, ako se broji koliko puta u sedam bacanja kovanice padne pismo, za 6 dobivenih pisma i jednu glavu logaritam izgleda iznosi 1,8. Za obratni slučaj logaritam izgleda će iznositi  $-1,8$ . Koristeći logaritam izgleda za mjerenje pozitivne klase, vidimo da je

logaritam pozitivan kada u skupu podataka ima više članova pozitivne klase nego negativne. Članovi sume u jednažbi (2.6) mogu se preurediti da ovise o logaritmu izgleda

$$\begin{aligned}
 y \log(\hat{p}) + (1 - y) \log(1 - \hat{p}) &= y \log(\hat{p}) - y \log(1 - \hat{p}) + \log(1 - \hat{p}) \\
 &= y(\log(\hat{p}) - \log(1 - \hat{p})) + \log(1 - \hat{p}) \\
 &= y \log\left(\frac{\hat{p}}{1 - \hat{p}}\right) + \log(1 - \hat{p}) \\
 &= y \log(\text{izg}) + \log(1 - \hat{p}).
 \end{aligned}$$

Međutim, zadnji član i dalje ovisi o vjerojatnosti. Ako se logaritmira izraz (2.8), dobiva se ovisnost vjerojatnosti o logaritmu izgleda

$$\begin{aligned}
 \log\left(\frac{\hat{p}}{1 - \hat{p}}\right) &= \log(\text{izg}) \\
 \frac{\hat{p}}{1 - \hat{p}} &= \exp(\log(\text{izg})) \\
 \hat{p} &= \exp(\log(\text{izg})) - \hat{p} \exp(\log(\text{izg})) \\
 \hat{p}(1 + \exp(\log(\text{izg}))) &= \exp(\log(\text{izg})) \\
 \hat{p} &= \frac{\exp(\log(\text{izg}))}{1 + \exp(\log(\text{izg}))}. \tag{2.9}
 \end{aligned}$$

Uvrštavajući taj izraz u sumu dobiva se

$$\begin{aligned}
 y \log(\text{izg}) + \log(1 - \hat{p}) &= y \log(\text{izg}) + \log\left(1 - \frac{\exp(\log(\text{izg}))}{1 + \exp(\log(\text{izg}))}\right) \\
 &= y \log(\text{izg}) + \log\left(\frac{1}{1 + \exp(\log(\text{izg}))}\right) \\
 &= y \log(\text{izg}) - \log(1 + \exp(\log(\text{izg}))),
 \end{aligned}$$

što rezultira logaritamskoj funkciji gubitka ovisnoj o logaritmu izgleda

$$J(y, \log(\text{izg})) = - \sum_{i=1}^N [y_i \log(\text{izg})_i - \log(1 + \exp(\log(\text{izg})_i))]. \tag{2.10}$$

Taj oblik će biti koristan kasnije. [3]

Svaki model sadrži parametre koje korisnik može mijenjati. Ti parametri zovu se hiperparametri i sprječavaju pretreniranje modela. Pretreniranje (eng. *overfit-*

ting) se događa kada se model previše prilagodi skup podataka na kojem je treniran, uključujući šum i nesustavne varijacije. Zbog toga model gubi sposobnost dobre generalizacije na nove, nepoznate podatke, što negativno utječe na njegovu točnost i performanse na testnim skupovima. Iz tog se razloga skup podataka dijeli na skup za treniranje i skup za testiranje. Modeli uče na skupu za treniranje, dok se njihova preciznost mjeri na testnom skupu. Ako model pokazuje visoku preciznost na skupu za treniranje, a nisku preciznost na testnom skupu, to ukazuje na pretreniranje. Kako bi se dobile točne metrike modela i izbjeglo pretreniranje, model ne smije dobiti nikakve informacije iz testnog skupa tijekom učenja.

Različiti algoritmi na različite načine minimiziraju funkcije gubitka, što dovodi do razlika u rezultatima i brzini predviđanja. Međutim, važno je napomenuti da programer koji nema unaprijed postavljene pretpostavke o skupu podataka ne bi trebao preferirati jedan algoritam nad drugim. Bez obzira na složenost algoritma, on ne mora nužno biti bolji u rješavanju problema. David Wolpert je tu pojavu demonstrirao u svom radu<sup>1</sup>, što je rezultiralo No Free Lunch teoremom. [2]

## 2.1 *Gradijentni spust*

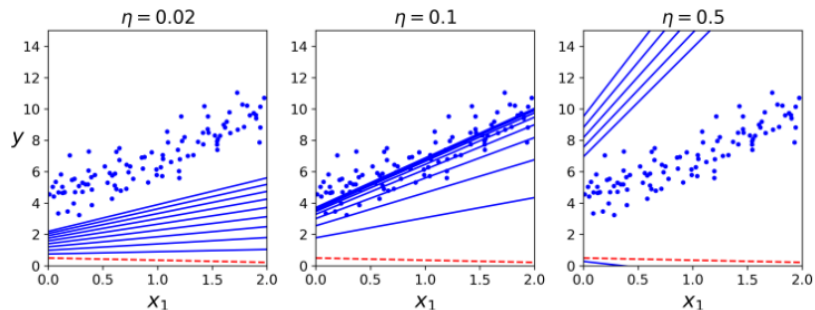
Algoritam gradijentnog spusta najjednostavnije se može razumjeti kao pronalazak najstrmijeg puta prema minimumu funkcije gubitka. Postupak gradijentnog spusta može se opisati sljedećim koracima:

1. Odabiru se nasumične vrijednosti za parametre  $\hat{\beta}$ .
2. Parcijalno se derivira funkcija gubitka s obzirom na svaki parametar, čime se dobiva gradijent funkcije gubitka  $\nabla_{\hat{\beta}} \mathcal{L}(y, \hat{y})$ .
3. Gradijent se množi stopom učenja  $\eta$ , čime se određuje veličina koraka promjene parametara.
4. Parametri se ažuriraju tako da se postojeće vrijednosti umanje za prethodno izračunati korak promjene.
5. Koraci 2.-4. se ponavljaju dok se ne postigne minimum funkcije gubitka.

---

<sup>1</sup>The Lack of A Priori Distinctions Between Learning Algorithms, D. Wolpert (1996)

Stopa učenja je hiperparametar koji određuje veličinu koraka promjene. Ako je stopa učenja prevelika, algoritam možda neće konvergirati prema minimumu, dok preniska stopa učenja može usporiti proces konvergencije. Uobičajeno je da algoritmi imaju zadanu stopu učenja od 0,1.



Slika 2.2: Utjecaj stope učenja na gradijentni spust [2]

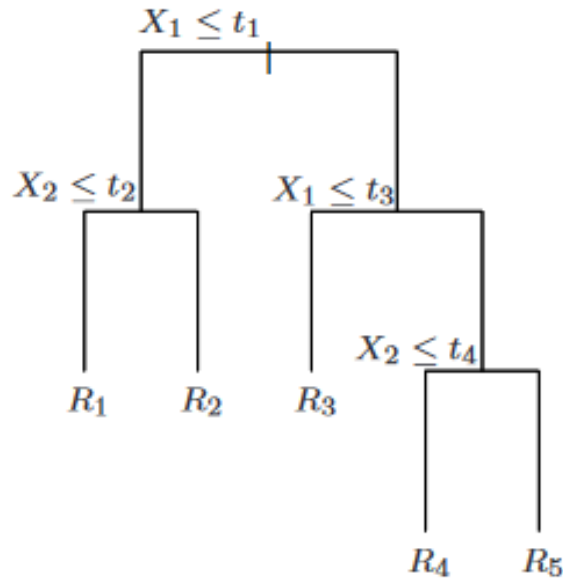
Problem gradijentnog spusta je što ne zna razlikovati lokalni minimum od globalnog. Zato uz gradijentni spust koristimo funkcije gubitka koje imaju jedan minimum, poput srednje kvadratne greške. [2]

Ako skup podataka ima velik broj primjera i značajki, gradijentni spust će biti dosta spor. Razlog tome leži u načinu računanja gradijenta koji uzima u obzir sve parametre i sve primjere. Stohastički i grupni gradijentni spust ubrzavaju taj proces nasumičnim odabirom primjera. Stohastički gradijentni spust nasumično bira jedan primjer po koraku pomoću kojeg računa gradijent, dok grupni gradijentni spust nasumično bira određeni postotak primjera. Jednim prolazom kroz cijeli skup podataka se na taj način uspijeva napraviti nekoliko koraka prema minimumu, ali svaki korak ne mora nužno voditi prema minimumu. Zbog nasumičnosti algoritama, neki koraci nas mogu udaljavati od minimuma, ali u prosjeku će voditi prema njemu.

Još jedna posljedica nasumičnosti stohastičkog i grupnog gradijentnog spusta je mogućnost izlaska iz lokalnih minimuma, što ovaj pristup čini povoljnijim u nekim situacijama. Algoritmi se mogu zaustaviti u točnom minimumu postepenim smanjivanjem stope učenja ili određivanjem broja ponavljanja algoritma, ali time se opet stvara mogućnost zaustavljanja u lokalnom minimumu. [2]

## 2.2 Stabla odlučivanja

Struktura stabla odlučivanja jednaka je strukturi binarnih stabala. Čvorovi određenom značajkom  $k$  i vrijednost  $t_k$  uspoređuju sve primjere. Podaci tada prolaze kroz novi čvor s drugim uvjetom ili dolaze do lista gdje se donosi konačni rezultat predviđanja.



Slika 2.3: Primjer stabla koji razdvaja podatke pomoću značajki  $X_1$  i  $X_2$  [1]

Vizualno čvorovi koji ispunjavaju uvjet prošlog čvora se nalaze na lijevoj strani, dok čvorovi koji ne ispunjavaju uvjet se nalaze na desnoj. Stabla biraju značajku i vrijednost značajke u čvorovima pomoću CART (eng. *Classification And Regression Tree*) algoritma i istoimene funkcije gubitka

$$J(k, t_k) = \frac{n_{\text{lijevo}}}{n} G_{\text{lijevo}} + \frac{n_{\text{desno}}}{n} G_{\text{desno}}, \quad (2.11)$$

gdje  $G$  predstavlja nečistoću čvora, a  $n$  predstavlja broj primjera u čvoru. Nečistoća  $i$ -tog čvora se računa pomoću gini vrijednosti

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2, \quad (2.12)$$

gdje  $p_{i,k}$  predstavlja postotak klase  $k$  među primjerima  $i$ -tog čvora. CART algoritam dijeli podatke do trenutka kad više ne može naći novu raspodjelu koja smanjuje nečistoću. Dijeljenje se može ranije zaustaviti hiperparametrima:

- `max_depth`- određuje maksimalanu dubinu stabla
- `min_samples_split`- određuje minimalan broj primjera koji mora biti u čvoru prije razdijele
- `min_samples_leaf`- određuje minimalan broj primjera koji se nalaze u listu
- `min_weight_fraction_leaf`- određuje minimalni postotak otežane sume svih težina podataka koji može biti u listu
- `max_leaf_nodes`- određuje broj mogućih listova u stablu

Uz CART algoritam postoje i C4.5 stabla odlučivanja koja koriste entropiju

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^n p_{i,k} \log_2(p_{i,k}), \quad (2.13)$$

pomoću koje se stvaraju više uravnotežena stabla. CART često grupira klase s više primjera na jednu stranu stabla što vodi neuravnoteženim stablima. Međutim, gini vrijednost se češće upotrebljava zbog bržeg izračuna uz beznačajnu razliku u preciznosti modela. [2]

Stabla razdvajaju podatke u različita odvojena područja, to jest listove  $R_j, j = 1, 2, \dots, J$ . Svako područje dodjeljuje podacima pripadnu vrijednost  $\gamma_j$  te se stabla mogu opisati na sljedeći način

$$T(x; \theta) = \sum_{j=1}^J \gamma_j I(x \in R_j), \quad (2.14)$$

gdje  $\theta = \{R_j, \gamma_j\}_1^J$  predstavlja parametre stabla. [1]

## 2.3 Gradient boosting

Gradient boosting stvara stabla odlučivanja koja su rezultat sume prošlih stabla

$$f_m(x) = \sum_{m=1}^M T(x; \theta), \quad (2.15)$$

s time da svako novo stablo u ansamblu mora imati parametre koji smanjuju grešku prošlog stabla

$$\hat{\theta}_m = \arg \min_{\theta_m} \sum_{i=1}^N \mathcal{L}(y_i, f_{m-1}(x_i) + T(x; \theta)). \quad (2.16)$$



Ako funkcija  $f(x)$  predviđa vrijednost  $y$ , njen gubitak se zapisuje kao

$$\mathcal{L}(f) = \sum_{i=1}^N \mathcal{L}(y_i, f(x_i)). \quad (2.17)$$

Cilj je minimizirati gubitak s ograničenjem da  $f(x)$  predstavlja sumu stabla odlučivanja zbog čega je problem (2.16) dosta složen. Taj problem se može riješiti numeričkom optimizacijom. Numeričkom optimizacijom ignoriramo ograničenje funkcije  $f(x)$  te se problem (2.17) svodi na

$$\hat{\mathbf{f}} = \arg \min_{\mathbf{f}} \mathcal{L}(\mathbf{f}). \quad (2.18)$$

”Parametri” rješenja  $\mathbf{f} \in \mathbb{R}^N$  su vrijednosti koje predviđa funkcija  $f$  za svaki primjer u skupu podataka. Numerička optimizacija rješava optimizaciju jednadžbe (2.17) pomoću sume komponentnih vektora

$$\mathbf{f}_M = \sum_{m=0}^M h_m, h_m \in \mathbb{R}^N, \quad (2.19)$$

gdje  $\mathbf{f}_0 = h_0$  označava prvo nagađanje, a svaki idući  $\mathbf{f}_m$  predstavlja zbroj prijašnjih parametarskih vektora. Razne metode numeričkih optimizacija se međusobno razlikuju pri načinu računanja vektora pomaka  $h_m$ , to jest koraka. [1]

Korak se može izračunati pomoću gradijentnog spusta kojem je svaki korak jednak

$$h_m = -\rho_m g_m, \quad (2.20)$$

gdje  $\rho_m$  predstavlja skalar, a  $g_m \in \mathbb{R}^N$  predstavlja gradijent. Svaka komponenta gradijenta iznosi

$$g_{im} = \left[ \frac{\delta \mathcal{L}(y_i, f(x_i))}{\delta f(x_i)} \right]_{f(x_i)=f_{m-1}(x_i)}. \quad (2.21)$$

Izračunom skalara  $\rho_m$  se dobiva konačna duljina koraka

$$\rho_m = \arg \min_{\rho} \mathcal{L}(\mathbf{f}_{m-1} - \rho g_m), \quad (2.22)$$

te se trenutno rješenje ažurira s dobivenim vrijednostima

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_m g_m. \quad (2.23)$$

Taj se postupak ponavlja za svaku novu iteraciju. [1]

Da je jedini zadatak modela prilagođavanje skupu za treniranje, koristio bi se samo gradijenti spust jer se lagano računa za bilo koju diferencijabilnu funkciju gubitka. Međutim, zadaća modela je i mogućnost predviđanja podataka koji nisu u skupu za treniranje. Zbog toga na svakom  $m$ -tom koraku se uvodi stablo čija predviđanja su približna predviđanjima negativnog gradijenta. Pripadni parametri stabla se dobivaju pomoću srednje kvadratne greške

$$\tilde{\theta} = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N (-g_{im} - T(x_i; \theta))^2. \quad (2.24)$$

Nakon podijele podataka na područja  $\tilde{R}_{jm}$ , to jest listove, vrijednost pridodana listu je ona koja najbolje ispravlja grešku u predviđanju prošlog stabla za sve primjere u listu

$$\tilde{\gamma}_{jm} = \arg \min_{\gamma_{jm}} \sum_{x_i \in \tilde{R}_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm}). \quad (2.25)$$

Stabla nađena ovom metodom nisu stabla koja rješavaju problem (2.16), ali daju rezultate koji su približni rezultatima tih stabla. [1]

Algoritam gradient boosta se može opisati sljedećim načinom:

1. Inicijaliziraj početno nagađanje  $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N \mathcal{L}(y_i, \gamma)$
2. Za  $m = 1$  do  $M$ :

- (a) Izračunaj odstupanja predviđanja svakog primjera

$$r_{im} = - \left[ \frac{\delta \mathcal{L}(y_i, f(x_i))}{\delta f(x_i)} \right]_{f(x_i)=f_{m-1}(x_i)} \quad (2.26)$$

- (b) Stvori stablo s  $J_m$  listova trenirano na odstupanjima  $r_m$
- (c) Za svaki list izračunaj pripadnu vrijednost  $\gamma$  pomoću jednadžbe (2.25)
- (d) Ažuriraj  $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

3. Izbaci konačno rješenje  $\hat{f}(x) = f_M(x)$

Za binarnu klasifikaciju je korisno koristiti jednadžbu gubitka (2.10). U tom slučaju početno nagađanje će biti jednako logaritmu omjera broja primjera pozitivne klase i

broja primjera negativne klase u skupu podataka. To jest, prvo nagađanje je logaritama izgleda. Također je korisno što je njena negativna derivacija, potrebna za korak 2.(a), jednaka

$$-\left[\frac{\delta J(y_i, \log(\text{izg}_i))}{\delta \log(\text{izg}_i)}\right]_{\log(\text{izg}_i)=\log(\text{izg}_i)_{m-1}} = \left(y_i - \frac{\exp(\log(\text{izg}_i))}{1 + \exp(\log(\text{izg}_i))}\right) = y_i - \hat{p}_i. \quad (2.27)$$

Vidljivo je iz jednadžbe (2.27) da je njena derivacija jednaka odstupanju u predviđanju vjerojatnosti. Novi problem koji se stvara je računanje pripadne vrijednosti listova  $\gamma$ . Naći minimum derivacije gubitka s uključenom  $\gamma$  kao što je traženo u (2.25) nije trivijalno. Međutim, moguće je aproksimirati vrijednost funkcije gubitka pomoću Taylorovog polinoma drugog reda oko nule s obzirom na  $\gamma$

$$J(y_i, f(x)_{m-1} + \gamma) \approx J(y_i, f(x)_{m-1}) + \frac{\delta}{\delta f(x)_{m-1}}(J(y_i, f(x)_{m-1}))\gamma + \frac{\delta^2}{2\delta f(x)_{m-1}^2}(J(y_i, f(x)_{m-1}))\gamma^2.$$

Tražena je prva derivacija Taylorovog polinoma, s obzirom na  $\gamma$ , kako bi se našla vrijednost pripadnog lista koja ispunjuje uvjet (2.25)

$$\frac{\delta}{\delta \gamma} J(y_i, f(x)_{m-1} + \gamma) = \frac{\delta}{\delta f(x)_{m-1}}(J(y_i, f(x)_{m-1})) + \frac{\delta^2}{\delta f(x)_{m-1}^2}(J(y_i, f(x)_{m-1}))\gamma.$$

Izjednačava se s nulom pošto je tražen minimum te se računa  $\gamma$

$$\frac{\delta}{\delta f(x)_{m-1}}(J(y_i, f(x)_{m-1})) + \frac{\delta^2}{\delta f(x)_{m-1}^2}(J(y_i, f(x)_{m-1}))\gamma = 0$$

$$\gamma = \frac{-\frac{\delta}{\delta f(x)_{m-1}}(J(y_i, f(x)_{m-1}))}{\frac{\delta^2}{\delta f(x)_{m-1}^2}(J(y_i, f(x)_{m-1}))}.$$

Druga derivacija jednadžbe gubitka je kompleksna, ali ju je moguće reducirati

$$\begin{aligned}
\frac{\delta^2}{\delta f(x)_{m-1}^2} (J(y_i, f(x)_{m-1})) &= -\frac{\exp(\log(\text{izg}))^2}{(1 + \exp(\log(\text{izg})))^2} + \frac{\exp(\log(\text{izg}))}{1 + \exp(\log(\text{izg}))} \\
&= \frac{-\exp(\log(\text{izg}))^2 + \exp(\log(\text{izg})) + \exp(\log(\text{izg}))^2}{(1 + \exp(\log(\text{izg})))^2} \\
&= \frac{\exp(\log(\text{izg}))}{(1 + \exp(\log(\text{izg})))^2} \\
&= \frac{\exp(\log(\text{izg}))}{1 + \exp(\log(\text{izg}))} \cdot \frac{1}{1 + \exp(\log(\text{izg}))} \\
&= \hat{p}(1 - \hat{p}).
\end{aligned}$$

Konačna vrijednost  $\gamma$  iznosi

$$\gamma = \frac{y - \hat{p}}{\hat{p}(1 - \hat{p})}. \quad (2.28)$$

Međutim, formula (2.28) daje rezultat za slučaj kada je samo jedan primjer u listu. Izvod se može analogno izvesti i u slučaju s više listova. Jedina razlika je u tome što je potrebno naći derivaciju s obzirom na  $\gamma$  od sume svih Taylorovih polinoma u listu. Tim postupkom se dobije sljedeći konačni izraz

$$\gamma = \frac{\sum_{i=1}^N (y_i - \hat{p}_i)}{\sum_{i=1}^N \hat{p}_i (1 - \hat{p}_i)}. \quad (2.29)$$

Rezultat završnog stabla će biti neka vrijednost koja je iznos logaritma izgleda. Ako je vrijednost pozitivna ili jednaka nuli, primjer je član pozitivne klase. Ako je vrijednost negativna, primjer je član negativne klase. [3]

Gradient boost, kao i drugi algoritmi, ima svoje hiperparametre za bolju preciznost i brzinu izvođenja programa. Poput gradijentnog otpadanja, može se smanjiti utjecaj doprinosa svakog stabla pomoću stope učenja  $\eta$ . Može se i ograničiti koliko novih stabla se stvara (određivanje broja  $M$ ). Treba napomenuti da  $\eta$  i broj stabla u ansamblu utječu jedan na drugoga. Definiranjem  $\eta$  jako male vrijednosti zahtijeva više stabla kako bi došlo do boljih predviđanja. Sama stabla unutar ansambla mogu se ograničiti postavljanjem maksimalne dubine, ograničavanjem broja listova, ograničavanjem broja primjera u listu te raznim drugim hiperparametrima. Kao i kod grupnog gradijentnog spusta, također je moguće uzeti neki postotak primjera pri računanju gradijenta. Time se znatno ubrzava algoritam i smanjuje mogućnost pretreniranja jer se više ne računa odstupanje od svakog primjera.

## 2.4 XGBoost

XGBoost (eng. *extreme gradient boosting*) je algoritam koji znatno unaprjeđuje gradient boosting. To omogućuje kvantilnom skicom podataka, paralelnim učenjem stabla odlučivanja, prepoznavanjem rijetkog skupa podataka i obrađivanjem podataka van RAM memorije. Osim što može učiti na postotku primjera skupa podataka, može koristiti i samo neke od značajki skupa za učenje. Korištenje te tehnike još više sprječava pretreniranje i ubrzava algoritam paralelnog računanja.

XGBoost ima drukčiji pristup stvaranja stabla. Uz normalnu funkciju gubitka, XGBoost također kažnjava kompleksnost stabla. Što je veća ukupna suma listova stabla  $\gamma$  ili što je veći broj listova  $J$ , više će se kažnjavati rezultat stabla.

$$\mathcal{L} = \sum_i l(y_i, \hat{y}) + \sum_m \Omega(f_m), \text{ gdje } \Omega(f) = \omega J + \frac{1}{2} \lambda \|\gamma\|^2. \quad (2.30)$$

Funkcija  $l$  je bilo koja diferencijabilna konveksna funkcija koja računa pogrešku između  $y$  i  $\hat{y}$ . Varijable  $\omega$  i  $\lambda$  su hiperparametri. Vrijednost pripadnog lista se može odrediti pomoću Taylorovog polinoma na način koji je prikazan kod gradient boosting klasifikatora. Recimo da je funkcija  $g$  prva derivacija funkcije  $l$ , a  $h$  je druga derivacija funkcije  $l$ . Optimalna vrijednosti lista  $j$  će biti

$$\gamma_j^* = -\frac{\sum_{i \in R_j} g_i}{\sum_{i \in R_j} h_i + \lambda}, \quad (2.31)$$

gdje je vidljiva sličnost jednadžbi (2.29). Jedina razlika je u dodatku hiperparametra  $\lambda$  u nazivniku, čija je uloga smanjenje sume svih listova. Funkcija gubitka pri računanju  $m$ -tog stabla se zapisuje kao

$$\begin{aligned} \mathcal{L}_m &= \sum_{i=1}^N l(y_i, \hat{y}_i^{(m-i)} + f(x_i)_{m-1}) + \Omega(f_m) \\ &\approx \sum_{i=1}^N \left[ l(y_i, \hat{y}) + g_i f(x_i) + \frac{1}{2} h_i f(x_i)^2 \right] + \Omega(f_m) \\ &\approx \sum_{i=1}^N \left[ (g_i f(x_i) + \frac{1}{2} h_i f(x_i)^2) \right] + \Omega(f_m) \\ &= \sum_{i=1}^N \left[ (g_i f(x_i) + \frac{1}{2} h_i f(x_i)^2) \right] + \omega J + \frac{1}{2} \sum_{j=1}^J \gamma_j^2 \end{aligned}$$

$$\mathcal{L}_m = \sum_{j=1}^J \left[ \gamma_j \sum_{i \in R_j} g_i + \frac{1}{2} \gamma_j^2 \sum_{i \in R_j} (h_i + \lambda) \right] + \omega J. \quad (2.32)$$

Uvrštavanjem (2.31) u (2.32) dobiva se točna vrijednost gubitka stabla m

$$\mathcal{L}_m = -\frac{1}{2} \sum_{j=1}^J \left[ \frac{(\sum_{i \in R_j} g_i)^2}{\sum_{i \in R_j} h_i + \lambda} \right] + \omega J. \quad (2.33)$$

Točna vrijednost gubitka stabla je korisna jer pomoću nje je moguće izračunati koliko se smanjuje funkcija gubitka ako list pretvorimo u čvor. Recimo da se list  $R$  podijeli na čvorove  $R_D$  i  $R_L$ , s time da vrijedi  $R = R_D \cup R_L$ . Smanjenje funkcije gubitka računa se oduzimanjem gubitka starog lista od zbroja gubitka novih listova

$$\mathcal{L}_{\text{razdijeli}} = \frac{1}{2} \left[ \frac{(\sum_{i \in R_D} g_i)^2}{\sum_{i \in R_D} h_i + \lambda} + \frac{(\sum_{i \in R_L} g_i)^2}{\sum_{i \in R_L} h_i + \lambda} - \frac{(\sum_{i \in R} g_i)^2}{\sum_{i \in R} h_i + \lambda} \right] - \omega. \quad (2.34)$$

Uobičajeno se koristi kriterij (2.34) kako bi se izračunao dobitak informacija koji je nastao razdjeljivanjem čvora. [4]

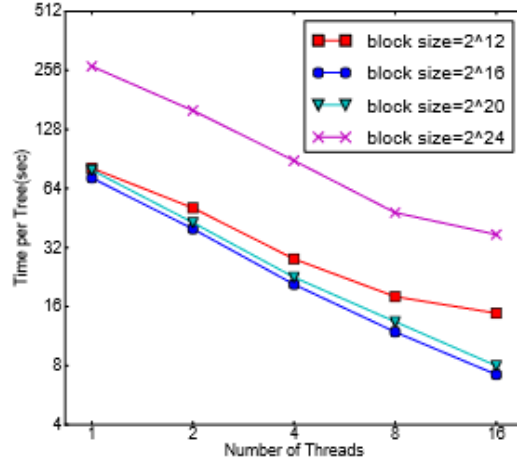
XGBoost upotrebljava dobitak (2.34) kada odlučuje u koji čvor će uputiti primjer čija vrijednosti značajke nedostaje. Promatra list kao da ima dva skupa podataka. Prvi  $R$  u kojem su svi primjeri i drugi  $R_k$  u kojem su primjeri čija vrijednost tražene značajke nije nula ili nepostojeća. Prate se sljedeći koraci

1. Izračunaj  $G = \sum_{i \in R} g_i$  i  $H = \sum_{i \in R} h_i$
2. Za  $k=1$  do  $M$ :
  - (a) Postavi  $G_L$  i  $H_L$  lijevog čvora na 0
  - (b) Za  $j$  u uzlazno sortiranom  $R_k$ :
    - i. Izračunaj  $G_L = G_L + g_j$ ,  $H_L = H_L + h_j$
    - ii. Izračunaj  $G_D = G - G_L$ ,  $H_R = H - H_L$
    - iii. Odredi dobitak pomoću (2.34) i zadrži rezultat ako je dao veći dobitak
  - (c) Postavi  $G_D$  i  $H_D$  desnog čvora na 0
  - (d) Za  $j$  u silazno sortiranom  $R_k$ :
    - i. Izračunaj  $G_D = G_D + g_j$ ,  $H_D = H_D + h_j$
    - ii. Izračunaj  $G_L = G - G_D$ ,  $H_L = H - H_D$

iii. Odredi dobitak pomoću (2.34) i zadrži rezultat ako je dao veći dobitak

Na ovaj način XGBoost provjerava u kojem slučaju je veći dobitak, kada se primjeri s nepoznatim vrijednostima nalaze u desnom ili kada se nalaze u lijevom čvoru. Algoritmi koji ne podržavaju nepostojeće vrijednosti, poput gradient boostinga, moraju popuniti vrijednosti značajke uglavnom s prosječnom vrijednošću ili medijanom. Na ovaj način XGBoost ne zadaje pretpostavljene vrijednosti nepostojećima i smanjuje na veličini podataka koje mora obrađivati pri diobi čvorova. [4]

XGBoost ostvaruje paralelno učenje podjelom skupa podataka na blokove. Svaka značajka se sprema u svoj blok. Unutar bloka, vrijednosti značajke su uređene. Blokovi se mogu opet razdijeliti u blokove koji sadržavaju dio primjera značajke. Takva struktura je korisna kod velikih skupova podataka jer onda više dretvi odjednom mogu pristupiti blokovima iste značajke te tako znatno ubrzati algoritme. Kad su blokovi pohranjeni na ROM memoriji komprimirani su radi bržeg čitanja podataka. XGBoost dodatno optimizira čitanje s ROM memorije u slučaju kada ima više uređaja za pohranjivanje. U tom slučaju blok se razdijeli preko više uređaja. Kad dretva zahtjeva pristup bloku, svi uređaji će istovremeno iščitavati podatke što je brže nego da se jednaka količina podataka iščitava iz samo jednog uređaja. Također, prioritizirana je predmemorija dretvi pošto je to najbrža memorija pristupna računalu. XGBoost alocira memoriju u predmemoriji kako bi sve  $g$  i  $h$  vrijednosti bloka bile brzo dostupne. Za tu optimizaciju je bitna i optimizacija veličine bloka. Ako je blok preveliki, dretva neće biti dovoljno opterećena što rezultira neefikasnim paralelnim učenjem. Ako je blok premali, neće biti dovoljno predmemorije za sve  $g$  i  $h$  vrijednosti bloka. Testiranjem je dokazano da blokovi s  $2^{16}$  primjera su optimalne veličine. [4]



Slika 2.4: Usporedba brzine XGBoost algoritma ovisno o veličini blokova pri primjeni na skupu podataka s deset milijuna primjera [4]

Stabla odlučivanja pohlepno traže mjesto na kojem će ostvariti najbolju diobu čvora. Provjeravaju između svakog primjera koliki je dobitak te se bira mjesto s najvećim dobitkom. Pošto se provjerava svaki primjer i svaka značajka, takvo stvaranje stabla odlučivanja je vremenski zahtjevno. Prvi način na koji XGBoost rješava taj problem je približnim dijeljenjem čvorova, što postiže kvantilnom skicom skupa podataka. Algoritam, s pretpostavkom da su vrijednosti značajke raspoređene s ne-kakvom distribucijom, rangira podatke pomoću druge derivacije funkcije gubitka  $h$ . Odredimo skup  $n$ -torki koji sadrže primjere, s pripadnom drugom derivacijom  $h$ ,  $\mathcal{D}_k = \{(x_{1k}, h_1), (x_{2k}, h_2) \dots (x_{nk}, h_n)\}$  gdje  $k$  označava značajku te je uzlazno uređen po  $x \in \mathcal{X} \implies [0, \infty)$ . Funkcija rangiranja djeluje na sljedeći način

$$r_k(z) = \frac{1}{\sum_{(x,h) \in \mathcal{D}_k} h} \sum_{(x,h) \in \mathcal{D}_k, x < z} h. \quad (2.35)$$

To jest, rangira prema omjeru sume drugih derivacija do vrijednosti značajke  $z$  i sume drugih derivacija u cijelom skupu. Algoritam sada traži vrijednosti skupa  $S_k = \{s_{k1}, s_{k2} \dots s_{kl}\}$  prema kojima može podijeliti čvor. Da neka vrijednost može ući u skup  $S_k$  mora ispunjavati sljedeći kriterij

$$|r_k(s_{k,i}) - r_k(s_{k,i+1})| < \epsilon, s_{k,1} = \min_i x_{ik}, s_{k,l} = \max_i x_{ik}. \quad (2.36)$$

Pomoću faktora približnosti  $\epsilon$  možemo procijeniti da će skup  $S_k$  sadržati otprilike  $\frac{1}{\epsilon}$ . Međutim, potrebno je još doraditi algoritam za slučaj kada skup podataka ima težine



pridodane svakom primjeru. To je XGBoost prvi uspio napraviti svojim težinskim kvantilnim skiciranjem. Definiraju se tri vrijednosti unutar domene  $\mathcal{X} \implies [0, \infty)$  koje koriste vrijednosti težina skupa  $w$

- Donja granica ranga  $r_{\mathcal{D}}^{-}(y) = \sum_{(x,w) \in \mathcal{D}, x < y} w$
- Gornja granica ranga  $r_{\mathcal{D}}^{+}(y) = \sum_{(x,w) \in \mathcal{D}, x \leq y} w$
- Težina ranga  $w_{\mathcal{D}} = r_{\mathcal{D}}^{+}(y) - r_{\mathcal{D}}^{-}(y) = \sum_{(x,w) \in \mathcal{D}, x=y} w$

Težina ranga se određuje jer je moguće da se iste vrijednosti istih težina nalaze u skupu  $\mathcal{D}$ . Kvantilni sažetak podataka s težinama zapisuje se kao n-torka  $Q(\mathcal{D}) = (S, \tilde{r}_{\mathcal{D}}^{+}(y), \tilde{r}_{\mathcal{D}}^{-}(y), \tilde{w}_{\mathcal{D}})$ .  $S$  je skup koji sadrži odabrane točke iz  $\mathcal{D}$  koje su uzlazno uređene ( $x_i < x_{i+1}$ ). Unutar  $S \implies [0, \infty)$  su funkcije  $\tilde{r}_{\mathcal{D}}^{+}, \tilde{r}_{\mathcal{D}}^{-}, \tilde{w}_{\mathcal{D}}$  te za njih vrijedi:

- $\tilde{r}_{\mathcal{D}}^{+}(x_i) \geq r_{\mathcal{D}}^{+}(x_i), \quad \tilde{r}_{\mathcal{D}}^{-}(x_i) \leq r_{\mathcal{D}}^{-}(x_i), \quad \tilde{w}_{\mathcal{D}}(x_i) \leq w_{\mathcal{D}}(x_i)$
- $\tilde{r}_{\mathcal{D}}^{-}(x_i) + \tilde{w}_{\mathcal{D}}(x_i) \leq \tilde{r}_{\mathcal{D}}^{-}(x_{i+1}), \quad \tilde{r}_{\mathcal{D}}^{+}(x_i) \leq \tilde{r}_{\mathcal{D}}^{+}(x_{i+1}) - \tilde{w}_{\mathcal{D}}(x_{i+1})$ .

Kvantilni sažetak definiran je unutar domene  $S$ . Za proširivanje domene funkcija na  $\mathcal{X}$ , potrebno je pratiti sljedeće korake

1. Ako  $y < x_1$ :  $\tilde{r}_{\mathcal{D}}^{+}(y) = 0, \tilde{r}_{\mathcal{D}}^{-}(y) = 0, \tilde{w}_{\mathcal{D}}(y) = 0$
2. Ako  $y > x_k$ :  $\tilde{r}_{\mathcal{D}}^{+}(y) = \tilde{r}_{\mathcal{D}}^{+}(x_k), \tilde{r}_{\mathcal{D}}^{-}(y) = \tilde{r}_{\mathcal{D}}^{-}(x_k), \tilde{w}_{\mathcal{D}}(y) = 0$
3. Ako  $y \in (x_i, x_{i+1})$ :  $\tilde{r}_{\mathcal{D}}^{+}(y) = \tilde{r}_{\mathcal{D}}^{+}(x_{i+1}) - \tilde{w}_{\mathcal{D}}(x_{i+1}), \tilde{r}_{\mathcal{D}}^{-}(y) = \tilde{r}_{\mathcal{D}}^{-}(x_i) + \tilde{w}_{\mathcal{D}}(x_i), \tilde{w}_{\mathcal{D}}(y) = 0$

Kvantilni sažetak možemo opisati kao  $\epsilon$  približan ako za bilo koji  $y \in \mathcal{X}$  vrijedi

$$\tilde{r}_{\mathcal{D}}^{+} - \tilde{r}_{\mathcal{D}}^{-} - \tilde{w}_{\mathcal{D}} = \epsilon w(\mathcal{D}). \quad (2.37)$$

Uz dovoljno mali skup  $\mathcal{D}$ , skup  $S$  kvantilnog sažetka  $Q(\mathcal{D})$  će biti sačinjen od svake točke unutar  $\mathcal{D}$ . Rezultat toga je nastanak 0-približnog sažetka jer su sve pripadne funkcije rangiranja jednake originalnom skupu. Sažeci dva različita skupa  $\mathcal{D}_1$  i  $\mathcal{D}_2$  se mogu spojiti u jedan sažetak  $Q(\mathcal{D})$ . Pošto su sve funkcije rangiranja aditivne, novi iznosi rangiranja sažetka  $Q(\mathcal{D})$  će biti zbroj iznosa sažetaka  $Q(\mathcal{D}_1)$  i  $Q(\mathcal{D}_2)$ . Međutim, faktor približnosti nije aditivan već se uzima vrijednost skupa s najvećim  $\epsilon$ . [4]

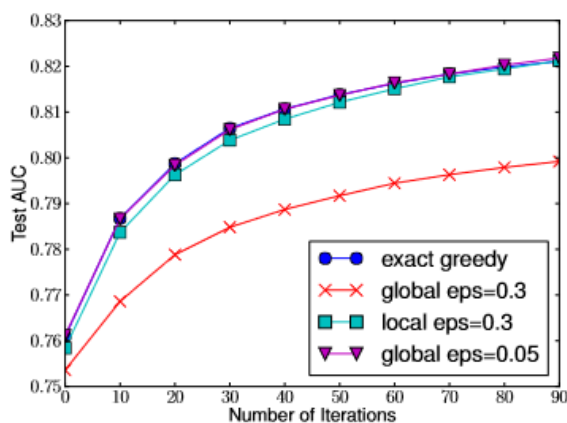
Najbitnije svojstvo kvantilnih sažetaka jest mogućnost njihovog skraćivanja, što omogućava brzo pronalaženje mjesta za diobu čvora. XGBoost pomoću svojeg algoritma upita traži  $x^*$  koji odgovara rangu  $0 \leq d \leq w(\mathcal{D})$  unutar sažetka  $\mathcal{Q}$ . Upit se označuje kao  $g(\mathcal{Q}, d)$  i izvodi se na sljedeći način

1. Ako vrijedi  $d \leq \frac{1}{2} [\tilde{r}_{\mathcal{D}}^-(x_1) + \tilde{r}_{\mathcal{D}}^+(x_1)]$ , vrati  $x_1$
2. Ako vrijedi  $d \geq \frac{1}{2} [\tilde{r}_{\mathcal{D}}^-(x_k) + \tilde{r}_{\mathcal{D}}^+(x_k)]$ , vrati  $x_k$
3. Nađi  $i$  za koji vrijedi  $\frac{1}{2} [\tilde{r}_{\mathcal{D}}^-(x_i) + \tilde{r}_{\mathcal{D}}^+(x_i)] \leq d \leq \frac{1}{2} [\tilde{r}_{\mathcal{D}}^-(x_{i+1}) + \tilde{r}_{\mathcal{D}}^+(x_{i+1})]$
4. Ako vrijedi  $2d < \tilde{r}_{\mathcal{D}}^-(x_i) + \tilde{w}_{\mathcal{D}}(x_i) + \tilde{r}_{\mathcal{D}}^+(x_{i+1}) - \tilde{w}_{\mathcal{D}}(x_{i+1})$ , vrati  $x_i$
5. Inače vrati  $x_{i+1}$

Vrijednost  $x^*$  koju vraća funkcija  $g(\mathcal{Q}, d)$  ispunjava uvjet

$$\tilde{r}_{\mathcal{D}}^+(x^*) - \tilde{w}_{\mathcal{D}}(x^*) - \frac{\epsilon}{2}w(\mathcal{D}) \leq d \leq \tilde{r}_{\mathcal{D}}^-(x^*) + \tilde{w}_{\mathcal{D}}(x^*) + \frac{\epsilon}{2}w(\mathcal{D}). \quad (2.38)$$

Sažetak  $\mathcal{Q}$  se skraćuje u novi sažetak  $\mathcal{Q}'$  u memoriju veličine  $b$ . Potrebno je odrediti novi  $S' = \{x'_1, x'_2, \dots, x'_{b+1}\}$  skup, dok se vrijednosti  $\tilde{r}_{\mathcal{D}}^+$ ,  $\tilde{r}_{\mathcal{D}}^-$ ,  $\tilde{w}_{\mathcal{D}}$  prepisuju. Članovi skupa  $S'$  se dobivaju algoritmom upita te se biraju oni za koje vrijedi  $x'_i = g(\mathcal{Q}, \frac{i-1}{b}w(\mathcal{D}))$ . Unutar novog sažetka je moguće uklanjati duplicirane vrijednosti radi optimalnijeg korištenja memorije. Ako je kvantilni sažetak  $\mathcal{Q}$  bio  $\epsilon$  približan, pripadno skraćenje bit će  $(\epsilon + \frac{1}{b})$  približno. To jest, smanjenjem veličine  $\mathcal{Q}$  se gubi na preciznosti. Sažetak  $\mathcal{Q}'$  sadrži u skupu  $S'$  potencijalna mjesta diobe čvora. U slučaju kada skup podataka nema zadane težine, XGBoost koristi drugu derivaciju  $h$  svakog primjera kao pripadnu težinu. Postupak određivanja potencijalnih kandidata za mjesto diobe može se odvijati prilikom kreiranja stabla (globalni način) ili prilikom diobe svakog čvora (lokalni način). Oba načina imaju približnu preciznost, ali globalnom načinu potreban je veći skup  $S$  da u jednakom broju ponavljanja stvori iste rezultate.



Slika 2.5: Usporedba preciznosti pohlepnog načina, lokalnog kvantilnog sažetka i globalnih kvantilnih sažetka s različitim veličinama skupa  $S$  [4]

Pomoću kvantilnih sažetaka XGBoost smanjuje broj vrijednosti koje mora pregledavati za diobu čvora. U praksi skup  $S$  sadrži 33 do 100 članova, to jest  $\epsilon \in [0,01, 0,03]$ , što rezultira drastično manjem broju provjera za skupove podataka koji imaju  $10^6+$  primjera.

## 2.5 Light gradient boosting machine

Light gradient boosting machine unaprijeđuje običan gradient boosting na dva načina. Prvo poreda primjere prema apsolutnoj vrijednosti njihovog gradijenta te sačuva  $(a \times 100)\%$  s najvećim gradijentom te nasumično odabere  $(b \times 100)\%$  ostalih primjera. Nasumično odabranim primjerima se vrijednost gradijenta uvećava pomoću konstante  $\frac{1-a}{b}$  pri računanju dobitka informacija. Tim načinom fokusira treniranje modela na primjerima koji daju više informacija. Taj postupak je nazvan gradient-based one-sided sampling, to jest jednostrano uzimanje uzorka temeljeno na gradijentu. [5]

Drugi način kojim se ubrzava gradient boosting dolazi s pretpostavkom da skupovi podataka sadrže isključive značajke. Točnije, neke značajke rijetko istovremeno imaju vrijednost različitu od nule. S tom pretpostavkom se mogu grupirati značajke i time značajno smanjiti dimenzionalnost problema s minimalnim gubitcima. Grupirane značajke u tom slučaju se nazivaju paket isključivih značajki. Značajke se grupiraju tako da se sastavi težinski graf gdje težine prikazuju koliko puta značajka nije bila međusobno isključiva s drugom značajkom. Značajke se silazno poredaju prema sumi svih bridova koje vode prema njima. Značajka ulazi u postojeći paket

ako je postotak sukoba manji od hiperparametra  $\gamma$ , inače se stvara novi paket. U paketu se mora raspoznati od koje značajke dolazi vrijednost. Taj problem se rješava tako da se domene značajki mijenjaju kako ne bi došlo do preklapanja. Recimo da značajka A koja sadrži vrijednosti u rasponu  $[0, 10)$  i značajka B koja sadrži vrijednosti u rasponu  $[0, 20)$  se nalaze u istom paketu. Njihovo preklapanje domena će se riješiti tako da se sve vrijednosti od B zbroje s 10, to jest s prvom vrijednosti koja se ne nalazi unutar domene od A. Značajka B sada sadrži vrijednosti u rasponu  $[10, 30)$  te je njihovo preklapanje razriješeno. [5]

Pomoću tih metoda LightGBM može drastično smanjiti dimenzionalnost problema uz zanemariv utjecaj na preciznost, dok rezultate postiže u značajno manjem vremenu.

## 2.6 CatBoost

CatBoost (eng. *categorical boost*) je još jedan algoritam koji implementira gradient boosting. Glavni naglasak algoritma je obrada kategoričkih vrijednosti u skupu podataka. Kategoričke značajke nisu kontinuirane te jedna moguća vrijednost ne govori o proporcionalnosti s ostalim mogućim vrijednostima. Primjeri kategoričkih značajki su spol, boja, vrsta i tako dalje. Tipično se kategoričke vrijednosti pretvaraju u numeričke. Najčešća metoda pretvorbe kategoričkih vrijednosti je one-hot kodiranje. Tom metodom svaka moguća vrijednost kategoričke značajke unutar skupa podataka postaje svoja značajka. Svako pojavljivanje te vrijednosti u originalnom stupcu se u pripadnoj novoj značajki označi s brojem 1, dok ostatak novih značajki se označi s 0. Problem u tom pristupu leži u tome što se dimenzionalnost skupa može jako povećati te nastaje rijedak skup podataka. CatBoost zato umjesto one-hot kodiranja koristi uređenu ciljanu statistiku. Umjesto da se stvaraju značajke za svaku moguću vrijednost, originalna kategorička vrijednost se zamjenjuje s numeričkom. Prvo se stvara permutacija skupa podataka  $\sigma$  pomoću kojeg CatBoost redom stvara vrijednosti za novu numeričku zamjenu kategoričke značajke. Primjeru  $x_i$  nova vrijednost značajke  $k$  ovisi o broju primjera unutar permutacije  $\sigma$  do  $i$  koji su imali istu kategoričku vrijednost

$$\hat{x}_{i,k} = \frac{\sum_{x_j \in \sigma(i-1)} I(x_j = x_i) y_j + ap}{\sum_{x_j \in \sigma(i-1)} I(x_j = x_i) + a}. \quad (2.39)$$

Računa se do traženog primjera kako algoritam ne bi svakom primjeru odavao vrijednost vlastitog traženog izlaza. Vidljivo je iz jednadžbe (2.39) da prvi primjeri kategoričkih vrijednosti će imati vrijednost koja ovisi samo o parametrima  $a$  i  $p$ . Parametar  $a$  je vrijednost koja je veća od 0, dok se za parametar  $p$  uglavnom uzima prosječna vrijednost očekivanog izlaza. [6]

CatBoost za razliku od ostalih algoritama do sad ne koristi uobičajena stabla odlučivanja već koristi nesvjesna stabla odlučivanja. U takvom stablu svaki čvor na istoj dubini stabla ima isti uvjet razdvajanja podataka. Nesvjesna stabla su manje precizna od običnih stabla. Međutim, algoritmima koji upotrebljavaju ansamble je poželjno koristiti slabije predviđače. Tom metodom Catboost dobiva više uravnotežena stabla, smanjuje pretreniranje i značajno ubrzava vrijeme testiranja. Prilikom računanja gradijenta primjera  $i$ , CatBoost i dalje uzima samo u obzir sve primjere do  $i$ -tog na isti način kao i kod kategoričkih značajki. Kako bi se smanjila varijabilnost, stvara se  $s$  permutacija skupa podataka te se uzima prosjek gradijenata za tražene vrijednosti. Za procijenu kandidata za diobu čvora se koristi sličnost kosinusa predviđene vrijednosti svih primjera i gradijenta [6]

$$\cos(\gamma, G) = \frac{\gamma \cdot G}{\|\gamma\| \cdot \|G\|}. \quad (2.40)$$

## 2.7 Slučajna šuma

Slučajna šuma je algoritam koji koristi ansambl stabla odlučivanja, ali ne radi preko boosting metode već preko bagging metode. Tom metodom rezultati više individualno stvorenih modela se spajaju te njihov zajednički rezultat stvara konačan rezultat. Slučajne šume pri stvaranju modela ne uzimaju originalan skup podataka. Uzima se skup podataka koji je jednake veličine kao i originalan, ali je svaki primjer nasumično odabran iz originalnog skupa podataka. Taj postupak se zove bootstrapping i njime novi skup podataka može sadržavati kopije nekog primjera, dok se neki primjeri neće prikazivati. Slučajna šuma, nakon stvaranja novog skupa podataka, stvara onoliko stabla koliko je određeno hiperparametrom. Pri dijeljenju čvorova stabla gleda se odabrana vrijednosti značajke koja je dala najbolje rezultate u nasumično odabranom podskupu značajki. U slučaju regresijskih problema se uzima prosjek rezultata svih stabla, dok se za klasifikacijske uzima većinsko predviđanje. [2]

## 3 Primjena modela

U ovom poglavlju uspoređuju se algoritmi gradient boosting, XGBoost, LightGBM, CatBoost i slučajna šuma na klasifikacijskim problemima. Uspoređivat će se pomoću četiri različite metrike, dok će se pomoću Shaplyjevih vrijednosti interpretirati njihove najznačajnije značajke i njihov utjecaj na rezultat.

### 3.1 Metrike

Najjednostavnija metrika koja se može gledati za ocjenu uspješnosti klasifikacijskog modela je točnost. Točnost govori koliki postotak predviđenih rezultata je svrstan u točnu klasu. Međutim, kod točnosti je problem što ne daje previše uvida u rezultate. Ako imamo skup podataka koji sadrži 99% primjera u negativnoj klasi, model može označiti da su svi primjeri članovi negativne klase te dobiva točnost od 99%. Zbog toga se koriste različite metrike za opisivanje rezultata modela.

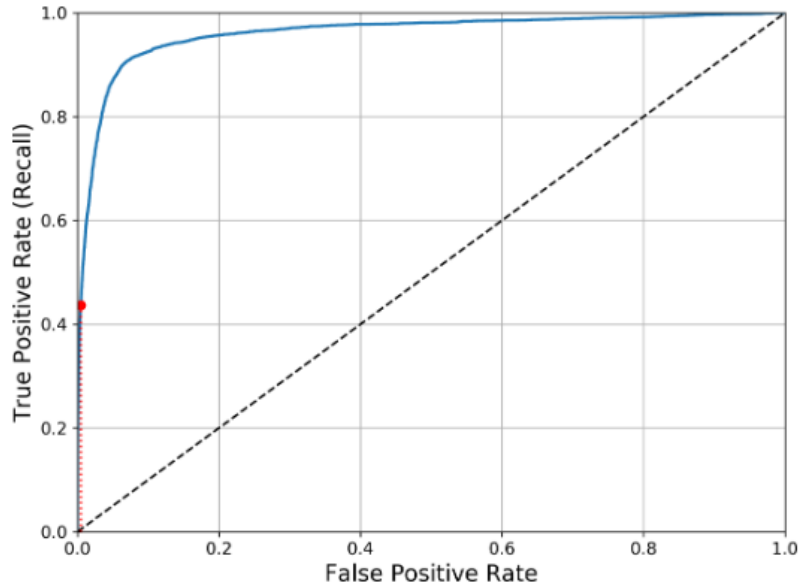
Bolje metrike koje ocjenjuju model su preciznost, opoziv i ROC-AUC vrijednost. Model pri stvaranju rezultata daje istinito i lažno pozitivno označene primjere uz istinito i lažno negativno označene primjere. Preciznost je omjer broja istinito pozitivno označenih primjera i ukupnog broja pozitivno označenih primjera

$$Pr = \frac{IP}{IP + LP}. \quad (3.1)$$

Opoziv je omjer broja istinito pozitivno označenih primjera i zbroja istinito pozitivno i lažno negativno označenih primjera

$$Op = \frac{IP}{IP + LN}. \quad (3.2)$$

Ako model cilja visoku preciznost, smanjuje se broj lažno pozitivno označenih primjera. Smanjenjem broja lažno pozitivno označenih primjera model će stvoriti više lažno negativno označenih primjera i time će se smanjiti vrijednost opoziva. Obratno vrijedi ako se model pokušava prilagoditi za visoki opoziv. Zato se uvodi dodatna metrika ROC (eng. *receiver operating characteristic*) krivulja. ROC krivulja predstavlja odnos opoziva i stope lažno pozitivno označenih primjera koja zapravo iznosi  $1 - Op$ .



Slika 3.1: Primjer ROC krivulje [2]

Crvena točka na slici 3.1 predstavlja iznos opoziva za danu toleranciju lažno pozitivno označenih primjera. ROC-AUC metrika je iznos površine ispod ROC krivulje na grafu ovisnosti opoziva o stopi lažno pozitivna. Idealnom modelu ROC-AUC vrijednost je 1, dok modelu koji nasumično nagađa iznosi 0,5. [2]

Navedenim metrikama uspoređivat će se rezultati modela na skupovima podataka. Svaki rezultat je dobiven peterostrukom unakrsnom validacijom. Ta metoda dijeli skup na pet dijela. Model se trenira na četiri dijela te se na petom testira. Postupak se ponavlja dok se model ne testira na svakom dijelu skupa podataka.

### 3.2 Shapleyjeve vrijednosti

Shapleyjeve vrijednosti su originalno zamišljene kako bi se izračunala vrijednost doprinosa igrača u poštenom dogovoru unutar igre. Unutar skupa  $N$  postoji  $n$  igrača.  $S$  definiramo kao podskup skupa  $N$  koji predstavlja jednu moguću koaliciju igrača. Odredimo funkciju  $v$  koja prima podskup  $S$ . Njena vrijednost će predstavljati vrijednost koalicije  $S$  unutar igre. Vrijednost igrača  $i$  možemo odrediti pomoću Shapleyjeve vrijednosti

$$\varphi_i(v) = \sum_{S \supset N \setminus \{i\}} \frac{|S|!(n - |S|)!}{n!} (v(S \cup \{i\}) - v(S)). \quad (3.3)$$

Jednadžba (3.3) provjerava sve moguće koalicije bez igrača  $i$  te računa doprinos igrača koaliciji. Doprinos svake koalicije je otežan uzimajući u obzir veličinu koalicije. Za primjenu u mehaničkom učenju, značajke zamjenjaju igrače, a rezultati modela predstavljaju dogovor unutar igre. Pomoću Shapleyjevih vrijednosti se može odrediti na koji način značajke doprinose pri kreiranju modela. Uvođenjem ansambla stabla odlučivanja se gubi interpretabilnost individualnih stabla. Uporabom Shapleyjevih vrijednosti se vraća mogućnost lakšeg shvaćanja donošenja odluke modela. Uvidom u Shapleyjeve vrijednosti je moguće i smanjiti dimenzionalnost problema izbacivanjem neznačajnih značajki. Bitno je napomenuti da Shapleyjeve vrijednosti modela služe za interpretaciju donošenja odluke modela te se iz njih ne može zaključiti prava priroda odnosa značajke i klase. [7]

### 3.3 Higgsov bozon

Ovaj skup podataka objavljen je 2014. godine na natjecanju na web stranici Kaggle, u suradnji s CERN institutom i The ATLAS experiment. Cilj natjecanja je bio stvoriti model koji indentificira Higgsove bozone. Unutar CERN-ovog sudarača čestica ATLAS detektor mjeri vrste čestice, energiju čestice i smjer gibanja čestica nastalih nakon sudaranja protona. Protoni se sudaraju svakih 50 nanosekundi te nakon sudara nastaju nestabilne čestice koje se brzo raspadaju. ATLAS detektor prati događaje čestica nastalih sudarom protona te pokušava detektirati Higgsov bozon. Higgsov bozon se može detektirati svojim raspadom u dvije  $\tau$  čestice. Međutim, detekcija raspada nije trivijalna. Detektor ne može raspoznati neutrino čestice čime se ometa detekciju mase čestice koje je potencijalni Higgsov bozon. Također, Z bozon, koji je slične mase, može se raspasti u dvije  $\tau$  čestice. Pošto je  $\tau$  čestica sama po sebi nestabilna, njeno prisustvo detektira se prisustvom elektrona ili miona s dva neutrina i prisustvom hadrona s neutrinom. ATLAS-ov klasifikator, treniran na simuliranim podacima, bira prostor bogat s događajima. U slučaju dovoljno velikog broja traženih događaja, označava se postojanje Higgsove čestice  $s$ . [8]

AMS (eng. *average median significance*) je određena mjera kvalitete klasifikatora

$$AMS = \sqrt{2 \left( (\mathbf{s} + \mathbf{b} + \mathbf{b}_r) \log \left( 1 + \frac{\mathbf{s}}{\mathbf{b} + \mathbf{b}_r} \right) - \mathbf{s} \right)}, \quad (3.4)$$



gdje  $\mathbf{s}$  i  $\mathbf{b}$  označavaju

$$\mathbf{s} = \sum_{i=1}^N w_i I(y_i = s) I(\hat{y}_i = s), \quad (3.5)$$

$$\mathbf{b} = \sum_{i=1}^N w_i I(y_i = b) I(\hat{y}_i = s). \quad (3.6)$$

Vrijednost  $\mathbf{b}_r = 10$  je regulacijska konstanta, a iznosi  $w_i$  predstavljaju težine primjera. Težine  $w$  u ovom skupu podataka nisu klasične težine gdje se želi istaknuti mali postotak primjera pozitivne klase. U ovom skupu podataka težine umanjuju signale  $s$  i povećavaju pozadinske signale  $b$ . Time se simulira stvarna učestalost pojave signala pri detektiranju Higgssovog bozona. AMS vrijednost kažnjava veliki broj lažno pozitivno označenih primjera te se zapravo traži veća preciznost modela. Međutim, AMS raste i s brojem pozitivno označenih primjera te se zato opoziv ne može zanemariti. Skup podataka za treniranje sadržava indentifikacijski stupac, 30 značajki, stupac pripadnih težina i stupac oznaka za sveukupno 250000 primjera. Unaprijed je određen skup za testiranje s 500000 primjera istih značajki, međutim ne sadržava težine i oznake. Sve značajke su tipa podataka float, a vrijednosti koje nedostaju označene su s  $-999.00$ . Značajke čija imena počinju s PRI označavaju primitivne vrijednosti koje je detektor izmjerio. Značajke čija imena počinju s DER označavaju izvedene vrijednosti koje su dobivene pomoću primitivnih vrijednosti. [8]

U ovom natjecanju se prvi put istaknuo XGBoost te mnoga najbolja rješenja su bila neka implementacija XGBoost algoritma. Radi smanjenja lažno pozitivno označenih primjera, prvih 10 – 20% primjera s najvećom vjerojatnošću su se označili s oznakom  $s$ . U mjerenjima danim u ovom radu se uzimalo 15% najvjerojatnijih primjera za izračun AMS rezultata. Hiperparametri su bili određeni funkcijom GridSearchCV iz paketa Scikit-learn. Ta funkcija uzima razne moguće vrijednosti hiperparametra te određuje koja kombinacija stvara model s najvećom točnosti. Hiperparametri za XGBoost su uzeti iz pojedničkog rješenja Kaggle natjecanja <sup>2</sup>.

<sup>2</sup><https://github.com/melisgl/higgsml/blob/master/xgboost-scripts/higgs-numpy.py>

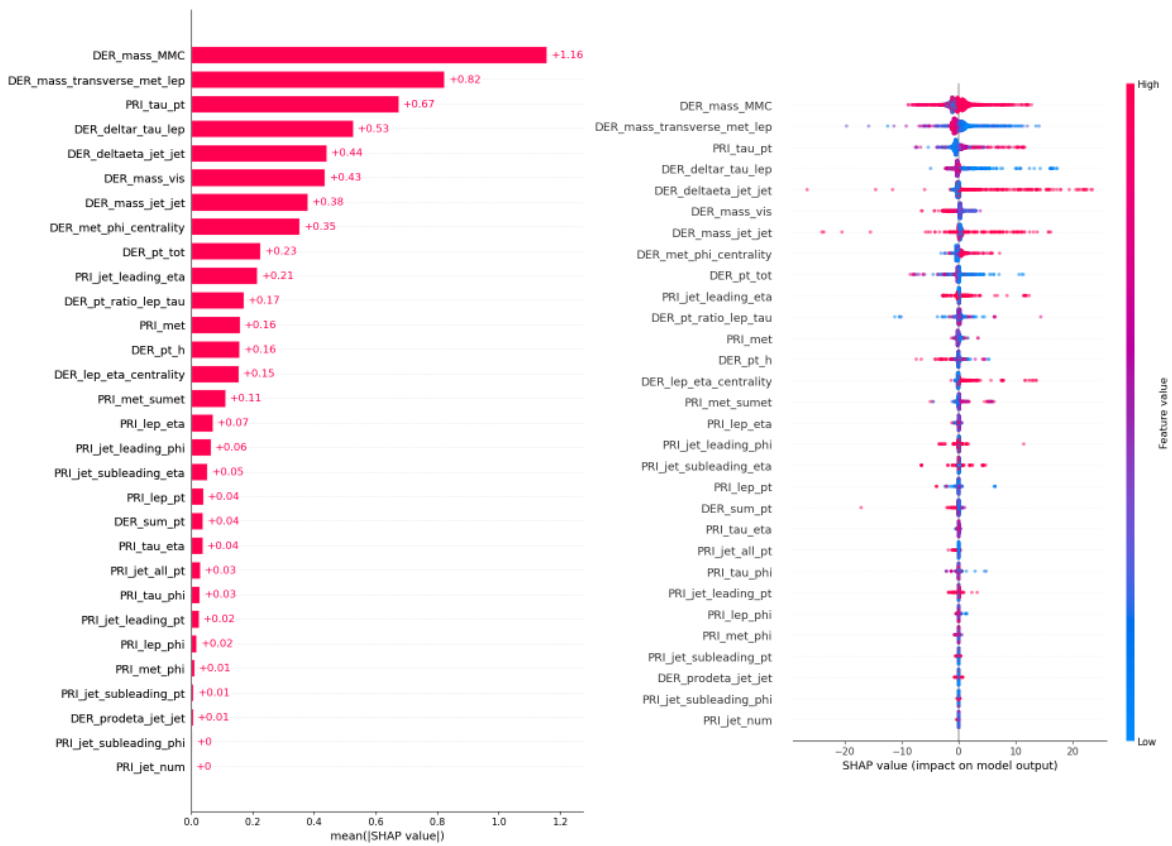
Algoritam	AMS	t [s]
Gradient boosting	3,05	110 ± 30
XGBoost	3,55	3,4 ± 0,3
LightGBM	3,10	1,6 ± 0,2
CatBoost	3,61	12,8 ± 0,3
Slučajna šuma	3,02	3,4 ± 0,2

Tablica 3.1: AMS rezultati i prosječne brzine treniranja modela

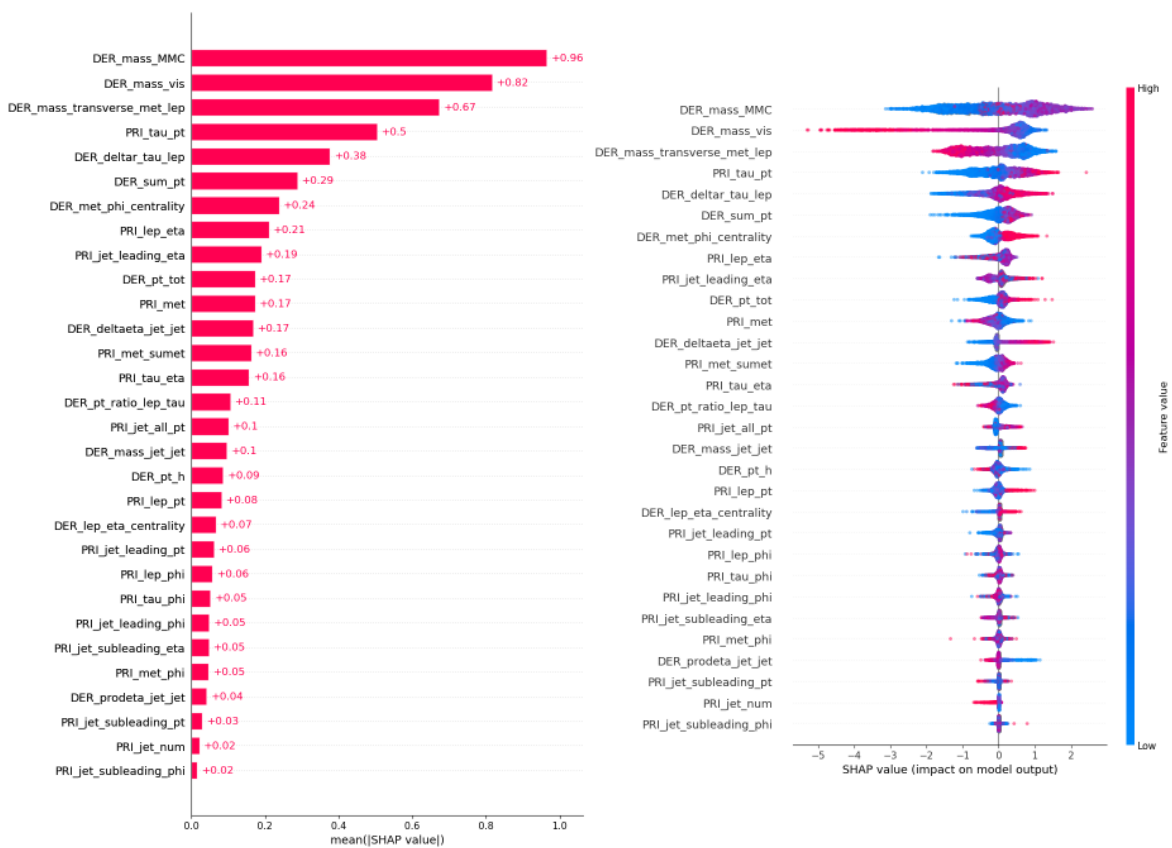
Algoritam	Točnost	Opoziv	Preciznost	ROC-AUC
Gradient boosting	0,695 ± 0,004	0,12 ± 0,01	0,94 ± 0,01	0,558 ± 0,006
XGBoost	0,824 ± 0,001	0,800 ± 0,005	0,719 ± 0,02	0,901 ± 0,001
LightGBM	0,710 ± 0,001	0,161 ± 0,005	0,95 ± 0,01	0,881 ± 0,004
CatBoost	0,794 ± 0,001	0,871 ± 0,003	0,649 ± 0,002	0,900 ± 0,001
Slučajna šuma	0,671 ± 0,001	0,039 ± 0,002	0,992 ± 0,003	0,873 ± 0,002

Tablica 3.2: Metrike ocjenjivanja modela na skupu podataka za Higgsov bozon

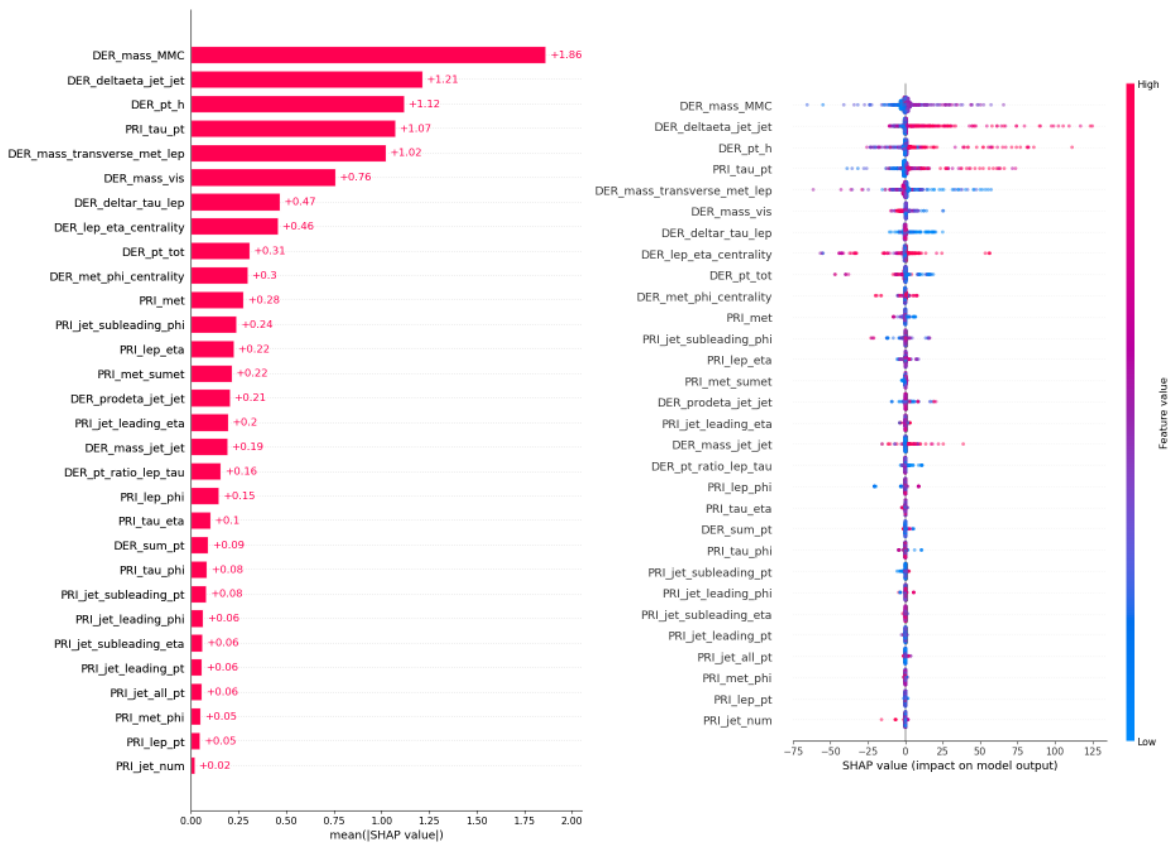
Iz tablice 3.1 je vidljivo da najbolje AMS rezultate daju algoritmi bazirani na gradient boostingu i da CatBoost daje najbolji rezultat među njima. Na originalnom natjecanju pobjednička rješenja su koristila samo XGBoost jer se CatBoost objavio četiri godine nakon natjecanja. Pomoću dodatnih metrika unutar tablice 3.2 je vidljivo zašto najbolje rezultate prikazuju CatBoost i XGBoost. Oba algoritma imaju dobru ravnotežu opoziva i preciznosti. Time je osigurana dobra ravnoteža velikog broja potencijalnih kandidata koji bi mogli biti Higgsov bozon i preciznosti pri klasificiranju. ROC-AUC rezultat gradient boostinga ukazuje da je neznatno bolji od nasumičnog nagađanja. Međutim, gradient boosting je i dalje ostvario dobar AMS rezultat. Svim modernijim inačicama gradient boostinga zajedničko je što su znatno brži od originalnog gradient boostinga te im je i brzina izvođenja programa znatno stabilnija. Značajne značajke određujemo prema prosječnoj apsolutnoj Shapleyjevoj vrijednosti koja je izračunata pomoću SHAP python paketa. Pčelinji grafovi prikazuju Shapleyjevu vrijednost značajke za svaki primjer. Bojom je označeno ako je vrijednost značajke nekog primjera niska ili visoka za tu značajku. Pčelinji grafovi također sivom bojom prikazuju Shapleyjeve vrijednosti primjera s nedostajućom vrijednosti značajke. Stupčasti grafovi prikazuju prosječnu apsolutnu Shapleyjevu vrijednost značajke. Oba grafa su poredana po značajnosti značajke. Određivale su se Shapleyjeve vrijednosti od 5000 nasumično odabranih primjera.



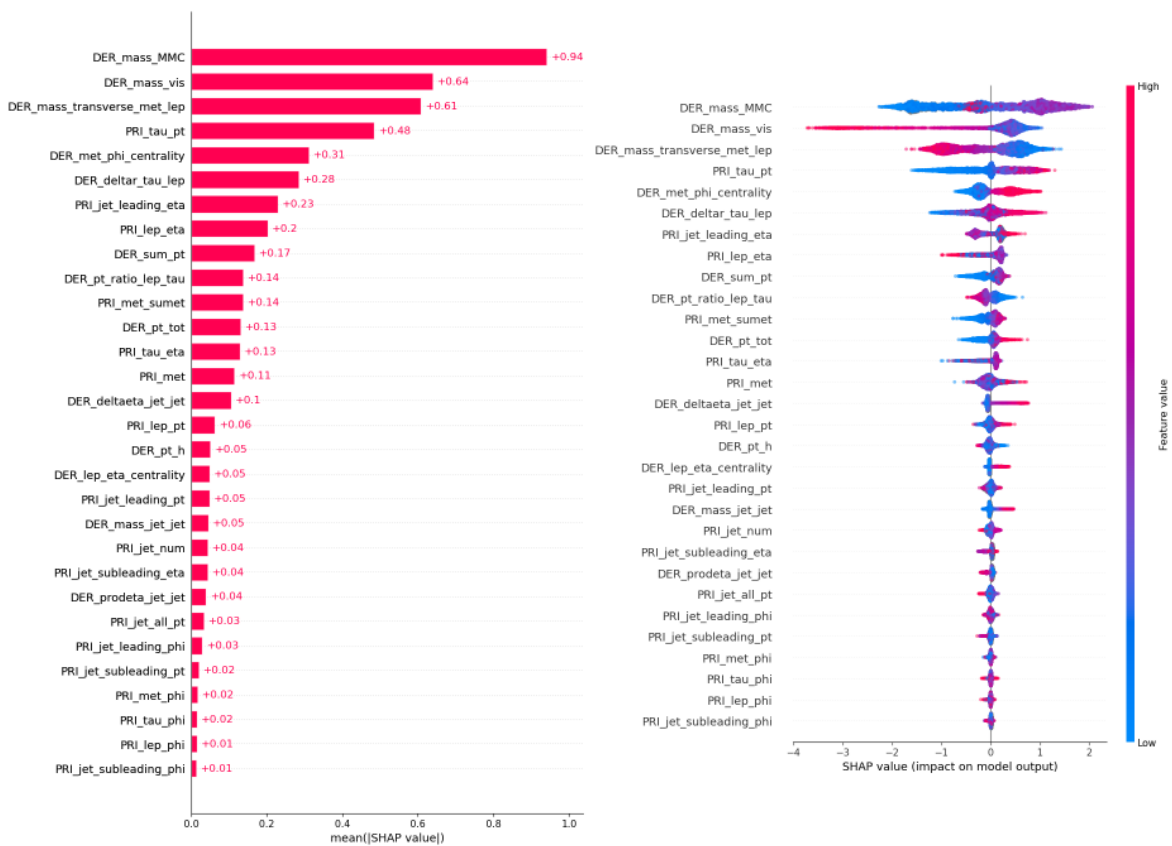
Slika 3.2: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za gradient boosting model



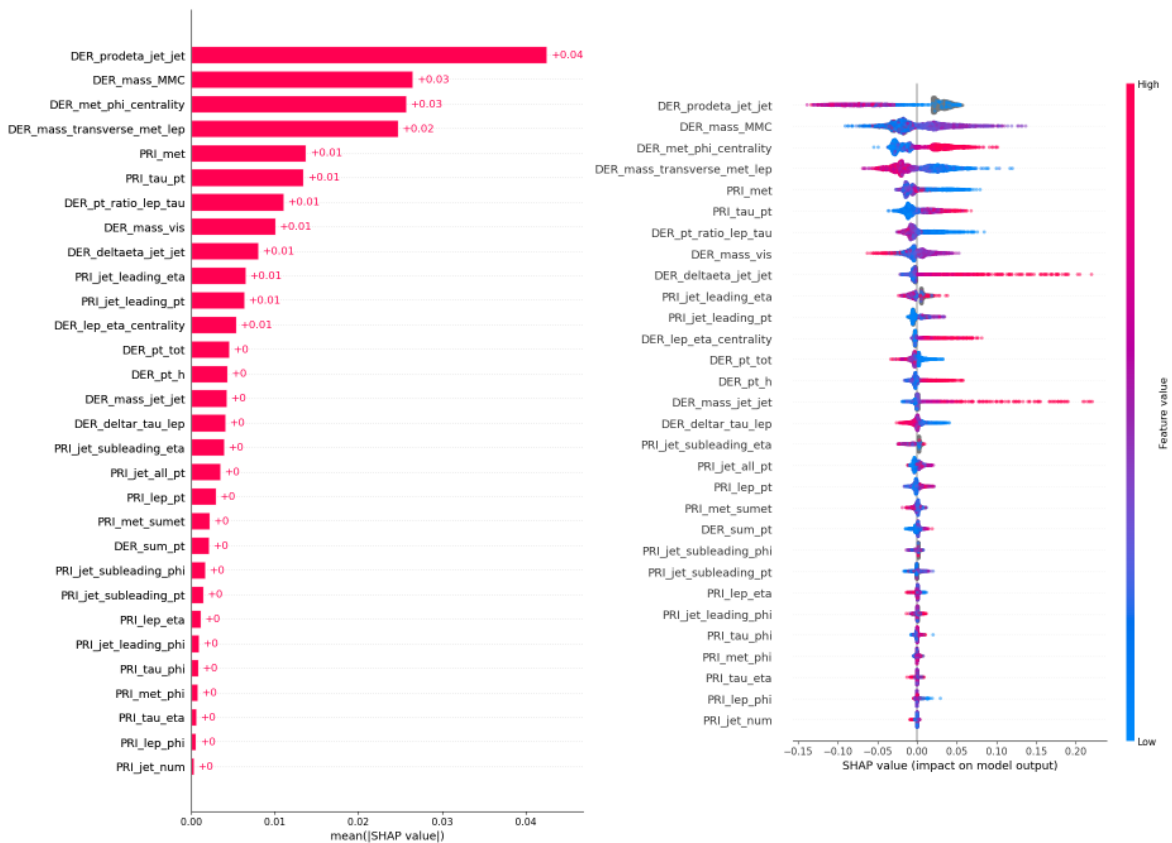
Slika 3.3: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za XGBoost model



Slika 3.4: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za LightGBM model

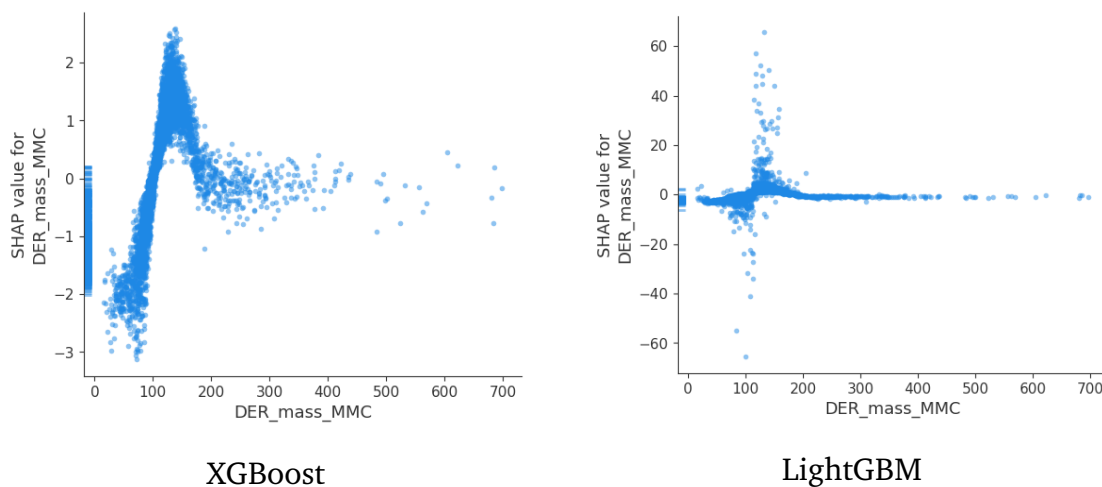


Slika 3.5: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za CatBoost model



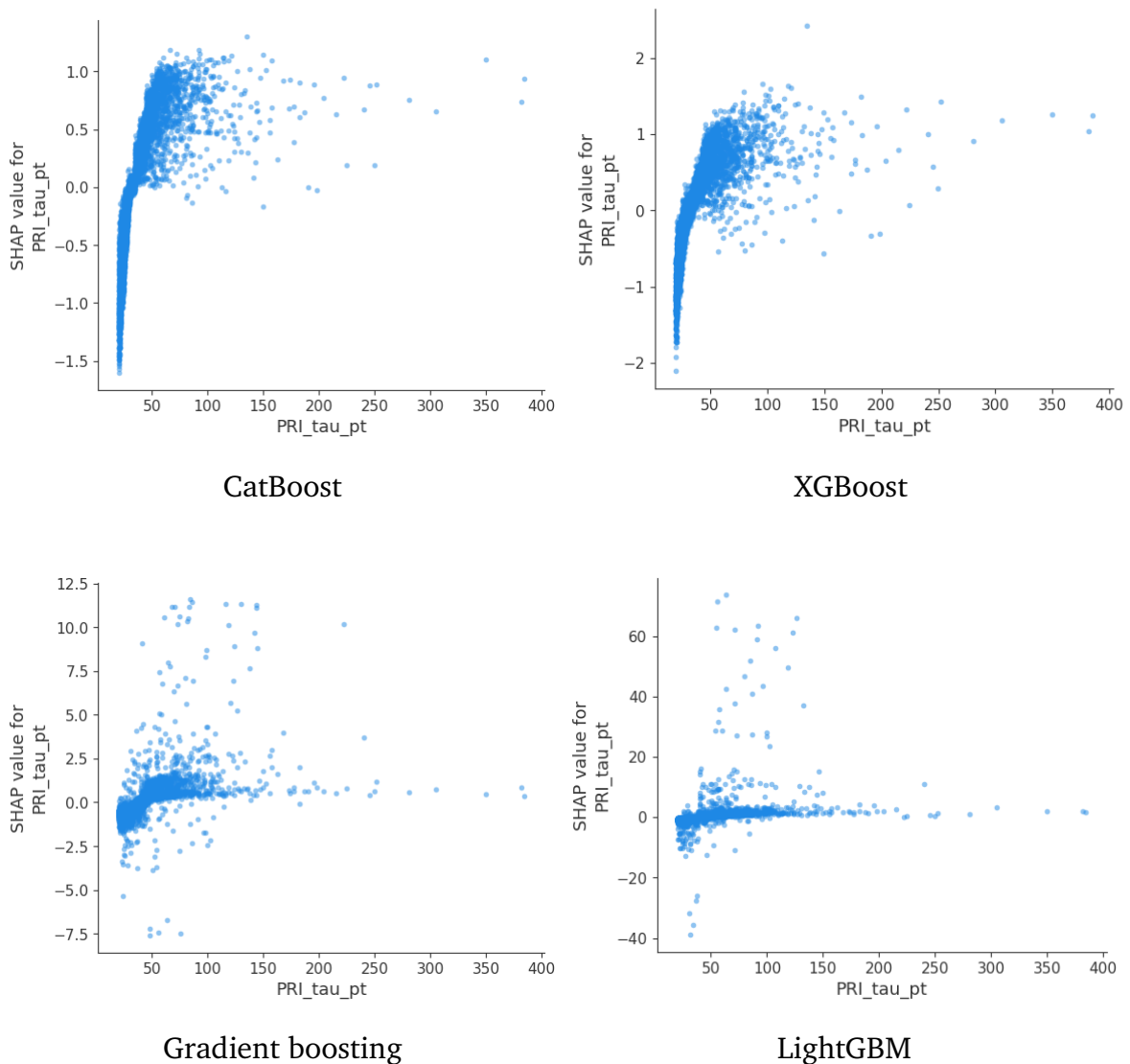
Slika 3.6: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za model slučajne šume

Svim modelima, izuzev slučajne šume, je značajka DER\_mass\_MMC najznačajnija. Ta značajka je procjena mase čestice koja je moguća Higgsova čestica izražena u  $GeV$ . Pomoću SHAP paketa je moguće doznati ovisnost Shapleyjeve vrijednosti o vrijednosti značajke.



Slika 3.7: Odnos Shapleyjeve vrijednosti značajke DER\_mass\_MMC o iznosu same značajke

Iz slike 3.7 vidljiv je mogući razlog zašto LightGBM postiže lošije rezultate. LightGBM vrijednostima iznosa [100, 200] daje puno veću važnost od XGBoosta što je vjerojatno uzrokovalo veći broj lažno pozitivno označenih primjera i gori AMS rezultat. Na slici 3.7 pravokutno su označene nedostajuće vrijednosti kojima oba algoritma daju otprilike jednaku važnost. Najznačajnija primitivna značajka većini modela je PRI\_tau\_pt koja označava iznos količine gibanja  $\tau$  čestice koja se raspala u hadrone. CatBoost i XGBoost, koji su mogli bolje rasporediti važnost vrijednosti ove značajke, su postigli bolji AMS rezultat.



Slika 3.8: Odnos Shapleyjeve vrijednosti značajke PRI\_tau\_pt o iznosu same značajke u raznim modelima

### 3.4 Plućna embolija

Plućna embolija je ozbiljan patofiziološki poremećaj koji nastaje uslijed opstrukcije plućnih arterija embolusom, najčešće trombom porijeklom iz venskog sustava donjih ekstremiteta ili zdjelice (duboka venska tromboza). Ovaj tromb putem venske cirkulacije dopijeva do desnog srca i dalje u plućnu arteriju, gdje uzrokuje prekid protoka krvi kroz plućni vaskularni sustav, što dovodi do ozbiljnih hemodinamskih i respiratornih posljedica. Skup podataka plućnih embolija, prikupljen unazad zadnjih 10 godina na Klinici za bolesti srca i krvnih žila u Kliničkoj bolnici Dubrava, sastoji se od 27 značajki i 764 primjera. Međutim, od 27 značajki, 15 njih su kategoričke značajke. Većina kategoričkih značajki unutar skupa imaju odgovore da ili ne, zamijenjene s 1 ili 0. Na grafovima ispod, PESI se odnosi na indeks težine plućne embolije, poznat kao Pulmonary Embolism Severity Index, dok je BMI, indeks tjelesne mase, važan za procjenu pretilosti. Malignancy označava prisutnost maligne bolesti, odnosno karcinoma. RDW ili širina distribucije crvenih krvnih stanica koristi se kao pokazatelj anemije, a GFR, što je brzina glomerularne filtracije, pokazuje funkciju bubrega. NOAC označava oralnu antikoagulacijsku terapiju, dok CHA<sub>2</sub>DS<sub>2</sub>VASc predstavlja skor za procjenu rizika od tromboembolije kod pacijenata s fibrilacijom atrijske. Hb se odnosi na koncentraciju hemoglobina u krvi, a CRP na C-reaktivni protein, koji je marker upale u organizmu. CAD je kratica za koronarnu bolest srca (eng. *Coronary Artery Disease*), dok DM (eng. *Diabetes Mellitus*) označava šećernu bolest. PAD se odnosi na bolest perifernih arterija, dok COPD označava kroničnu opstruktivnu plućnu bolest. Na kraju, HF (eng. *Heart Failure*) predstavlja zatajenje srca. One-hot kodiranje se primijenilo na značajke koje sadrže stringove ili koje imaju više od dvije kategoričke vrijednosti. Iz tog razloga broj značajki porastao je na 31.

U skupu podataka prevladavaju žene, koje čine 57,3% uzorka (438 od 764 pacijenata). Prosječna dob pacijenata iznosi 69,5 godina (SD = 15,2), s rasponom od 19 do 97 godina. Prosječna visina iznosi 1,69 m (SD = 0,093), dok prosječna težina iznosi 81,5 kg (SD = 17,5), s rasponom težine od 43 kg do 150 kg. Srednji BMI pacijenata iznosi 28,6 (SD = 5,5), a vrijednosti se kreću od 16,5 do 58,5. Prosječna vrijednost hemoglobina (Hb) je 131 g/L (SD = 19,2), dok je prosječni hematokrit (Hct) 0,396 (SD = 0,059). Srednja vrijednost eritrocita (E) je 4,43 (SD = 0,65). Srednja vrijednost D-dimera iznosi 8,0 mg/L (SD = 8,45), s rasponom od 0,4 do 39 mg/L. Kod 42,5% pacijenata troponin je bio pozitivan. Prosječni Wellsov skor iznosi 5,0 (SD =

2,4), dok je 9,6% pacijenata imalo povijest duboke venske tromboze (DVT). Srednja vrijednost *NTproBNP* iznosi 2544,7 pg/mL (SD = 4954,0), s rasponom od 10 do 28462 pg/mL. *CHA2DS2VASc* skor iznosi prosječno 3,43 (SD = 1,6), dok 9,6% pacijenata ima perifernu arterijsku bolest (PAD). Povijest hipertenzije ima 59,4% pacijenata, a 20,3% ima malignitet. Što se tiče srčanog ritma, 574 pacijenta (75,1%) imaju sinusni ritam, dok 190 pacijenata (24,9%) ima zabilježenu fibrilaciju atrijsku. U praćenju, 44,2% pacijenata (338) je preminulo. Prosječan indeks šoka za cijeli uzorak iznosi 0,87 (SD = 0,32), s rasponom od 0,3 do 3,0.

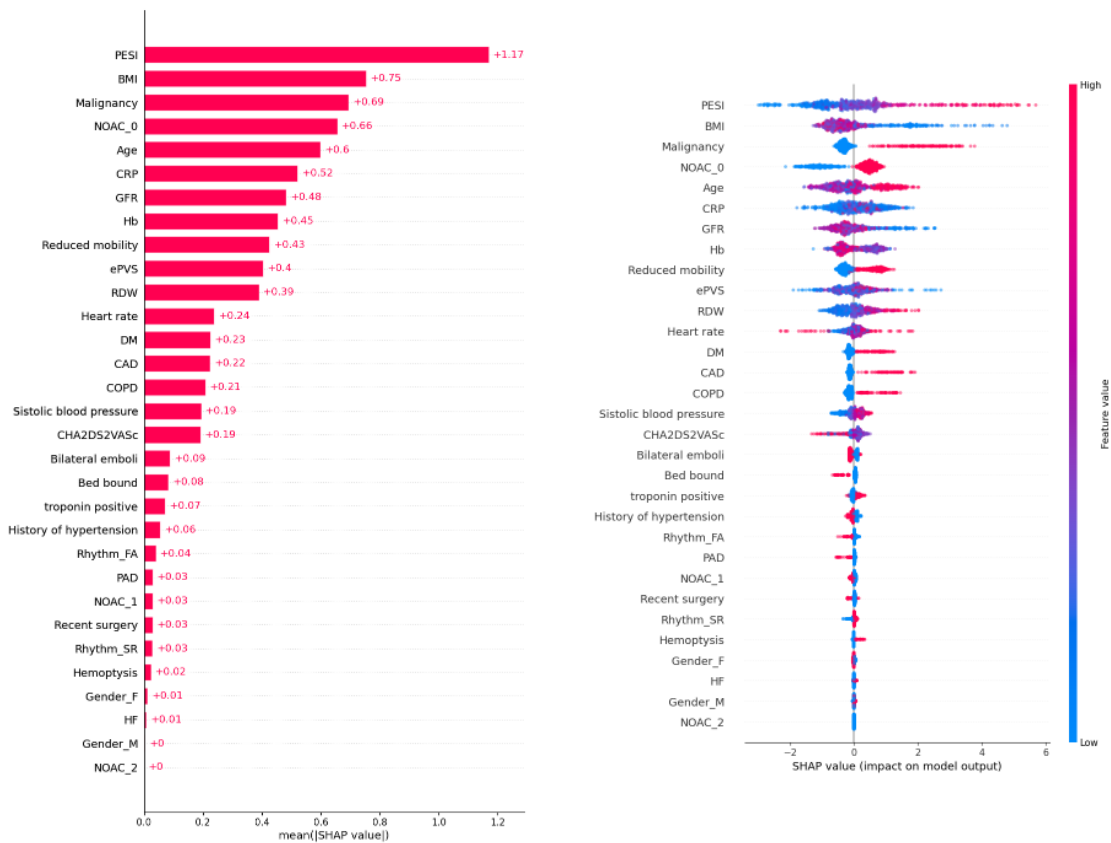
Cilj skupa podataka je pomoću vrijednosti značajki predvidjeti smrtni ishod pacijenta kroz prosječni period praćenja od 5 godina. Hiperparametri su dobiveni pomoću funkcije *GridSearchCV*. Nelinearni obrazac povezanosti pokazao se izazovnim za frekventističke statističke analize. [9, 10]

Algoritam	Točnost	Opoziv	Preciznost	ROC-AUC
Gradient boosting	0,79 ± 0,02	0,75 ± 0,07	0,78 ± 0,04	0,79 ± 0,02
XGBoost	0,77 ± 0,03	0,75 ± 0,10	0,75 ± 0,07	0,86 ± 0,03
LightGBM	0,80 ± 0,03	0,77 ± 0,08	0,78 ± 0,08	0,86 ± 0,03
CatBoost	0,79 ± 0,02	0,76 ± 0,09	0,77 ± 0,07	0,87 ± 0,03
Slučajna šuma	0,81 ± 0,03	0,79 ± 0,07	0,78 ± 0,06	0,88 ± 0,03

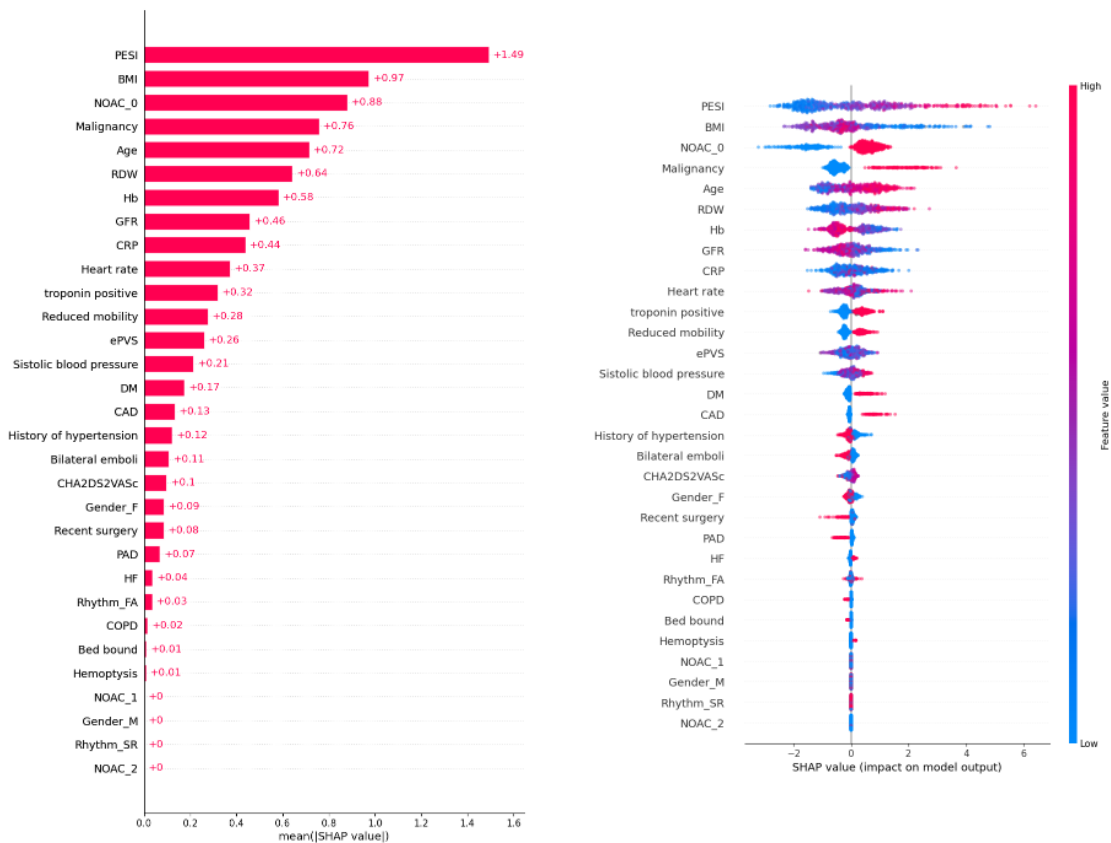
Tablica 3.3: Metrike ocjenjivanja modela na skupu podataka za plućnu emboliju

U ovom slučaju, slučajna šuma ima bolje rezultate od ostalih algoritama u svim metrikama, što nam potvrđuje prije spomenut *No Free Lunch* teorem. Makar je slučajna šuma dala najgore rezultate na prošlom skupu podataka, to ne znači da će njena primjena biti najgora na svim skupovima podataka. Najveća razlika između algoritama je u ROC-AUC rezultatu gdje gradient boosting najviše odstupa od ostalih algoritama. Važnost značajki se može iščitati s pčelinjih i stupčastih grafova *SHAP* paketa.

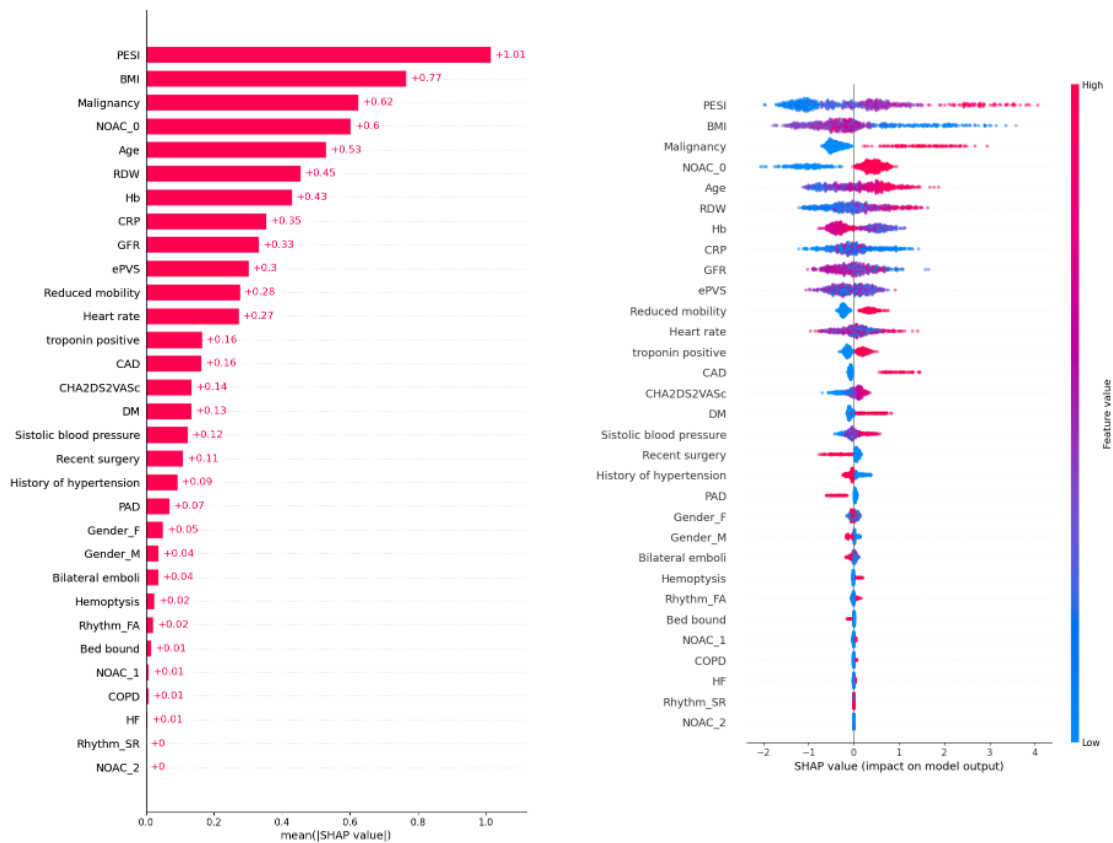




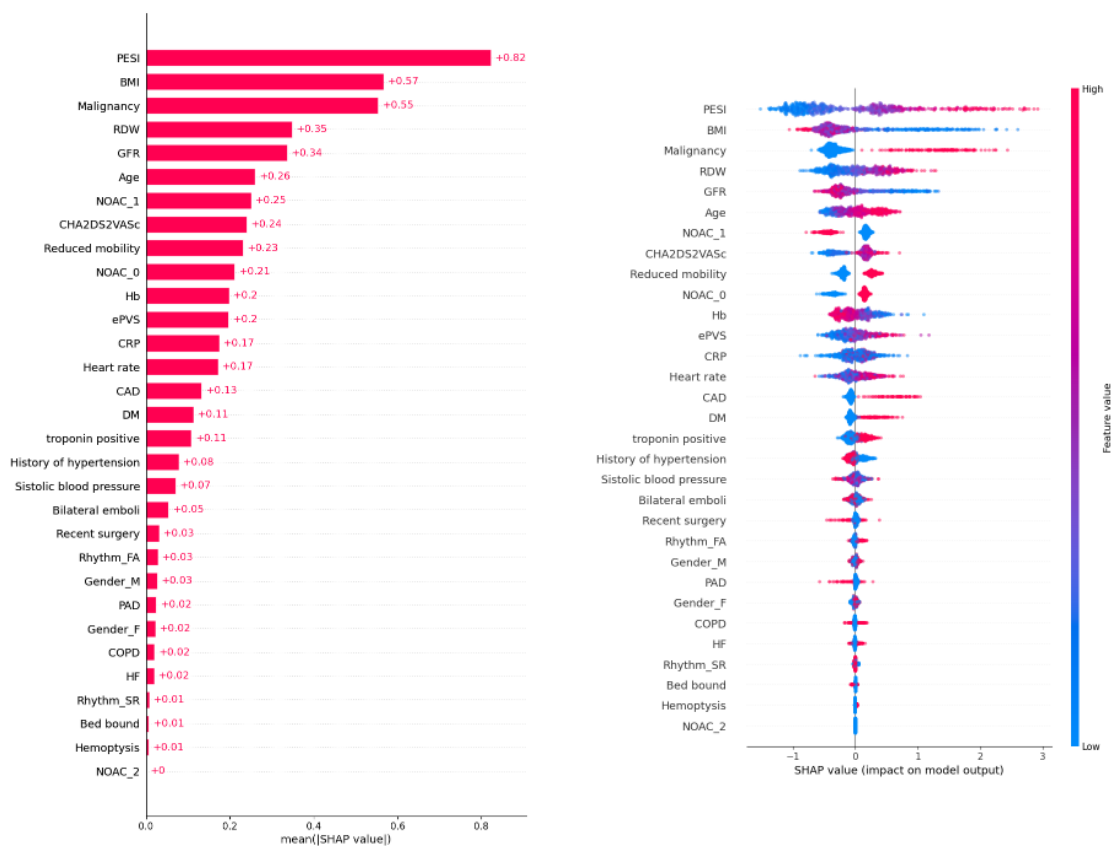
Slika 3.9: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za gradient boosting model



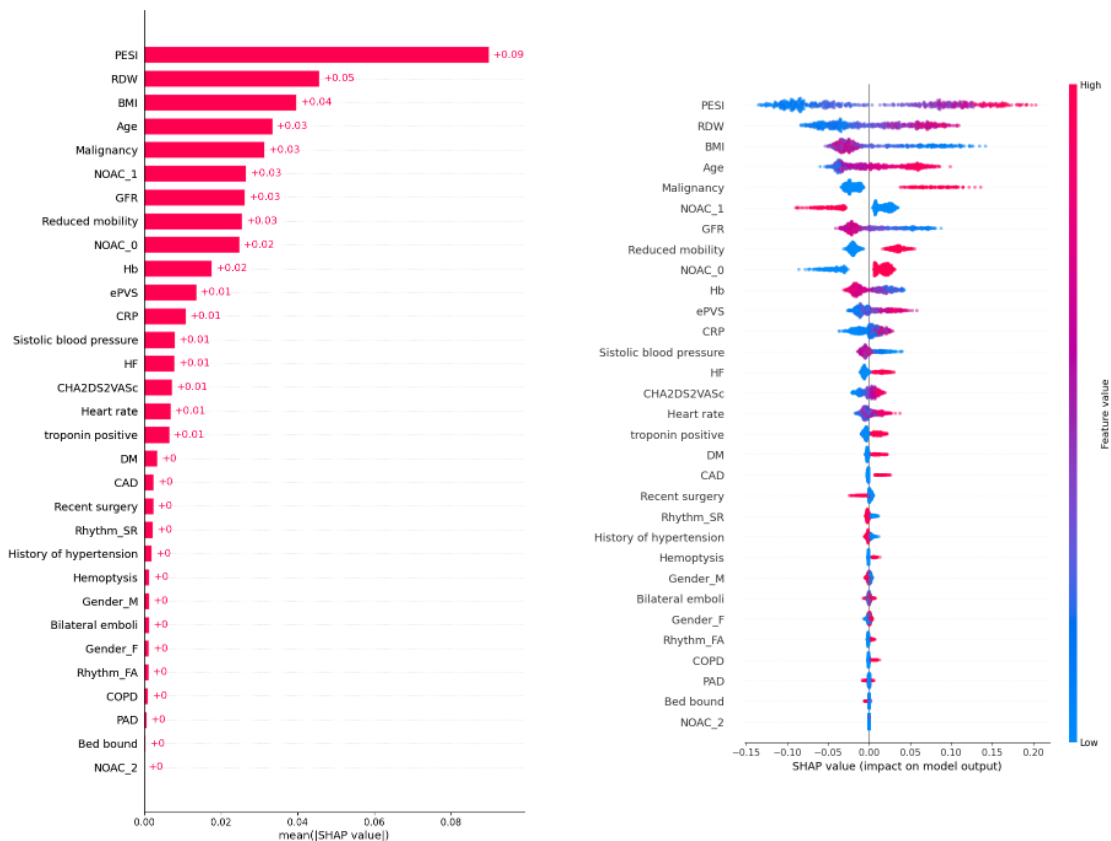
Slika 3.10: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za XGBoost model



Slika 3.11: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za LightGBM model

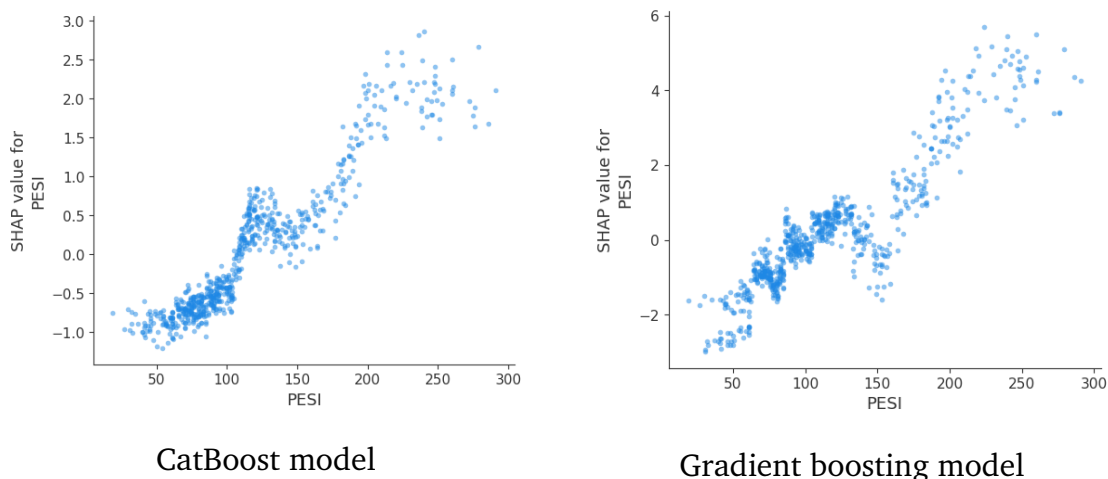


Slika 3.12: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za CatBoost model



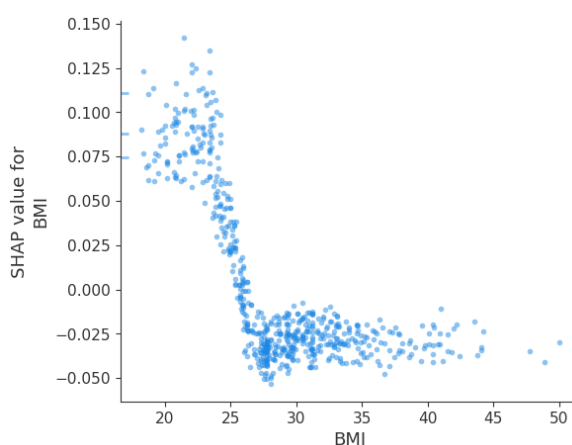
Slika 3.13: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za model slučajne šume

Oba modela pokazuju da su PESI, BMI i Malignancy najznačajnije značajke, s dominantnim utjecajem na predikciju ishoda. Veća PESI vrijednost ukazuje na rizičnijeg pacijenta te Shapleyjeve vrijednosti svih modela su također rasle s PESI značajkom. Vidi se i jasna separacija prisutnosti maligne bolesti, pri čemu crvene točke (više vrijednosti značajke) ukazuju na lošije ishode kod pacijenata s malignitetom. Također, značajke poput RDW i GFR imaju značajan utjecaj, dok značajke poput Hemoptysis i Bed bound imaju zanemariv utjecaj. Iako se raspodjela SHAP vrijednosti za pojedine značajke blago razlikuje između dva modela, ukupna interpretacija ostaje vrlo slična, što ukazuje na dosljednost između različitih modela.



Slika 3.14: Odnos Shapleyjeve vrijednosti PESI značajke o iznosu PESI

Druga najznačajnija značajka je BMI (eng. *body mass index*) te u svakom modelu niža BMI vrijednost doprinosi vjerojatnosti da pacijent neće preživjeti do sljedeće kontrole.



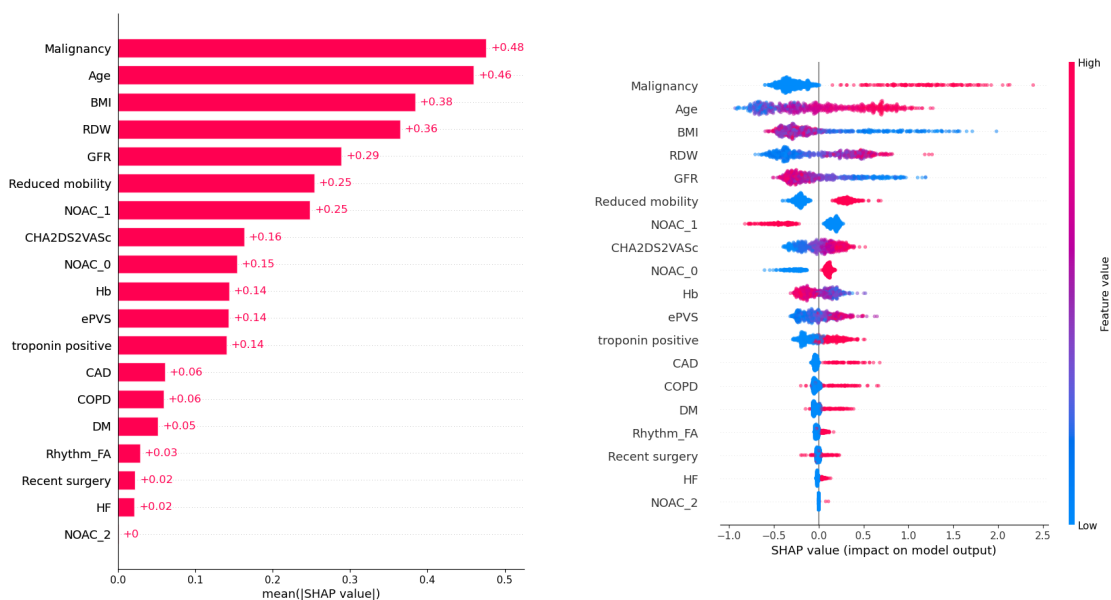
Slika 3.15: Odnos Shapleyjeve vrijednosti BMI značajke o iznosu BMI u modelu slučajne šume

Prema BMI kategorijama ljudi s BMI vrijednošću većom od 25 imaju prekomjernu težinu. Na slici 3.15 se može uočiti da pacijenti s BMI vrijednošću većom od 25 su prema modelu slučajne šume više vjerojatni da će preživjeti do sljedeće kontrole od ljudi koji su umjerene težine ili su pothranjeni.

Smanjenjem broja značajki skupa podataka se može provjeriti ako je došlo do pretreniranja. Izbacilo se sveukupno 12 značajki raznih značajnosti te se testiralo na CatBoost modelu i modelu slučajne šume istom metodom kao i u prijašnjim testiranjima.

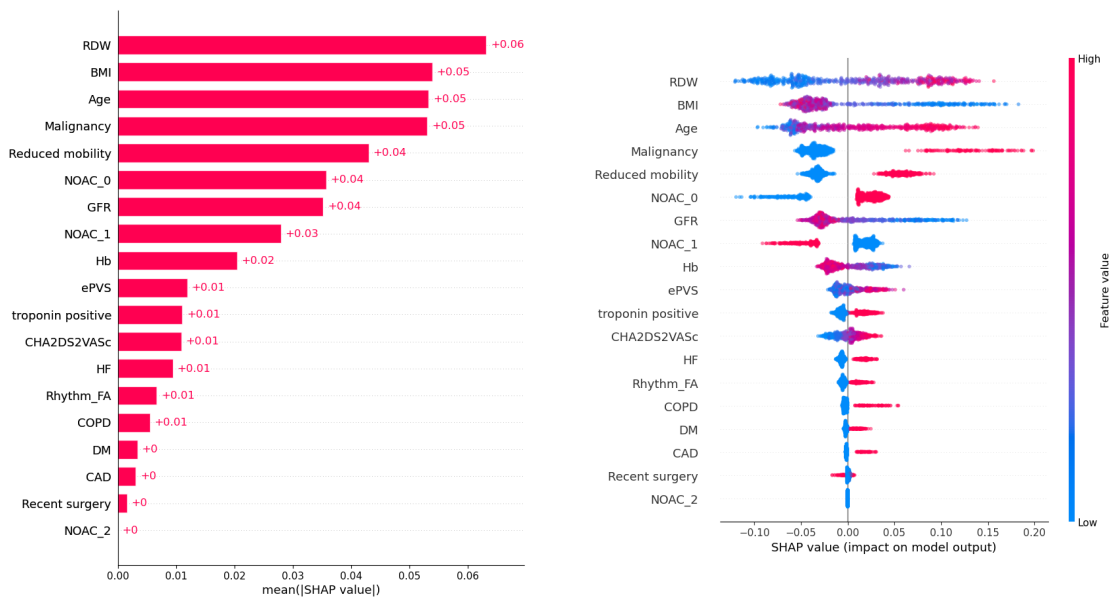
Algoritam	Točnost	Opoziv	Preciznost	ROC-AUC
CatBoost	0,78 ± 0,03	0,74 ± 0,10	0,77 ± 0,07	0,87 ± 0,04
Slučajna šuma	0,79 ± 0,03	0,75 ± 0,08	0,78 ± 0,07	0,87 ± 0,03

Tablica 3.4: Metrike ocjenjivanja modela na smanjenom skupu podataka za plućnu emboliju



Slika 3.16: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za CatBoost model treniran na manjem skupu podataka

Najviše SHAP vrijednosti zabilježene su kod značajki Malignancy, Age i BMI, pri čemu Malignancy ima najvišu SHAP vrijednost (+0.48), što ukazuje na izrazito jak utjecaj maligne bolesti na ishod pacijenata. Na desnom grafu vidljiva je jasna separacija između prisutnosti maligne bolesti (crvene točke) i pozitivnih SHAP vrijednosti, što indicira da pacijenti s malignitetom imaju lošije ishode. Ova separacija posebno je izražena kod značajki poput Age i BMI, gdje više vrijednosti tih značajki (prikazane crvenom bojom) imaju pozitivan utjecaj na model, što upućuje na viši rizik za pacijente s tim faktorima rizika.



Slika 3.17: Stupčasti (lijevo) i pčelinji (desno) SHAP graf za model slučajne šume treniran na manjem skupu podataka

Uspoređivanjem metrika unutar tablica 3.3 i 3.4 vidljivo je da su se metrike blago pogoršale ili su ostale nepromijenjene. To ukazuje da modeli dobiveni originalnim skupom podataka nisu pretrenirani. Uz to, demonstrirano je kako modeli trenirani na manjim skupovima podataka mogu postići približne rezultate kao modeli trenirani na većim skupovima. Zato je ponekad poželjno u slučaju velike dimenzionalnosti skupa podataka izbaciti neke značajke kako bi se uštedjelo na resursima.

Na slikama 3.16 i 3.17 može se uočiti da se poredak značajki promijenio, međutim nije došlo do drastičnih promjena u poretku. Značajke su neovisno o skupu podataka zadržale približnu važnost što dokazuje da algoritmi doista uče i prepoznaju bitne značajke.

## 4 Zaključak

U ovom diplomskom radu izložili smo matematičku osnovu algoritama iz strojnog učenja koja se temelji na paradigmi gradient boostinga. U svrhu prikaza efikasnosti, algoritmi su primijenjeni na problemu klasifikacije na dva različita skupa koji sadrže stvarne (mjerene) podatke. Modeli su se uspoređivali s prosječnim vrijednostima zadanih metrika. Prvi skup podataka Higgsovog bozona je demonstrirao razlike u brzinama i rezultatima između algoritama. Prikazano je kako su CatBoost i XGBoost imali najbolje rezultate prema AMS metrici. Također, u tablici 3.1 uočeno je poboljšanje u efikasnosti novijih algoritama od originalnog gradient boostinga. Brzine treniranja modela bile su od 10 do skoro 100 puta brže od originalnog algoritma. Stabilnost treniranja je isto poboljšana što je vidljivo u standardnim devijacijama. Drugi skup podataka ukazao je na važnost mjerenja svih modela na svakom skupu. Unatoč što je slučajna šuma imala najgori rezultat u skupu podataka Higgsovog bozona, u skupu podataka za plućnu emboliju je dala najbolje rezultate. Skup podataka za plućnu emboliju je također demonstrirao kako algoritmi ne zahtijevaju velik broj primjera da stvore precizna predviđanja. LightBGM nažalost nije u potpunosti pokazao svoju efikasnost jer zadani skupovi podataka nisu bili toliko rijetki da bi se mogla uočiti značajna razlika.

Algoritmi temeljeni na gradient boostingu su dokazali svoju konkurentnost i razlog zašto u brojnim natjecanjima i istraživanjima se upravo takvi algoritmi i koriste.

# Dodaci

## Dodatak A Kod za dobivanje rezultata na skupu podataka za Higgsov bozon

### A.1 Gradient boosting

```
import numpy as np
import pandas as pd
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, roc_auc_score, recall_score,
                                precision_score

pd.set_option('future.no_silent_downcasting', True)
podaci = pd.read_csv("training.csv")

y_train = podaci["Label"] #oznake
x_train = podaci.iloc[:, 1:31] #znacajke
#ispuni nedostajuće vrijednosti s medijanom znacajke
for i in x_train.columns:
    x_train[i] = x_train[i].fillna(x_train[i].median())
y_train = LabelEncoder().fit_transform(y=y_train)

podaci['test_Weight'] = podaci['Weight'] * 550000 / len(podaci["Label"
])

s = np.sum(podaci[podaci['Label']==1]['test_Weight'])
b = np.sum(podaci[podaci['Label']==0]['test_Weight'])

clf = GradientBoostingClassifier(n_iter_no_change=10, validation_fraction
                                =0.15)

param = {
    "n_estimators": [100, 150, 200],
    "min_samples_leaf": [100, 150, 200],
    "learning_rate": [0.1, 0.15],
    "max_depth": [5, 7, 10],
}
```



```

grid=GridSearchCV(clf,param_grid=param,n_jobs=-1,verbose=3)
grid.fit(x_train,y_train,sample_weight=podaci["test_Weight"])
print(grid.best_estimator_)

acc=[]
ra=[]
re=[]
pr=[]

clf=GradientBoostingClassifier(max_depth=10,min_samples_leaf=100,
                               n_estimators=200,n_iter_no_change=
                               10,validation_fraction=0.15)

for z in range(5):
    print("\n",z)
    train,test=train_test_split(podaci,test_size=0.1)
    x_train_val=train.iloc[:,1:-3]
    y_train_val=train.iloc[:,-2]
    weight=train.iloc[:,-1]
    x_test=test.iloc[:,1:-3]
    y_test=test.iloc[:,-2]
    y_train_val=LabelEncoder().fit_transform(y=y_train_val)
    y_test=LabelEncoder().fit_transform(y=y_test)
    for i in x_train_val.columns:
        x_train_val[i]=x_train_val[i].fillna(x_train_val[i].median())
        x_test[i]=x_test[i].fillna(x_test[i].median())
    clf.fit(x_train_val,y_train_val,sample_weight=weight)
    rezultati=clf.predict(x_test)

    acc.append(accuracy_score(y_true=y_test,y_pred=rezultati))
    ra.append(roc_auc_score(y_true=y_test,y_score=rezultati))
    re.append(recall_score(y_true=y_test,y_pred=rezultati))
    pr.append(precision_score(y_true=y_test,y_pred=rezultati))
print("Accuracy= ",acc)
print("ROC_AUC= ",ra)
print("Recall= ",re)
print("Precision= ",pr)
clf.fit(x_train,y_train,sample_weight=podaci["test_Weight"])

#Proizvedi rezultat za procjenu na web stranici Kaggle
test_data=pd.read_csv("test.csv")

```

```

x_test=test_data.iloc[:,1:]
for i in x_test.columns:
    x_test[i]=x_test[i].fillna(x_test[i].median())
predictions=clf.predict(x_test)
rezultati_proba=clf.predict_proba(x_test)
predictions_proba=[i[0] for i in rezultati_proba]
n=0.15*len(predictions_proba)
results=pd.DataFrame({"EventId":test_data["EventId"],"Proba":
                      predictions_proba})
results=results.sort_values(by="Proba")
results=results.reset_index(drop=True)
results=results.assign(RankOrder=range(1,550001))
results=results.assign(Class="b")
for i in range(len(predictions_proba)):
    if (results.iloc[i,2]<n):
        results.iloc[i,3]="s"

results=results.sort_values(by="EventId",ascending=True)
results=results.reset_index(drop=True)
results=results.drop("Proba",axis=1)

results.to_csv("submission_GB.csv",index=False)

```

## A.2 XGBoost

```

import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_validate

podaci= pd.read_csv("training.csv")
podaci=podaci.replace(-999.00,np.nan) #zamjeni -999.00 s indikatorom
                                     za nedostajuce vrijednosti

y_train=podaci["Label"] #oznake
x_train=podaci.iloc[:,1:31] #znacajke
y_train=LabelEncoder().fit_transform(y=y_train)

```

```

podaci['test_Weight'] = podaci['Weight'] * 550000 / len(podaci["Label"
])
s = np.sum(podaci[podaci['Label']=='s']['test_Weight'])
b = np.sum(podaci[podaci['Label']=='b']['test_Weight'])

clf=xgb.XGBClassifier(n_estimators=360,max_depth=6,eta=0.15,
                      scale_pos_weight=b/s,objective="
                      binary:logitraw",eval_metric=["auc"
                      ,"ams@0.15"])

cv=cross_validate(clf,X=x_train,y=y_train,scoring=["accuracy","roc_auc
","recall","precision"],params={"
                      sample_weight":podaci["test_Weight"
                      ]})

print("Accuracy= ",cv["test_accuracy"])
print("ROC_AUC= ",cv["test_roc_auc"])
print("Recall= ",cv["test_recall"])
print("Precision= ",cv["test_precision"])

clf.fit(x_train,y_train,sample_weight=podaci["test_Weight"])

#Proizvedi rezultat za procjenu na web stranici Kaggle
test=pd.read_csv("test.csv")
test=test.replace(-999.00,np.nan)
x_test=test.iloc[:,1:]
rezultati_proba=clf.predict_proba(x_test)

rezultati_proba=clf.predict_proba(x_test)
predictions_proba=[i[0] for i in rezultati_proba]
n=0.15*len(predictions_proba)
results=pd.DataFrame({"EventId":test["EventId"],"Proba":
                      predictions_proba})
results=results.sort_values(by="Proba")
results=results.reset_index(drop=True)
results=results.assign(RankOrder=range(1,550001))
results=results.assign(Class="b")
for i in range(len(predictions_proba)):
    if (results.iloc[i,2]<n):
        results.iloc[i,3]="s"

```

```

results=results.sort_values(by="EventId",ascending=True)
results=results.reset_index(drop=True)
results=results.drop("Proba",axis=1)

results.to_csv("submission_XGB.csv",index=False)

```

### A.3 LightGBM

```

import numpy as np
import pandas as pd
import lightgbm as lgbm
from sklearn.model_selection import GridSearchCV, cross_validate
from sklearn.preprocessing import LabelEncoder

pd.set_option('future.no_silent_downcasting', True)
podaci = pd.read_csv("training.csv")
podaci=podaci.replace(-999.00,np.nan) #zamjeni -999.00 s indikatorom
                                     za nedostajuce vrijednosti
y_train=podaci["Label"] #oznake
x_train=podaci.iloc[:,1:31] #znacajke
y_train=LabelEncoder().fit_transform(y=y_train)

podaci['test_Weight'] = podaci['Weight'] * 550000 / len(podaci["Label"
                                                         ])
s = np.sum(podaci[podaci['Label']=="s"]['test_Weight'])
b = np.sum(podaci[podaci['Label']=="b"]['test_Weight'])

clf=lgbm.LGBMClassifier()
param={
    "n_estimators": [100, 150, 200],
    "min_child_samples": [100, 150, 200],
    "learning_rate": [0.1, 0.15],
    "max_depth": [10, 12, 14],
    "num_leaves": [10, 20, 30]
}
grid=GridSearchCV(clf, param_grid=param, n_jobs=-1, verbose=3)
grid.fit(x_train, y_train, sample_weight=podaci["test_Weight"])
print(grid.best_estimator_)

```

```

clf=lgbm.LGBMClassifier(learning_rate=0.15, max_depth=14,
                        min_child_samples=100,
                        n_estimators=200, num_leaves=30)
cv=cross_validate(clf,X=x_train,y=y_train,scoring=["accuracy","roc_auc",
          "recall","precision"],params={"sample_weight":podaci["test_Weight"]})

print("Accuracy= ",cv["test_accuracy"])
print("ROC_AUC= ",cv["test_roc_auc"])
print("Recall= ",cv["test_recall"])
print("Precision= ",cv["test_precision"])

clf.fit(x_train,y_train,sample_weight=podaci["test_Weight"])

#Proizvedi rezultat za procjenu na web stranici Kaggle
test_data=pd.read_csv("test.csv")
test_data=test_data.replace(-999.00,np.nan)
x_test=test_data.iloc[:,1:]
rezultati_proba=clf.predict_proba(x_test)
predictions_proba=[i[0] for i in rezultati_proba]
n=0.15*len(predictions_proba)
results=pd.DataFrame({"EventId":test_data["EventId"],"Proba":
                    predictions_proba})
results=results.sort_values(by="Proba")
results=results.reset_index(drop=True)
results=results.assign(RankOrder=range(1,550001))
results=results.assign(Class="b")
for i in range(len(predictions_proba)):
    if (results.iloc[i,2]<n):
        results.iloc[i,3]="s"

results=results.sort_values(by="EventId",ascending=True)
results=results.reset_index(drop=True)
results=results.drop("Proba",axis=1)

results.to_csv("submission_LGBM.csv",index=False)

```

## A.4 CatBoost

```
import pandas as pd
import numpy as np
import catboost as cb
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV, cross_validate

pd.set_option('future.no_silent_downcasting', True)
podaci = pd.read_csv("training.csv")
podaci = podaci.replace(-999.00, np.nan) #zamjeni -999.00 s indikatorom
                                         za nedostajuce vrijednosti

y_train = podaci["Label"] #oznake
x_train = podaci.iloc[:, 1:31] #znacajke
y_train = LabelEncoder().fit_transform(y=y_train)
clf = cb.CatBoostClassifier(loss_function="Logloss", scale_pos_weight=b/s
                             , verbose=0)

param = {
    "n_estimators": [100, 150, 200],
    "learning_rate": [0.1, 0.15, 0.2],
    "max_depth": [6, 7, 8, 9, 10],
    "min_child_samples": [10, 20, 30, 40],
    "nan_mode": ["Min", "Max"]
}

clf = clf.grid_search(param_grid=param, X=train_pool, cv=5)
print(clf)

clf = cb.CatBoostClassifier(loss_function="Logloss", scale_pos_weight=b/s
                             , min_data_in_leaf= 10, depth= 9,
                             learning_rate= 0.1, iterations=200)
cv = cross_validate(clf, X=x, y=y, scoring=["accuracy", "roc_auc", "recall", "
                                         precision"], params={"sample_weight"
                                         : podaci["test_Weight"]})

print("Accuracy= ", cv["test_accuracy"])
print("ROC_AUC= ", cv["test_roc_auc"])
print("Recall= ", cv["test_recall"])
print("Precision= ", cv["test_precision"])
clf.fit(train_pool)

#Proizvedi rezultat za procjenu na web stranici Kaggle
```

```

test_data=pd.read_csv("test.csv")
test_data=test_data.replace(-999.00,np.nan)
x_test=test_data.iloc[:,1:]
test_pool=cb.Pool(x_test)
rezultati=clf.predict(test_pool)
rezultati_proba=clf.predict_proba(test_pool)
predictions_proba=[i[0] for i in rezultati_proba]
n=0.15*len(predictions_proba)
results=pd.DataFrame({"EventId":test_data["EventId"],"Proba":
                      predictions_proba})
results=results.sort_values(by="Proba")
results=results.reset_index(drop=True)
results=results.assign(RankOrder=range(1,550001))
results=results.assign(Class="b")
for i in range(len(predictions_proba)):
    if (results.iloc[i,2]<n):
        results.iloc[i,3]="s"

results=results.sort_values(by="EventId",ascending=True)
results=results.reset_index(drop=True)
results=results.drop("Proba",axis=1)

results.to_csv("submission_CB.csv",index=False)

```

## A.5 Slučajna šuma

```

import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, cross_validate
from sklearn.preprocessing import LabelEncoder

pd.set_option('future.no_silent_downcasting', True)
podaci= pd.read_csv("C:\\Alan\\Diplomski\\higgs\\training.csv")
podaci=podaci.replace(-999.00,np.nan) #zamjeni -999.00 s indikatorom
                                     za nedostajuce vrijednosti

y_train=podaci["Label"] #oznake
x_train=podaci.iloc[:,1:31] #znacajke

```

```

y_train=LabelEncoder().fit_transform(y=y_train)

podaci['test_Weight'] = podaci['Weight'] * 550000 / len(podaci["Label"
])

clf=RandomForestClassifier()
param={
    "n_estimators": [50, 100, 150, 200],
    "criterion": ["gini"],
    "min_samples_leaf": [10, 50, 100],
    "max_depth": [6, 8, 10],
    "class_weight": ["balanced", None],
    "n_jobs": [-1]
}
grid=GridSearchCV(clf, param_grid=param, n_jobs=-1, verbose=3)
grid.fit(x_train, y_train, sample_weight=podaci["test_Weight"])
print(grid.best_estimator_)

clf=RandomForestClassifier(class_weight='balanced', max_depth=10,
                           min_samples_leaf=10, n_estimators=50, n_jobs=-1
                           )
cv=cross_validate(clf, X=x_train, y=y_train, scoring=["accuracy", "roc_auc",
            "recall", "precision"], params={"sample_weight": podaci["test_Weight"]
            })
print("Accuracy= ", cv["test_accuracy"])
print("ROC_AUC= ", cv["test_roc_auc"])
print("Recall= ", cv["test_recall"])
print("Precision= ", cv["test_precision"])
clf.fit(x_train, y_train, sample_weight=podaci["test_Weight"])

test_data=pd.read_csv("test.csv")
test_data=test_data.replace(-999.00, np.nan)
x_test=test_data.iloc[:, 1:]
rezultati_proba=clf.predict_proba(x_test)
predictions_proba=[i[0] for i in rezultati_proba]
n=0.15*len(predictions_proba)
results=pd.DataFrame({"EventId": test_data["EventId"], "Proba":
                    predictions_proba})
results=results.sort_values(by="Proba")
results=results.reset_index(drop=True)

```



```

results=results.assign(RankOrder=range(1,550001))
results=results.assign(Class="b")
for i in range(len(predictions_proba)):
    if (results.iloc[i,2]<n):
        results.iloc[i,3]="s"

results=results.sort_values(by="EventId",ascending=True)
results=results.reset_index(drop=True)
results=results.drop("Proba",axis=1)

results.to_csv("submission_RF.csv",index=False)

```

## Dodatak B Kod za dobivanje rezultata na skupu podataka za plućnu emboliju

### B.1 Gradient boosting

```

import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV,
                                cross_validate
from sklearn.metrics import accuracy_score, roc_auc_score,
                                recall_score, precision_score

import lightgbm as lgb
import numpy as np
data=pd.read_excel("Embolije.xlsx",usecols="B,C,F,H,K,M,P,R,S,U,V,X:AA
                                ,AF:AI,AV:AZ,BF,BH,BY,BZ")

data=pd.get_dummies(data,columns=["Gender","Rhythm","NOAC"],dtype=int)
y=data["Death at follow-up"]
data=data.drop("Death at follow-up",axis=1)
data["Death at follow-up"]=y
data=data.replace("#NULL!",pd.NA)

train,test=train_test_split(data,test_size=0.2,random_state=42)

x_train=train.iloc[:, :-1]
y_train=train.iloc[:, -1]

```

```

x_test=test.iloc[:, :-1]
y_test=test.iloc[:, -1]
# popuni nedostajuce varijable
for i in x_train.columns:
    x_train[i]=x_train[i].fillna(x_train[i].median())
    x_test[i]=x_test[i].fillna(x_test[i].median())

clf=GradientBoostingClassifier()
param={
    "n_estimators": [100, 150, 200, 250, 300],
    "min_samples_leaf": [5, 6, 7, 8, 9, 10, 11, 12],
    "learning_rate": [0.1, 0.15, 0.2, 0.25, 0.3]
}
grid=GridSearchCV(clf, param_grid=param, n_jobs=-1)
grid.fit(x_train, y_train)
print(grid.best_estimator_)

acc=[]
ra=[]
re=[]
pr=[]

clf=GradientBoostingClassifier(learning_rate=0.2, min_samples_leaf=10,
                               n_estimators=200)
for z in range(5):
    train, test=train_test_split(data, test_size=0.1)

    x_train=train.iloc[:, :-1]
    y_train=train.iloc[:, -1]
    x_test=test.iloc[:, :-1]
    y_test=test.iloc[:, -1]
    for i in x_train.columns:
        x_train[i]=x_train[i].fillna(x_train[i].median())
        x_test[i]=x_test[i].fillna(x_test[i].median())
    clf.fit(x_train, y_train)
    rezultati=clf.predict(x_test)
    print("\n", z)
    acc.append(accuracy_score(y_true=y_test, y_pred=rezultati))
    ra.append(roc_auc_score(y_true=y_test, y_score=rezultati))
    re.append(recall_score(y_true=y_test, y_pred=rezultati))

```

```

    pr.append(precision_score(y_true=y_test,y_pred=rezultati))
print("Accuracy= ",acc)
print("ROC_AUC= ",ra)
print("Recall= ",re)
print("Precision= ",pr)

```

## B.2 XGBoost

```

import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV,
                                cross_validate
from sklearn.metrics import accuracy_score, roc_auc_score,
                                recall_score, precision_score

import xgboost as xgb
data=pd.read_excel("Embolije.xlsx",usecols="B,C,F,H,K,M,P,R,S,U,V,X:AA
                                ,AF:AI,AV:AZ,BF,BH,BY,BZ")

data=pd.get_dummies(data,columns=["Gender","Rhythm","NOAC"],dtype=int)
y=data["Death at follow-up"]
data=data.drop("Death at follow-up",axis=1)
data["Death at follow-up"]=y
data=data.replace("#NULL!",pd.NA)

train,test=train_test_split(data,test_size=0.2,random_state=42)

x_train=train.iloc[:, :-1]
y_train=train.iloc[:, -1]
x_test=test.iloc[:, :-1]
y_test=test.iloc[:, -1]

clf=xgb.XGBClassifier()
param={
    "n_estimators": [100, 150, 200, 250, 300],
    "learning_rate": [0.1, 0.15, 0.2, 0.25, 0.3],
    "max_depth": [3, 4, 5, 6, 7, 8, 9, 10],
    "objective": ["binary:logistic"]
}
grid=GridSearchCV(clf,param_grid=param,n_jobs=-1)

```

```

grid.fit(x_train,y_train)
print(grid.best_params_)

clf=xgb.XGBClassifier(learning_rate=0.3,max_depth=8,n_estimators=200,
                      objective="binary:logistic")
cv=cross_validate(clf,data.iloc[:, :-1],y,scoring=["accuracy","roc_auc",
          ,"recall","precision"])
print("Accuracy= ",cv["test_accuracy"])
print("ROC_AUC= ",cv["test_roc_auc"])
print("Recall= ",cv["test_recall"])
print("Precision= ",cv["test_precision"])

```

### B.3 LightGBM

```

import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV,
                                cross_validate
from sklearn.metrics import accuracy_score, roc_auc_score,
                                recall_score, precision_score

import lightgbm as lgb
import numpy as np
data=pd.read_excel("Embolije.xlsx",usecols="B,C,F,H,K,M,P,R,S,U,V,X:AA
                ,AF:AI,AV:AZ,BF,BH,BY,BZ")

data=pd.get_dummies(data,columns=["Gender","Rhythm","NOAC"],dtype=int)
y=data["Death at follow-up"]
data=data.drop("Death at follow-up",axis=1)
data["Death at follow-up"]=y
data=data.replace("#NULL!",pd.NA)

train,test=train_test_split(data,test_size=0.2,random_state=42)

x_train=train.iloc[:, :-1]
y_train=train.iloc[:, -1]
x_test=test.iloc[:, :-1]
y_test=test.iloc[:, -1]

clf=lgb.LGBMClassifier()
param={

```

```

    "n_estimators": [100, 150, 200],
    "learning_rate": [0.1, 0.15, 0.2],
    "max_depth": [6, 7, 8, 9, 10],
    "min_child_samples": [10, 20, 30, 40],
    "num_leaves": [10, 20, 31, 40],
}
grid=GridSearchCV(clf, param_grid=param, n_jobs=-1)
grid.fit(x_train, y_train)
print(grid.best_params_)

clf=lgb.LGBMClassifier(learning_rate=0.15, max_depth=9,
                       min_child_samples=40, n_estimators=
                       100, num_leaves=20)
cv=cross_validate(clf, data.iloc[:, :-1], y, scoring=["accuracy", "roc_auc",
                                                    "recall", "precision"])

print("Accuracy= ", cv["test_accuracy"])
print("ROC_AUC= ", cv["test_roc_auc"])
print("Recall= ", cv["test_recall"])
print("Precision= ", cv["test_precision"])

```

## B.4 CatBoost

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV,
                                   cross_validate
from sklearn.metrics import accuracy_score, roc_auc_score,
                             recall_score, precision_score
import catboost as cb
data=pd.read_excel("Embolije.xlsx", usecols="B,C,F,H,K,M,P,R,S,U,V,X:AA
                  ,AF:AI,AV:AZ,BF,BH,BY,BZ")

data=pd.get_dummies(data, columns=["Gender", "Rhythm", "NOAC"], dtype=int)
y=data["Death at follow-up"]
data=data.drop("Death at follow-up", axis=1)
data["Death at follow-up"]=y
data=data.replace("#NULL!", pd.NA)

```

```

param={
    "n_estimators": [100, 150, 200],
    "learning_rate": [0.1, 0.15, 0.2],
    "max_depth": [6, 7, 8, 9, 10],
    "min_child_samples": [10, 20, 30, 40],
    "nan_mode": ["Min", "Max"]
}

train, test = train_test_split(data, test_size=0.1)

x_train = train.iloc[:, :-1]
y_train = train.iloc[:, -1]

clf = cb.CatBoostClassifier(verbose=0).grid_search(param, x_train, y_train
                                                , verbose=False)

print(clf)

clf = cb.CatBoostClassifier(min_data_in_leaf=10, depth=7, learning_rate=0.
                            1, iterations=100, verbose=0, nan_mode
                            ="Min")

cv = cross_validate(clf, data.iloc[:, :-1], y, scoring=["accuracy", "roc_auc"
                                                    , "recall", "precision"])

print("Accuracy= ", cv["test_accuracy"])
print("ROC_AUC= ", cv["test_roc_auc"])
print("Recall= ", cv["test_recall"])
print("Precision= ", cv["test_precision"])

```

## B.5 Slučajna šuma

```

import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV,
                                     cross_validate
from sklearn.metrics import accuracy_score, roc_auc_score,
                                     recall_score, precision_score
from sklearn.impute import SimpleImputer
data = pd.read_excel("Embolije.xlsx", usecols="B,C,F,H,K,M,P,R,S,U,V,X:AA
                                     ,AF:AI,AV:AZ,BF,BH,BY,BZ")
data = pd.get_dummies(data, columns=["Gender", "Rhythm", "NOAC"], dtype=int)
y = data["Death at follow-up"]

```

```

data=data.drop("Death at follow-up",axis=1)
data["Death at follow-up"]=y
data=data.replace("#NULL!",pd.NA)

train,test=train_test_split(data,test_size=0.2,random_state=42)

x_train=train.iloc[:, :-1]
y_train=train.iloc[:, -1]
x_test=test.iloc[:, :-1]
y_test=test.iloc[:, -1]
clf=RandomForestClassifier()
param={
    "n_estimators": [100, 150, 200, 250, 300],
    "min_samples_leaf": [5, 6, 7, 8, 9, 10, 11, 12],
    "max_depth": [3, 4, 5, 6, 7, 8, 9, 10]
}
grid=GridSearchCV(clf, param_grid=param, n_jobs=-1)
grid.fit(x_train, y_train)
print(grid.best_estimator_)
clf=RandomForestClassifier(max_depth=9, min_samples_leaf=5,
                           n_estimators=250)
cv=cross_validate(clf, data.iloc[:, :-1], y, scoring=["accuracy", "roc_auc",
                                                    "recall", "precision"])
print("Accuracy= ", cv["test_accuracy"])
print("ROC_AUC= ", cv["test_roc_auc"])
print("Recall= ", cv["test_recall"])
print("Precision= ", cv["test_precision"])

```

## Dodatak C Funkcija za dobivanje SHAP grafova

```

import shap
import matplotlib.pyplot as plt

def plot_shap_summary(clf, x_train):

    # u slucaju velikog broja primjera
    #x_train=shap.utils.sample(x_train, 5000)
    interpreter=shap.Explainer(clf, x_train)

```

```

explanation=interpreter(x_train,check_additivity=False)# dodati
                [:,:,1] u slucaju da je
                klasifikator slucajna cuma

shap_values = pd.DataFrame(explanation.values, columns=x_train.
                           columns)

shap.plots.bar(explanation,max_display=100)

shap.plots.beeswarm(explanation,max_display=100,alpha=0.5)

# Izracunaj prosjecnu apsolutnu vrijednost svih znacajki i spremi
                imena 5 najznacajnijih
mean_abs_shap_values = shap_values.abs().mean().sort_values(
                ascending=False)
top_features = mean_abs_shap_values.head(5).index

# Nacrtaj i spremi grafove ovisnosti najznacajnijih znacajki
for feature in top_features:
    plt.figure()
    shap.dependence_plot(feature, shap_values.values, x_train,
                interaction_index=None,
                show=False,alpha=0.5)
    plt.savefig(feature,bbox_inches = 'tight')

```



## Literatura

- [1] Hastie T.; Tibshirani R.; Friedman J. The elements of statistical learning - data mining, inference, and prediction 2nd ed. New York : Springer, 2009.
- [2] Gron A. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems 2nd ed. Sebastopol : O'Reilly, 2019.
- [3] All You Need to Know about Gradient Boosting Algorithm Part 2. Classification, (7.02.2022), Towards Data Science, <https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-2-classification-d3ed8f56541e>, 22.08.2024.
- [4] Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, str. 785.794.
- [5] Ge, K.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree // Advances in Neural Information Processing Systems, 30(2017), str. 3146.-3154.
- [6] Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: unbiased boosting with categorical features // Advances in Neural Information Processing Systems, 31(2018), str. 6638.-6648.
- [7] Curiel, I.J. Cooperative game theory and applications. Doktorski rad. Nijmegen: Sveučilište Radboud, 1988.
- [8] Higgs Boson Machine Learning Challenge, (12.05.2014.), Kaggle, <https://www.kaggle.com/c/higgs-boson/data>, 25.08.2024.
- [9] Pavlov M, Novak A, Radonic V, Letilovic T, Hadzibegovic I, Jurin I. Body mass index and long-term survival following acute pulmonary embolism: insights from machine learning models. European Heart Journal: Acute Cardiovascular Care. 2024 Apr;13(Supplement\_1):zuae036-092.

- [10] Pavlov M, Barić D, Novak A, Manola Š, Jurin I. From statistical inference to machine learning: a paradigm shift in contemporary cardiovascular pharmacotherapy. *British journal of clinical pharmacology*. 2024 Mar;90(3):691-9.