Klasifikacija gamma čestica generiranih Corsika programom

Ema Dogančić, Anastasija Jezernik, Ana Peterfaj, Maja Tonček

***Sažetak*—U ovom radu rješavamo problem binarne klasifikacije gamma i pozadinskih signala generiranih Cherenkov teleskopom. Dataset je preuzet s UCI repozitorija. Na njemu primjenjujemo neke poznatije algoritme strojnog učenja kao što su metoda potpornih vektora, slučajne šume, metoda najbližih susjeda, naivni Bayesov klasifikator te jedan od novijih algoritama - XGBoost. Koristimo AUC i ROC krivulju kao mjeru uspješnosti modela te za usporedbu s ostatkom modela. Najbolji rezultat daju XGBoost i slučajne šume.**

Ključne riječi—Gamma čestice, binarna klasifikacija, SVM, Random Forest, KNN, Bayes, XGBoost, AUC, ROC

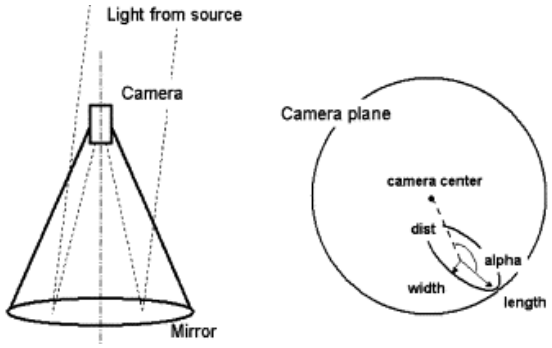
I. Uvod

Astronomija je jedna od opservacijskih znanosti u kojoj je čest problem iz velikog broja podataka prepoznati mali broj zanimljivih događaja koji se suptilno razlikuju od pozadinske buke. U ovom ćemo radu pokušati razraditi problem binarne klasifikacije: na temelju značajki instance signala odrediti je li to gamma signal ili neki signal pozadine.

II. Opis problema i skup podataka

Podaci „*MAGIC Gamma Telescope Data Set*“ su preuzeti sa stranice UCI repozitorija [3].

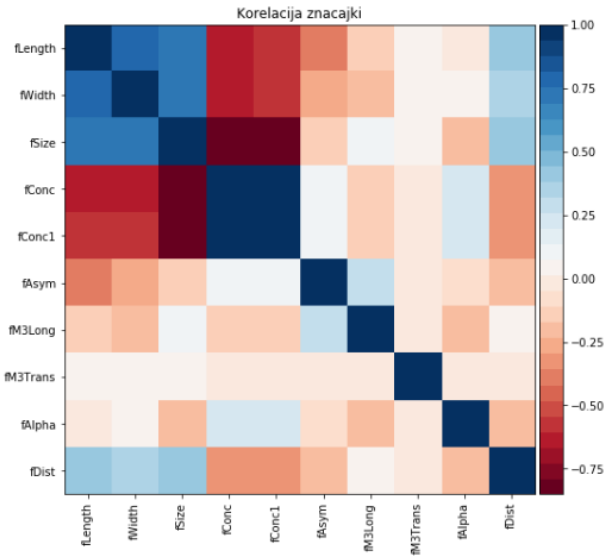
Podaci su generirani Monte Carlo programom Corsika kako bi se simulirala registracija čestica visoke energije, tj. gamma čestica u Cherenkov gamma teleskopu pomoću tehnike snimanja. Cherenkov teleskop promatra visoko energetske zrake koje se razvijaju u atmosferi te su inicirane gamma česticama. To zračenje propušta kroz atmosferu te se registrira u detektoru, čime se omogućava rekonstrukcija parametara zraka. Dostupne informacije se sastoje od impulsa koje su fotoni ostavili na kameri. Ti fotoni se skupljaju u uzorke koje nazivamo slika zračenja. Signali su grupirani u obliku elipse, pa značajke instance signala u datasetu zapravo predstavljaju neka dimenzionalna svojstva signala u odnosu na tu elipsu (Slika 1).



Slika 1. Cherenkov teleskop: neki parametri slike

Svaka instanca signala je određena sa 10 značajki, te značajkom koja određuje pripadnost klasi. Sve značajke (osim klase) su numeričke i neprekidne. Dataset se sastoji od 19020 instanci, od kojih 12332 pripadaju gamma klasi ('g'), a 6688 pripadaju klasi pozadinskih signala ('h'). Odnos tih klasa je otprilike 65%-35%. Nema vrijednosti koje nedostaju.

Slika 2. predstavlja matricu korelacije značajki. Možemo primijetiti da su značajke fConc i fConc1 snažno pozitivno korelirane, što je i razumno za očekivati jer je fConc omjer zbroja dva najviša piksela i značajke fSize, dok je fConc1 omjer najvišeg piksela i značajke fSize. Isto tako, vidi se da je fSize snažno negativno korelirana s fConc i fConf1, što također ima smisla zbog već navedenih definicija značajki fConc i fConc1.



Slika 2. Matrica korelacije značajki

III. Prethodna istraživanja

Objavljena su barem tri znanstvena rada koja su na navedenom skupu podataka provodila algoritme strojnog učenja. U radovima koje smo mi proučile omjer skupa za treniranje i testiranje je 2:1, pa je i naša raspodjela bila približno jednaka tome. Kratki opisi i glavni zaključci radova navedeni su u sljedećim odlomcima.

A. *Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope ([8])*

U ovom istraživanju primjenjivale su se razne metode: klasifikacijska stabla (C5.0, CART, slučajne šume), kernel metode, metoda najbližih susjeda, umjetne neuronske mreže (pet različitih metoda), composite probabilities, direct selection, linearna diskriminantna analiza te metoda potpornih

vektora. Koristilo se šest različitih mjera uspješnosti, koje su sve na neki način opisivale odnos false positive i true positive stopa, premda nijedna od njih nije bila ROC krivulja, koju smo mi na kraju odabrale za našu mjeru. Primjerice, jedna mjera je srednja vrijednost true positive stopa, ili u znanstvenom radu ε_γ, dobivenih interpoliranjem ROC krivulje u vrijednostima 0.01, 0.02 i 0.05 false positive stope, ili ε_p. Kao najbolji algoritam pokazale su se slučajne šume, koje su u pet od šest mjera uspješnosti ostvarile najbolje rezultate.

B. *Experminetal study of leaf confidences for random forests ([9])*

Ovdje se primjenjuje metoda slučajnih šuma ograničenih veličina sa težinama temeljenim na leaf levels of confidence. Rezultati su dani za šume s rangom veličina 20-80 i pokazalo se da ova metoda daje bolje rezultate u odnosu na slučajne šume bez težina, no da se poboljšanje smanjuje kako se veličina šuma povećava. Mjera uspješnosti bila je neprekidna greška (continuous error) koja je koristila jednu posebnu funkciju za rad sa stablima.

C. *Softening Splits in Decision Trees Using Simulated Annealing ([10])*

U ovom radu proučava se metoda klasifikacijskih stabala (C4.5) s "mekim" splitovima i načinu treniranja podataka koje koristi simulirano kaljenje. Metoda je uspoređivana s C5.0 i CART metodama klasifikacijskih stabala. I ovdje se koristilo šest nestandardnih mjera uspješnosti. Pokazalo se da ova metoda ima sličnu točnost kao i C5.0 s omekšanim stablima, no da daje puno bolje rezultate u usporedbi s CART metodom s neomekšanim stablima.

IV. PRISTUP PROBLEMU

Budući da se radi o problemu binarne klasifikacije, koristit ćemo sljedeće algoritme strojnog učenja: metoda potpornih vektora, slučajne šume, metoda najbližih susjeda, naivni Bayes i XGBoost. Dataset dijelimo na train i test skup u omjeru 70-30. Kao mjeru uspješnosti modela ćemo koristiti ROC krivulju (varijacija te mjere je korištena i u dosadašnjim istraživanjima).

Kako je broj značajki malen (10), nećemo smanjivati dimenzionalnost. Već smo prije spomenuli da su sve značajke neprekidne, pa ih nećemo ni mijenjati.

Napomenimo još da je razlog zašto koristimo upravo ROC krivulju taj što se smatra da je klasificiranje pozadinskog signala kao gamma signala (false positive) gore od klasificiranja gamma signala kao pozadinskog (false negative).

A. *XGBoost*

XGBoost je algoritam strojnog učenja koji se temelji na ansamblu stabla odluke te optimiziranju gradient boosting algoritma koristeći paralelno procesiranje, „obrezivanje stabla“, rješavanje problema nedostajućih podataka te ugrađenu regularizaciju potrebnu za izbjegavanje overfittanja.

Budući da naš dataset nema nedostajećih podataka, fokusirali smo se na odabir parametara samog modela. Algoritam ima ugrađenu cross-validation metodu u svakoj iteraciji koju smo koristili za podešavanje parametara.

U sljedećem odjeljku ćemo samo nabrojiti parametre koje smo podešavali uz kratko obrazloženje zašto smo ih odabrali. Detaljniji opis svih parametara se može naći u [1].

Parametri koje ne podešavamo, nego sami zadajemo su:

- booster = gbtree (verzija modela koji koristi stabla; postoji i linearna verzija)
- nthread = 4 (koliko dretvi se koristi za paralelno procesiranje)
- objective = binary:logistic (jer je u pitanju problem klasifikacije)
- val_metric = auc (metrika koja se koristi za validacijski dataset)
- scale_pos_weight = 1 (zbog nebalansiranosti dataseta)

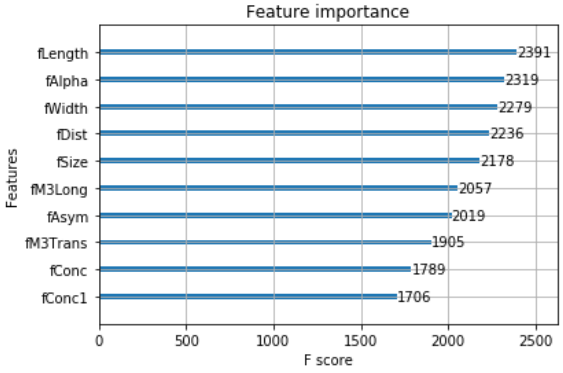
Parametri koje podešavamo (koristeći grid search i primjer sa stranice [2]):

- min_child_weight i max_depth – rješavaju problem overfittanja i underfittanje,
- subsample i colsample_bytree – rješavaju problem overfittanja
- gamma – parametar koji kontrolira kompleksnost modela,
- lambda i alpha – parametri regularizacije.

Krenuli smo s početnim modelom gdje smo pogađali parametre:

```
XGBClassifier(
        learning_rate =0.1,
        n_estimators=1000,
        max_depth=5,
        min_child_weight=1,
        gamma=0,
        subsample=0.8,
        colsample_bytree=0.8,
        objective= 'binary:logistic',
        nthread=4,
        scale_pos_weight=1,
        seed=27)
```

Graf *feature importance* početnog modela možemo vidjeti na Slici 3.

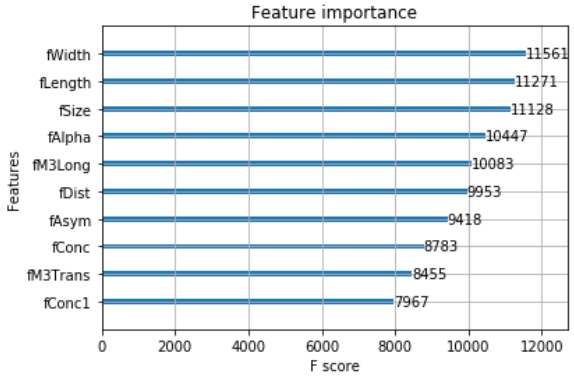


Slika 3. Feature Importance graf početnog XGBoost modela

Nakon podešavanja svih parametara, dobivamo sljedeći model:

```
XGBClassifier(
        learning_rate =0.01,
        n_estimators=5000,
        max_depth=7,
        min_child_weight=1,
        gamma=0,
        subsample=0.9,
        colsample_bytree=0.6,
        reg_alpha = 1.5e-05,
        reg_lambda = 1,
        objective= 'binary:logistic',
        nthread=4,
        scale_pos_weight=1,
        seed=27)
```

Graf *feature importance* konačnog modela možemo vidjeti na Slici 4.

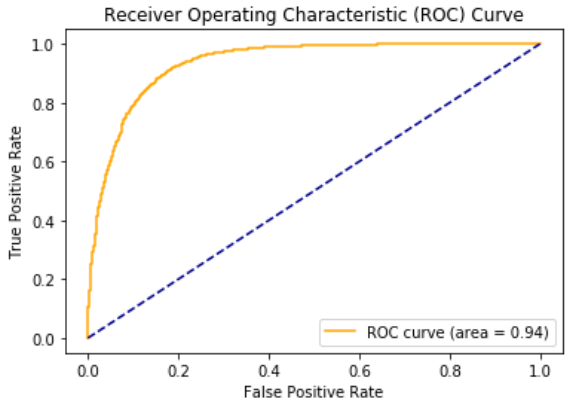


Slika 4. Feature Importance graf konačnog XGBoost modela

Možemo primijetiti da se redoslijed najbitnijih značajki malo izmijenio, ali nije došlo do ekstremnih promjena.

Podešavanjem parametara je vrijednost AUC skočila s 0.933 na 0.936, što i nije velika razlika, ali to je i za očekivati, budući da je već početna AUC vrijednost poprilično velika.

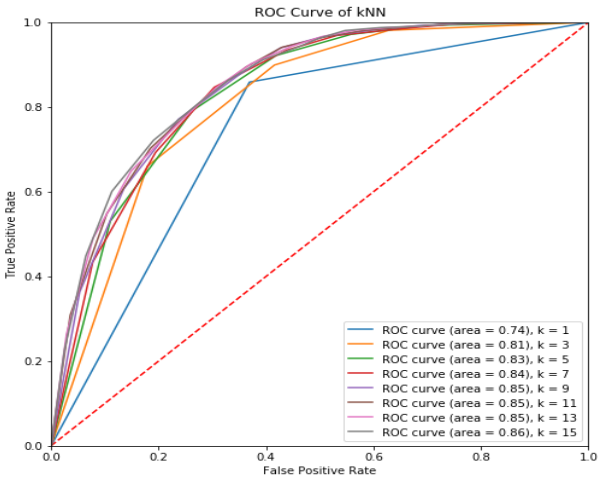
Na Slici 5. je prikazana ROC krivulja konačnog XGBoost modela.



Slika 5. ROC krivulja XGBoost modela

B. Metoda najbližih susjeda (k-nn)

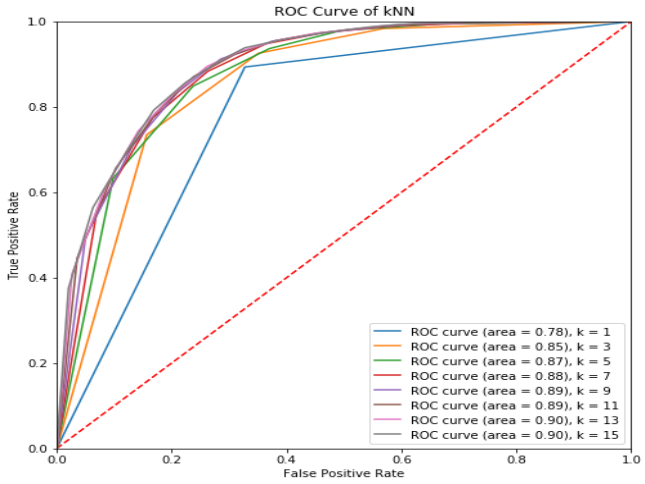
Kod metode najbližih susjeda radimo podešavanje parametra *k*, odnosno varijable `n_neighbors` klasifikatora `KNeighborsClassifier`. S obzirom da imamo dvije klase koristimo neparne vrijednosti za *k* kako ne bismo dobili neriješen slučaj. Prvo unutar for-petlje za *k* između 1 i 15 uključivo radimo odabir modela. Za to smo dataset podijelili u 3 skupa: skup za treniranje (50%), validacijski skup (20%) i testni skup (30%). Modele treniramo na skupu za treniranje, zatim odabiremo najbolji model ovisno o tome kakvu ROC krivulju i AUC vrijednost dobivamo na validacijskom skupu i njega onda testiramo na testnom skupu. Tu smo najbolju vrijednost AUC-a dobili za *k* = 15 i iznosi 0.86 (Slika 6).



Slika 6. ROC krivulje za nenormalizirane podatke

S obzirom da k-nn uglavnom radi bolje na normaliziranim podacima, uz pomoć `StandardScaler`-a normaliziramo podatke i dobivamo da je ponovno vrijednost *k* = 15, ali ovaj put AUC iznosi 0.9 (Slika 7). Nakon ovoga radimo još i odabir modela pomoću funkcije `GridSearchCV` kako bismo pokušali još malo povećati AUC. Ponovno podešavamo parametar *k*, odnosno `n_neighbors`, ali ovaj put u obzir uzimamo i druge parametre modela k-nn, a to su `weights` i `p`, gdje je `p` vrijednost potencije u Minkowski mjeri. Najbolju vrijednost AUC za *k*-nn algoritam dobivamo 0.912 za sljedeći model:

```
KNeighborsClassifier(
        algorithm='auto',
        leaf_size=30,
        metric='minkowski',  metric_params=None,
        n_jobs=None,
        n_neighbors=35,
        p=1,
        weights='distance')
```



Slika 7. ROC krivulje za normalizirane podatke

C. *Slučajne šume*

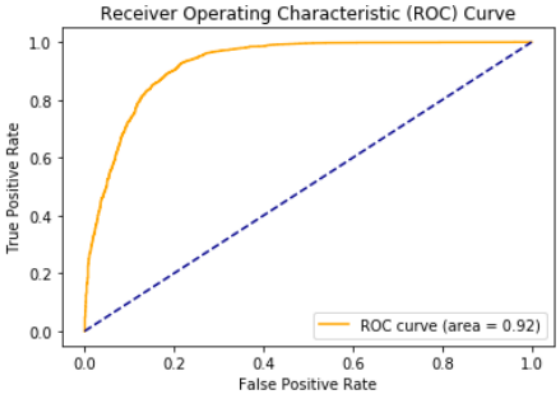
S obzirom da je XGBoost također ansambl temeljen na stablima odlučivanja kao i slučajne šume, zanimalo nas je koliko će slična biti dobivena AUC vrijednost. Radili smo grid search za podešavanje nekoliko parametara i najbolja vrijednost AUC koju dobivamo je ispada jako blizu one u XGBoost-u, a to je 0.937 za dubinu od 32 te broj stabala u šumi 600.

D. *Metoda potpornih vektora*

Metoda potpornih vektora (SVM) je algoritam strojnog učenja koji nalazi hiper-ravninu koja ima najveću marginu razdvajanja. Koristili smo RBF kernel prema naputcima Andrewa Nga na Coursera Machine Learning tečaju.

Početni model s automatskim parametrima `C` i `gamma` daje nam vrijednost AUC `= 0.754.` Pomoću `GridSearchCV` s 3-fold cross validation pretražujemo parametre na skupu `(1.0, 10.0, 100.0, 1000.0)` za `C` i `(0.001, 0.01, 0.1, 1.0)` za `gamma` i dobivamo najbolji rezultat za `C = 1.0` i `gamma = 0.001,` time smo AUC povećali na 0.873.

Nakon normalizacije podataka uz pomoć `StandardScaler`-a, ponovno pokrećemo grid search i ovaj put dobivamo najbolji rezultat za `C = 100.0` i `gamma = 0.1.` Za te vrijednosti parametara na normaliziranim podacima, dobili smo `AUC = 0.923`.



Slika 8. ROC krivulja za metodu potpornih vektora

E. *Naivni Bayes*

Naivni Bayes je algoritam nadziranog učenja koji se temelji na Bayesovom teoremu s „naivnom“ pretpostavkom (u stvarnom svijetu) o uvjetnoj nezavisnosti između svakog para značajki uz danu vrijednost klase. Za naivnog Bayesa vrijedi da može pristojno klasificirati instance, ali da loše procjenjuje vjerojatnost pripadnosti nekoj klasi ([11]), a kako su za računanje AUC potrebne te vjerojatnosti, te zbog prisutnosti visoko koreliranih značajki, nismo očekivale jako dobar rezultat od ove metode.

Postoje različite varijante naivnog Bayesa, a mi smo koristile Gaussovu. Primjenjujući ju na sve značajke, dobile smo rezultat AUC = 0.76. Pogledavši važnost značajki, testirale smo metodu kombinirajući značajnije značajke te dobile da naivni Bayes postiže najveći AUC (0.81) uz samo dvije značajke, fAlpha i fWidth. Ispod je dana matrica konfuzije.

	gamma	**hadron**
gamma	929	1072
hadron	328	3377

V. Zaključak

Rezultati su prikazani sljedećom tablicom:

	XGBoost	**k-nn**	**RF**	**SVM**	**NB**
AUC	0.936	0.912	0.937	0.923	0.814

Vidimo da slučajne šume daju najbolji rezultat, ali vrlo blizu je i XGBoost, razlika je samo 0.1%. Dobre rezultate daju metoda potpornih vektora, kao i k-nn. Kao što smo i očekivale, najgori rezultat smo dobile koristeći naivni Bayesov model, budući da model pretpostavlja nezavisnost podataka, dok mi imamo blagu zavisnost.

VI. Mogući daljni rad

Budući da smo radile na originalnom skupu podataka, bez ikakvih izmjena, u daljnjem radu bi se moglo ispitati kakve rezultate bi dali algoritmi strojnog učenja na transformiranim podacima (primjerice logaritamska transformacija), također dodavanje novih značajki kao i odabir značajki bi moglo poboljšati rezultate.

Također, uz već korištene evaluacijske metode, u daljnjem radu bi se mogle koristiti i neke druge metode, primjerice bootstrap.

References

[1] https://xgboost.readthedocs.io/en/latest/parameter.html

[2] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[3] https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope

[4] https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-algorithm-using-python-and-sci

[5] https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

[6] https://www.ritchieng.com/machine-learning-efficiently-search-tuning-param/

[7] https://stackoverflow.com/questions/51459406/apply-standardscaler-in-pipeline-in-scikit-learn-sklearn

[8] https://www.sciencedirect.com/science/article/pii/S0168900203025051

[9] http://uivty.cs.cas.cz/~savicky/papers/softening.pdf

[10] https://www.researchgate.net/publication/303539515_Experimental_study_of_leaf_confidences_for_random_forest

[11] https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes