Identifikacija miona u CMS detektoru pri proton-proton sudarima energije \(\sqrt{s}=13\) TeV metodama strojnog učenja

Šarić, Toni

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, University of Split, Faculty of science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:806797

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET

Diplomski rad

Identifikacija miona u CMS detektoru pri proton-proton sudarima energije $\sqrt{s} = 13\ \mathrm{TeV}$ metodama strojnog učenja

Toni Šarić

Split, srpanj 2020.

Zahvale

Zahvaljujem svojim mentorima doc. dr. sc. Marku Kovaču i doc. dr. sc. Hrvoju Kaliniću na nesebičnoj pomoći, prenesenom znanju i konstruktivnim kritikama pri izradi ovog diplomskog rada i što su uvijek imali strpljenja za moje mnogobrojne upite.

Zahvaljujem i svim profesorima i asistentima na suradnji i prenesenom znanju.

Zahvaljujem svim prijateljima, kolegama i cimeru koji su me pratili tijekom cijelog mog studiranja i učinili ga zanimljivim, zabavnim i nezaboravnim.

Posebnu zahvalnost dugujem svojoj obitelji (roditeljima, bratu i sestri) koja je uvijek uz mene, koja me podržava i ohrabruje na putu prema cilju. Od srca zahvaljujem i svojoj djevojci na pruženoj ljubavi, potpori i razumijevanju. Sve ovo, bez njih ne bi bilo moguće.

Temeljna dokumentacijska kartica

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za fiziku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Diplomski rad

IDENTIFIKACIJA MIONA U CMS DETEKTORU PRI PROTON-PROTON SUDARIMA ENERGIJE $\sqrt{s} = 13$ TEV METODAMA STROJNOG UČENJA

Toni Šarić

Sveučilišni diplomski studij Fizika, smjer Računarska fizika

Sažetak:

U ovom diplomskom radu napravljena je identifikacija objekata dobivenih simulacijom Drell-Yan procesa. Rekonstruirani objekti podijeljeni su u dvije klase: u klasu signalnih i u klasu pozadinskih objekata koji su zatim spremljeni u skup podataka. Signalni objekti su mioni najvjerojatnije nastali u primarnom verteksu, dok su pozadina objekti pogrešno rekonstruirani kao mioni. Analize podataka na CERN-u nerijetko koriste takozvane *cut based ID* metode za identifikaciju objekata, stoga ovaj rad razmatra mogućnost zamjene tog pristupa pristupom strojnog učenja. Strojno učenje uči model razlikovati objekte treniranjem na već klasificiranim podacima da bi bio sposoban identificirati signal u eksperimentalnim podacima prikupljenim radom CMS detektora. Želeći imitirati ovaj proces, kreirani skup podataka dijeli se na skup podatka za treniranje i na skup za testiranje. Glavnu prepreku u ostvarenju cilja predstavlja neuravnoteženost klasa jer je broj pozadinskih objekata uvelike, približno 14 puta, manji od signalnih. Obrađena su dva pristupa problemu neuravnoteženosti podataka. Prvi pristup je težinsko treniranje koje mijenja težine funkciji pogreške. Drugi pristup je balansiranje klasa dodavanjem ili izbacivanjem objekata skupa podataka za treniranje.

Ključne riječi:	CERN, CMS, identifikacija, rekonstrukcija, Drell-Yan, strojno učenje, klasifikacija, balansiranje
Rad sadrži:	69 stranica, 29 grafičkih prikaza, 10 tablica, 19 literaturnih navoda i 2 koda. Izvornik je na hrvatskom jeziku.
Mentor:	doc. dr. sc. Marko Kovač
Komentor:	doc. dr. sc. Hrvoje Kalinić
Ocjenjivači:	doc. dr. sc. Marko Kovač, doc. dr. sc. Hrvoje Kalinić, prof. dr. sc. Nikola Godinović
Rad prihvaćen:	24.6.2020.

Rad je pohranjen u Knjižnici Prirodoslovno – matematičkog fakulteta, Sveučilišta u Splitu.

Basic documentation card

University of Split
Faculty of Science
Department of Physics
Ruđera Boškovića 33, 21000 Split, Hrvatska

Master thesis

IDENTIFICATION OF MUONS IN THE CMS DETECTOR IN PROTON-PROTON COLLISIONS AT $\sqrt{s} = 13$ TEV USING MACHINE LEARNING METHODS

Toni Šarić

University graduate study programme Physics, orientation Computational Physics

Abstract:

In this thesis, the identification of objects obtained by simulation of the Drell-Yan process is made. The reconstructed objects were divided into two classes: the signal object class and the background object class, which were then stored in a data set. Signal objects are muons most likely formed in the primary vertex, while background objects are inaccurately reconstructed as muons. Data analysis at CERN often use so-called *cut-based ID* methods to identify objects, therefore this paper considers the possibility of replacing this approach with a machine learning strategy. Machine learning teaches the model to distinguish objects by training on already classified data to be able to identify the signal in the experimental data collected by the operation of the CMS detector. Wanting to mimic this process, the created data set is divided into a training data set and a test data set. In achieving the goal, the main impediment were the imbalance of classes which means the number of background objects is much, approximately 14 times, less than the signal. Two approaches to the problem of data imbalance are discussed. The first approach is weight balancing which changes the weights of the error function. The second approach is to balance classes by adding or removing objects from a training data set.

Keywords: CERN, CMS, identification, reconstruction, Drell-Yan, machine learning, classification, balancing

Thesis consists of: 69 pages, 29 figures, 10 tables, 19 references and 2 codes.
Original language: Croatian

Mentor: Assist. Prof. Dr. Marko Kovač

Co-mentor: Assist. Prof. Dr. Hrvoje Kalinić

Reviewers: Assist. Prof. Dr. Marko Kovač,
Assist. Prof. Dr. Hrvoje Kalinić,
Prof. Dr. Nikola Godinović

Thesis accepted: June, 24, 2020.

Thesis is deposited in the library of the Faculty of Science, University of Split.

Sažetak

U ovom diplomskom radu napravljena je identifikacija objekata dobivenih simulacijom Drell-Yan procesa. Rekonstruirani objekti podijeljeni su u dvije klase: u klasu signalnih i u klasu pozadinskih objekata koji su zatim spremljeni u skup podataka. Signalni objekti su mioni najvjerojatnije nastali u primarnom verteksu, dok su pozadina objekti pogrešno rekonstruirani kao mioni. Analize podataka na CERN-u nerijetko koriste takozvane *cut based ID* metode za identifikaciju objekata, stoga ovaj rad razmatra mogućnost zamjene tog pristupa pristupom strojnog učenja. Strojno učenje uči model razlikovati objekte treniranjem na već klasificiranim podacima da bi bio sposoban identificirati signal u eksperimentalnim podacima prikupljenim radom CMS detektora. Želeći imitirati ovaj proces, kreirani skup podataka dijeli se na skup podatka za treniranje i na skup za testiranje. Glavnu prepreku u ostvarenju cilja predstavlja neuravnoteženost klasa jer je broj pozadinskih objekata uvelike, približno 14 puta, manji od signalnih. Obrađena su dva pristupa problemu neuravnoteženosti podataka. Prvi pristup je težinsko treniranje koje mijenja težine funkciji pogreške. Drugi pristup je balansiranje klasa dodavanjem ili izbacivanjem objekata skupa podataka za treniranje.

Ključne riječi: CERN, CMS, identifikacija, rekonstrukcija, Drell-Yan, strojno učenje, klasifikacija, balansiranje

Sadržaj

Popis slika

Popis tablica

Uvod

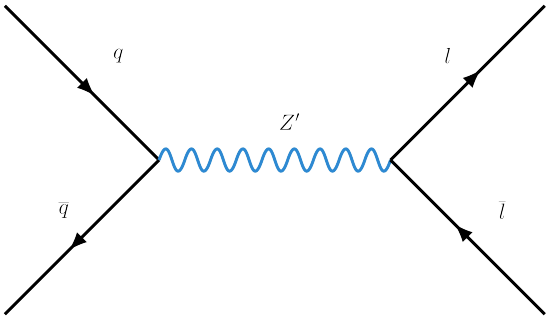
Veliki hadronski sudarivač (eng. Large Hadron Collider - LHC) generira, sa svojih milijardu proton-proton sudara u sekundi, nevjerojatnih 25 GB podataka svake sekunde odnosno na stotine petabajta godišnje. Svakih 25 nanosekundi detektor zabilježi u prosjeku 1000 čestica. Pred tim enormnim brojem događaja dugačak je put *online* i *offline* postupka odabira događaja. *Online* postupak odabira događaja vrlo je sofisticirana *real-time* filtracija podataka, koja u djeliću sekunde odlučuje hoće li događaj biti odbačen ili sačuvan za daljnju analizu, što u konačnici svodi na oko 100 događaja po sekundi ([1]). *Offline* postupak odabira događaja, za razliku od *online*, nije ograničen vremenom i u principu se može podijeliti na dva dijela:

- Odabir događaja (eng. event selection) i rekonstrukciju objekata. Ovo će biti opisano u poglavlju 3.
- Postupak klasifikacije rekonstruiranih objekata će biti opisan u poglavlju 4.

Klasifikacija objekata vjerojatno je najvažniji zadatak fizike elementarnih čestica na kojeg je do sad primijenjeno strojno učenje. Compact Muon Solenoid (CMS) sve više se oslanja na tehnike strojnog učenja za identifikaciju objekata i klasifikaciju događaja kako bi se smanjili mnogobrojni pozadinski procesi koji kontaminiraju signale od interesa.

U ovom radu analiziran je Drell-Yan proces koji opisuje nastanak električki suprotno nabijenog, masivnog leptonskog para u hadron-hadron sudaru putem *s* kanala izmjenom odgovarajućeg baždarnog bozona kao na slici 1. Zašto Drell-Yan i zašto baš mioni? Poopćenjem definicije Drell-Yan procesa otvaraju se vrata istraživanju nekih novih, teških, nepoznatih (van Standardnog Modela) bezbojnih čestica - posrednih bozonskih stanja pri nastanku leptonskog para. Drugim riječima, osim već poznatih baždarnih bozona $W^{\pm}$, Z^0 i H^0 posrednik također može biti bilo koja druga, do sad nepoznata, bezbojna čestica. Također, Drell-Yan

proces osigurava relativno čisto konačno stanje s dva miona. Mioni, zbog svoje približno 200 puta veće mase od elektrona, manje su podložni radioaktivnom gubitku energije u materijalu detektora, dublje mogu prodrijeti pa ih možemo bolje mjeriti.



Slika 1: Feynmanov dijagram Drell-Yan procesa.

Mioni koji dolaze iz primarnog verteksa su signalni mioni (eng. prompt muons), dok pozadinom, između ostalog, smatramo mione nastale nekim drugim procesima poput raspada teških mezona ili pogreškom u rekonstrukciji gdje su mlazovi čestica rekonstruirani kao mioni. Kako razlikovati te procese? Kako odrediti koji je signalni, a koji pogrešno identificirani mion kad u konačnici, nakon sudara, imamo samo rekonstruirane objekte s pridruženim im značajkama poput pseudorapiditeta, količine gibanja i drugih? Napretek analiza podataka koje se rade na CERN-u tijekom koraka identifikacije čestica koristi takozvani *cut based ID* gdje se identifikacija primjerice miona temelji na skupu rezova na različite varijable. Ovaj rad istraživa mogućnost zamjene takvog pristupa, odnosno *cut based ID* pristupa pristupom koji se temelji na strojnom učenju. Simulacije se koriste da bi se vidjelo kako bi interesantan signal izgledao u detektoru. Samo u simuliranim podacima znamo koji su signalni, a koji pozadinski objekti i te podatke koristimo za trening odnosno za izgradnju modela. Zatim ćemo taj model primijeniti na eksperimentalne podatke gdje ne možemo znati što je signal, a što pozadina, a model će nam pokušati dati što točniju/precizniju klasifikaciju objekata. Pitanja na koja ćemo se usredotočiti kroz dio rada vezan uz strojno učenje, između ostalih, su: (i) kako izgraditi model za klasifikaciju objekata, zatim (ii) kako ga evaluirati i (iii) kako usporediti više različitih modela. Analizirati ćemo i usporediti rezultate raznih metoda nadziranog strojnog učenja. Fokus će biti na klasifikaciji, točnije na analizi neuravnoteženog skupa podataka u kojem je jedan tip događaja višestruko brojniji od drugog. U problemima ovakvog tipa, većinska klasa označava uobičajene, a manjinska neuobičajene događaje (anomalije), poput pogrešaka, prevara ili pozadinskih objekata pri proton-proton sudarima. Zbog čega naglašavamo kakav je skup podataka? Takav skup za sobom povlači potencijalne probleme, kojima je potrebno pristupiti na vrijeme i na ispravan način. Glavni problem je pristranost modela/klasifikatora prema većinskoj klasi

iz razloga što je upravo na takvom, nebalansiranom skupu, model izgrađen. Da bi se uspješno odradila analiza i evaluacija modela, potrebno je uvesti osnove statističke analize, ali i njene metode primjerenije neuravnoteženom skupu podataka s kojim smo suočeni. Dva pristupa su odrađena. Prvi pristup je težinsko treniranje u kojem mijenjamo težine funkciji pogreške, tako da pogreške različitih klasa različito pridonose ukupnim pogreškama (imaju različit težinski faktor) - pri čemu neki algoritmi imaju ugrađene parametre za to, a drugi ne. Drugi pristup je balansiranje klasa gdje ćemo mijenjati ulazni skup podataka.

Poglavlje 1

Standardni model

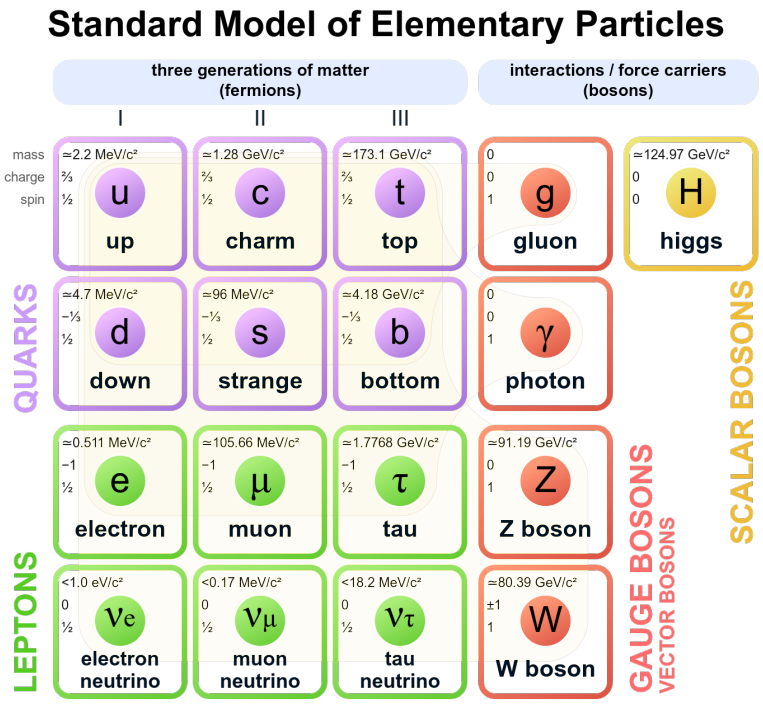
Slijedi kratak pregled Standardnog modela fizike elementarnih čestica što uključuje opis fundamentalnih čestica kao i fundamentalnih međudjelovanja. Postoje četiri temeljna međudjelovanja: elektromagnetsko, jako, slabo i na posljetku gravitacijsko koje se zbog svog vrlo slabog intenziteta u fizici elementarnih čestica može zanemariti. Opisan je Drell-Yan proces čijom Monte Carlo (MC) simulacijom je dobiven trening skup podataka. Naglasak je stavljen na jako međudjelovanje, gluone i kvarkove jer su glavni konstituenti spomenutog procesa.

1.1 Fundamentalne čestice

U Standardnom modelu, sva poznata materija sačinjena je od tri vrste elementarnih čestica: kvarkova, leptona i posrednika sile baždarnih bozona. Fundamentalne čestice mogu biti ili građevni blokovi materije pa se nazivaju fermioni ili posrednici međudjelovanja u kojem slučaju se zovu bozoni. Čak su i sile među česticama opisane izmjenom čestica.

Fizika nastoji odrediti strukturu materije i kako njene fundamentalne sastavnice međudjeluju. Energija pri kojoj se promatra fizikalni sustav diktira koliko kompleksan model je potrebno izgraditi. Promotrimo li svijet oko nas, u našem svakidašnjem životu, čini se kao da je sastavljen od svega nekoliko čestica. Atom je vezano stanje električki negativno nabijenog elektrona, pozitivnog protona i neutralnog neutrona. Elektroni su, zbog suprotnog električnog naboja od jezgre, elektrostatskom silom vezani za nju. Unutar same jezgre, protoni i neutroni su vezani jakom nuklearnom silom. U fundamentalna međudjelovanja još ubrajamo slabu silu koja je odgovorna za nuklearni β raspad radioaktivnih izotopa i

nuklearnu fuziju. Pri oba procesa nastaje praktički bezmasena čestica - elektronski neutrino (ν_{e}). Većina fizikalnih fenomena, naročito pri svakidašnjim energijama, može se opisati pomoću netom nabrojanih čestica: protona, elektrona, neutrona i elektronskog neutrina koje međudjeluju elektromagnetskom, slabom i jakom silom. Toj podjeli još pripada gravitacija, iako slaba i uvijek privlačna, odgovorna je za velike strukture u svemiru. Da sumiramo, sa samo četiri čestice i četiri sile može se kreirati veoma jednostavan fizikalni model. Međutim, pri višim energijama, otkriva se detaljnija struktura. Protoni i neutroni su zapravo vezana stanja manjih, za sad se smatra, fundamentalnih čestica - kvarkova. Proton, na primjer, se sastoji od dva up-kvarka (u) i jednog down (d) - uud. Elektron, elektronski neutrino, up i down kvark skupno se nazivaju *prvom generacijom* čestica i s trenutačnim znanjem smatraju se elementarnim. Doduše, promatraju li se međudjelovanja u visoko-energetskim sudarima, daljnja kompleksnost se otkriva. Za svaku od četiri čestice prve generacije, postoje dodatne dvije kopije koje se razlikuju samo po masi. Uzmimo primjer miona koji je zapravo masivnija verzija elektrona s masom $\mathrm{m}_{\mu} \approx 3500\ \mathrm{m}_{e}$. Tih dodatnih osam čestica zovu se *drugom* i *trećom* generacijom. Šest različitih tipova kvarkova se nazivaju okusi (eng. flavors) - up (u), down (d), charm (c), strange (s), top (t), bottom (b). Također postoji šest odgovarajućih antikvarkova - koji se od svojih čestica razlikuju samo po električnom naboju. Slično, postoji šest leptona, neke, poput elektrona (e) i elektronskog neutrina (ν_e) smo već spomenuli, preostaje još druga i treća generacija leptona, a to su: mion (μ) i mionski neutrino (ν_μ), tau (τ) i tau neutrino (ν_τ) i također šest odgovarajućih antileptona. Na slici 1.1 po stupcima su prikazane generacije fermiona. Napomena: fermioni su čestice polucjelobrojnog spina, a bozoni cjelobrojnog.



Slika 1.1: Standardni model elementarnih čestica: 12 fundamentalnih fermiona i 5 fundamentalnih bozona. Izvor: [2]

1.2 Fundamentalna međudjelovanja

Međudjelovanja među česticama određena su preko četiri fundamentalne sile: gravitacijska, elektromagnetska, jaka i slaba. U pogledu fizike elementarnih čestica, gravitacija se zbog svoje ekstremno male jakosti može zanemariti. Svojstva dvanaest fundamentalnih fermiona su kategorizirana po tipu međudjelovanja koje mogu osjetiti, kao što je prikazano u tablici 1.1. Sve čestice osjećaju slabo međudjelovanje jer imaju okus (eng. flavour charge), neutrini ne osjećaju elektromagnetsko jer nemaju električnog naboja dok u jakom međudjelovanju sudjeluju samo kvarkovi jer imaju boju (eng. color charge). Vidjet ćemo malo kasnije kakvo ograničenje boja nameće.

				Jaka	Elekromagnetska	Slaba
Kvarkovi	u	c	t	✓	✓	✓
	d	s	b			
Leptoni	e^-	μ^-	τ^-		✓	✓
	ν_e	ν_μ	ν_τ			✓

Tablica 1.1: Sile koje osjećaju različite čestice.

U fizici elementarnih čestica, sve sile opisane su kvantnom teorijom polja (eng. Quantum Field Theory - QFT) i to izmjenom odgovarajuće spin-1 čestice, znane kao baždarni bozon (eng. gauge boson). U kvantnoj elektrodinamici (eng. Quantum Electrodynamics - QED) baždarni bozon je poznati spin-1 foton. Kad pričamo o jakom međudjelovanju u kvantnoj kromodinamici (eng. Quantum Chromodynamics - QCD) silu prenosi gluon, koji je poput fotona bezmasen. Slaba međudjelovanja posredovana su nabijenim $\mathbf{W}^{\pm}$ bozonima, koji su otprilike osam puta masivniji od protona te neutralnim **Z** bozonom.

Standardni Model matematički je opisan pomoću teorija grupa. Lagrangian Standardnog modela je invarijantan na grupu simetrije $SU(3) \otimes SU(2) \otimes U(1)$. $SU(n)$ je kolekcija svih unitarnih $n \times n$ matrica determinante 1. $U(1)$ opisuje QED, $SU(2) \otimes U(1)$ elektroslabo međudjelovanje, a $SU(3)$ QCD. Higgsov mehanizam razbija elektroslabu simetriju, ali osigurava mehanizam po kojem sve druge čestice dobivaju masu i opisuje postojanje masivnih vektorskih bozona (W i Z).

QCD je teorija jakih međudjelovanja između kvarkova i gluona, zajedničkog imena partona. Koncept boje (eng. color) razriješio je dilemu inkonzistentnosti Paulijevog principa isključenja na trima identičnim kvarkovima u istom stanju u barionu. Princip isključenja kaže da identični kvarkovi ne mogu biti u istom kvantnom stanju kao na primjer u Δ^{++} barionu koji je sastavljen od tri up-kvarka. Dva pozitivna znaka označavaju dvije jedinice pozitivnog električnog naboja, jedan up-kvark nosi naboj +2/3 slijedom čega 2/3 + 2/3 + 2/3 = 6/3 $\equiv$ 2. Kvark može imati jednu od tri boje (*red*, *green*, *blue*), što zapravo nema veze s bojom kakvu poznajemo, dakle sa valnom duljinom elektromagnetskog zračenja od približno 400 do 700 nm. *Color charge* također ima korisno svojstvo, a to je da su sve slobodne čestice u *color singlet* stanju - bezbojne su. Partoni unutar hadrona posjeduju dvije osnovne karakteristike: asimptotska sloboda i zatočenje (eng. confinement). Kad su partoni unutar hadrona blizu (veća energija), jaka sila slabi, ponašaju se kao slobodne čestice - asimptotska sloboda. Suprotno, kad se partoni udaljavaju jaka sila se povećava, pa su partoni zatočeni unutar hadrona. Sve obojane čestice, poput kvarkova i gluona, su uvijek vezane unutar

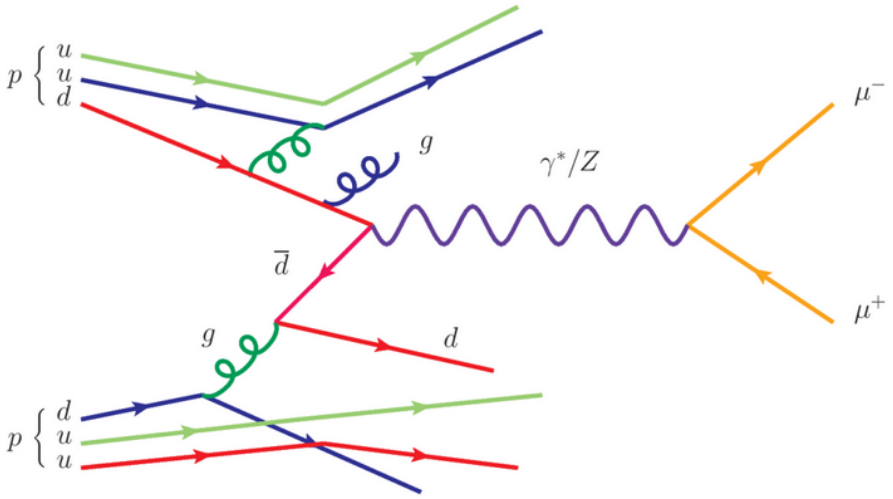
bezbojnog stanja, ne mogu se uočiti kao samostalne čestice. Hadronska stanja koja su do sad uočena su mezoni koji se sastoje od kvarka i antikvarka, barioni sastavljeni od tri kvarka i antibarioni od tri antikvarka.

1.3 Drell-Yan produkcija

Produkcija masivnog leptonskog para preko posrednika Z bozona ili virtualnog fotona γ^*, anhilacijom kvark-antikvark para,

$$q\bar{q} \to Z/\gamma^* \to l^+l^-, \tag{1.3.1}$$

naziva se Drell-Yan proces. Jedan je od *benchmarka* za potvrdu Standardnog modela pri TeV energijskoj skali, a 1970. godine predvidjeli su ga fizičari Drell i Yan [3]. U ovoj analizi, kvark i antikvark nastali su hadron-hadron sudarom, a leptonski par koji ćemo promatrati, u konačnom stanju se sastoji od miona i antimiona $Z/\gamma^* \to \mu^+\mu^-$. Velika prednost je relativno čisto konačno stanje koje uključuje samo raspadnute leptone. Osigurava nam precizno mjerenje funkcije strukture kvarka, parton pljuskova nastalih zbog kvarka, kao i svojstava samog događaja. Skica procesa može se vidjeti na slici 1.2.



Slika 1.2: Feynmanov dijagram Drell-Yan procesa. Izvor: [4]

Kao što ubrzane električki nabijene čestice zrače fotone, tako i ubrzani kvarkovi zrače gluone. Međutim, za razliku od električki nenabijenih fotona, gluoni i sami nose *color charge* pa mogu dalje emitirati radijaciju što dovodi do parton pljuska. Pri niskim energija, barioni sadrže tri valentna kvarka, a mezoni dva (kvark i antikvark), dok pri višim energijama primjećujemo more nevalentnih kvarkova uz dodatak valentnih kvarkova. Rekli smo da su hadroni bezbojni, stoga direktni

dokaz boje ne postoji. Međutim, indirektni dokaz dobiva se mjerenjem udarnog presjeka koje ovisi o slaganju boje kvarka i antikvarka prilikom njihove anhilacije. Ovo slaganje boja potrebno je za Drell-Yan proces u kojem su, u ovom slučaju, posrednici gluoni. Budući da gluoni samointeragiraju, što će se dogoditi pokušamo li razdvojiti dva obojena objekta kao npr. dva gluona ili dva kvarka? Energija QCD polja će se povećati i zbog tog se kvarkovi ne mogu razdvojiti iz hadrona.

Drell-Yan proces također se može promatrati kao QED kvark-antikvark anhilacija u di-leptonski par posredstvom virtualnog fotona. Takvim su ga promatrali Drell i Yan davne 1970. godine. Napredak QCD teorije unaprijedio je Drell-Yan proces s kvark-gluon i gluon-gluon subprocesima. Ako je masa di-leptona velika u usporedbi s masom nukleona, Heisenbergova relacija neodređenosti kaže da je vrijeme međudjelovanja kratko. Prema tome, nema mogućnosti da par kvark-antikvark međudjeluje sa drugim komponentama početnog hadrona. Druge partone se tada gleda kao promatrače. Međutim, ako je vrijeme međudjelovanja dugo u usporedbi sa vremenom anhilacije, promatrači se mogu presložiti u hadrone [5].

Zbog zatočenja boje jake sile, ne postoji detektor koji ima sposobnost promatranja kvarkova i gluona u izoliranom stanju. Kako nam tu pomaže Drell-Yan mehanizam? Osim što omogućava analiziranje vezanih gibanja i prostornih distribucija kvarkova i gluona unutar bezbojnog hadrona, također otvara mogućnost proučavanja nastanka masivnog leptonskog para putem bezbojnih, posrednih bozonskih stanja, poput $W^{\pm}$ i Z^0 bozona kao i Higgsovog bozona H^0. Čak štoviše, ističe Yan, poopćenjem definicije Drell-Yan procesa, on potencijalno postaje kanal otkrivanja nekih novih, teških, nepoznatih (van Standardnog modela) bezbojnih (eng. color neutral) čestica. Zamjenom virtualnog fotona s teškim baždarnim bozonom slabe interakcije, poput $W^{\pm}$ i Z^0 bozona, Drell-Yan proces uistinu je postao vodeći produkcijski mehanizam u otkrivanju $W^{\pm}$ i Z^0 bozona u UA1 [6] i UA2 [7] [8] eksperimentima. Zbog nabijene prirode $W^{\pm}$ produkcije, poopćeni Drell-Yan proces postaje izvor informacija pri razdvajanju okusa kvarkova, zato što je u proton-proton i proton-antiproton sudaru QCD podproces *flavor sensitive*: $\mathrm{u\bar{d}} \to \mathrm{W}^+$, $\mathrm{d\bar{u}} \to \mathrm{W}^-$ i $\mathrm{q\bar{q}} \to \mathrm{Z}$.

Za potrebe ovog rada samo dva visoko-energetska, izolirana miona suprotnog naboja, ponekad u pratnji s fotonima ili gluonima početnog i konačnog stanju mogu biti klasificirana kao signalni događaj. Drugi fizikalni procesi koji također mogu pružiti dva visoko-energetska miona prema [9] su:

- QCD hadronski proces u kojem zbog bottom i charm kvarkova intenzivno nastaju teški mezoni, poluleptonski se raspadaju na dva miona u velikom broju događaja. Mioni nastali raspadom piona i kaona relativno su meki što znači da imaju mali p_T i kao takvi ne prežive zahtjeve pri odabiru događaja.

- Proces Z $\rightarrow \tau^+\tau^- \rightarrow \mu^+\mu^- +$ X može imitirati background. Omjer grananja (eng. Branching Ratio - BR) za Z raspad na nabijeni leptonski par je 3.3 $\%$, a na $\tau \rightarrow \nu_\tau \mu \nu_\mu$ je 17 $\%$. Pojasnimo, BR govori vjerojatnost da se masivnija čestica, u ovom slučaju Z bozon, raspadne na točno određeni skup lakših čestica.
- W+ mlazovi (eng. jets) u kojima se W raspada mionskim kanalom može imitirati pozadinu, ukoliko jedan mlaz, najčešće vodeći, imitira mion.
- Izvor di-miona također mogu biti $\mathrm{t\bar{t}}+$ mlazovi u kojima mioni nastaju leptonskim raspadom W-a proizašlog iz $\mathrm{t\bar{t}}$ raspada.
- Razne di-bozon produkcije također daju obol pozadini svojim leptonskim raspadom.

Poglavlje 2

Eksperimentalna aparatura

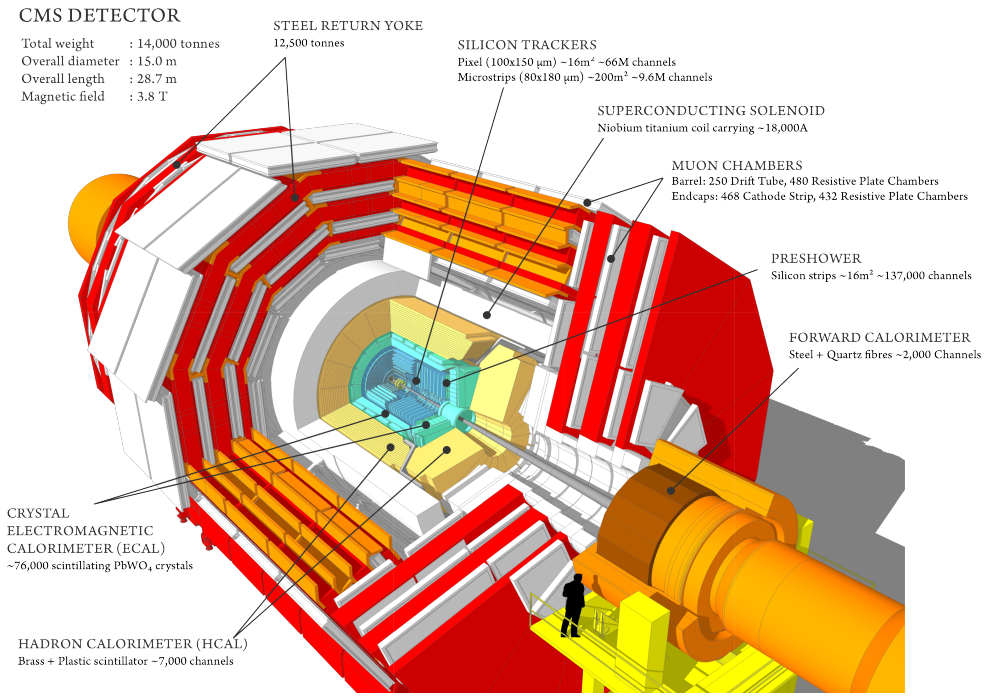
Stotinu metara ispod zemlje, u CERN-u, Europskoj organizaciji za nuklearna istraživanja, na Francusko-Švicarskoj granici, pokraj Ženeve, nalazi se 27 km dugačak, najveći i najsnažniji sudarivač čestica na svijetu imena Large Hadron Collider (LHC). Unutar akceleratora, dvije visoko-energetske zrake čestica putuju brzinom vrlo blizu brzini svjetlosti prije nego ih se sudari. Jako magnetsko polje supravodljivih magneta usmjerava zrake duž prstena, u suprotnim smjerovima, u odvojenim cijevima unutar vrlo visokog vakuuma. Materijali od kojeg su načinjeni elektromagneti ohlađeni su na iznimno niske temperature, -271.3 $^{\circ}$C, nalaze se u supravodljivom stanju te kao takvi provode struju bez otpora ili gubitka energije. Centralni događaj jesu proton-proton sudari energije 13 TeV u sustavu centra mase, luminoziteta do 10^{34} cm^{-2} s^{-1}. Pri tom luminozitetu, detektori svake sekunde bilježe oko milijardu neelastičnih sudara, što u prosjeku daje 1000 čestica iz više od 20 neelastičnih sudara za svaki interval križanja paketa snopova od 25 ns. Takav enorman broj događaja reducira se pomoću online postupka odabira zvanog okidač (eng. trigger) koji ih svede na oko 100 po sekundi za daljnju pohranu i analizu. Očekivani udarni presjek je otprilike 100 mb i u načelu opisuje vjerojatnost sudara dviju čestica, u našem slučaju dvaju protona. Polumjer atomske jezgre je reda veličine 10^{-14} m, onda je udarni presjek reda veličine 10^{-28} m^2, od tud dolazi definicija jednog barna od 10^{-28} m^2. Za detaljniji opis poslužite se referencom [1] odakle sam i preuzeo podatke o detektoru opisanom u ovom poglavlju.

Jedan od detektora na LHC eksperimentu jest Compact Muon Solenoid (CMS), (slika 2.1) čiji su osnovni ciljevi proučavanje čestične fizike na TeV energijskoj skali kao i promatranje prirode loma elektroslabe simetrije za koju je, pretpostavlja se, odgovoran Higgsov mehanizam. Detektor je dizajniran poput višeslojnog filtera s ciljem iskorištavanja različitih svojstava čestica proizašlih iz proton - pro-

ton sudara. Zaustavljajući, prateći i mjereći njihovu energiju i količinu gibanja dolazi se do informacija o njihovom identitetu. Da bi navedeno bilo ostvarivo, CMS treba sadržavati:

- Detektor tragova za precizno mjerenje količine gibanja (pododjeljak 2.1)
- Elektromagnetski kalorimetar (ECAL) za mjerenje energije elektrona i fotona (pododjeljak 2.2)
- Hadronski kalorimetar (HCAL) za mjerenje energije hadrona (kraj pododjeljka 2.2)
- Mionski sustav za detekciju miona (pododjeljak 2.3).

što se može i vidjeti na slici 2.1.



Slika 2.1: Presjek CMS detektora. LHC zrake putuju u suprotnim smjerovima prije nego ih se sudari u centralnom dijelu CMS-a. Izvor: [10]

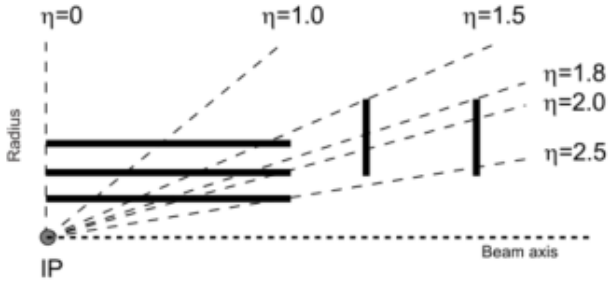
Shodno navedenome, prvo što CMS treba je snažno magnetsko polje. Što nabijena čestica ima veću količinu gibanja, to će na nju, magnetsko polje imati slabiji učinak, na način da će njena putanja biti manje zakrivljena. Za precizno mjerenje čestica velike količine gibanja, poput miona, potrebno je snažno magnetsko polje. Količina gibanja se može mjeriti kako unutar zavojnice, pomoću detektora tragova tako i izvan pomoću mionskih komora.

CMS detektor dugačak je 21.6 m, promjera 14.6 m i ukupne mase 12500 t. Glavni dio CMS detektora je supravodljivi solenoid duljine 13 m, unutarnjeg promjera 6 m, koji osigurava magnetsko polje jačine do 4 T. Unutar volumena solenoida nalazi se detektor tragova (eng. tracker), homogeni elektromagnetski kalorimetar te hadronski kalorimetar. Putanja nabijenih čestica mjeri se u detektoru tragova, koji pokriva puni kut azimuta ϕ definiranog u xy ravnini od r osi, gdje je r radijalna koordinata u toj ravnini te pseudorapiditet $|\eta| < 2.5$ definiran kao $\eta \equiv -\ln\tan(\theta/2)$. Polarni kut theta θ mjeri se suprotno od smjera kazaljke na satu od smjera zrake. Ishodište koordinatnog sustava postavlja se u točki sudara, os ordinata vertikalno prema gore, apscisa radijalno prema središtu, a z os u smjeru zrake. Prema tome, energija i količina gibanja transverzalni putanji upadne zrake označavaju se s E_T odnosno s p_T te se računaju iz x i y komponenti. Mioni se mjere u intervalu $|\eta| < 2.4$ pomoću detekcijskih ploča izrađenih koristeći tri tehnologije: posmične cijevi (eng. Drift Tubes - DT), komore s katodnim trakama (eng. Cathode Strip Chambers - CSC) i komore s otpornim pločama (eng. Resistive Plate Chambers - RPC).

2.1 Detektor tragova

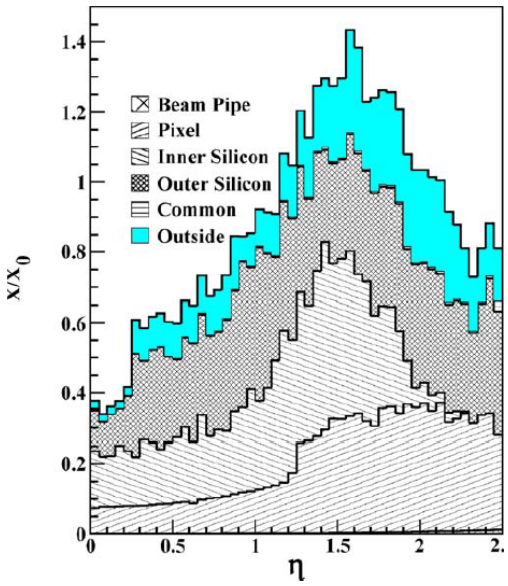
Detektor tragova prvi je na udaru čestica nastalih u sudarima. Napravljen je u cijelosti od silicija, u dva dijela: silicijski piksel detektor tragova (eng. silicon pixel tracker) i silicijski detektor tragova s mikrotrakama (eng. silicon strip tracker). Nabijene čestice gibaju se spiralno u CMS-ovom magnetskom polju, detektori mjere poziciju čestica što omogućava rekonstrukciju putanja. Zakrivljenost njihovih putanja omogućava mjerenje količine gibanja - što je više zakrivljena, čestica ima manju količinu gibanja. Transverzalna količina gibanja može biti preko 1 GeV, a pseudorapiditet u rasponu od $|\eta| \leq 2.5$ što je upravo područje detekcije koje osigurava detektor tragova.

Piksel detektor najbliži je točki sudara i kao takav veoma je bitan pri rekonstrukciji čestica kratkog života. Zbog same blizine mjestu sudara suočen je s enormnim brojem čestica; na udaljenosti od svega 8 cm biti će ih oko 20 milijuna po cm^2 svake sekunde. Svaki sloj (ukupno ih je četiri) podijeljen je na segmente, male silicijske senzore veličine 100 $\times$ 150 μm, poput pločica. Nabijena čestica prolazeći kroz detektor tragova pogodi elektron, preda mu dovoljno energije da se odvoji od atoma silicija, stvori elektron šupljina par što u konačnici pobuđuje piksel koji odaje trag čestice. Kako je detektor napravljen od 2D pločica i ima nekoliko slojeva, može se stvoriti trodimenzionalna slika ([1]). Slika 2.2 geometrijski prikazuje pseudorapiditet.



Slika 2.2: Geometrijski prikaz piksel detektora. Izvor: [1]

Prošavši kroz piksel detektor, čestice nailaze na 10 slojeva silicijskog detektora s trakama. Silicijski senzori s trakama rade na istom principu kao piksel senzori; nabijena čestica prolaskom kroz materijal izbija elektron, što unutar električnog polja izazove mali puls struje koji traje svega nekoliko nanosekundi. Puls se zatim uveća, što daje udar (eng. hit) i kao rezultat dobijemo rekonstruirani trag. Od detektora tragova se očekuje da međudjeluje što je manje moguće sa česticama, dok je kalorimetar dizajniran da zaustavi čestice. Međutim, sav taj materijal silicijskog detektora tragova koji se nalazi ispred ECAL-a predstavlja problem. Na slici 2.3 može se vidjeti ovisnost prijeđene udaljenosti elektrona i fotona kroz detektor tragova o pseudorapiditetu η prije nego dospiju do ECAL-a.



Slika 2.3: Količina materijala ispred ECAL-a u jedinicama radijacijske duljine. Izvor: [11]

Debljina detektora tragova kojeg prođu elektroni i fotoni kreće od oko $0.35\ X_0$ za centralni pseudorapiditet ($\eta = 0$) i raste do $1.4\ X_0$ kako se približava ECAL-u

te pada do $0.8\ X_0$ pri $\eta = 2.5$. Zakočnim zračenjem (njem. bremsstrahlung) elektroni gube energiju u materijalu detektora tragova. Emitirani fotoni mogu stvoriti elektron-pozitron par, što dovodi do ranog razvoja elektromagnetskog pljuska.

2.2 Elektromagnetski kalorimetar (ECAL)

Za potpunu sliku događaja u LHC-u, osim količine gibanja potrebna je i energija upadnih čestica. Sljedeći sloj detektora napravljen od veoma gustog ($8.3\ \mathrm{g/cm^3}$) transparentnog olovo - tungsten ($\mathrm{PbWO_4}$) materijala radijacijske duljine $X_0 = 0.89\,\mathrm{cm}$ te Mollierovog radijusa $R_m = 2.19\ \mathrm{cm}$ naziva se elektromagnetski kalorimetar. $\mathrm{PbWO_4}$, prolaskom elektrona, fotona i mlazova čestica stvara scintilacijsko svjetlo proporcionalno energiji upadne čestice. Scintilacija je pojava pri kojoj neki materijali ozračeni ionizirajućim zračenjem stvaraju fotone vidljive svjetlosti. Centralni barrel (EB) pokriva pseudorapiditet u rasponu $|\eta| < 1.479$, dok endcap (EE) pokriva $1.479 < |\eta| < 3.0$. Higgsov raspad na dva fotona vrlo lako mogu imitirati određeni pozadinski procesi. Na primjer, neutralni pion kreiran u inicijalnom proton-proton sudaru može se praktički odmah raspasti na dva usko razmaknuta fotona. Zbog tog ECAL također sadrži *preshower* detektor, prije endcapa, čiji je cilj identificirati neutralne pione u endcap regiji u rasponu pseudorapiditeta $1.653 < |\eta| < 2.6$ i tako omogućiti CMS-u razlikovanje samostalnih visoko-energetskih fotona od dva usko razmaknuta fotona. *Preshower*, svojim gustim olovnim materijalom inicira elektromagnetski pljusak upadnih fotona/elektrona, da bi pomoću silicijskih trakastih senzora mjerio poziciju pljuska (stoga i upadne čestice) i iz tog zaključio depozit energije upadne čestice (jer je broj čestica u pljusku proporcionalan energiji upadne čestice). Sloj iza ECAL-a, hadronski kalorimetar (HCAL) dizajniran je da detektira hadrone, čestice sastavljene od kvarkova, koje interagiraju jakom silom. Jedino neutrini i mioni prođu iza HCAL-a, potonji se zatim prate u mionskim komorama. Neutrini su električki neutralni pa ne međudjeluju i pobjegnu detekciji, ali se njihova prisutnost indirektno primjećuje iz nedostajuće transverzalne energije u događaju. Drugim riječima, čestice koje prolete kroz CMS ostavljaju karakterističan uzorak u različitim slojevima, što omogućava njihovu identifikaciju.

2.3 Mionski sustav

Detekcija miona je, kao što i samo ime detektora implicira, jedan od najvažnijih zadataka CMS-a. Predviđeni raspad Higgs bozona na ZZ^* i zatim na 4 leptona smatra se zlatnim kanalom. Zašto to mislimo? Osim što ih je relativno jednostavno detektirati, mioni mogu, bez međudjelovanja, prodrijeti kroz nekoliko metara debljine željeza. Manje su podloži radioaktivnom gubitku energije u ma-

terijalu detektora tragova od elektrona jer su oko 200 puta masivniji. Stoga ih CMS-ovi kalorimetri neće zaustaviti kao što zaustave većinu drugih čestica. Shodno tome, komore za detekciju miona se stavljaju na sam kraj eksperimenta, izvan solenoida, gdje će mioni najvjerojatnije biti jedine čestice.

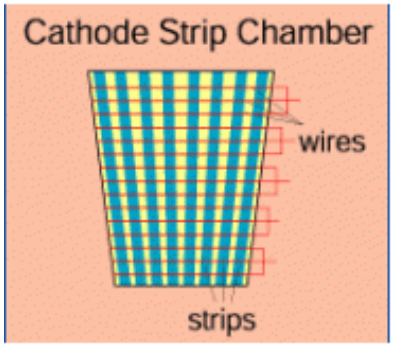
Tri tipa plinskih ionizacijskih komora, kojih ima 1400, sačinjavaju CMS mionski sustav: 250 komora s posmičnim cijevima i 540 komora s katodnim trakama prate tragove čestica, dok 610 komora s otpornim pločama gradi sustav okidača koji odlučuje zadržati ili ne promatrani mion. Detaljni opis je dostupan na referenci [1].

Posmične cijevi (DT)

Prolaskom miona ili neke druge električki nabijene čestice, kroz volumen ispunjen plinom, izbija se elektron iz atoma plina kojeg električno polje usmjerava prema pozitivno nabijenim žicama - anodama. Gledajući mjesto udara elektrona o žicu i računajući početnu udaljenost miona od žice dobivamo dvije koordinate položaja miona. Kao takav, DT detektor pokriva pseudorapiditet u intervalu $|\eta| < 1.2$.

Komore s katodnim trakama (CSC)

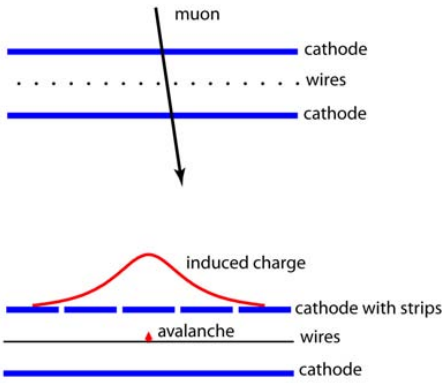
U dvije *endcaps* regije CMS-a, gdje je stopa miona i pozadine visoka, magnetsko polje jako i neuniformno, mionski sustav koristi komore s katodnim trakama koje pokrivaju pseudorapiditet u intervalu $0.9 < |\eta| < 2.4$. Sastoje se od redova pozitivno nabijenih anodnih žica ukrštenih s negativno nabijenim bakrenim katodnim trakama unutar volumena plina, kao što se može vidjeti na slici 2.4.



Slika 2.4: Sastav komora s katodnim trakama. Izvor: [10]

Shema CSC-a na slici 2.5 prikazuje princip rada. Mion prolaskom kroz volumen plina izbija elektrone koji nahrle prema anodnim žicama. Pozitivni ioni udaljava-

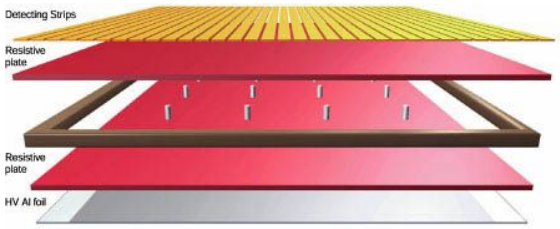
jući se od anode prema katodi induciraju pozitivan puls na trakama pod pravim kutom na smjer žica. Okomitost žica i traka osigurava dvije koordinate položaja za svaku česticu koja prođe kroz komoru. Šest slojeva CSC modula osigurava robusnu i preciznu identifikaciju miona kao i sparivanje njihovih tragova s onima iz detektora tragova.



Slika 2.5: Shematski prikaz CSC modula. Izvor: [1]

Komore s otpornim pločama (RPC)

Kad LHC dosegne svoj puni luminozitet, mionski sustav gubi sposobnost točnog mjerenja vremena križanja snopova, stoga se u endcap i u barrel regije uvodi dodatni sustav okidača koji se sastoji od komora s otpornim pločama (RPC). RPC je komora s dvostrukim procjepom, brzi plinski detektor, koji paralelno sa DT i CSC tvori mionski sustav okidača. Osim što je brz i neovisan, osigurava oštar p_T prag u širokom spektru pseudorapiditeta $|\eta| < 1.6$. Sastoji se od dvije paralelne ploče: pozitivno nabijene anode te negativne katode, obje napravljene od veoma otpornog plastičnog materijala odvojenog volumenom plina; kao na slici 2.6. Sličan princip rada kao i kod DT i CSC komora vrijedi i ovdje; mion izbija elektrone iz atoma plina, ti elektroni sudarima s drugim atomima izbijaju još elektrona i tako nastane lavina elektrona. Vanjske metalne trake nakon kratke ali precizne vremenske odgode detektiraju signale (elektrone) koji nesmetano prođu kroz elektronski providne elektrode. Udarac uzrokuje veoma brzi izračun količine gibanja miona, kojeg okidač koristi da trenutačno odluči treba li ili ne zadržati podatke.



Slika 2.6: Slojevi RPC modula. Izvor: [10]

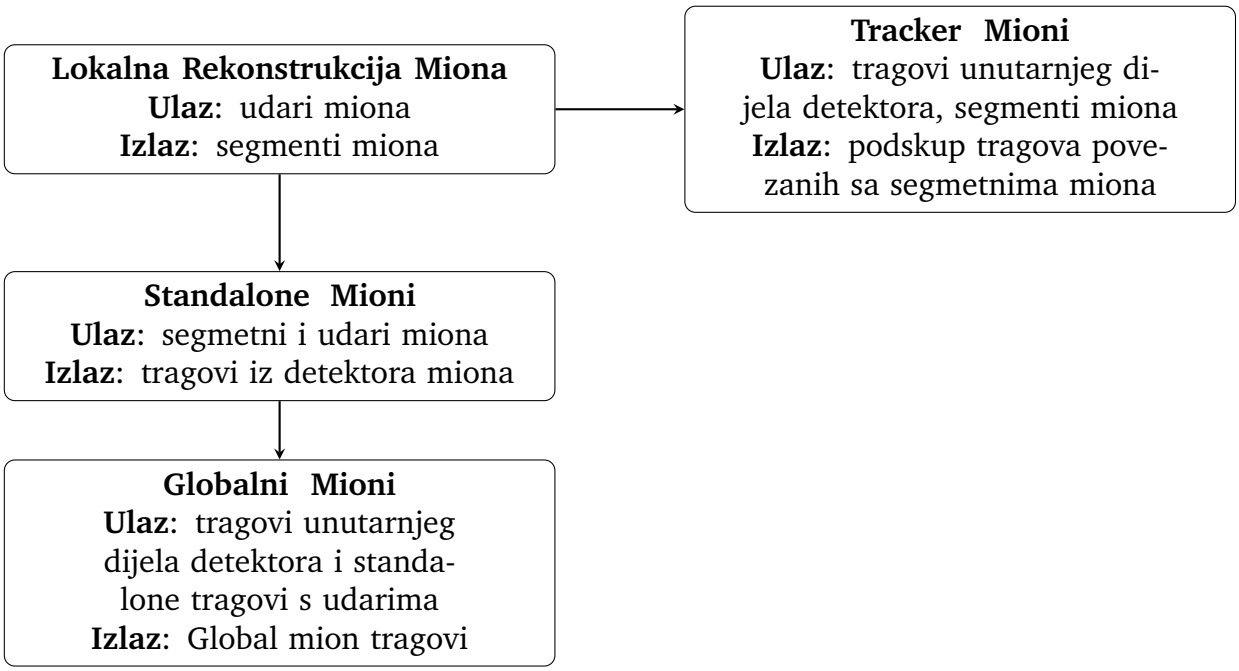
Ponovimo, DT i CSC pokrivaju pseudorapiditet u intervalu $|\eta| < 1.2$ odnosno $0.9 < |\eta| < 2.4$, upotpunjene sa RPC-om u intervalu $|\eta| < 1.6$.

Poglavlje 3

Rekonstrukcija miona

Rekonstrukcija i identifikacija miona jedna je od glavnih zadaća CMS-a, zbog čega ćemo ukratko objasniti dva pristupa rekonstrukciji tragova čije korake možete popratiti na dijagramu 3.1.

3.1 Rekonstrukcija tragova miona



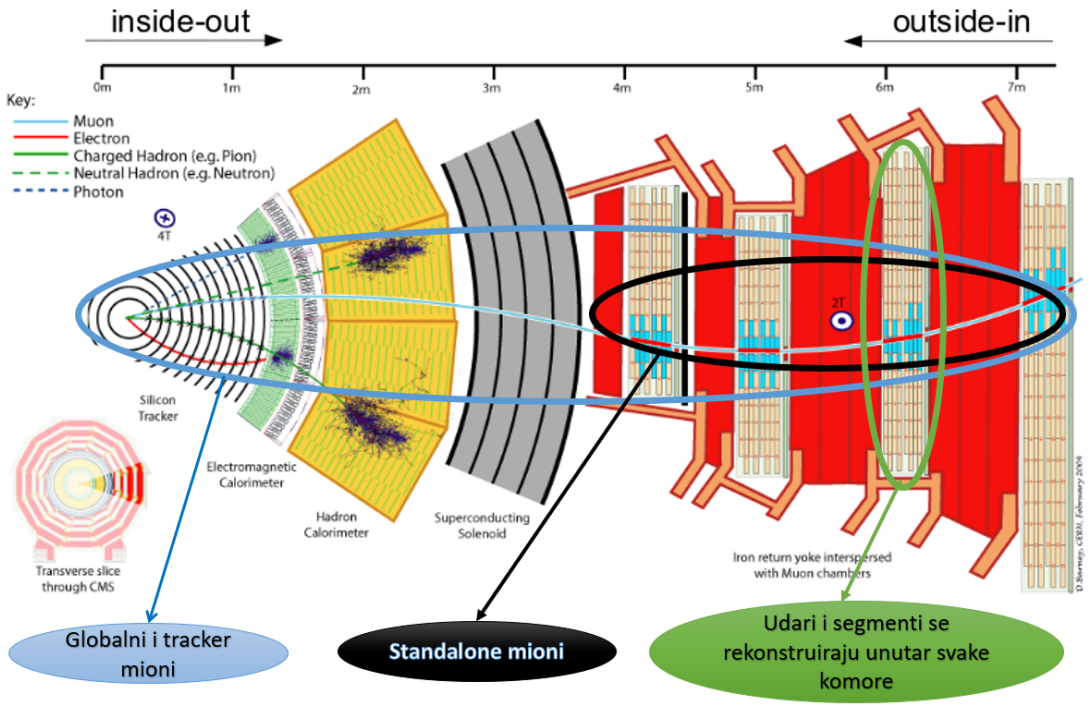
Slika 3.1: Pregled rekonstrukcije miona. Izvor: [12].

Lanac rekonstrukcije miona započinje "lokalnom rekonstrukcijom". Mioni, kao i sve druge električki nabijene čestice, prolaskom kroz subdetektor ioniziraju plin u komorama što rezultira stvaranjem električnih signala na žicama i trakama. Iz električnih signala rekonstruiraju se udari, a zatim se unutar svake komore povezuju da bi se dobili segmenti.

Tragovi se prvenstveno rekonstruiraju u unutarnjem dijelu detektora tragova (*tracker track*) i u mionskom sustavu (*standalone muon track*) nezavisno, na osnovi kojih se formiraju dva komplementarna pristupa rekonstrukciji miona koji na kraju vode do tri različita tipa mionskih kandidata: tracker, standalone i global. Važno je naglasiti da navedena tri tipa nisu međusobno isključiva - jedan mion može biti u sve tri kategorije.

- **Standalone** mionski tragovi rekonstruiraju se isključivo informacijama iz mionskih komora. Segmenti rekonstruirani u mionskim komorama se koriste za generiranje *seedova* sastavljenih od vektora položaja i smjera i procjene transverzalne količine gibanja.
- **Globalni mioni** (*outside-in*) nastaju povezivanjem standalone mionskih tragova s rekonstruiranim tragom iz unutarnjeg dijela detektora tragova.
- **Tracker mioni** (*inside-out*) nastaju pristupom suprotnim global mion rekonstrukciji. Svaka trag unutarnjeg dijela detektora tragova s $p_T > 0.5\,\mathrm{GeV}$ i ukupnom količinom gibanja $p > 2.5\,\mathrm{GeV}$ razmatra se kao mogući kandidat za mion te se tek onda ekstrapolira u mionski sustav. Ako barem jedan mionski segment poveže ekstrapolirani trag, odgovarajući trag unutarnjeg dijela detektora tragova klasificiramo kao tracker mion. Međutim, navedeni kriterij na količinu gibanja vrlo je nizak pa se ne koristi bez dodatnih zahtjeva.

Tri faze rekonstrukcije miona mogu se vidjeti i na slici 3.2.



Slika 3.2: Presjek CMS detektora sa označene tri faze rekonstrukcije miona.

Skoro 99 % miona nastalih unutar geometrijskog prihvaćanja mionskog sustava bivaju rekonstruirani ili kao globalni ili kao tracker mioni, često kao oboje. Globalni i tracker mioni koji dijele isti trag unutarnjeg dijela detektora tragova spajaju se u jedinstvenog kandidata. Tracker mion algoritam koristan je za identifikaciju miona s malom transverzalnom komponentom količine gibanja ($p_T < 5$ GeV) koji vjerojatno neće ostaviti dovoljno signala u mionskoj komori da bi standalone mion bio rekonstruiran. Za više informacija pogledajte [13] odakle je i preuzet opis rekonstrukcije i podjela mionskih tragova.

3.2 Monte Carlo Gen Matching

Svaki događaj sadrži kolekciju rekonstruiranih objekata i kolekciju Monte Carlo generiranih objekata. Monte Carlo Gen Matching (MCGM) odvija se sljedećim algoritmom:

Algorithm 1 Monte Carlo Gen Matching

```
for all reconstructed objects do
    if (p_T > p_{T,threshold}) then
        for all generated objects do
            if (object == muon AND object status == 1) then
            Find nearest reconstructed muon in dR; (eq:3.2.1)
            if dR > 0.3 then
                Reconstructed muon → Background.
            if PromptFinalState then
                TruePromptMuon → Signal
            else if Mion from τ then
                Reject (neither signal, nor background)
            else
                Background
```

Pojasnimo ga u kratkim crtama: Iteracijom po svim rekonstruiranim objektima odbacuju se objekti s transverzalnom komponentom količine gibanja manjom od granične vrijednosti. Iteracijom po generiranim objektima traže se mioni statusa 1. Status 1 objekti su objekti koji nisu raspadnuti (eng. decayed) ili fragmentirani i kao takvi predstavljaju konačno stanje dobiveno generatorom. Nakon što pronađemo generirani mion statusa 1, tražimo mu najbliži rekonstruirani objekt unutar stošca dR definiranog:

$$d_R = \sqrt{d\phi^2 + d\eta^2}. \tag{3.2.1}$$

Ukoliko je dR veći od 0.3 rekonstruirani objekt se klasificira kao pozadina. Ukoliko generirani objekt ima svojstvo *PromptFinalState*, tada je rekonstruirani objekt *TruePromptMuon* i klasificiramo ga kao signal. *PromptFinalState* imat će objekti statusa 1 koji su direktno povezani s hard scattering procesom u interakciji, tj. koji dolaze iz primarnog verteksa i nisu nastali raspadom neke druge čestice. U slučaju da je generiran objekt mion iz raspada τ leptona, tada se odbacuje. U svim preostalim slučajevima, rekonstruirani objekt klasificira se kao pozadina.

Prošavši kroz sve rekonstruirane objekte, algoritam povezivanjem rekonstruiranih objekata s MC generiranim objektima odaje njihov identitet što nas u konačnici dovodi do dvije klase objekata unutar jednog velikog skupa podataka: klasa signalnih objekata i klasa pozadinskih objekata. Skup podataka spremljen je u ROOT tip datoteke nazvan "**train.root**" veličine 7 GB i kao takav sastoji se od preko 66 milijuna rekonstruiranih objekata s 33 karakteristične značajke. Zbog iznimno velike količine podataka, računalno je veoma zahtjevno, na slabijim konfiguracijama čak i nemoguće, učitati ih istovremeno u radnu memoriju za daljnju analizu. Shodno tome podaci se učitavaju u dijelovima/komadima nad kojima se obavlja predselekcija određena sljedećim uvjetom:

```
('n_obs_int' > 1) & ('mu_dz' < 1) & ('mu_dxy' < 0.5) & (('
    is_tracker_mu' == True) | ('is_global_mu' == True))
```

Ovo je standardni skup rezova (eng. loose cuts) primijenjen u velikom broju analiza na CMS-u gdje koristeći *dz* i *dxy* rezove odabiremo samo mione koji su najvjerojatnije nastali u primarnom verteksu, tj. nisu došli raspadom neke druge čestice nastale u primarnom verteksu. Također se ograničimo samo na mione rekonstruirane u dijelu detektora za tragove ili globalne mione. U varijablu nazvanu "*features*" spremljena je lista značajki koje će biti zadržane. To su najvažnije varijable što se tiče miona kao na primjer: kvaliteta traga, broj signala u slojevima detektora tragova i izolacija miona, tj. varijable koje opisuju postoje li neke druge čestice oko traga miona. Cijela lista značajki prikazana je ispod:

```
features = ['glb_valid_mu_hits','tk_valid_hits','tk_valid_pixel_hits'
    ,'mu_pf_photon_iso','mu_pf_charged_had_iso','mu_pf_neutral_had_iso
    ','mu_rho','mu_sip','mu_dxy','mu_dz']
```

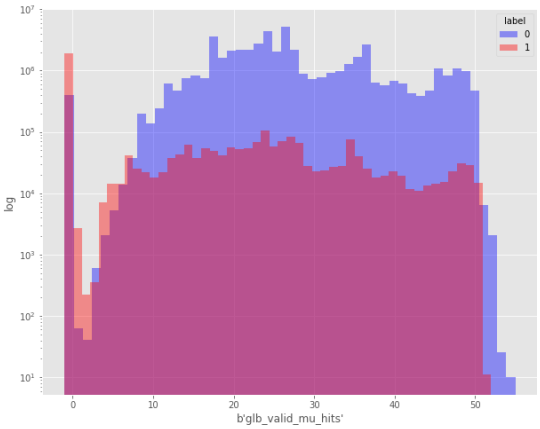
U sljedećem odjeljku prikazati ćemo distribuciji spomenutih značajki za rekonstruirane signalne i pozadinske objekte. Konačno, uređeni skup podataka sprema se u *Pandas DataFrame* nazvan jednostavno "**df**". Kompletan Python kod učitavanja i predselekcije dostupan je u dodatku A.1.

3.3 Vizualizacija skupa podataka

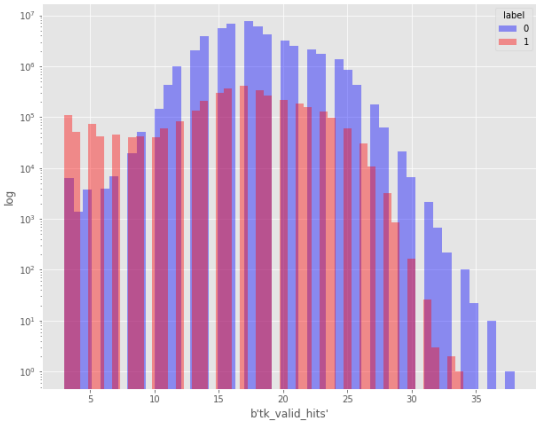
Pogledajmo sad distribuciju značajki našeg skupa podataka. Napomenimo da smo signalne rekonstruirane objekte označili s 0 a pozadinske s 1 značajkom "label". Zbog čega smo tako napravili objasnit ćemo u sljedećem dijelu rada. Ova infomacija nam je dostatna za prikazati i razumjeti histograme skupa podataka u nastavku. Svim histogramima y os sadrži logaritamsku skalu gustoće objekata zbog velikog nesrazmjera distribucije podataka.

Osnovne značajke

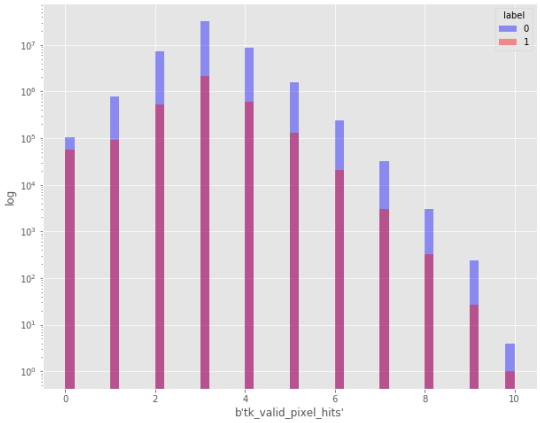
Na slici 3.3 nalaze se histogrami osnovnih značajki miona koji prikazuju distribuciju signalnih i pozadinskih objekata skupa podataka.



(a) Broj valjanih signala globalnih miona u mionskim komorama.



(b) Broj signala u detektoru s mikrotrakama.

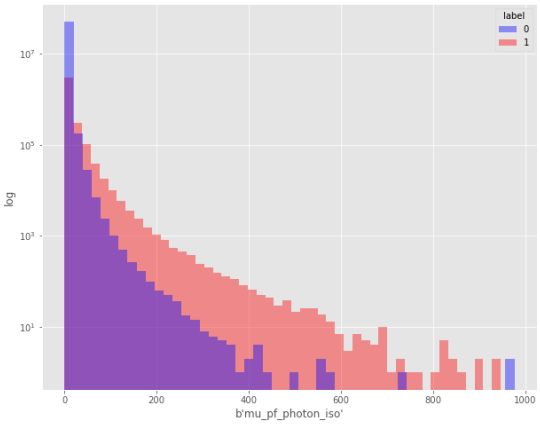


(c) Broj signala u piksel dijelu detektora tragova.

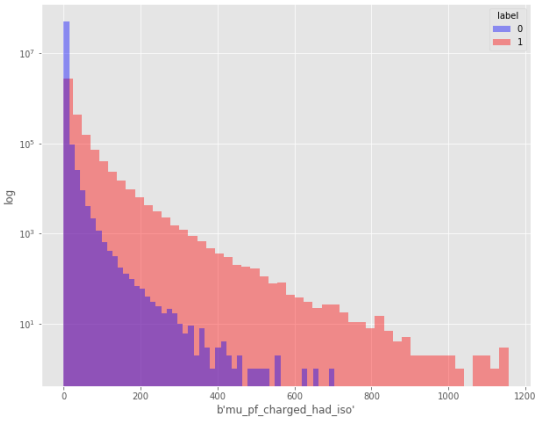
Slika 3.3: Histogrami osnovnih značajki miona.

Izolacijske varijable

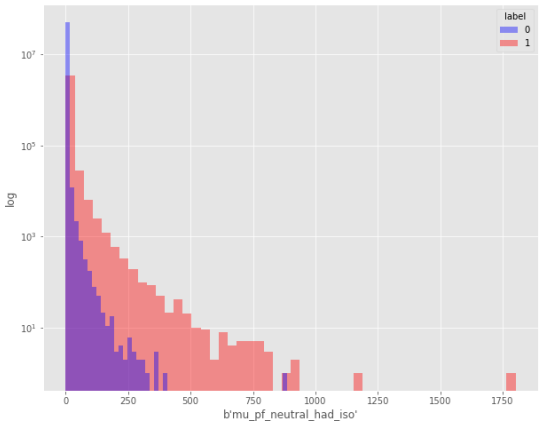
Na slici 3.4 prikazani su histogrami izolacijskih varijabli miona:



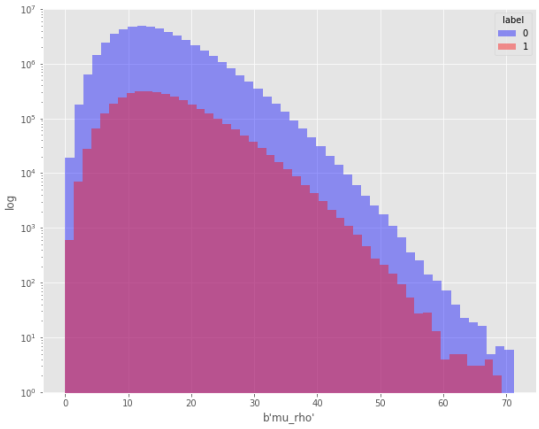
(a) Informacije o prisutstvu fotona oko PF miona.



(b) Informacije o prisutstvu nabijenih hadrona oko PF miona.



(c) Informacije o prisutstvu neutralnih hadrona oko PF miona.

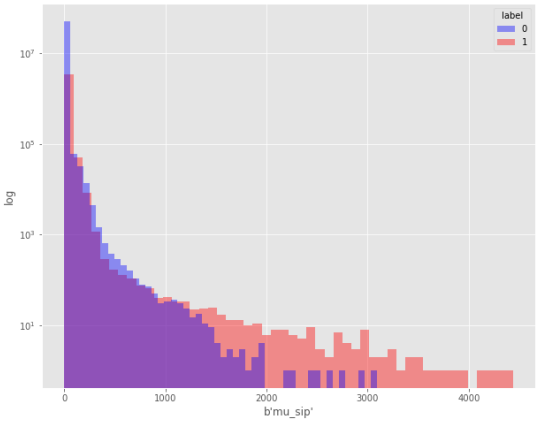


(d) Varijabla koja opisuje gustoću energije čestice.

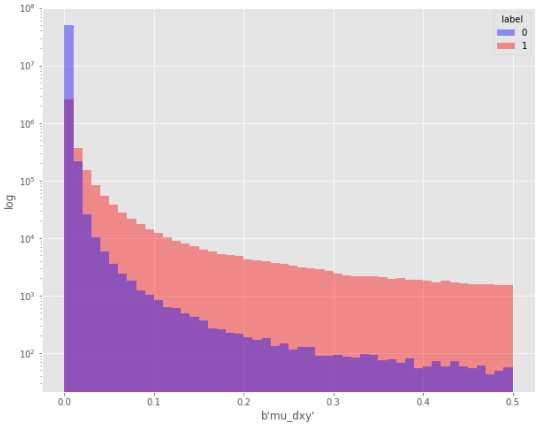
Slika 3.4: Histogrami izolacijskih varijabli miona koje daju informacije o verteksu.

Varijable parametra sudara

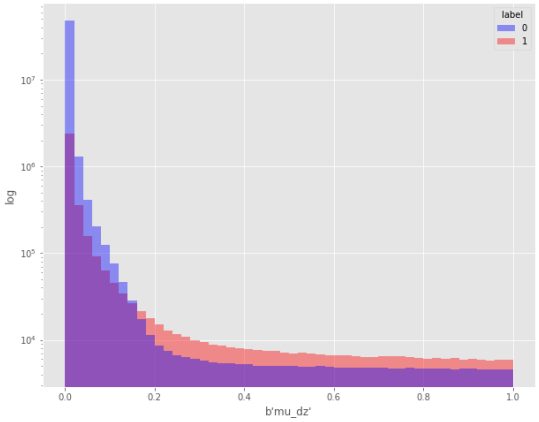
Na slici 3.5 prikazani su histogrami značajki signifikantnosti parametra sudara (eng. Significance Impact Parameter - SIP) miona:



(a) Signifikantnost parametra sudara miona.



(b) Transverzalna komponenta parametra sudara miona.



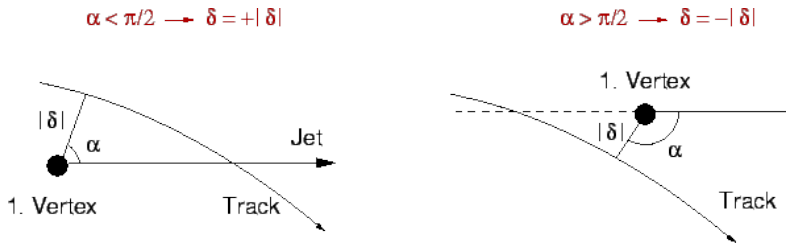
(c) Longitudinalna komponenta parametra sudara miona.

Slika 3.5: Histogrami SIP značajki miona.

Signifikantnost parametra sudara

$$d_{SIP} = \frac{d}{\sigma_d} \tag{3.3.1}$$

definirana je kao omjer vrijednosti parametra sudara s njegovom procijenjenom neodređenošću i koristi se kao opservabla. Parametar sudara definiran je kao najkraća transverzalna udaljenost putanje do točke primarnog verteksa (eng. distance of closest approach - DCA). Predznak parametra sudara putanje je pozitivan ako je kut između npr. mlaza čestica i linije koja povezuje primarni verteks s točkom DCA manji od 90° (kao na lijevom dijelu slike 3.6), inače je negativan (desni dio slike 3.6).



Slika 3.6: Skica rekonstrukcije parametra sudara miona. Izvor: [14]

Parametar sudara se koristi za odbacivanje *heavy flavour* pozadine kao i pozadine koju sačinjavaju lagani kvarkovi. d_{xy} i d_z su transverzalna i longitudinalna komponenta parametra sudara. Apsolutna vrijednost mora biti manja od 0.5 cm za d_{xy} i 1 cm za d_z.

Pogledamo li detaljnije sliku 3.5a primijetit ćemo da nemali broj signalnih miona ima vrlo visoku SIP vrijednost što nije očekivano. Informaciju iznosim kao svoje zapažanje i sumnju u valjanost simulacije te smatram da bi bilo korisno dodatno istražiti uzrok, ali to nije tema ovog rada.

Kratki osvrt

Napravimo kratki osvrt dosadašnjeg rada. Provukavši rekonstruirane generirane objekte kroz Monte Carlo Gen Matching algoritam dobili smo dvije klase objekata: signalne i pozadinske. Koji nam je sljedeći korak? Što možemo zaključiti iz tih podataka, možemo li sačuvati neko znanje koje će nam u budućnosti biti od koristi? Sad na red dolazi strojno učenje (engl. Machine Learning) koje počiva na ideji da sustav može naučiti iz podataka, identificirati uzorke i donositi odluke uz minimalnu intervenciju čovjeka. Kako to primijeniti na naš problem? Model ćemo trenirati na simuliranim podacima, dakle na rekonstruiranim objektima jer samo u simuliranim podacima znamo koji objekt je signal, a koji pozadina da bi bio u mogućnosti klasificirati neke nove, nepoznate podatke, kad nam to bude potrebno. Da bi se imitirao ovakav slučaj, dostupni podaci će se podijeliti u omjeru 75:25 u korist trening skupa. Svi modeli će biti trenirani koristeći istu podjelu i zatim evaluirani na test skupu podataka. Test skup podataka imitira nove nepoznate podatke. Kad budemo zadovoljni performansama modela na test skupu, možemo ga primijeniti na eksperimentalnim podatcima prikupljenim radom CMS detektora. Krenimo sa strojnim učenjem.

Poglavlje 4

Klasifikacija podataka

4.1 Uvod u strojno učenje

Strojno učenje (eng. Machine Learning - ML u daljnjem tekstu) je disciplina umjetne inteligencije koja sustavima pruža mogućnost automatskog učenja i poboljšanja kvalitete iz iskustva bez da budu eksplicitno programirana. Sljedećim je riječima Tom Mitchell 1998. godine postavio problem učenja:

> *A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.*

Proces započinje dohvatom i analizom podataka na osnovi kojih algoritmi strojnog učenja izgrađuju matematički model te pomoću kojih se mogu zaključiti znanja ili pronaći uzorak skupa podataka s ciljem predviđanja svojstava nekih novih skupova podataka. Primarni cilj jest omogućiti računalu automatsko učenje bez posredovanja čovjeka. Vrste strojnog učenja najčešće se dijele u tri skupine:

- **Bez nadzora** - ne postoji vanjski učitelj na temelju čijih informacija pokušavamo učiti.
- **Učenje podrškom** - podrazumijeva kontinuirano učenje pri kojem, u svakom trenutku nemamo informaciju (od učitelja) je li odluka ispravna ili ne, ali povremeno se dobije povratna informacija u obliku nagrade koju kumulativno želimo maksimizirati.
- **Učenje pod nadzorom** - postoji vanjski učitelj na temelju čijih informacija možemo prilagoditi rezultate.

Osnovnu terminologiju preuzeti ćemo od [15], poslužite se poveznicom za podrobnija objašnjenja. Mi ćemo koristiti učenje pod nadzorom. Za tu svrhu definirajmo notaciju: $x^{(i)}$ su ulazne varijable odnosno atributi ili značajke podataka (eng. features), $y^{(i)}$ je izlaz ili ciljana varijabla (eng. target, label) koja se pokušava predvidjeti. Uređeni par $\left(x^{(i)}, y^{(i)}\right)$ naziva se primjer za treniranje (ulaz uparen s odgovarajućim izlazom), a lista m primjera za treniranje $\left\{\left(x^{(i)}, y^{(i)}\right); i = 1, ..., m\right\}$ skup za treniranje - *dataset* iz kojeg će se učiti. Sa X se označava prostor ulaznih vrijednosti, a s Y prostor izlaznih. Cilj problema nadziranog strojnog učenja je, za dani trening skup, naučiti funkciju $h : X \mapsto Y$ tako da je $h(X)$ dobar prediktor za odgovarajuću vrijednost y. Funkcija h iz povijesnih razloga naziva se **hypothesis** i izgrađuje se tijekom treninga. Kad je izlazna varijabla koja se pokušava predvidjeti kontinuirana, problem je regresijske prirode, dok ukoliko y poprima malen broj diskretnih vrijednosti, klasifikacijski problem je u pitanju.
Kako reprezentirati hipotezu h? Najčešće se započinje od linearne funkcije po x:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{4.1.1}$$

θ_i su parametri/težine koje parametriziraju prostor linearnih funkcija preslikavanja iz X u Y. Postavljajući $x_0 = 1$, prema [15], općeniti oblik može se zapisati kao:

$$\hat{\mathbf{y}} = h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T \cdot \mathbf{x} \tag{4.1.2}$$

gdje je desna strana zapisana u vektoriziranoj formi sa sljedećim varijablama:

- θ je vektor parametara modela koji uz član koji označava pristranost sadrži i težine značajki θ_1 do θ_n.
- n je broj ulaznih varijabli.
- θ^T je transponirani θ (vektor redak umjesto vektor stupca).
- $\mathbf{x}$ je vektor značajki, koji sadrži x_0 do x_n, uz x_0 uvijek jednako 1.
- $h(x)$ je *hypothesis* funkcija za model parametara θ.

Sad kad je problem postavljen, nameće se pitanje kako odabrati parametre θ, kako naučiti model? Prisjetimo se da trenirati model znači pronaći parametre takve da model najbolje paše na podatke iz skupa za treniranje - da je $h(x)$ što bliže y. Da bi to mogli izmjeriti potrebno je definirati mjeru koliko dobro (ili loše) model opisuje trening podatke.

Funkcija pogreške (eng. Cost Function)

Funkcija pogreške kvantificira grešku između predviđene i očekivane vrijednosti te je predstavi u obliku jednog realnog broja. Ona za svaku vrijednost od θ mjeri koliko je $h(x^{(i)})$ blizu odgovarajućem $y^{(i)}$. Intuitivan odabir u euklidskom prostoru je euklidska udaljenost ili srednja kvadratna udaljenost. Prema [15] definirana je sljedećim izrazom:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{4.1.3}$$

Ovo je svima poznata metoda najmanjih kvadrata (eng. least square function) - način pripasavanja funkcije kojim se želi zadržati minimalna udaljenost modela od točaka. Cilj je pronaći vrijednosti θ koje minimiziraju $J(\theta)$. Tada se vrijednost koju daje naziva *cost*, *loss* ili *error*.

Gradijentalni spust (eng. Gradient Descent)

Na koji način i zbog čega minimizirati funkciju pogreške? Zato što manja pogreška između predviđene i očekivane vrijednosti signalizira da je model obavio dobar posao u učenju. Gradijentalni spust jedan je od najpopularnijih iterativnih optimizacijskih algoritama prvog reda za pronalaženje minimuma funkcije. Minimum funkcije pogreške definirane jednadžbom 4.1.3 se traži poduzimanjem koraka proporcionalnih negativnom gradijentu funkcije u danoj točki i to iteracijom sljedeće formule:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{4.1.4}$$

i istovremenim ažuriranjem svih θ_j:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \tag{4.1.5}$$

Normalna jednadžba

Gradijentalni spust jedan je način minimiziranja $J(\theta)$. Postoji matematička jednadžba koja direktno pronalazi vrijednosti θ bez iterativnog algoritma. Eksplicitno minimizira funkciju pogreške uzimajući njenu derivaciju po θ i izjednačujući je s nulom. Poznata je kao normalna jednadžba i prema [15] dana je sljedećim izrazom:

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X} \right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y} \tag{4.1.6}$$

u kojem je:

- $\hat{\theta}$ vrijednost θ koja minimizira funkciju pogreške,
- **X** matrica dimenzije $m \times (n+1)$ gdje je m broj uzoraka, a
- **y** vektor izlaznih vrijednosti $y^{(i)}, i = 1, ..., m$.

Tip problema koji zahtijeva primjenu strojnog učenja može imati matricu **X** od nekoliko desetaka milijuna redaka, što je kod nas slučaj, stoga računanje matrice $\mathbf{X}^T\mathbf{X}$, a zatim i traženje njenog inverza može biti računalno vrlo zahtjevno. Naravno, numerički je stabilnije rješavati sustav jednadžbi $\mathbf{X}^T\mathbf{X}\theta = \mathbf{X}^T y$. Međutim, računalno je jeftinije (brže) pronaći rješenje gradijentalnim spustom.

Prebacimo sad fokus na tip problema s kojim smo suočeni u ovom radu; na klasifikacijski problem. Sve rečeno o regresiji vrijedi i ovdje, osim što vrijednosti y koje se nastoje predvidjeti poprimaju malen broj diskretnih vrijednosti. Naglasak će biti na binarnoj klasifikaciji u kojoj y može poprimiti samo dvije vrijednosti: 0 i 1. Konkretno, želimo napraviti klasifikator objekata gdje je $x^{(i)}$ neka značajka (npr. mu_dxy), a y će biti 0 ukoliko je objekt mion, a 1 inače. 0 se naziva negativnom klasom, a 1 pozitivnom. U sljedećem odjeljku detaljnije pogledajmo s kakvim podacima ćemo raditi.

4.2 Informacije o podacima

Rekonstrukcijom objekata nastalih simulacijom Drell-Yan procesa dobili smo skup podataka za treniranje. Skup podataka za treniranje sadrži već klasificirane objekte drugim riječima već identificirane čestice. Na tim simuliranim podacima trenirati ćemo model da bismo u budućnosti bili sposobni klasificirati nove, nepoznate eksperimentalne podatke.

Uvijek je pametan potez prije treninga promotriti skup podataka ne bi li se pronašle nekakve nepravilnosti, anomalije ili pogreške na koje će trebati obratiti pozornost ili čak promijeniti pristup rješavanja problema. S tim na umu, već smo u odjeljku 3.3 prikazali distribuciju značajki skupa podataka za treniranje za signalne i pozadinske objekte. Na slici 4.1 može se vidjeti koji tip podataka je sadržan u skupu podataka za treniranje, zatim koliko imamo rekonstruiranih objekata i koliko memorije nam zauzima naš skup. Biblioteka Pandas [16] automatski detektira tipove podataka prilikom učitavanja što u nekim slučajevima može rezultirati većim *dataframeom* nego je dostupna memorija u našem okruženju. Za numeričke vrijednosti Pandas će kreirati stupce s vrijednostima *int32, int64, float32, float64* što možemo i vidjeti na slici 4.1a. Ukoliko su nam poznate minimalne i maksimalne vrijednosti svake pojedine značajke (svakog stupca) i ukoliko njihov raspon može stati u neku sažetiju vrstu vrijednosti, pretvoriti ćemo varijablu te značajke u tu sažetiju vrstu vrijednosti. Značajkama

"*tk_valid_hits*", "*tk_valid_pixel_hits*" i "*label*" pretvorili smo vrstu vrijednosti iz *int32* u *uint8* dok značajki "*glb_valid_mu_hits*" pretvaramo u *int8* jer ipak sadrži negativne vrijednosti. Postupak upravo napravljen poznat je kao *downcasting* odnosno iznuda najsažetije vrste vrijednosti, bez gubitka preciznosti, koja može obuhvatiti očekivani raspon brojeva. S *DataFramea* od 3.8 GB dospjeli smo na 3.0 GB kao na slici 4.1b. Nije velika ušteda memorije, ali je svakako vrijedno napraviti.

```
df.info()
executed in 486ms, finished 20:18:43 2020-05-20
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54179399 entries, 0 to 54179398
Data columns (total 11 columns):
b'glb_valid_mu_hits'        int32
b'tk_valid_hits'            int32
b'tk_valid_pixel_hits'      int32
b'mu_pf_photon_iso'         float64
b'mu_pf_charged_had_iso'    float64
b'mu_pf_neutral_had_iso'    float64
b'mu_rho'                   float64
b'mu_sip'                   float64
b'mu_dxy'                   float64
b'mu_dz'                    float64
label                       int64
dtypes: float64(7), int32(3), int64(1)
memory usage: 3.8 GB
```

(a) Prije *downcastinga*

```
df.info()
executed in 358ms, finished 19:31:01 2020-05-16
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54179399 entries, 0 to 54179398
Data columns (total 11 columns):
b'glb_valid_mu_hits'        int8
b'tk_valid_hits'            uint8
b'tk_valid_pixel_hits'      uint8
b'mu_pf_photon_iso'         float64
b'mu_pf_charged_had_iso'    float64
b'mu_pf_neutral_had_iso'    float64
b'mu_rho'                   float64
b'mu_sip'                   float64
b'mu_dxy'                   float64
b'mu_dz'                    float64
label                       uint8
dtypes: float64(7), int8(1), uint8(3)
memory usage: 3.0 GB
```

(b) Nakon *downcastinga*

Slika 4.1: Tip podataka sadržan u skupu podataka za treniranje.

Slika 4.2 po stupcima, tj. po značajkama prikazuje osnovne informacije o skupu podataka poput broja rekonstruiranih objekata, srednje vrijednosti te značajke, njene standardne devijacije, minimalne vrijednosti itd.

```
df.describe()
executed in 1m 1.96s, finished 22:07:26 2020-06-05
```

	b'glb_valid_mu_hits'	b'tk_valid_hits'	b'tk_valid_pixel_hits'	b'mu_pf_photon_iso'	b'mu_pf_charged_had_iso'	b'mu_pf_neutral_had_iso'	b'mu_rho'	b'mu_sip'	b'mu_dxy'	b'mu_dz'	label
count	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07	5.417940e+07
mean	2.653065e+01	1.752045e+01	3.062367e+00	1.343987e+00	1.586395e+00	5.727284e-01	1.472672e+01	1.805250e+00	2.140664e-03	1.060589e-02	6.469579e-02
std	1.102748e+01	3.407826e+00	7.590516e-01	5.944490e+00	9.495778e+00	2.944501e+00	6.431333e+00	1.118630e+01	1.416627e-02	5.967611e-02	2.459883e-01
min	-1.000000e+00	3.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.573874e-06	2.838686e-11	3.750277e-11	0.000000e+00
25%	2.000000e+01	1.500000e+01	3.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.007930e+01	3.554904e-01	2.384321e-04	7.289061e-04	0.000000e+00
50%	2.500000e+01	1.700000e+01	3.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.380487e+01	6.400902e-01	5.719309e-04	1.783423e-03	0.000000e+00
75%	3.400000e+01	1.900000e+01	3.000000e+00	8.994267e-01	5.883179e-01	0.000000e+00	1.837918e+01	1.093097e+00	1.225048e-03	4.016518e-03	0.000000e+00
max	5.500000e+01	3.800000e+01	1.000000e+01	9.761140e+02	1.155839e+03	1.801540e+03	7.117231e+01	4.435118e+03	4.999874e-01	9.999972e-01	1.000000e+00

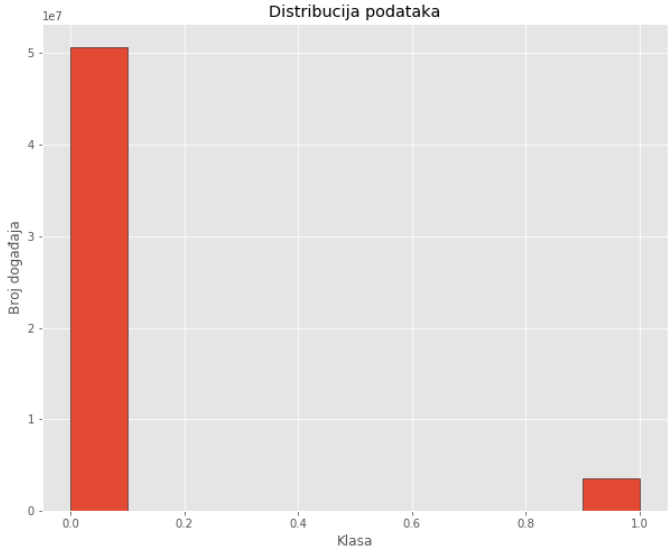
Slika 4.2: Osnovne informacije o skupu podataka.

Znajući da su signalni podaci označeni s 0, a pozadinski s 1 atributom **label**, jednostavno je izračunati zastupljenost ili drugim riječima omjer količine podataka te dvije klase - slika 4.3.

```
1   1 - df["label"].mean()
executed in 150ms, finished 00:06:06 2020-04-23
0.9353042103697016
```

Slika 4.3: Odnos zastupljenosti klasa u podacima.

Grafički prikaz omjera je na slici 4.4.



Slika 4.4: Histogram zastupljenosti objekata dvije klase podataka.

Klasi signalnih objekata pripada **93.5 %** podataka, što znači da su podaci približno u omjeru **14 : 1**. Izgleda da je skup podataka koji planiramo koristiti za treniranje modela neuravnotežen; klase nisu balansirane. Imamo mnogo više uzoraka jedne klase, a to za sobom povlači određene probleme, odnosno mijenja način na koji ćemo pristupiti problemu. Dva ćemo pristupa rješavanju problema neuravnotežene prirode prezentirati i obraditi u nadolazećim poglavljima. Ta dva pristupa su:

- Težinsko treniranje u kojem mijenjamo težine funkciji pogreške, tako da pogreške različitih klasa različito pridonose ukupnim pogreškama (poglavlje 4.4).
- Balansiranje skupa podataka za treniranje da bi dobili uravnotežene klase (poglavlje 4.6).

Neuravnoteženi skup podataka je takav skup u kojem se uzorci jedne klase pojavljuju u malom broju naspram drugih klasa, u ovom slučaju naspram druge

klase. Veoma su česti prilikom proučavanja bolesti ili poremećaja koji su jako rijetko zastupljeni u odnosu na zdravu populaciju ili pri detekciji prevara. Ovakvi skupovi podataka predstavljaju problem pri klasifikaciji zbog sljedećih razloga: (i) izlaz je pristran jer je klasifikator osjetljiviji pri detektiranju većinske klase i (ii) skup podataka je neuravnotežen zbog same prirode problema; manjinska klasa se jednostavno rjeđe događa ili je njene uzorke teže prikupiti u stvarnom svijetu.

Problem koji iz toga može nastati jest da točnost (eng. Accuracy) definirana količnikom točnih predviđanja i ukupnog broja predviđanja, neće ispravno pokazivati stvarnu performansu modela naročito kad postoji rizik uz lažno negativna (eng. False Negative - **FN**) i lažno pozitivna (eng. False Positive - **FP**) predviđanja. Većina algoritama strojnog učenja pružaju najbolje rezultate kad rade na podacima u kojima je broj uzoraka pojedine klase približno jednak iz razloga što su dizajnirani da maksimiziraju točnost i reduciraju grešku.

Sljedeći korak je podijeliti skup podataka na dva dijela: skup za treniranje i skup za testiranje. To ćemo učiniti pomoću "*train_test_split*" funkcije iz biblioteke scikit-learn [17]. Skup za treniranje će zadržati informaciju o labeli. Skup za testiranje neće, već će je u konačnici istrenirani model pokušati predvidjeti. Drugim riječima, model se trenira na klasificiranim podacima, a zatim neviđene podatke nastoji klasificirati. Nakon toga slijedi evaluacija modela kroz metrike odabrane ovisno o tipu problema. Krenimo na prvi algoritam.

Dummy Classifier

Skup podataka sadrži dvije klase: signalne i pozadinske objekte, problem je binarne klasifikacijske prirode, slijedom čega bi pametan potez bio isprobati trivijalan algoritam *Dummy Classifier* iz scikit-learna [17]. Dummy Classifier je tip klasifikatora koji predviđa klasu podataka na temelju najzastupljenije klase. Koristi se samo kao osnovni, temeljni model - onaj s kojim ćemo uspoređivati sve druge modele. Zapravo, pruža nam najnižu prihvatljivu točnost za ovaj skup podataka i početnu točku od koje ćemo pokušati izgraditi uspješniji model od osnovnog. Također, može poslužiti kod problema neuravnoteženih klasa gdje se želi pokazati kako *misleading* najčešće korištena metrika točnost može biti. Strategija "*most frequent*" uvijek predviđa signalnu klasu i tako postavljen klasifikator osigurava točnost od **93.5**% što je i očekivano znajući da je omjer podataka približno $14:1$ u korist signalne klase (pogledaj sliku 4.5).

```
Dummy_Clf = DummyClassifier(strategy="most_frequent").fit(X_train,y_train)
y_pred = Dummy_Clf.predict(X_test)
print('Accuracy score: ', accuracy_score(y_test,y_pred))
print("Precision score: ", precision_score(y_test,y_pred))
print("Recall score: ", recall_score(y_test,y_pred))
print("F1 score: ", f1_score(y_test,y_pred))
print("Confusion matrix:\n ", confusion_matrix(y_test,y_pred))
```

executed in 28.0s, finished 14:56:52 2020-04-30

```
Accuracy score:  0.9353041931066051
Precision score:  0.0
Recall score:  0.0
F1 score:  0.0
Confusion matrix:
  [[12668555        0]
 [  876295        0]]
```

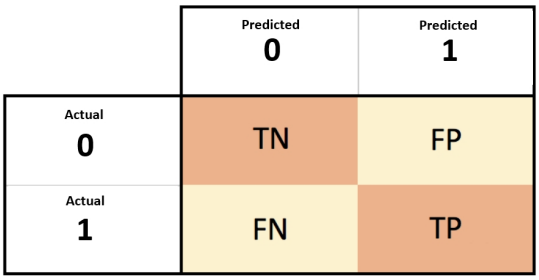
Slika 4.5: Točnost, preciznost, osjetljivost i F1 rezultat Dummy Classifier-a. Također je prikazana i matrica zabune.

Ukoliko uzmemo Dummy Classifier kao temeljni model, uz minimalan trud, ostvarujemo vrlo dobar rezultat - sa stajališta točnosti. Međutim, zatraži li netko od nas predviđanje manjinske klase, točnost će biti mizernih **6.47**%. Zbog toga točnost nije adekvatna metrika za neuravnotežen skup podataka. Osim toga, točnost jednako vrednuje lažno pozitivne FP i lažno negativne FN rezultate što zavisno o tipu problema možda nije poželjno. Očigledno točnost, iako vrlo jednostavna i praktički najintuitivnija metrika, pogrešan je odabir za evaluaciju performansi modela treniranog na neuravnoteženom skupu podataka, stoga ćemo u sljedećem poglavlju uvesti nove metrike koje će nam pružiti bolji uvid i koje neće dati "iskrivljenu" statistiku i lažnu nadu u uspjeh klasifikatora.

4.3 Statističke metode

4.3.1 Matrica zabune, Preciznost, Osjetljivost, F1 rezultat

Možda na prvi pogled jednostavna, međutim, veoma korisna metrika jest matrica zabune (eng. Confusion matrix) koja pruža bolji uvid ne samo u performansu modela, već i koje klase su ispravno predviđene, koje ne i koji tip greške je napravljen (detalji u opisu slike 4.6). Nadalje, omogućuje izračun ostalih metrika opisanih u nastavku. Nama je zapravo potreban jedan broj da bismo mogli optimizirati/učiti, a upravo matrica zabune će nam poslužiti da odredimo koji je to broj. Pogledajmo koje sve metrike možemo izvući iz nje.



Slika 4.6: Matrica zabune - tablica koja prikazuje tipove točnih predviđanja (**T**rue **N**egative i **T**rue **P**ositive) i tipove netočnih (**F**alse **N**egative i **F**alse **P**ositive).

Vidjeli smo na primjeru Dummy Classifiera da detekcija signalne klase nije problem i da fokus treba prebaciti na identifikaciju "pozitivne" (manjinske) klase - pozadine. Također smo zaključili kako točnost nije najbolja metrika evaluacije nebalansiranog skupa podataka, stoga predstavimo metrike koje će nam pružiti bolji uvid. **Osjetljivost** (eng. Recall ili Sensitivity ili True Positive Rate) opisuje sposobnost modela da pronađe sve relevantne podatke - udio ispravno predviđenih pozitivnih (pozadinskih) objekata u ukupnom broju pozitivnih objekata. Prema [15] definirana je sljedećim izrazom:

$$Recall = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives} \tag{4.3.1}$$

Primijetite, ukoliko sve objekte algoritam klasificira kao pozadinu, vrijednost osjetljivosti postaje 1.0. Savršeni klasifikator? Naravno da ne. U svim područjima znanosti, gdje ni strojno učenje nije izuzetak, vrijedi svojevrsno načelo neodređenosti koje u principu kaže da treba balansirati između osjetljivosti i preciznosti i da se ne može zahtijevati istovremeno povećanje i preciznosti i osjetljivosti. Povećanjem osjetljivosti, smanjuje se preciznost definirana (prema [15]) na sljedeći način:

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives} \tag{4.3.2}$$

Dakle, **preciznost** (eng. Precision) je mjera egzaktnosti klasifikatora; koliko je rezultat pouzdan kad model kaže taj objekt pripada toj klasi (koliko je objekata stvarno pozadina za koje je algoritam predvidio da su pozadina).

Specifičnost (eng. Specificity ili True Negative Rate) je, također prema [15] definirana:

$$Specificity = \frac{True\ Negatives}{True\ Negatives\ +\ False\ Positives} \tag{4.3.3}$$

Suočeni s problemima kod kojih je poznato što se želi maksimizirati, smjer rješavanja problema je prilično očit. Isprobati razne modele, modificirati im parametre i hiperparametre ili uzorkovati skup podataka za treniranje dok se ne

postigne zadovoljavajući rezultat željene metrike. Primjera radi, na preliminarnom pregledu bolesnika s mogućom dijagnozom zloćudnog tumora, potrebno je odlučiti koju vrstu greške bi radije napravili: (i) Zdravom pojedincu pogrešno dijagnosticirati tumor ili (ii) bolesnom pojedincu nedijagnosticirati tumor? Cilj će, naravno biti zadržati sve pacijente koji zapravo imaju bolest (želimo postići što je moguće veću osjetljivost), nauštrb smanjenju preciznosti - zadržali smo neke koji nisu bolesni. Međutim, ako se želi ostvariti optimalna ravnoteža ove dvije metrike koristi se **F1 rezultat**. F1 rezultat je harmonijska srednja vrijednost osjetljivosti i preciznosti definirana sljedećom jednadžbom:

$$F_1\ Rezultat = 2 * \frac{Precision * Recall}{Precision\ +\ Recall} \tag{4.3.4}$$

Harmonijska srednja vrijednost se koristi zato što kažnjava ekstremne vrijednosti. Klasifikator s preciznošću 1.0 i osjetljivosti od 0.0 ima "jednostavnu" srednju vrijednost od 0.5, a F1 rezultat od 0.0. Shodno tome, ako želimo naučiti model koji će jednako vrednovati osjetljivost i preciznost, možemo odabrati F1 rezultat kao funkciju koju želimo maksimizirati.

Za odabranu klasu, različite kombinacije osjetljivosti i preciznosti imaju različita značenja:

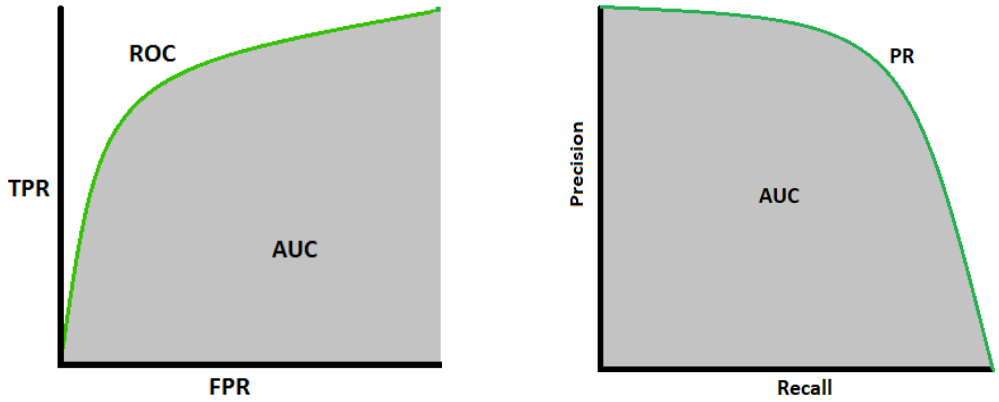
- Visoka osjetljivost i visoka preciznost $\rightarrow$ Model vrlo dobro upravlja/rukovodi klasom.
- Visoka osjetljivost + niska preciznost $\rightarrow$ Klasa je dobro identificirana, ali podaci sadrže i objekte druge klase u sebi.
- Niska osjetljivost + visoka preciznost $\rightarrow$ Klasa nije dobro identificirana, ali u slučajevima kad jest vrlo smo sigurni/pouzdani.
- Niska osjetljivost + niska preciznost $\rightarrow$ Loša performansa modela.

Vidjeli smo koje se sve metrike mogu iz matrice zabune izračunati. Zadovoljavanje dvaju kriterija: TP i FP (čim više TP, a čim manje FP) ili njihovih relativnih vrijednosti TPR i FPR (jednadžba 4.3.5) grafički možemo prikazati **Receiver Operating Characteristics - ROC** krivuljom. Metrike koje je sačinjavaju definirane su elementima matrice zabune. Međutim, ukoliko nas zanima kako postići što manje FN rezultata, bolji odabir bit će **Precision-Recall**(**PR**) metrika jer ne koristi TN rezultate za izračun. Detaljnije u nastavku.

4.3.2 Pregled ROC i PR krivulja

ROC krivulja jedna je od najvažnijih evaluacijskih metrika i tehnika vizualizacije performansi klasifikacijskih modela. ROC krivulja, kao krivulja vjerojatnosti,

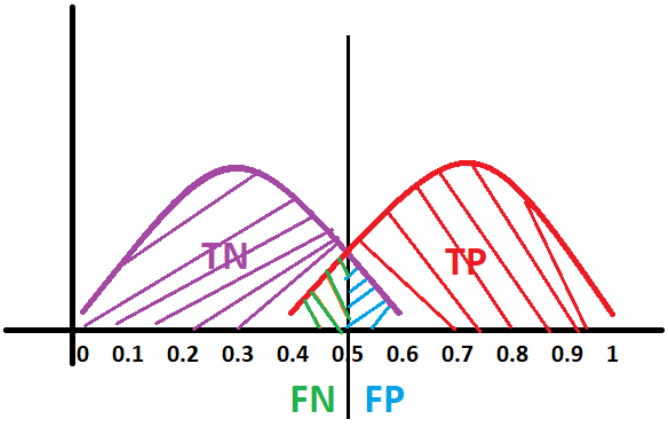
sumira kompromis TPR i FPR odnosno prikazuje kako se njihov odnos mijenja variranjem praga vjerojatnosti. Prag vjerojatnosti (eng. threshold) je granična vrijednost iznad koje će podaci biti klasificirani kao da pripadaju pozitivnoj klasi/jedinicama, a ispod koje negativnoj odnosno nulama. ROC krivulja prikazuje **TPR** na y-osi, a **FPR** na x-osi kao funkciju praga vjerojatnosti modela klasifikacije pozitivne klase - slika 4.7a. Jednostavnije rečeno, ROC kaže koliko dobro model može razlikovati dvije klase (signal/pozadina).



(a) Skica ROC krivulje **(b)** Skica PR krivulje

Slika 4.7: Skica ROC i PR krivulja.

Kako je ROC krivulja vjerojatnosti, pametno je pogledati distribucije tih vjerojatnosti na slici 4.8:



Slika 4.8: Ljubičastom bojom označena je krivulja distribucije negativne klase (signal), a crvenom pozitivne (pozadina). Vertikalna linija na vrijednosti apscise 0.5 je prag vjerojatnosti. Sve desno od linije (iznad praga) bit će razvrstano kao Positive, samo TP uzorci biti će uistinu Positive, zato ih nazivamo True Positive. Svi negativni uzorci, koji su zbog tako odabranog praga svrstani kao Positive nazivamo False Positive. Sve lijevo od linije biti će razvrstano kao Negative, samo uzorci koji su doista Negative biti će True Negative, dok pozitivne uzorke svrstane kao Negative nazivamo False Negative.

Osjetljivost i specifičnost inverzno su proporcionalne metrike. Snižavanjem praga vjerojatnosti, model predviđa više pozitivnih rezultata (više TP, ali i više FP) pa prema tome osjetljivost raste. Međutim, specifičnost se smanjuje što se na grafu 4.7a vidi kao pomicanje desno i gore duž krivulje. Slično, povećanjem praga vjerojatnosti predviđa se više negativnih rezultata i pritom raste specifičnost, a smanjuje se osjetljivost. *False Positive Rate* (FPR) prema [15] definiran je sljedećim izrazom:

$$FPR = 1 - Specificity = \frac{False\ Positives}{True\ Negatives\ +\ False\ Positives} \tag{4.3.5}$$

Dakle, povećanjem TPR, FPR također raste. Vidimo da sad promatramo samo pozitivne rezultate.

Ostali smo dužni opisati spomenutu PR metriku, metriku koja je prikladnija neuravnoteženim podacima od ROC metrike, naročito u slučaju kad pozitivne klase imamo u manjini jer je ROC neosjetljiva na neuravnoteženost klasa. Uvodimo je jer se i osjetljivost (4.3.1) i preciznost (4.3.2) fokusiraju na pozitivnu klasu (manjinsku), što je i poželjno jer ukoliko imamo veliki broj uzoraka klase 0 (signala) manje smo zainteresirani za predviđanje te klase. Ključ u preciznosti i osjetljivosti jest nekorištenje *True Negatives* rezultata za njihov izračun. Zanimaju nas samo točna predviđanja manjinske klase, klase 1. PR krivulja prikazuje **Preciznost** na y-osi, a **Osjetljivost** na x-osi kao na slici 4.7b. Idealni model ima Osjetljivost 1 i Preciznost 1, dakle, nalazi se u točci (1,1). Model "bez tehnike" je onaj koji ne može razlikovati klase i dobije se kad algoritmu Dummy Classifier u argumentu "strategy" postavimo vrijednost "stratified" koji mu kaže da predviđanja radi na osnovi zastupljenosti klasa u skupu podataka. Njegov prikaz bit će vodoravna linija sa preciznošću proporcionalno broju pozitivnih uzoraka u skupu podataka. Kada dakle koristiti ROC, a kad PR Curve? Jesse Davis [18] kaže:

> However, ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew in the class distribution. [...] Precision-Recall (PR) curves, often used in Information Retrieval , have been cited as an alternative to ROC curves for tasks with a large skew in the class distribution.

Iako su ROC i PR krivulje korisni dijagnostički alati, nekad može biti zahtjevno usporediti dva ili više klasifikatora samo na osnovi njihovih krivulja. Umjesto promatranja krivulje, može se izračunati površina ispod krivulje koja će za model dati jedan rezultat za sve vrijednosti praga vjerojatnosti. Naziva se **AUC (Area Under Curve)**, pruža pak mjeru separabilnosti i kvantificira ROC i PR krivulje jednim jedinim brojem. Što je viši AUC, to model bolje predviđa nule kao nule, a jedinice kao jedinice. AUC blizu 1.0 znači da model ima odličnu mjeru

separabilnosti. AUC vrlo blizu 0.5 znači da model nema sposobnost diskriminacije klasa, dok AUC od približno 0 znači da model predviđa negativnu klasu kao pozitivnu i obrnuto. Dakle, AUC će nam reći koji model nam može dati više TP rezultata uz manje FP.

Sad kad smo upoznati s drugim metrikama koje bolje odgovaraju našem tipu problema, diskutirajmo koju metriku koju želimo maksimizirati. Možda ste već naslutili da je to F1 rezultat jer jednako vrednuje osjetljivost i preciznost. No, u kojoj mjeri želimo postići balans, želimo li ipak nešto veću osjetljivost od preciznosti? Odgovor na ta pitanja i njima slična, znat ćemo tek kad dobijemo stvarne (eksperimentalne) podatke i opis zadatka/problema koji trebamo riješiti. Ukoliko imamo veliki broj miona (kao što je sad slučaj) i želimo imati čišći signal, nije nam problem pogrešno odbaciti poneki mion da bismo što više pozadine eliminirali (veća osjetljivost i manja preciznost). Ukoliko tragamo za novom česticom, želimo imati što je više podataka moguće, tj. signala, što uzrokuje i više pozadine. Dakle, sve ovisi o tipu analize za koju nam treba identifikacija objekata. Kad budemo znali što nam treba, jednostavno ćemo na PR krivulji odrediti koji prag vjerojatnosti uzeti i kakav omjer preciznosti i osjetljivosti želimo.

Posvetimo se sad rješavanju problema. Prvi pristup koji smo rekli isprobati jest težinsko treniranje u kojem ćemo mijenjati težine funkciji pogreške da bi pogreške različitih klasa različito pridonosile ukupnim pogreškama. Neki algoritmi će se sami pobrinuti za to, a nekima ćemo morati eksplicitno zadati.

4.4 Težinsko treniranje

Krenimo s najjednostavnijim algoritmom, logističkom regresijom, statističkim modelom korištenim za modeliranje vjerojatnosti pojedine klase. Iako relativno jednostavan algoritam preciznosti mu ne manjka kao što ćemo i vidjeti.

4.4.1 Logistička regresija

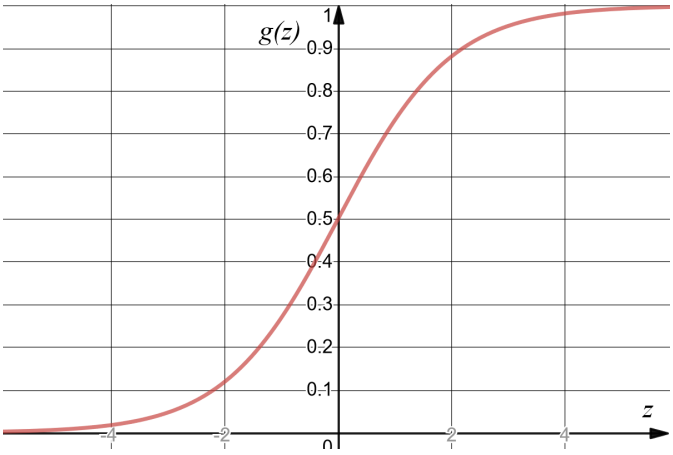
Mogli smo klasifikacijskom problemu pristupiti s linearnom regresijom, ali nema smisla da $h_\theta(x)$ može poprimiti vrijednosti veće od 1 ili manje od 0 kad znamo da $y \in \{0, 1\}$, stoga ćemo promijeniti formu naše hipoteze $h_\theta(x)$:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{4.4.1}$$

gdje se

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4.4.2}$$

naziva logistička funkcija ili sigmoidalna funkcija. Na slici 4.9 je graf $g(z)$:



Slika 4.9: Sigmoidalna funkcija.

Primijetite da $g(z)$ teži u 1 za $z \to \infty$, a $g(z)$ teži u 0 za $z \to -\infty$. Štoviše, $g(z)$ pa tako i $h(x)$, je uvijek ograničena između 0 i 1.

Po završetku treninga, netom istrenirani model poslužit će za predviđanje klasa gdje se dobivene predviđene vrijednosti pomoću sigmoidalne funkcije preslikavaju na skalu vjerojatnosti od 0 do 1 formulom 4.4.1. Da bi ih se preslikalo u diskretne klase (signal i pozadina u ovom slučaju), odabire se prag vjerojatnosti iznad kojeg će objekt biti klasificiran kao pozadina (pripada klasi 1), a ispod kojeg pripada signalnoj klasi (klasi 0). Odabrani prag može biti kao u formuli 4.4.3:

$$\begin{aligned} p >= 0.5,\ klasa = 1, \\ p < 0.5,\ klasa = 0 \end{aligned} \tag{4.4.3}$$

Kako se vjerojatnost približava jedinici, model je sve sigurniji da objekt doista pripada klasi 1 to jest pozadinskom procesu. Pogledajmo primjer kako odrediti kojoj klasi pripada objekt:

$$z = \theta_0 + \theta_1 mu\_sip + \theta_2 mu\_pf\_photon\_iso + ... \tag{4.4.4}$$

$$P(klasa = 1) = \frac{1}{1 + e^{-z}} \tag{4.4.5}$$

Zamislimo da nam je sigmoidalna funkcija dala rezultat od 0.7. Koja bi interpretacija tog rezultata bila? Može se reći, da je vjerojatnost 70% da je taj objekt pozadina i ukoliko je granica odluke postavljena na 0.5, objekt bi bio klasificiran kao pozadina (klasa 1).

Još preostaje prilagoditi funkciju pogreške za klasifikacijski problem. Ona je prema [15] definirana:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \tag{4.4.6}$$

$$Cost(h_\theta(x), y) = -log(h_\theta(x)) \text{ if } y = 1 \tag{4.4.7}$$

$$Cost(h_\theta(x), y) = -log(1 - h_\theta(x)) \text{ if } y = 0 \tag{4.4.8}$$

što u konačnici daje:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \tag{4.4.9}$$

Kako minimizirati funkciju pogreške za klasifikacijski problem? Na isti način kao i kod linearne regresije - gradijentalni spust. Nažalost, ne postoji jednadžba koja eksplicitno računa vrijednosti θ koje bi minimizirale grešku (ne postoji ekvivalent normalne jednadžbe za klasifikaciju). Međutim, funkcija pogreške je konveksna, stoga će gradijentalni spust (ili neki drugi optimizacijski algoritam) uvijek pronaći globalni minimum tvrdi [15].

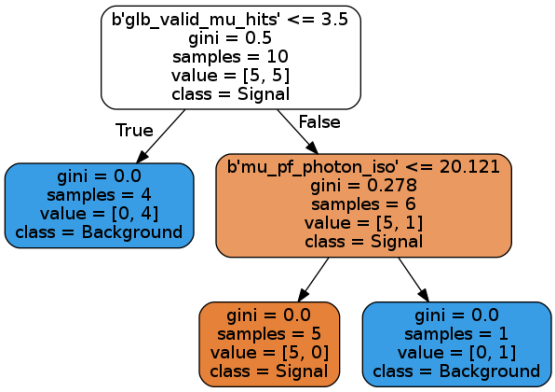
Opće poznato je da u većini slučajeva, postavi li se kompleksno pitanje većoj nasumičnoj grupi ljudi, kumulativni odgovor biti će točniji nego li odgovor jednog eksperta. Slično se može primijeniti na zbir prediktora. Tehnike osnaživanja (eng. ensemble learning) u strojnom učenju kombiniraju više modela u jedan optimalni prediktivni model da bi se smanjila varijanca, pristranosti ili unaprijedila sveukupna performansa predikcije. S tim na umu, predstaviti ćemo modele temeljene na stablima odluke.

4.4.2 Stablo odluke

Vjerojatno najpopularniji model temeljen na stablima odluke (eng. Tree Based) je Random Forest, ali za njegovo bolje razumijevanje prvo moramo pojasniti što su stabla odluke (eng. Decision Tree). Stablo odluke je poprilično intuitivan model kojeg je najlakše zamisliti kao seriju da/ne pitanja o podacima koja na kraju dovode do predviđene klase (do kontinuirane vrijednosti ukoliko se radi o regresijskom problemu). Dakle, klasifikaciju radi na isti način kao i ljudi: postavlja niz upita o podacima dok ne dođe do odluke. Tehnički detalji stabla odluke kriju se iza konstruiranja pitanja čiji odgovori vode do najvećeg smanjenja Gini

indeksa nečistoće (eng. Gini Impurity index) (jednadžba 4.4.10). Pojasnimo. To znači da stablo odluke nastoji formirati čvor koji sadrži velik broj uzoraka jedne klase pronalazeći vrijednosti u značajkama koje dobro razdvajaju podatke u klase. Čvor nazivamo svaku točku grananja (svaku točku odluke). Algoritam se zapravo na svakom čvoru pita sljedeće pitanje: *Koja značajka će mi omogućiti da dane podatke razdvojim na dvije grupe koje su što je moguće više različite i da su elementi rezultirajuće podgrupe što je moguće više slični jedno drugom?*

Primjera radi, na slici 4.10 odabrano je 10 (5 signalnih i 5 pozadinskih) objekata iz dostupnog skupa podataka i kreirano prvo stablo odluke:



Slika 4.10: Primjer dijela stabla odluke.

Na koji način stablo odluke sa slike 4.10 predviđa klasu? Za početak definirajmo elemente stabla odluke:

- **Ishodišni** ili temeljni čvor (eng. root node) je korijen stabla.
- **List** čvor (eng. leaf node) je čvor koji nema (čvorova) djece i ne pita više pitanja.
- **Dijete** je onaj čvor koji ima barem jedan čvor nivo iznad sebe u stablu i s kojim je povezan odgovarajućom granom.
- **Roditelj** je čvor koji ima barem jedan čvor povezan granom u nivou ispod sebe.

Sad možemo popratiti na koji način stablo odluke sa slike 4.10 predviđa klasu. Objekt je detektiran i treba ga klasificirati. Kreće se od ishodišnog čvora dubine 0, na samom vrhu: ovaj čvor postavlja pitanje je li vrijednost *glb_valid_mu_hits* manja ili jednaka 3.5. Ako je, pomičemo se dolje lijevo u čvor dijete na dubini 1, lijevo. U ovom slučaju, ovo je list i ne pita više pitanja: stablo odluke za taj čvor predviđa klasu *Background*. Neka je sada detektiran još jedan objekt, ovaj put

je vrijednost *glb_valid_mu_hits* veća od 3.5. Pomičemo se dolje desno do dijete čvora, koji nije list stoga se postavlja još jedno pitanje - je li *mu_pf_photon_iso* manji ili jednak 20.121? Ukoliko je, taj objekt je Signal (dubina 2, lijevo). Ako nije, onda je Background (dubina 2, desno). Veoma jednostavno.

Svi čvorovi osim listova sadrže 5 elemenata:

1. Pitanje o podacima zasnovano na vrijednostima značajke. Svako pitanje ima ili TOČNO ili NETOČNO odgovor, na osnovi kojeg se pomiče niz stablo.
2. **gini**: Gini nečistoća čvora koja se računa prema sljedećoj formuli:

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2, \tag{4.4.10}$$

   gdje je $p_{i,k}$ omjer instanci klase *k* u trening instancama i-tog čvora.
3. **samples**: Broj uzoraka u čvoru.
4. **value**: Broj uzoraka u svakoj klasi.
5. **class**: Većinska klasa. U slučaju lista, ovo je predviđanje za sve uzorke u čvoru.

Listovi nemaju pitanja jer je to mjesto gdje se rade konačna predviđanja. Kako se računa Gini nečistoća? Čvor je čist (gini=0) ako sve instance treninga na tom čvoru pripadaju istoj klasi. Primjer je čvor dubine 1 - lijevo, gdje sva četiri objekta pripadaju Background klasi pa je gini indeks = 0. Jednadžba 4.4.10 pokazuje kako trening algoritam računa gini indeks G_i i-tog čvora. Primjer: dubina 1 - desno, ima gini indeks jednak $1 - (5/6)^2 - (1/6)^2 \approx 0.278$.

Na slici 4.10 težinska Gini nečistoća zadnjeg sloja je nula što znači da je svaki čvor potpuno čist i ne postoji šansa da je objekt pogrešno klasificiran. Iako se to čini kao pozitivna stvar, to može dovesti do pretreniranja (eng. overfitting) jer čvorovima, tj. stablu nije ograničena maksimalna dubina (broj nivoa). Cilj strojnog učenja je stvoriti model koji radi zadovoljavajuće dobro na podacima koje nikad nije vidio. Pretreniranje se javlja kad se istrenira jako fleksibilan model, koji ima mnogo slobodnih parametara i koji nauči i šum(pozadinu) koji je prisutan, nauštrb nemogućnosti generalizacije na novim podacima. Fleksibilan model će stoga imati visoku varijancu. S druge strane, nefleksibilan model imati će veliku pristranost, pretpostaviti će jednostavnije veze među podacima nego stvarno jesu. Ukoliko se modelu ne ograniči maksimalna dubina, beskonačno je fleksibilan; savršeno razvrstava zadani problem i stablo može rasti sve dok se ne dobije po jedan list za svaki objekt.

Kao alternativan pristup postavljanju granice na maksimalnu dubinu stabla, reducirajući time varijancu (poželjno), ali povećavajući pristranost (nepoželjno), može se kombinirati više stabala odluke u ansambl - grupu prediktora što je poznatije kao Random Forest algoritam.

4.4.3 Random Forest

Random Forest algoritam je ansambl stabala odluke treniranih *bagging* (*bootstrap aggregating*) metodom. To je metoda treninga više individualnih prediktora, istim trening algoritmom, na različitim nasumičnim podskupima podataka za treniranje. Ti podskupi generirani su izvlačenjem uzoraka s ponavljanjem (eng. boostraping). Jednom kad su svi prediktori istrenirani, ansambl može napraviti predviđanje jednostavno sakupljajući predviđanja svih prediktora. Dodatna nasumičnost uvedena je traženjem najbolje značajke za račvanje(na tom čvoru) unutar nasumičnog podskupa značajki, a ne unutar potpunog skupa značajki kao što je slučaj kod stabla odluke. Time Random Forest rezultira većom raznolikosti stabala - manjom međusobnom korelacijom. Bitno je naglasiti da Random Forest trenira svako stablo nezavisno čime je model manje podložan pretreniranju. Dakle, stabla nisu samo trenirana na različitim podskupima podataka, nego i koriste različite značajke za donošenje odluke.

4.4.4 LightGBM

LightGBM je *gradient boosting framework* koji za učenje koristi algoritme temeljene na stablima. Neke od prednosti prema službenoj dokumentaciji ([19]) su: veća brzina treniranja, bolja efikasnost i manja upotreba memorije što se postiže grupiranjem vrijednosti kontinuiranih značajki u diskretne binove korištenjem *histogram based* algoritma, ali i ponovnom upotrebom binova kroz više iteracija. Također, veoma bitno, LightGBM gradi stabla ***leafwise*** a ne *levelwise* odnosno *depthwise* kako to rade ostali algoritmi temeljeni na stablima.

4.4.5 XGBoost - eXtreme Gradient Boosting

XGBoost algoritam, poput Random Forest algoritma, koristi ansambl tehniku učenja, predviđanje radi kombinirajući rezultate više individualnih prediktora. Međutim, za razliku od Random Forest algoritma, XGBoost sekvencijalno dodaje prediktore u ansambl, gdje svaki novi ispravlja greške prethodnog. Mogućnost pretreniranja kontrolira se odgovarajućim parametrom (`max_depth` - maksimalna dubina stabla).

XGBoost također ima sposobnost korištenja algoritma temeljenog na histogramu,

koji grupira kontinuirane značajke u diskretne binove, što se postiže jednostavno postavljajući parametar `tree_method` na `hist`. Ovaj način izgradnje stabla dozvoljava nam kontrolu nad načinom dodavanja novih čvorova u stablo:

- `grow_policy = depthwise` (zadano): dijeljenje na čvorovima bliže samom korijenu ili drugim riječima, izgradnja listova završiti će se na istom nivou.
- `grow_policy = lossguide`: Preferira se dijeljenje stabla na čvorovima s najvećom deltom gubitka (eng. highest loss change). Ovo je ***leafwise*** metoda izgradnje stabla i s ovim XGBoost imitira ponašanje LightGBM-a.

Grow_policy zahtijeva definiranost još dva parametra:

- `max_leaves`: Maksimalan broj čvorova koji se mogu dodati. Relevantno samo za `lossguide`.
- `max_depth`: maksimalna dubina. Ponaša se uobičajeno ako je `grow_policy` postavljen na `depthwise`. Ako je `grow_policy` `lossguide`, `max_depth` se može postaviti na nulu čime se postiže neograničenost dubine.

Sljedeće postavke za XGBoost i LightGBM smo koristili:

- XGBoost (hist+depthwise): `tree_method=hist, grow_policy=depthwise, max_depth=8`
- XGBoost (hist+lossguide): `tree_method=hist, grow_policy=lossguide, max_depth=0, max_leaves=255`
- LightGBM: `max_depth=0, num_leaves=255`

Parametar `num_leaves` LightGBM modela istovjetan je parametru `max_leaves` XGBoosta. Glavni parametar za kontrolu kompleksnosti modela baziranog na stablu odluke. Teoretski, postavljanjem `num_leaves = 2^(max_depth) - 1` možemo osigurati isti broj listova kao u *depthwise* stablu. Međutim, stvarnost je drukčija. *Leafwise* stablo je tipično mnogo dublje od *depthwise* stabla za fiksan broj listova. Neograničena dubina može dovesti do pretreniranja na manjim skupovima podataka, tvrdi [19].

Da ne pišemo cijele nazive modela, uvesti ćemo sljedeće kratice: Logistic Regression (LogReg), Random Forest (RF), Decision Tree (DT), XGBoost (XGB), LightGBM (LGBM). Slovo B ili W ispred imena modela znači da smo modelu modificirali parametar kojim regulira težine pridružene određenoj klasi. Istrenirajmo netom objašnjene modele, testirajmo ih na test skupu podataka i pri-

kažimo rezultate. Primjerak koda kojeg sam napisao za treniranje, evaulaciju i usporedbu modela strojnog učenja dostupan je u dodatku A.2.

4.5 Rezultati težinskog treniranja

U tablicama 4.1, 4.2, 4.3 i na slici 4.11 nalaze se rezultati modela kojima nismo modificirali parametre zbog neuravnoteženosti skupa podataka. Tu specifično mislim na parametre koji se tiču težine klasa, što znači da je svim klasa pridjeljena vrijednost težine jedan. Algoritmi prema inicijalnim postavkama odabiru vrijednost praga vjerojatnosti od 0.5. Njihovi rezultati prikazani su u tablici 4.1.

Model	Precision	Recall	F1-Score	Time[min]
LogReg	0.8920	0.6183	0.7303	24
DT	0.8002	0.8116	0.8058	52
RF	0.9205	0.8143	0.8642	42
XGB[1]	**0.9222**	0.8285	0.8728	12
XGB[2]	0.9206	0.8336	0.8749	10
LGBM	0.9171	**0.8375**	**0.8755**	13

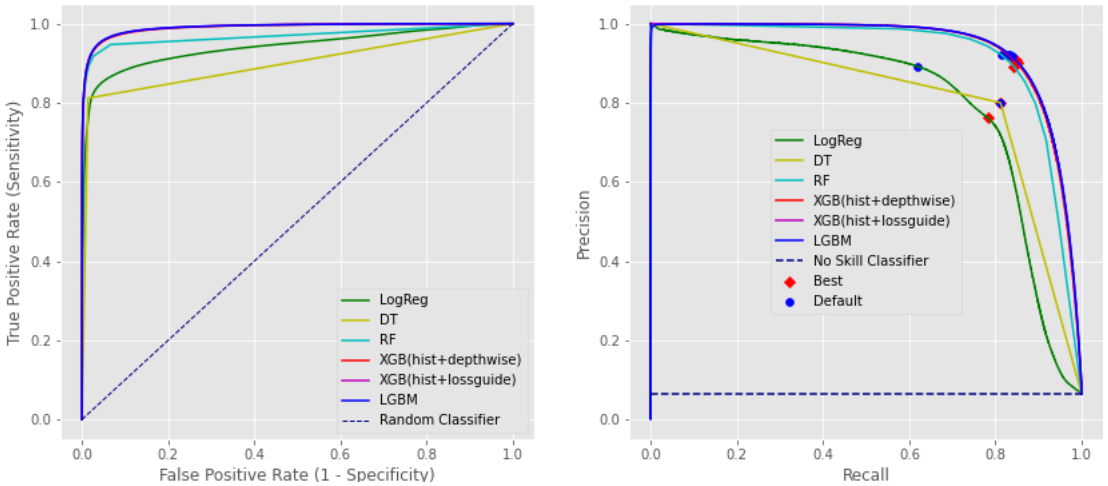
[1] *hist+depthwise*
[2] *hist+lossguide*

Tablica 4.1: Usporedba performansi ML modela s pragom vjerojatnosti od 0.5 bez uključenog težinskog parametra. Zadnji stupac sadrži vremena treninga modela.

Možemo za početak komentirati rezultate svima poznate Logističke regresije. Osjetljivost od 61.83% je niska, što znači da postoji velik broj *False Negative* rezultata - pozadinskih objekata koji su pogrešno identificirani kao signalni. Preciznost je zadovoljavajuća, 89.2%. Već smo pojasnili da visoka preciznost i niska osjetljivost karakteriziraju model koji nije uspješno identificirao pozadinu, ali u slučajevima kad jest vrlo smo sigurni u taj rezultat. Kod nebalansiranog skupa podataka, poželjnije bi bilo povećati osjetljivost nauštrb preciznosti jer nije "velika" pogreška odbaciti poneki signal (mion - FP) jer ih ionako ima 14 puta više nego pozadine. Veći problem je ako se neki pozadinski objekt proglasi signalom (mionom - FN). Kako povećati osjetljivost klasifikatora? U nadolazećem tekstu ćemo vidjeti da postoje dva načina.

Promotrimo ROC i PR krivulje na slici 4.11 i primijetimo da prag vjerojatnosti od 0.5 nije optimalna vrijednost (plavi krug na slici 4.11b), vrijednost s kojom se postiže ravnoteža preciznosti i osjetljivosti. Pomicanjem praga vjerojatnosti

može se postići istodobno povećanje i preciznosti i osjetljivosti. Prisjetimo se F1 metrike koja nam daje njihovu harmonijsku srednju vrijednost. Odaberemo li graničnu vrijednost koja maksimizira F1 rezultat, možemo postići željenu ravnotežu. Ta granična vrijednost označena je crvenim dijamantom na slici 4.11b.



(a) ROC krivulja **(b)** PR krivulja

Slika 4.11: ROC i PR krivulje ML modela bez uključenog težinskog parametra. Plavi krug označava prag vjerojatnosti od 0.5, a crveni dijamant prag koji daje optimalnu F1 mjeru.

Neće svi modeli imati isti prag vjerojatnosti koji će maksimizirati F1 rezultat, stoga ćemo ga navesti u tablici 4.2 zajedno s ostalim metrikama izračunatim primjenom dobivenoga praga.

Model	Precision	Recall	F1-Score	Threshold
LogReg	0.7613	0.7824	0.7717	0.2019
DT	0.8002	0.8116	0.8058	1.0
RF	0.8920	0.8426	0.8666	0.4
XGB[1]	0.9066	0.8441	0.8742	0.4347
XGB[2]	**0.9073**	0.8467	0.8759	0.4424
LGBM	0.9024	**0.8514**	**0.8761**	0.4377

[1] *hist+depthwise*
[2] *hist+lossguide*

Tablica 4.2: Usporedba performansi ML modela bez uključenog težinskog parametra. Zadnji stupac prikazuje vrijednost praga vjerojatnosti koja daje maksimum F1 rezultata za pojedini model.

Pogledajmo sad rezultat Logističke regresije u tablici 4.2. Snižavanjem praga vjerojatnosti s 0.5 na 0.219 dobili smo povećanje osjetljivosti na 78.24% ali na-uštrb preciznosti koja se smanjila na 76.13%. Međutim, sveukupno F1 rezultat se povećao s 0.7303% na 0.7717%. Sa stajališta F1 rezultata možemo reći da nam je drugi model uspješniji. Iz samog grafa teško je zaključiti koji model ima najbolje performanse. U tome će nam pomoći ROC i PR AUC tablica 4.3:

Model	ROC AUC	PR AUC
LogReg	0.9407	0.8044
DT	0.8988	0.8120
RF	0.9686	0.9219
XGB[1]	0.9897	0.9388
XGB[2]	0.9902	0.9406
LGBM	**0.9904**	**0.9407**

[1] *hist+depthwise*
[2] *hist+lossguide*

Tablica 4.3: ROC i PR AUC tablica ML modela bez uključenog težinskog parametra.

Najveći ROC i PR AUC postiže LightGBM algoritam uz zanemarivu prednost nad XGBoost(hist+lossguide) modelom. Također osigurava najveću F1 mjeru, međutim, potrebno mu je ∼30% više vremena za trening. Neznatno lošije rezultate pružaju XGBoost(hist+depthwise) i Random Forest modeli. Logistička regresija i Decision Tree ipak slabije klasificiraju objekte.

Ponovimo, u prethodnim tablicama nalaze se rezultati modela kojima nismo modificirali parametre zbog neuravnoteženosti skupa podataka. Pokušajmo sad primijeniti težinsko treniranje na način da modelima modificiramo parametar odgovoran za težine klasa. Pogledajmo kako to odraditi za svaki model jer različiti modeli mogu imati različite nazive parametara i različite načine balansiranja:

- Balanced Logistic Regression (BLogReg) : `class_weight = "balanced"`
- Weighted Random Forest (WRF): `class_label : weight`
- Balanced Random Forest (BRF): `class_weight = "balanced"`, model nasumično poduzorkuje svaki *bootstrap* uzorak da bi ga balansirao.
- Balanced XGBoost (BXGB): `scale_pos_weight = weight`
- Balanced LightGBM (BLGBM): `is_unbalance = True`

Želimo li eksplicitno težine pridružiti klasama, to ćemo napraviti u formi rječnika {class_label : weight}. Težinu ćemo izračunati na sljedeći način:

$$weight = \frac{count(Negativni\ uzorci)}{count(Pozitivni\ uzorci)} \tag{4.5.1}$$

`Balanced` način pomoću vrijednosti y automatski prilagođava težine obrnuto proporcionalno frekvenciji klase zadanih ulaznih podataka na sljedeći način:

$$\frac{n\_samples}{n\_classes \cdot np.bincount(y)} \tag{4.5.2}$$

Opcija `balanced_subsample` ista je kao i `balanced` osim što se težine računaju na *boostrap* uzorku prilikom kreiranja svakog stabla.

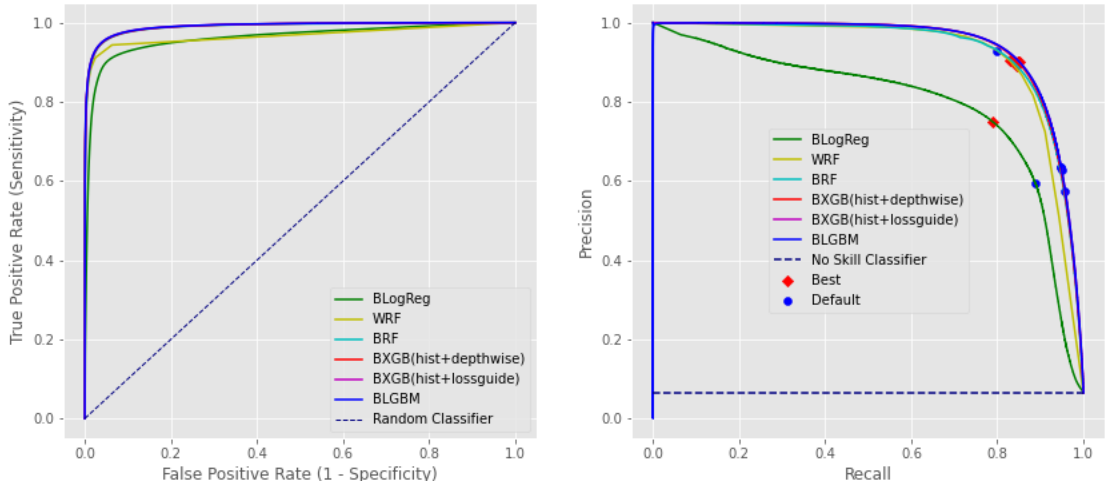
U tablici 4.4 nalaze se rezultati modela s uključenim težinskim treniranjem uz prag vjerojatnosti od 0.5.

Model	Precision	Recall	F1-Score	Time[min]
BLogReg	0.5944	0.8883	0.7122	20
WRF	0.9296	0.7997	**0.8598**	36
BRF	0.5730	0.9550	0.7162	41
BXGB[1]	0.6361	0.9461	0.7608	10
BXGB[2]	0.6348	0.9482	0.7605	11
BLGBM	0.6291	0.9493	0.7567	12

[1] *hist+depthwise*
[2] *hist+lossguide*

Tablica 4.4: Usporedba performansi ML modela s uključenim težinskim parametrom i s pragom vjerojatnosti od 0.5. Zadnji stupac sadrži vremena treninga modela.

Pogledajmo i njihove ROC i PR krivulje na slici 4.12. Sličnu situaciju smo već imali priliku vidjeti. Prag vjerojatnosti od 0.5 nije najbolji izbor ako želimo izvući najviše vrijednosti preciznosti i osjetljivosti odnosno maksimum F1 mjere. Međutim, prisjetimo se što smo željeli postići uključivanjem težinskog parametra? Namjera nam je bila penalizirati pogrešnu klasifikaciju manjinske klase i tim postupkom smanjiti broj FN rezultata - povećati osjetljivost klasifikatora što smo i postigli.



(a) ROC krivulja **(b)** PR krivulja

Slika 4.12: ROC i PR krivulje ML modela s uključenim težinskim parametrom. Plavi krug označava prag vjerojatnosti od 0.5, a crveni dijamant graničnu vrijednost s optimalnom F1 mjerom.

Promotrimo ROC i PR AUC tablicu 4.5 prije nego komentiramo sveukupne rezultate:

Model	ROC AUC	PR AUC
BLogReg	0.9612	0.7975
WRF	0.9668	0.9203
BRF	0.9893	0.9327
BXGB[1]	0.9901	0.9388
BXGB[2]	**0.9905**	**0.9402**
BLGBM	0.9905	0.9400

[1] *hist+depthwise*
[2] *hist+lossguide*

Tablica 4.5: ROC i PR AUC tablica ML modela s uključenim težinskim parametrom.

Balansirani XGBoost(hist+lossguide), model koji uz korištenje algoritma temeljenog na histogramu, gradi stablo *leafwise* postiže najveći ROC i PR AUC rezultat uz zanemarivu prednost nad balansiranim LightGBM modelom. Informacija o tom rezultatu nam je vrlo bitna jer omjer preciznosti i osjetljivost, tj. F1 mjere vrlo lako podesimo odabirom praga vjerojatnosti kad ustanovimo koji model ima najveću površinu ispod ROC ili PR krivulje.

Što zapravo balansirani modeli naprave? Možda će biti lakše objasniti uz matrice zabune prije uvođenja parametra `scale_poss_weight` (tablica 4.6):

		Predviđeno modelom	
		Signal	Pozadina
Stvarna klasa	Signal	12605534	63021
	Pozadina	145859	730436

Tablica 4.6: Matrica zabune XGBoost modela.

i poslije (tablica 4.7):

		Predviđeno modelom	
		Signal	Pozadina
Stvarna klasa	Signal	12190600	477955
	Pozadina	45349	830946

Tablica 4.7: Matrica zabune Balanced XGBoost modela.

Balanced XGBoost model smo, izričito mu zadajući težine klasa, napravili osjetljivijim na određenu pogrešku što znači da neće kažnjavati jednako pogrešnu klasifikaciju objekta iz negativne kao i iz pozitivne klase; penalizirati će pogrešnu klasifikaciju manjinske klase i time smanjiti broj FN rezultata nauštrb povećanju FP. Tim postupkom se povećava osjetljivost klasifikatora, ali smanjuje preciznost. Porast FP rezultata uzrokuje više "odbačenih" miona; miona za koje tvrdimo da su pozadina, no to nam ne predstavlja veliki problem jer svakako imamo velik broj signalnih podataka.

4.6 Balansiranje skupa podataka metodama uzorkovanja

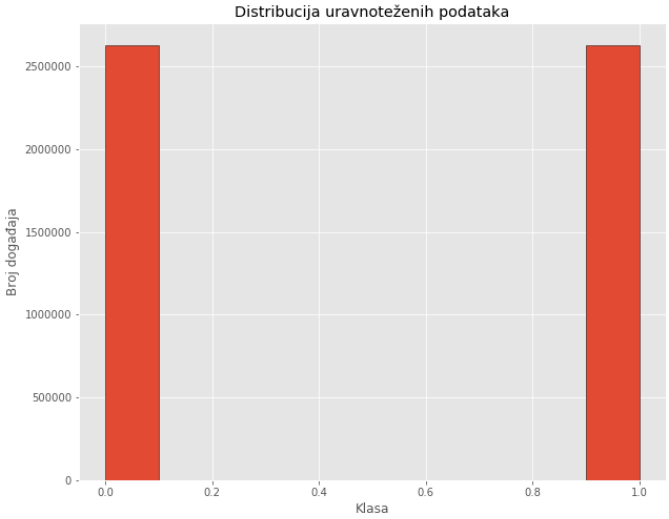
Problem neuravnoteženih klasa vrlo često može rezultirati velikom pristranošću prema većinskoj klasi što smanjuje performanse modela i povećava broj *false negative* rezultata. Drugi pristup koji je već spomenut, a nije još obrađen jest balansiranje ulaznog skupa podataka da bi se dobile uravnotežene klase metodama uzorkovanja (eng. resampling). Tri tehnike balansiranja smo obradili:

- poduzorkovanje većinske klase (eng. undersampling)
- naduzorkovanje manjinske klase (eng. oversampling)
- kombinacija pod i naduzorkovanja.

Metode balansiranja mijenjaju kompoziciju neuravnoteženog skupa podataka dodajući ili oduzimajući primjere iz skupa za treniranje. Najprije ćemo dati kratki uvid u različite metode, a zatim ih, u sljedećem poglavlju primijeniti na našem skupu podataka te prikazati i analizirati dobivene rezultate. Nakon balansiranja, na novom skupu podataka primijeniti ćemo XGBoost(hist+lossguide) algoritam za učenje s postavkama: `tree_method=hist, grow_policy=lossguide, max_depth=0, max_leaves=255`. Zbog konzinstencije usporedbe, isti algoritam s istim inicijalnim postavkama će se primijeniti na svim metodama balansiranja.

4.6.1 Poduzorkovanje

Za definiciju poduzorkovanja može se uzeti postupak odbacivanja instanci većinske klase dok se ne postigne ravnoteža s manjinskom ili drugim riječima, dok se ne postigne približno sličan ili jednak (ovisno o željama) broj objekata obje klase (kao na slici 4.13). U slučaju velikog broja objekata ili ukoliko su podaci većinske klase dovoljno "blizu", ova metoda može dati iznimno kvalitetne rezultate. Dva tipa poduzorkovanja su obrađena.



Slika 4.13: Distribucija uravnoteženih podataka

Nasumično poduzorkovanje

Nasumično poduzorkovanje (eng. Random Undersampling - RUS) najjednostavnija je tehnika poduzorkovanja gdje se nasumičnim odabirom odbacuju objekti većinske klase iz trening skupa podataka dok se ne postigne isti broj objekata što za sobom povlači potencijalni gubitak vrijednih informacija i u konačnici dovodi do eventualnog podtreniranja (eng. underfitting) i slabe procjene na test skupu podataka.

NearMiss-1

NearMiss je napredniji pristup poduzorkovanja od nasumičnog odbacivanja objekata jer uključuje svojevrsno učenje ili istraživanje pri identifikaciji redundantnih objekata za brisanje ili korisnih za ne brisanje. Postoji mnogo metoda poduzorkovanja koje koriste ovaj pristup. Odabrana je jedna imena NearMiss, a koje postoje tri verzije NearMiss-1, NearMiss-2 i NearMiss-3. NearMiss-1(u grafovima NM1US) odabire objekte iz većinske klase koji imaju najmanju prosječnu udaljenost do tri najbliža objekta iz manjinske klase (prema [20]). Udaljenost se mjeri kao Euklidska udaljenost u prostoru značajki (eng. feature space). Prema početnim postavkama ova metoda poduzorkuje većinsku klasu dok ne sadrži isti broj objekata kao manjinska.

4.6.2 Naduzorkovanje

Naduzorkovanje povećava težinu manjinske klase replicirajući objekte iz manjinske klase. Iako se na taj način ne proširuje znanje o skupu podataka, javlja se problem pretreniranja. Zbog prirode problema u ovom radu (velik broj objekata), trening na naduzorkovanim podacima će zahtijevati jako puno vremena ili čak neće biti moguć za određene algoritme, stoga će se prikazati dva tipa naduzorkovanja: nasumično i SMOTE.

Nasumično naduzorkovanje

Nasumično naduzorkovanje (eng. Random Oversampling - ROS) vrlo jednostavno nasumičnim odabirom duplicira objekte iz manjinske klase, što za rezultat ima vjerojatnije pojavljivanje pretreniranja, iako s tim ne dodaje modelu nikakvu novu informaciju.

SMOTE

Želi li se zaobići problem pretreniranja, može se uvesti **S**ynthetic **M**inority **O**versampling **TE**chnique (**SMOTE**). Novi objekti se mogu sintetizirati iz postojećih objekata manjinske klase. SMOTE generira sintetičke podatke na osnovi sličnosti između postojećih objekata manjinske klase u prostoru značajki. Da bi stvorio sintetičke instance, pronalazi K najbližih susjeda svake manjinske instance, nasumično odabire jednu od njih i zatim računa linearnu interpolaciju za stvaranje nove instance manjinske klase u susjedstvu tvrdi [21]. Na ovaj način može se stvoriti proizvoljan broj instanci. Prednost ovog pristupa, može se reći, jest što generira nove, vjerodostojne, instance manjinske klase koje su relativno blizu postojećim manjinskim primjerima u prostoru značajki. Niti jedna tehnika nije idealna, nedostatak ovog pristupa je što se sintetičke instance generiraju bez uzimanja u obzir većinske klase, što može dovesti do generiranja moguće dvosmislenih objekata.

4.6.3 Kombinacija pod/nad uzorkovanja

Oba tipa uzorkovanja mogu biti vrlo učinkovita kad se koriste zasebno, iako još bolje rezultate često daje konjukcija pod i naduzorkovanja. Kao što je opisano u [22], preporučuje se prvo koristiti nasumično poduzorkovanje za smanjivanje broja instanci u većinskoj klasi, a zatim iskoristiti SMOTE i naduzorkovati manjinsku klasu za postizanje ravnoteže. Redoslijed izvođenja je nebitan jer se procedure izvode na različitim podskupima trening podataka; nasumično poduzorkovanje na većinskoj, a SMOTE na manjinskoj klasi. Postoje kombina-

cije poduzorkovanja i naduzorkovanja koje su dokazano efektnije. Dva primjera su kombinacija SMOTE-a s Tomek Links poduzrokovanjem i SMOTE-a s Edited Nearest Neighbors poduzorkovanjem. Imbalanced-learn Python biblioteka [23] osigurava implementaciju obje kombinacije. Nažalost, zbog velike količine podataka i ograničene računalne moći, za ovaj rad nije bilo moguće primijeniti neku od dvije navedene predefinirane kombinacije pod/naduzorkovanja. Međutim, dao sam si slobodu isprobati dvije različite kombinacije omjera nasumičnog poduzorkovanja većinske klase i primjene SMOTE tehnike na manjinskoj klasi. Prva kombinacija je SMOTE tehnikom generirati sintetičke instance manjinske klase dok ne postigne 25% većinske, a zatim Random Undersamplerom nasumično odbaciti instance većinske dok se ne izjednače (nazvali smo je SMOTE_25+RUS). Druga je istovjetna prvoj osim što se generiraju uzorci manjinske dok se ne postigne 50% većinske klase (nazvali smo je SMOTE_50+RUS).

4.7 Rezultati balansiranja skupa podataka za treniranje

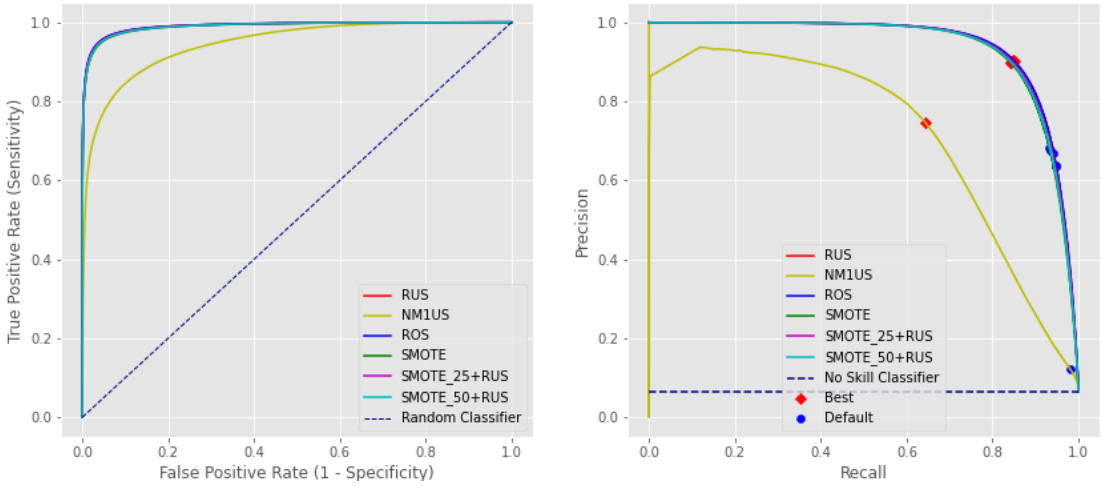
U prethodnom poglavlju, postavili smo teorijska polazišta za tri osnovna tipa balansiranja trening skupa podataka. Sad smo spremni primijeniti ih na našem problemu. Ponavljam, zbog konzistencije rezultata, samo XGBoost(hist+lossguide) model je treniran na balansiranom skupu podataka za treniranje i to na šest različitih načina balansiranja. Prag vjerojatnosti je 0.5. Promotrimo tablicu 4.8 s rezultatima sad već dobro poznatih metrika.

Metoda	Precision	Recall	F1-Score	Time[min]
RUS	0.6342	0.9485	0.7601	2
NM1US	0.1218	0.9809	0.2166	75
ROS	0.6363	**0.9481**	0.7615	25
SMOTE	0.6817	0.9322	0.7875	84
SMOTE_25+RUS	0.6677	0.9394	0.7806	81
SMOTE_50+RUS	0.6765	0.9352	0.7851	77

Tablica 4.8: Usporedba performansi XGBoost modela treniranog na balansiranom (različitim metodama) skupu podataka za treniranje. Prag vjerojatnosti je 0.5. Zadnji stupac sadrži zbroj vremena potrebnih za balansiranje i trening modela.

Vjerojatno prvo što se primijeti je NearMiss-1 metoda poduzorkovanja. Osjetljivost je vrlo visoka 98.09%, međutim, preciznost je iznimno niska, svega 12.18%. Najveći F1 rezultat od 78.75% postignut je SMOTE metodom naduzorkovanja.

Promotrimo li sliku 4.14 primijetiti ćemo sličnu situaciju kao i u prethodnom odjeljku. Granična vrijednost, ona koja maksimizira F1 mjeru, nije odabrana, što znači da bismo njenom promjenom mogli postići još bolje rezultate, sa stajališta F1 mjere.



(a) ROC krivulja **(b)** PR krivulja

Slika 4.14: ROC i PR krivulje XGBoost modela treniranog na balansiranom (različitim metodama) skupu podataka za treniranje. Plavi krug označava graničnu vrijednost od 0.5, a crveni dijamant graničnu vrijednost s optimalnom F1 mjerom.

Nezahvalno je samo na osnovi slike 4.14 pokušati odrediti najuspješniju metodu balansiranja skupa podataka za treniranje, stoga konačno pogledajmo ROC i PR AUC rezultat svih tipova i metoda balansiranja u tablici 4.9:

Metoda	ROC AUC	PR AUC
RUS	0.9905	0.9403
NM1US	0.9455	0.7184
ROS	**0.9905**	**0.9404**
SMOTE	0.9892	0.9348
SMOTE_25+RUS	0.9899	0.9378
SMOTE_50+RUS	0.9895	0.9361

Tablica 4.9: ROC i PR AUC tablica XGBoost modela treniranog na balansiranom (različitim metodama) skupu podataka za treniranje.

XGBoost algoritam treniran na nasumično naduzorkovanim (ROS) podacima,

daje najbolji rezultate i za ROC AUC i za PR AUC. Međutim, s grafa 4.14 i iz tablice 4.9 jasno nam je da sve metode balansiranja (izuzev NM1US) osiguravaju vrlo slične rezultate. Kao prednost vidim razliku vremena potrebnog kako za balansiranje skupa podataka za treniranje tako i za trening modela. Nasumično poduzorkovanje uvjerljivo najkraće traje, što je i očekivano.

Sad kad su svi rezultati predstavljeni, kako odlučiti koju kombinaciju algoritma i metode balansiranja odabrati? Više govora o tome slijedi u zaključku.

Zaključak

U ovom smo diplomskom radu napravili model za identifikaciju miona koristeći mione dobivene iz simulacije Drell-Yan procesa. S ciljem zamjene *cut based ID* metode identifikacije miona, predstavili smo pristup temeljen na strojnom učenju. Treniranjem modela na već klasificiranim objektima, želimo sačuvati znanje u svrhu identifikacije miona u eksperimentalnim podatcima gdje, za razliku od simulacije, ne znamo što je signal, a što pozadina. Da bismo imitirali ovaj proces, kreirani skup podataka podijelili smo na skup za treniranje i na skup za testiranje u omjeru 75:25. Zbog nebalansiranosti skupa podataka u kojem je broj signalnih objekata približno 14 puta veći od pozadinskih, primorani smo modificirati način treniranja i evaluacije modela. U tu svrhu, predstavili smo dva moguća rješenja: težinsko treniranje u odjeljcima 4.4 i 4.5 i balansiranje skupa podataka obrađeno u odjeljcima 4.6 i 4.7. Težinskim treniranjem, različito kažnjavajući pogreške različitih klasa povećali smo osjetljivost klasifikatora pri detekciji manjinske klase. Da osiguramo pravilnu usporedbu, koristimo ROC i PR evaluacijske metrike. ROC i PR krivulje nam grafički prikazuju koji omjer preciznosti i osjetljivosti klasifikatora možemo postići različitim pozicioniranjem na krivulji, tj. različitim odabirom praga vjerojatnosti. Kakav omjer zahtijevamo ovisi o fazi analize podataka u kojoj se nalazimo, o vrsti problema kojeg rješavamo ali i o dostupnom skupu podataka. Pouzdaniju usporedbu različitih algoritama uspostavili smo ROC i PR AUC metrikom, dakle jednim brojem koji kvantificira tu krivulju, računa iznos površine ispod nje i definira sposobnost raspoznavanja dviju klasa.

Primijenili smo 5 različitih algoritama nadziranog strojnog učenja započevši s često korištenom Logističkom Regresijom. Zatim smo se prebacili na Stablo Odluke i algoritme temeljene na njemu. Tu već Random Forest pokazuje zavidno poboljšanje, stoga je očekivano da *boosting* algoritmi poput LightGBM i XGBoosta ostvare za nijansu još bolje rezultate. Čak štoviše, *boosting* algoritmi, osim poboljšanja evaluacijskih metrika, uvelike smanjuju vrijeme potrebno za trening modela. No među samim *boosting* algoritmima nema velikih razlika jer se ipak

temelje na istim algoritmima za treniranje. Drugi pristup problemu nebalansiranosti skupa podataka je upravo njegovo balansiranje na način da se izbaci višak objekata većinske klase, doda objekte u manjinsku ili odradi kombinacija navedena dva pristupa. Zbog konzistencije usporedbe različitih metoda balansiranja, koristili smo isti XGBoost model pri istim inicijalnim postavkama. Ove metode pokazuju usporedivo slične performanse evaluacijskih metrika, dok je razlika tek primjetna u vremenu potrebnom za samo balansiranje skupa podataka za treniranje u kojem nasumično poduzorkovanje (RUS) višestruko nadmašuje sve druge metode. Štoviše, vrijeme potrebno za trening ovisi jesmo li poduzorkovali ili naduzorkovali, tj. smanjili ili povećali skup podataka za treniranje. Međutim, na umu treba imati da poduzorkovanje izbacuje objekte većinske klase što može rezultirati gubitkom vrijednih infomacija, a naduzorkovanje riskira pretreniranje. Konačni komentar na sve rezultate možemo dati pogledamo li tablice 4.3, 4.5 i 4.9. XGBoost i LightGBM očekivano pružaju najbolje rezultate koji se tek neznatno razlikuju primijenimo li ili ne težinsko treniranje i balansiramo li ili ne skup podataka za treniranje.

Kao moguće proširenje i poboljšanje rezultata vidim korištenje *grid search* metode za optimizaciju hiperparametara jer se u njima krije stvarna moć boosting algoritama. Također, vjerujem da bi korištenje dubokih neuronskih mreža bio izuzetno pametan potez zbog veličine samog skupa podataka. Postoji li mogućnost primjene istreniranog modela na nekom drugom zadatku? Kao prednost Drell-Yan procesa istaknuli smo relativno čisto konačno stanje koje uključuje leptone. Mnoge analize koriste spomenutu *cut based ID* metodu identifikacije leptona. Promatramo li samo kanal raspada Higgsa poput $H \rightarrow ZZ \rightarrow 4l$, imamo pristup istom čistom finalnom stanju kao u Drell-Yan procesu. Zamjena *cut based* identifikacije miona metodom strojnog učenja mogla bi uvelike smanjiti broj pozadinskih događaja u $H \rightarrow ZZ \rightarrow 4l$ uz istovremeno zadržavanje istog broja signalnih događaja što je iznimno važno u kanalima proizvodnje Higgsovog bozona koji trenutno nemaju veliku statistiku.

Dodatak A

Dodatak

A.1 Importing data

```
import uproot as up
import pandas as pd
import pickle
import numpy as np


def import_data(file):
    tree = file['ntuplizer/tree']
    frames = []
    for array in tree.iterate(entrysteps=100000):
        tmp = pd.DataFrame.from_dict(array)
        tmp = tmp[
            (tmp[b'n_obs_int'] > 1) & (tmp[b'mu_dz'] < 1) & (tmp[b
                                'mu_dxy'] < 0.5) &
            ((tmp[b'is_tracker_mu'] == True) | (tmp[b'is_global_mu
                                '] == True))]

        # Drop muons from Tau
        tmp = tmp[tmp[b'matchedToGenMu'] != 2]
        tmp['label'] = np.where(tmp[b'matchedToGenMu'] == 1, 0, 1)

        features = [
            b'glb_valid_mu_hits', b'tk_valid_hits', b'
                                tk_valid_pixel_hits',
            b'mu_pf_photon_iso', b'mu_pf_charged_had_iso', b'
                                mu_pf_neutral_had_iso',
            b'mu_rho', b'mu_sip', b'mu_dxy', b'mu_dz']
        features.append("label")
        tmp = tmp[features]

        tmp[b'glb_valid_mu_hits'] = pd.to_numeric(
```

```
            tmp[b'glb_valid_mu_hits'], downcast="signed")
        tmp[b'tk_valid_hits'] = pd.to_numeric(
            tmp[b'tk_valid_hits'], downcast="unsigned")
        tmp[b'tk_valid_pixel_hits'] = pd.to_numeric(
            tmp[b'tk_valid_pixel_hits'], downcast="unsigned")
        tmp["label"] = pd.to_numeric(tmp["label"], downcast="
                                    unsigned")

        frames.append(tmp)
    df = pd.concat(frames, ignore_index=True)
    filename = "Data/Finalized_dataset"
    df.to_pickle(filename)


def main():
    file = up.open(
    '/eos/user/m/mkovac/www/Studenti/2016/Muon-MVA/
                                    DY_2016_10_10_2019/train.root')
    import_data(file)

if __name__ == "__main__":
    main()
```

Code Listing A.1: Importing data

A.2 Train model

```
# Import libraries
import uproot as up
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle
import warnings
import timeit
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve, roc_curve,
                                    roc_auc_score
from sklearn.metrics import confusion_matrix, auc
from sklearn.metrics import classification_report
from numpy import nanargmax
from collections import Counter
pd.set_option("display.max_columns", 700)
%matplotlib inline
plt.style.use('ggplot')
warnings.filterwarnings('ignore')


# Balance train subset and return it
def balance_data(X_train, y_train):
    # Import wanted sampler
```

```
24      from imblearn.under_sampling import RandomUnderSampler
25      features = X_train.columns
26      start_time = timeit.default_timer()
27      # Balance training subset
28      X_train, y_train = RandomUnderSampler().fit_resample(X_train,
                                    y_train)
29      print("Balancing took: ",
30            round((timeit.default_timer() - start_time)/60), "
                                    minutes \n")
31      print("After balancing train set:", Counter(y_train))
32      # Convert it to pandas dataframe
33      X_train = pd.DataFrame(X_train, columns=features)
34      y_train = pd.DataFrame(y_train)
35      return X_train, y_train
36
37
38  # train model, save it and return it
39  def train_model(model, model_name, X_train, y_train):
40      start_time = timeit.default_timer()
41      model.fit(X_train, y_train)
42      print(model_name, " training time: ",
43            round((timeit.default_timer() - start_time)/60), "
                                    minutes \n")
44      pickle.dump(model, open("saved_models/%s.sav" % model_name, "
                                    wb"))
45      return model
46
47
48  # print model's performance metrics
49  def performance_metrics(model, model_name, X_test, y_test):
50      y_pred = model.predict(X_test)
51      y_pred_proba = model.predict_proba(X_test)[:, 1]
52      print(model_name, " metrics:")
53      print(classification_report(y_test, y_pred, digits=4))
54      print("Confusion matrix:\n", confusion_matrix(y_test, y_pred),
                                     "\n")
55
56
57  # plot model's ROC & PR Curve, find threshold which maximize F1
                                    Score
58  # Calculate ROC & PR AUC
59  def plot_roc_pr_curve(trained_model, model_name, X_test, y_test):
60      print(model_name, " ROC & PR:")
61      y_pred_proba = trained_model.predict_proba(X_test)[:, 1]
62      fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))
63      # ROC
64      fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
65      axes[0].plot(fpr, tpr, color='m', label=model_name)
66      print("ROC AUC: ", roc_auc_score(y_test, y_pred_proba))
67      axes[0].plot([0, 1], [0, 1], color='navy',
68                   lw=1, linestyle='--', label="Random Classifier")
69      axes[0].set(xlabel='False Positive Rate (1 - Specificity)',
```

```
                ylabel='True Positive Rate (Sensitivity)')
    axes[0].legend()
    # PR
    precision, recall, thresholds = precision_recall_curve(y_test,
                                    y_pred_proba)
    axes[1].plot(recall, precision, color='m', label=model_name)
    print("PR AUC: ", auc(recall, precision))
    # convert to f1 score
    fscore = (2 * precision * recall) / (precision + recall)
    # locate the index of the largest f score
    ix = nanargmax(fscore)-1
    # locate the index of threshold=0.5
    ix05 = (np.abs(thresholds - 0.5)).argmin()
    print("Precision={:.4f}, Recall={:.4f}, "
          "F1-Score={:.4f}, Best Threshold={:.4f}\n"
          .format(precision[ix+1], recall[ix+1], fscore[ix+1],
                                    thresholds[ix]))
    axes[1].scatter(recall[ix+1], precision[ix+1], marker='D',
                    color='red', label='Best')
    axes[1].scatter(recall[ix05+1], precision[ix05+1], marker='o',
                    color='blue', label='Default')
    axes[1].plot([0, 1], [y_test.mean(), y_test.mean()], color='
                                    navy',
                 linestyle='--', label='No Skill Classifier')
    axes[1].set(xlabel='Recall', ylabel='Precision')
    axes[1].legend()
    fig.savefig("saved_pictures/"+model_name+"_ROC_PR_Curve.png")


def main():
    # Import dataset
    df = pd.read_pickle("Data/Finalized_dataset")
    print("Number of instances in dataset:", Counter(df["label"]))
    features = df.columns
    # Split data into train and test subset
    X_train, X_test, y_train, y_test = train_test_split(
        df[features.drop("label")], df["label"],
        random_state=1, stratify=df['label'])
    # Delete dataset to release memory
    del df
    # Uncomment next line to balance training subset
    # X_train, y_train = balance_data(X_train, y_train)
    # Import desired classifier
    from xgboost.sklearn import XGBClassifier
    model = XGBClassifier(n_jobs=-1, tree_method="hist",
                          grow_policy="lossguide", max_depth=0,
                                    max_leaves=255)
    # Uncomment next line to train classifier
    trained_model = train_model(model, "XGB_leaf", X_train,
                                    y_train)
    # Uncomment next line to print classifier's perfomace metrics
    performance_metrics(trained_model, "XGB_leaf", X_test, y_test)
```

```
117     # Uncomment next line to plot classifier's ROC&PR curves
118     plot_roc_pr_curve(trained_model, "XGB_leaf", X_test, y_test)
119
120
121 if __name__ == "__main__":
122     main()
```

Code Listing A.2: Train model

Code Listings

Bibliografija

[1] T. C. Collaboration, S. Chatrchyan *et al.*, "The CMS experiment at the CERN LHC," *Journal of Instrumentation*, vol. 3, no. 08, pp. S08 004–S08 004, aug 2008. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F3%2F08%2Fs08004

[2] W. Commons, "File:standard model of elementary particles.svg — wikimedia commons, the free media repository," 2020, [Online; accessed 13-May-2020]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Standard_Model_of_Elementary_Particles.svg&oldid=416441632

[3] S. D. Drell and T.-M. Yan, "Massive lepton-pair production in hadron-hadron collisions at high energies," *Phys. Rev. Lett.*, vol. 25, pp. 316–320, Aug 1970. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.25.316

[4] "Drell-yan," https://www.quantumdiaries.org/2015/05/18/dy-resummation/, accessed: 2020-05-12.

[5] I. R. Kenyon, "The drell-yan process," *Reports on Progress in Physics*, vol. 45, no. 11, pp. 1261–1315, nov 1982. [Online]. Available: https://doi.org/10.1088%2F0034-4885%2F45%2F11%2F002

[6] G. Arnison *et al.*, "Experimental observation of isolated large transverse energy electrons with associated missing energy at $\sqrt{s} = 540gev$," 01 1983.

[7] M. Banner *et al.*, "Observation of single isolated electrons of high transverse momentum in events with missing transverse energy at the cern pp collider," *Physics Letters B*, vol. 122, no. 5, pp. 476 – 485, 1983. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0370269383916052

[8] P. Bagnaia *et al.*, "Evidence for $z^0 \to e^+e^-$ at the cern pp collider," *Physics Letters B*, vol. 129, no. 1, pp. 130 – 140, 1983. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037026938390744X

[9] M. Jindal *et al.*, "Drell–yan process at large hadron collider," *PRAMANA c Indian Academy of Sciences*, vol. 7615, 03 2011.

[10] "CMS detector design," http://cms.web.cern.ch/news/cms-detector-design, accessed: 2019-07-20.

[11] S. Baffioni *et al.*, "Electron reconstruction in CMS," CERN, Geneva, Tech. Rep. CMS-NOTE-2006-040, Feb 2006. [Online]. Available: http://cds.cern.ch/record/934070

[12] "Muon analysis," https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookMuonAnalysis, accessed: 2019-07-31.

[13] A. Sirunyan *et al.*, "Performance of the CMS muon detector and muon reconstruction with proton-proton collisions at $\sqrt{s} = 13$ TeV," *Journal of Instrumentation*, vol. 13, no. 06, pp. P06 015–P06 015, jun 2018. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F13%2F06%2Fp06015

[14] "Impact parameter," https://www.desy.de/~ameyer/hq/node25.html, accessed: 2020-06-25.

[15] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st ed. O'Reilly Media, Inc., 2017.

[16] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[17] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 233–240. [Online]. Available: https://doi.org/10.1145/1143844.1143874

[19] "Lightgbm," https://lightgbm.readthedocs.io/en/latest/index.html, accessed: 2020-03-06.

[20] "Undersampling algorithms for imbalanced classification," https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/, accessed: 2020-03-06.

[21] "Smote oversampling for imbalanced classification," https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/, accessed: 2020-03-06.

[22] K. W. Bowyer *et al.*, "SMOTE: synthetic minority over-sampling technique," *CoRR*, vol. abs/1106.1813, 2011. [Online]. Available: http://arxiv.org/abs/1106.1813

[23] G. Lemaître *et al.*, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html