

Spis treści

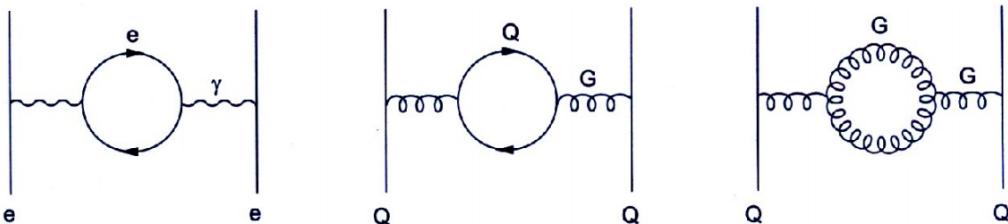
1 Fizyka dżetów cząstek	2
1.1 Chromodynamika kwantowa	2
1.2 Plazma kwarkowo-gluonowa	3
1.3 Dżety	4
1.4 Dżety b	5
1.5 Eksperyment ALICE	6
2 Uczenie maszynowe	9
2.1 Wzmacniane drzewa decyzyjne	9
2.2 Sieci neuronowe	10
2.3 Dyskusja użycia dwóch algorytmów	15
3 Dane	16
4 Analiza	19
4.1 Dobór metryki	19
4.2 Wyniki dla poszczególnych modeli	22
4.3 Korelacje predykcji modeli	27
4.4 Analiza istotności zmiennych w BDT	28
4.5 Analiza wpływu zmiennych na predykcje modeli	31
5 Podsumowanie i wnioski	35
Dodatki	36
Dodatek A Metryki	36
Dodatek B Skróty i oznaczenia	37

23 1 Fizyka dżetów cząstek

24 1.1 Chromodynamika kwantowa

25 Chromodynamika kwantowa (ang. *Quantum Chromodynamics – QCD*) to kwantowa teoria
26 pola opisująca oddziaływanie silne [1]. Wprowadza ona dla kwarków nową liczbę kwantową
27 nazywaną kolorem lub ładunkiem kolorowym, który jest odpowiednikiem ładunku elektrycznego
28 w elektrodynamice kwantowej (ang. *Quantum Electrodynamics – QED*), ale w przeciwnieństwie
29 do niego może przyjmować 3 różne wartości (i trzy antywartości dla antykwarków). Elementarne
30 oddziaływanie w obu teoriach przenoszone są przez bezmasowe bozony pośredniczące: w *QED*
31 jest to elektrycznie obojętny foton a w *QCD* gluony, które występują w 8 odmianach i są
32 kolorowo naładowane, przez co możliwe jest oddziaływanie zachodzące między dwoma gluonami.
33 Kwarki i gluony zbiorczo nazywane są partonami.

34 Próżnia, w rozumieniu klasycznym będąca zupełnie pusta, w teoriach kwantowych wypeł-
35 niona jest pojawiającymi i znikającymi wirtualnymi cząstками. Cząstki te ekranują ładunek
36 próbny umieszczony w kwantowej próżni, wywołując zjawisko polaryzacji próżni (analogiczne
37 do polaryzacji dielektryków), które efektywnie zmniejsza pole wytwarzane przez ten ładunek.
38 Siła tego efektu zależy od liczby ekranujących cząstek, czyli pośrednio od skali odległości. Skala
39 ta wyznaczona jest przez długości fali próbującej cząstki, zatem także jej energię (im więk-
40 sza energia, tym mniejsza długość fali i mniejsza ilość ekranujących cząstek obserwowanych
41 w pobliżu rzeczywistego ładunku, zatem tym słabszy efekt ekranowania i większy efektywny
42 ładunek). Prowadzi to do zależnej od energii stałej sprzężenia α , którą nazywamy efektywną
43 lub biegącą stałą sprzężenia (ang. *running coupling constant*).



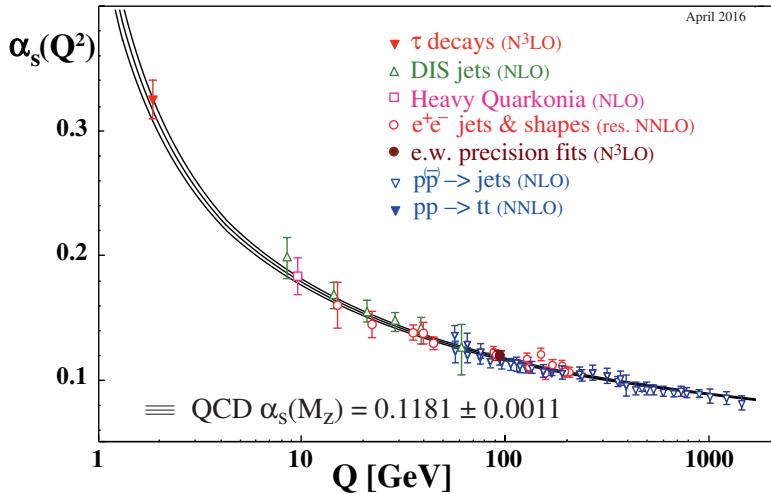
Rysunek 1: Diagramy Feynmana opisujące polaryzację próżni w *QED* (lewy) i *QCD* (środkowy i prawy). Rysunki lewy i środkowy są swoimi odpowiednikami w tych dwóch teoriach, natomiast prawy, w którym oddziałują jedynie bozony pośredniczące nie ma swojego odpowiednika w *QED*. Źródło: [1]

44 Zarówno pary elektron-pozyton jak i kwark-antykwark działają ekranującą kolejno ładunek elektryczny i kolorowy. Jednak jak zostało to już wspomniane, w przypadku *QCD* możliwe
45 jest także samooddziałanie gluonów, przez co dopuszczalne są diagramy Feynmana jak ten
46 przedstawiony na Rys. 1 po prawej. Pętle gluonowe działają anty-ekranującą – zwiększą efek-
47 tywną wartość silnej stałej sprzężenia, ponadto jest to efekt dominujący nad przyczynkiem od
48 par kwark-antykwark, co sprawia że zależność biegącej stałej sprzężenia w *QCD* jest odwrotna
49 i dużo silniejsza niż w przypadku *QED*. Wartość α_{em} maleje od wartości $\frac{1}{128}$ przy energiach ok.
50 90 GeV do $\frac{1}{137}$ przy energii bliskiej zeru, co oznacza zmianę o kilka procent. Tymczasem α_S
51 rośnie w miarę zbliżania się do niskich energii od wartości $\alpha_S \lesssim 0.1$ dla $E \gtrsim 100$ GeV do $\alpha_S > 1$
52 dla energii poniżej 200 MeV (Rys. 2). Prowadzi to do dwóch zjawisk charakterystycznych dla
53 chromodynamiki kwantowej:

- 55 • asymptotyczna swoboda (ang. *asymptotic freedom*) [2], [3] – dla wysokich energii ($\gtrsim 100$
56 GeV) silna stała sprzężenia jest mała (w tym zakresie energii możliwe jest stosowanie

57 rachunku perturbacyjnego) i kwarki wewnątrz hadronów zachowują się jak cząstki quasi-
 58 swobodne.

- 59 • uwiecznienie koloru (ang. *colour confinement*) – przy zwiększaniu odległości między par-
 60 tonami siła oddziaływanego rośnie do nieskończoności, dlatego nigdy nie obserwuje się
 61 swobodnych cząstek obdarzonych ładunkiem kolorowym a jedynie w postaci związanej w
 62 kolorowo obojętnych hadronach.



Rysunek 2: Zależność silnej stałej sprzężenia od przekazu czteropędu. Źródło: [4].

63 1.2 Plazma kwarkowo-gluonowa

64 Odkrycie asymptotycznej swobody pozwoliło na sprawdzenie przewidywań *QCD* w warun-
 65 kach bardzo wysokich temperatur oraz gęstości. Dla wystarczająco dużych gęstości hadrony
 66 zaczynają na siebie zatoczyć, prowadząc do stworzenia stanu, w którym poszczególne hadrony
 67 przestają być odróżnialne [5]. Zasugerowane zostało także istnienie przejścia fazowego w tem-
 68 peraturze porównywalnej z masą pionów oraz w temperaturze niższej, ale przy odpowiednio
 69 dużych gęstościach [6]. Stan materii powstały po osiągnięciu, któregoś z tych warunków nazy-
 70 wany jest plazmą kwarkowo-gluonową (ang. *quark-gluon plasma – QGP*). Obecnie przewiduje
 71 się, że materia w takim stanie istniała w pierwszych ułamkach sekund po Wielkim Wybuchu
 72 [7] oraz, że może się znajdować w jądrach gwiazd neutronowych [8].

73 Obecnie aby uzyskać dostęp do materii w stanie plazmy kwarkowo-gluonowej potrzebne są
 74 wysokoenergetyczne zderzenia cząstek. Powszechnie mówi się o niej w kontekście zderzeń cięż-
 75 kich jonów, chociaż istnieją także prace doszukujące się obecności *QGP* w mniejszych systemach
 76 np. w zderzeniach proton-proton [9], [10].

77 Cechą charakterystyczną *QGP* jest obecność wolnych kwarków i gluonów. Ze względu na
 78 uwiecznienie koloru w każdym innym stanie materii są one zawsze związane i tworzą hadrony.
 79 Wolne kwarki i gluony powstające w zderzeniach muszą zatem przejść przez proces hadroni-
 80 zacji, w którym rekombinują one ze spontanicznie wytworzonymi nowymi partonami, tworząc
 81 hadrony. W wyniku tego procesu, z każdego partonu obecnego w początkowym etapie zderzenia
 82 może powstać wiele cząstek poruszających się podobnym kierunkiem, tworząc stożek z wierzchoł-
 83 kiem blisko punktu interakcji wiązek. Taki stożek skolimowanych cząstek nazywany jest dżetem
 84 cząstek.

85 1.3 Dżety

86 Przedstawiona powyżej definicja dżetu nie jest precyzyjna z punktu widzenia pracy ekspery-
87 mentalnej. W detektorze obserwuje się tylko cząstki w stanie końcowym, nie jest zatem możliwe
88 przyporządkowanie cząstki do dżetu według jej pochodzenia. W związku z tym, konieczne jest
89 użycie algorytmu klasteryzującego, dostającego na wejściu tylko obserwowalne eksperymentalnie
90 cząstki. To jakie dżety zostaną zaobserwowane w danym zdarzeniu zależy od użytego
91 algorytmu. Oznacza to, że precyzyjną definicję dżetu stanowi algorytm klasteryzujący wraz z
92 parametrami.

93 Obecnie najpowszechniej stosowanym algorytmem jest algorytm *anti-kt* [11]. Jest to algo-
94 rytm iteracyjny, który łączy kolejno pary cząstek o najmniejszej wartości wielkości

95 $d_{ij} = \min(1/k_{t,i}^2, 1/k_{t,j}^2) \frac{\Delta_{ij}^2}{R^2}$, gdzie $\Delta_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2$, natomiast $k_{t,i}$, η_i , ϕ_i to kolejno
96 pęd poprzeczny, *pseudopospieszeńszość* (ang. *pseudorapidity*)¹ i kąt azymutalny i -tej cząstki. R
97 jest parametrem określającym rozmiar dżetu i zwykle przyjmuje wartości 0.2, 0.4 lub 0.7. Tak
98 określona miara, według której łączone są cząstki sprawia, że w pierwszej kolejności łączone są
99 cząstki z najbliższego otoczenia i koncentrują się one wokół tzw. twardych cząstek (tj. tych o
100 dużym k_t).

101 Eksperymentalne ograniczenia związane z obserwacją tylko końcowego stanu oddziaływań
102 nie występują w analizie danych z symulacji Monte Carlo (MC), gdzie ma się dostęp do peł-
103 nej informacji na temat historii każdej cząstki. Nie należy jednak używać jej do klasteryzacji
104 dżetów, gdyż utracona zostałaby cecha odpowiedniości między obiektymi nazywanymi dżetami
105 w symulacji i w eksperymencie, która to cecha jest niewątpliwie jedną z podstawowych wyma-
106 gań stawianych przed dobrą symulacją. Właściwym podejściem jest rekonstrukcja dżetów przy
107 pomocy takiego samego algorytmu jak w przypadku danych eksperymentalnych.

108 Dżety wykorzystuje się w badaniach plazmy kwarkowo-gluonowej. Dają one pośredni wgląd
109 we właściwości *QGP* na podstawie wpływu jaki wywiera na oddziałujące z nią partony. Przy-
110 kładową obserwablą mierzoną w zderzeniach ciężkich jonów jest czynnik modyfikacji jądrowej
111 R_{AA} (ang. *nuclear modification factor*), który jest miarą strat energii przez parton przecho-
112 dzający przez medium. Jest to stosunek pędowych rozkładów dżetów cząstek zmierzonych w
113 zderzeniach ciężkich jąder oraz w zderzeniach pp (przemnożonych przez liczbę binarnych zde-
114 rzeń nukleon-nukleon przewidywanych przez model teoretyczny). Odchylenia od wartości 1 dla
115 wysokich p_T są oznaką modyfikacji pędów dżetów przez gęste medium (w stosunku do zderzeń
116 pp gdzie *QGP* nie powstaje), jest to tzw. tłumienie dżetów (ang. *jet quenching*). W zakresach
117 energii mierzonych na LHC obserwuje się silne tłumienie dżetów [12] (CSM), [13] (ALICE).

118 Oprócz globalnego wpływu medium na dżety, analizuje się także różnice między dżetami
119 pochodząymi z gluonów oraz kwarków o różnych zapachach (ang. *flavours*). Modele teore-
120 tyczne przewidują między innymi większe straty energii w wyniku interakcji z *QGP* dla dżetów
121 gluonowych niż kwarkowych [14] oraz zależność strat energii od masy partonu [15] – w tym
122 przypadku precyzyjne pomiary rozróżniające typy dżetów pozwalają lepiej zrozumieć mecha-
123 nizm odpowiadający za straty energii przez partony. Zagadnienie rozpoznania z jakiego rodzaju
124 partonu powstał dany dżet, nazywane jest identyfikacją lub tagowaniem dżetu. Ważną rolę
125 w studiowaniu tego problemu odgrywają symulacje MC, które pozwalają określić wydajności
126 poszczególnych technik tagowania dżetów na podstawie znajomości kanału produkcji każdej
127 symulowanej cząstki.

¹ $\eta = -\ln[\tan(\frac{\theta}{2})]$, gdzie θ jest kątem między wektorem pędu cząstki a osią wiązki

128 1.4 Dżety *b*

129 1.4.1 Właściwości

130 Poza badaniami właściwości *QGP*, szczególne znaczenie ma identyfikacja dżetów pochodzących
131 z ciężkich kwarków: *b* i *c*. Są one ważnym elementem w poszukiwaniu łamania symetrii *CP* w
132 rozpadach hadronów *B* i *D* oraz innych sygnatur tzw. *Nowej Fizyki* wykraczającej poza ramy
133 Modelu Standardowego. Kwarki *piękne* pojawiają się także często w kanałach rozpadu cząstek
134 takich jak bozon Higgsa i kwark *t*.

135 Identyfikacja dżetów *b* jest sporym wyzwaniem ze względu na zdecydowanie częściej wy-
136 stępujące dżety lekkie, tj. powstałe z hadronizacji kwarków *u,d,s* lub gluonów. Rozpoznawanie
137 dżetów *b* bazuje na charakterystycznych właściwościach hadronów zawierających kwark piękny:
138 relatywnie długim czasie życia oraz (w mniejszym stopniu) na ich półleptonowych rozpadach
139 o względnej częstotliwości rozpadu w tym kanale (ang. *branching ratio*) na poziomie 10% [4].

140 1.4.2 Przegląd algorytmów używanych do identyfikacji dżetów *b* na LHC

141 Używane w eksperymentach na LHC: ATLAS, CMS i ALICE algorytmy można podzielić na
142 trzy kategorie: wykorzystujące wtórne wierzchołki, informację o odległości najbliższego zbliżenia
143 cząstek tworzących dżet (ang. *Distance of Closest Approach – DCA*, *Impact Parameter – IP*)
144 oraz identyfikujące produkty półleptonowych rozpadów pięknych lub powabnych hadronów.
145 Dokładne opisy omawianych algorytmów można znaleźć w: [16], [17] (ATLAS), [18], [19] (CMS),
146 [20], [21] (ALICE).

147 Najprostszym algorytmem jest dyskryminacja na podstawie istotności statystycznej (wynik
148 pomiaru podzielony przez jego niepewność) odległości wtórnego wierzchołka od wierzchołka
149 pierwotnego *L*. Jest to metoda wykorzystywana w każdym z trzech wymienionych eksperymentów
150 (ATLAS: algorytm SV0, CMS: algorytm SSV, ALICE). Może być ona rozszerzona poprzez
151 użycie dodatkowych zmiennych opisujących wtórny wierzchołek jak na przykład jego masa,
152 ułamek niesionej przez niego całkowitej energii dżetu (ATLAS: SV1) lub użycie "pseudowierz-
153 chołków" (kombinacji dwóch cząstek o dużych *DCA*) w celu poprawienia wydajności detekcji o
154 przypadki, w których wtórny wierzchołek nie został zrekonstruowany (CMS: CSV).

155 Algorytmy wykorzystujące informację o poszczególnych cząstkach mogą zasadniczo bazo-
156 wać albo na sumie logarytmów prawdopodobieństw pochodzenia każdej cząstki z pierwotnego
157 wierzchołka (ATLAS: IP3D, CMS: JP) lub tym samym prawdopodobieństwie ale dla wybranej,
158 np. drugiej lub trzeciej cząstki na liście posortowanej według malejącego *IP* (ALICE i CMS:
159 TC). Bardziej złożonym podejściem, w którym cząstki nie są traktowane jako niezależne, jest
160 użycie rekurencyjnych sieci neuronowych (ATLAS: RNNIP).

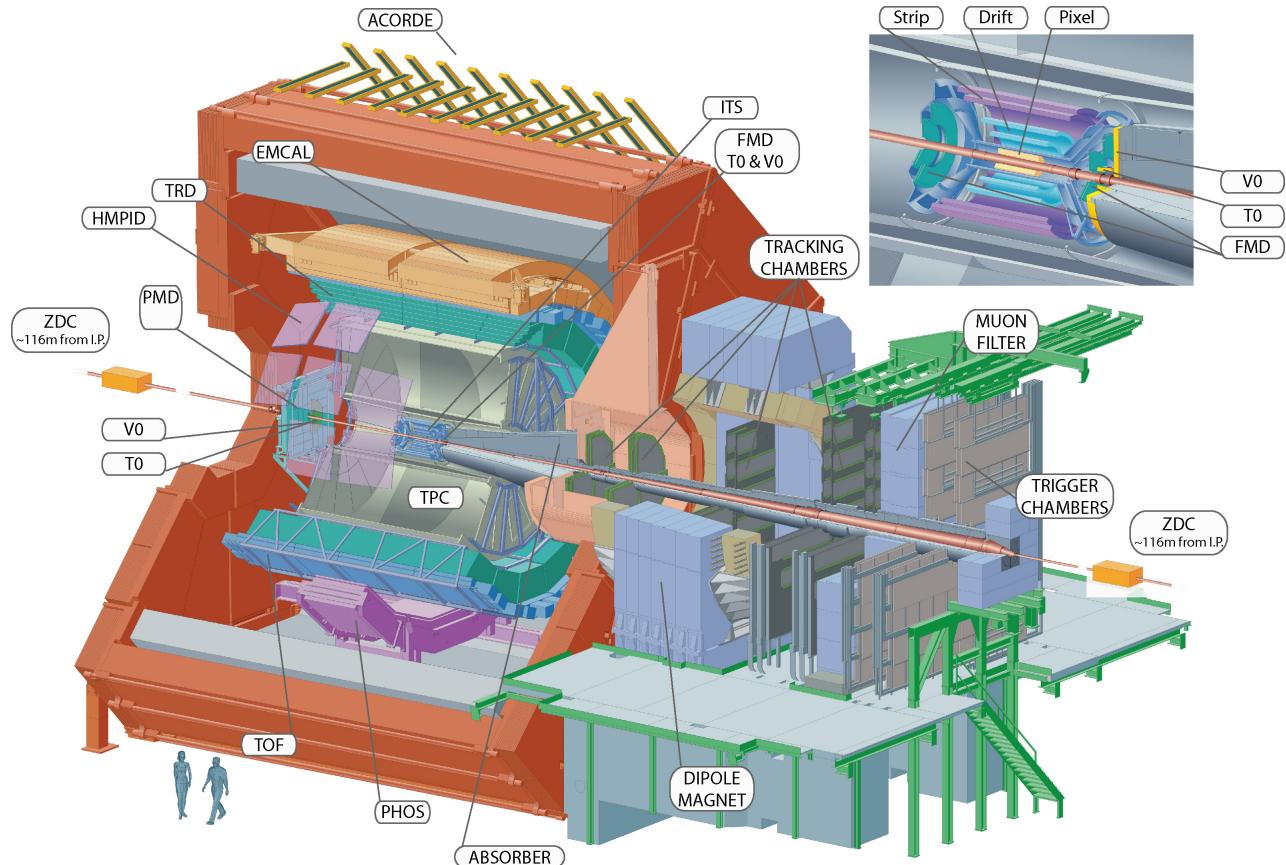
161 Do wykorzystania półleptonowego kanału rozpadu ciężkich hadronów do identyfikacji dżetów
162 *b* w eksperymentach ATLAS (SMT) i CMS (SE, SM) użyto wzmacnianych drzew decyzyjnych
163 trenowanych na kilku ręcznie zdefiniowanych w tym celu zmiennych takich jak pęd leptonu
164 transwersalny względem osi dżetu.

165 Znaczącą poprawę zdolności predykcyjnej można uzyskać łącząc kilka różnych modeli. Al-
166 gorytm łączący może pobierać na wejściu albo tylko predykcje klasyfikatorów niższego poziomu
167 (CMS: cMVAv2) lub dodatkowo także ich zmienne wejściowe (ATLAS: MV2, DL1). Do scalania
168 używane są zwykle wzmacniane drzewa decyzyjne lub sieci neuronowe.

169 Przykład innego podejścia zaprezentowała współpraca przy eksperymencie LHCb, gdzie
170 wykorzystano dwa zestawy wzmacnianych drzew decyzyjnych operujących na zmiennych zwią-
171 zanych z wtórnymi wierzchołkami. Pierwszy zapewnia separację dżetów lekkich od ciężkich a
172 drugi odróżnia dżety *b* od *c*. Do wyboru punktu pracy, zamiast jednowymiarowego rozkładu
173 predykcji używany jest dwuwymiarowy rozkład wag przypisany przez oba klasyfikatory [22].

1.5 Eksperyment ALICE

Eksperyment ALICE [23], [24] jest jednym z czterech największych eksperymentów na LHC. Jest on dedykowany zderzeniom ciężkich jonów (w LHC są to jony ołowiu PbPb), ale mierzone są także mniejsze systemy, tj. proton-proton pp (głównie jako referencję dla pomiarów PbPb) oraz proton-ołów p+Pb, które dostarczają także okazji do badania asymetrycznych zderzeń. Cechą charakterystyczną pomiarów ciężkojonowych jest ich znacznie większa niż w przypadku zderzeń pp krotność, tj. liczba cząstek wyprodukowana w pojedynczym zderzeniu. W przypadku zderzeń PbPb może powstawać nawet do 8000 naładowanych cząstek na jednostkę pseudopospieszności η . Detektor ALICE został zoptymalizowany do mierzeniach takich przypadków, jak również pod kątem rekonstrukcji i identyfikacji cząstek o szerokim zakresie pędów (100 MeV/c – 100 GeV/c).



Rysunek 3: Schemat detektora ALICE. Źródło: [25]

Detektor ALICE jest urządzeniem złożonym z wielu subdetektorów, schematycznie przedstawionych na Rys. 3. Można je podzielić według pełnionej w pomiarach roli. ITS, TPC, TRD oraz TOF pokrywają pełen kat azymutalny oraz zakres pseudopospieszności $|\eta| < 0.9$.

- Detektory śladowe – mierzące trajektorie cząstek zakrzywiane w polu magnetycznym o wartości $B = 0.5$ T.
 - Inner Tracking System (ITS) – zespół krzemowych detektorów śladowych znajdujący się najbliżej miejsca interakcji wiązek. Składa się on z 6 cylindrycznych warstw o promieniach od 4 cm do 43 cm, wykonanych w trzech różnych technologiach. Jego główną rolą jest rekonstrukcja pierwotnego oraz wtórnego wierzchołków. Bierze także udział w rekonstrukcji trajektorii i strat energetycznych cząstek, szczególnie tych niskopędowych, które nie docierają do dalej położonych detektorów.

- 196 – Time Projection Chamber (TPC) – długa na 5 m i o takiej średnicy komora projekcji
 197 czasowej. Jest to główny detektor śladowy ALICE, wraz z ITS służy do wyznaczania
 198 trajektorii cząstek i na ich podstawie również wierzchołków zderzenia. Elektryny
 199 uwolnione ze zjonizowanego przez poruszające się w nim naładowane cząstki gazu
 200 dryfują wzdłuż kierunku wiązki w stronę końcowych elektrod. Następnie są tam
 201 zbierane dostarczając informacji o dwóch współrzędnych toru cząstki: odległości od
 202 wiązki i kącie azymutalnym. Trzecia składowa trajektorii jest otrzymywana na pod-
 203 stawie czasu dotarcia elektronów do elektrod. TPC jest najwolniejszym detektorem
 204 ALICE (ze względu na ograniczający czas dryfu elektronów wynoszący $\sim 90 \mu\text{s}$),
 205 użycie detektora tego typu podyktowane jest jego zdolnością do mierzenia śladów
 206 tysięcy cząstek spodziewanych w centralnych zderzeniach PbPb.
 207 Znajomość toru ruchu cząstki pozwala na wyznaczenie jej pędu. Oprócz dokładnej
 208 trajektorii każdej cząstki próbkiowej do 159 razy, TPC mierzy straty energii cząstek
 209 dE/dx . Pozwala to na ich identyfikację na podstawie zależności Bethego-Blocha,
 210 najwyższą zdolność rozdzielczą TPC osiąga dla cząstek o $p_T < 1 \text{ GeV}$.

211 • detektory służące identyfikacji cząstek (ang. *particle identification – PID*)

- 212 – Transition Radiation Detector (TRD) – detektor wykrywający promieniowanie przej-
 213 ścia, służy głównie do odróżniania wysokopędowych ($p_T > 1 \text{ GeV}$) elektronów od
 214 pionów. Promieniowanie przejścia emitowane jest podczas przechodzenia relatywi-
 215 stycznych cząstek przez granicę ośrodków, jego intensywność jest proporcjonalna do
 216 czynnika Lorentza γ , co pozwala na odróżnienie cząstek o tym samym pędzie na pod-
 217 stawie różnicy mas (elektryny są ponad 250 razy lżejsze od pionów). TRD oprócz
 218 identyfikacji elektronów uczestniczy także w rekonstrukcji śladów wysokopędowych
 219 cząstek i może być użyty w systemie wyzwalania.
 220 – Time-Of-Flight (TOF) – detektor czasu przelotu o zdolności rozdzielczej $\sim 80 \text{ ps}$.
 221 Pozwala na separację pionów i kaonów o pędach do ok. 2.5 GeV i protonów do 4 GeV .
 222 – High-Momentum Particle Identification Detector (HMPID) – detektor typu RICH
 223 (ang. *ring-imaging Cherenkov*), wykrywający fotony emitowane podczas przejścia
 224 przez ośrodek naładowanej cząstki o prędkości większej od prędkości fazowej światła
 225 w tym ośrodku (promieniowanie Cherenkowa). Na podstawie kąta pod jakim emitowane
 226 są fotoniki określana jest prędkość cząstki. HMPID pozwala na identyfikację pio-
 227 nów, kaonów i protonów o $p_T > 1 \text{ GeV}$. Pokrywa przestrzeń kątów: $1.2^\circ < \phi < 58.8^\circ$
 228 oraz $|\eta| < 0.6$ (5% akceptancji TPC).

229 • kalorymetry

- 230 – Photon Spectrometer (PHOS) – elektromagnetyczny kalorymetr o wysokiej rozdziel-
 231 czości energetycznej i przestrzennej (podzielony na kryształy o rozmiarze poprzecznym
 232 $2.2 \times 2.2 \text{ cm}$, co odpowiada przestrzeni fazowej $\Delta\eta \times \Delta\phi = 0.004 \times 0.004$).
 233 Pokrywa zakres pseudopospieszności $|\eta| < 0.12$ i kąta azymutalnego równy 100° .
 234 PHOS ma za zadanie identyfikację i pomiar czteropędów fotonów, w szczególności
 235 tych niepochodzących z rozpadu innych cząstek (ang. *direct photons*) oraz lekkich
 236 mezonów neutralnych (np. π^0) przez dwufotonowy kanał rozpadu.
 237 – Electromagnetic Calorimeter (EMCal) – drugi elektromagnetyczny kalorytmeter ALICE
 238 o mniejszej ziarnistości ($\Delta\eta \times \Delta\phi = 0.014 \times 0.014$), ale dużo większej akceptancji
 239 ($|\eta| < 0.7$, $\Delta\phi = 107^\circ$). EMCal poprawia możliwości ALICE w zakresie pomiarów
 240 tłumienia dżetów, pozwalając na wyznaczanie neutralnej składowej energii dżetów

(energii niesionej przez neutralne cząstki). Dzięki innej charakterystyce dla elektrownów i hadronów (elektrony typowo deponują niemal całą energię a hadrony tylko niewielką część) pozwala je odróżnić na podstawie stosunku zmierzonej w nim energii do wyznaczonego wcześniej pędu E/p . EMCAL może być użyty także w szybkim systemie wyzwalania, do selekcji przypadków z dżetami oraz wysokoenergetycznymi fotonami i elektronami.

- Muon spectrometer – spektrometr mionowy, złożony z dwóch pasywnych absorberów, znajdujących się między nimi 10 warstw detektora śladowego oraz komór systemu wyzwalającego na końcu. Przedni absorber, gruby na 4 metry ($\sim 60X_0$) wykonany z betonu i grafitu, zatrzymuje hadrony oraz miony o niższych energiach (np. z rozpadów pionów i kaonów). Jest on zoptymalizowany aby minimalizować rozpraszanie mionów i zapewnić ochronę pozostałych detektorów ALICE przed wtórnymi cząstками powstałymi w jego materiale. Komory śladowe mają zdolność rozdzielczą ok. $100 \mu\text{m}$, co pozwala osiągnąć wysoką rozdzielcość przy wyznaczaniu masy niezmienniczej rzędu $100 \text{ MeV}/c^2$. Spektrometr mionowy służy głównie do mierzenia mezonów wektorowych (ω , ϕ , J/Ψ , Υ) rozpadających się w kanale $\mu^+\mu^-$.
- Detektory przednie, wyznaczające min. centralność zderzeń oraz płaszczyznę reakcji.
 - ZDC – zespół czterech kalorymetrów (po dwa do pomiaru protonów i neutronów) pokrywających inną przestrzeń fazową ponieważ tory protonów są odchylane przez pole magnetyczne, mierzących energię nukleonów nieuczestniczących w zderzeniu tzw. obserwatorów, co pozwala na określenie liczby nukleonów oddziałujących, tzw. uczestników. Znajdują się one 116 m od miejsca interakcji.
 - PMD – detektor gazowy mierzący krotkości oraz rozkład przestrzenny fotonów
 - FMD – krzemowy detektor paskowy mierzący precyzyjnie liczbę naładowanych cząstek w zakresie pseudopospieszności wykraczającym poza akceptancję detektora ITS.
 - V0 – liczniki scyntylacyjne położone po obu stronach detektora, używane w systemie wyzwalania o minimalnym obciążeniu (ang. *minimum bias trigger*) – wymóg obecności sygnału w obu detektorach pozwala na odrzucenie przypadków tła z oddziaływania wiązki protonów z reszkami gazu obecnymi w rurach późniowych.
 - T0 – zespół dwóch liczników Cherenkowa, który dostarcza dokładny czas interakcji potrzebny dla detektora TOF, pozwala także na śledzenie świetlności w czasie rzeczywistym.

273 2 Uczenie maszynowe

274 Uczenie maszynowe jest bardzo szerokim i obecnie dynamicznie się rozwijającym obszarem
275 nauki. Występuje w wielu odmianach łącząc w sobie w zależności od wariantu wiele dziedzin
276 takich jak matematyka (statystyka, algebra) informatyka (algorytmika, teoria informacji) a
277 także elementy robotyki i sterowania. Dziedzinami, w których jest najczęściej wykorzystywane
278 są min. widzenie maszynowe, przetwarzanie języka naturalnego, autonomiczne roboty i pojazdy,
279 systemy decyzyjno - eksperckie, optymalizacyjne oraz rekomendacyjne.

280 W tej pracy wykorzystywana jest gałąź uczenia maszynowego nazywana uczeniem nadzorowanym lub "uczeniem z nauczycielem" (ang. *supervised learning*), gdzie uczenie występuje na podstawie poprawnie oznaczonych przykładów. Terminami bliskoznacznymi dla tak rozumianego uczenia maszynowego są uczenie statystyczne (ang. *statistical learning*) i rozpoznawanie wzorców (ang. *pattern recognition*).

285 Problem identyfikacji dżetów jest klasycznym przykładem zagadnienia klasyfikacji, gdzie
286 poprawna odpowiedź jest jedną ze skończonej ilości opcji (klas) w przeciwieństwie do regresji,
287 gdzie szukana odpowiedź algorytmu ma charakter ciągły.

288 Występuje wiele algorytmów uczenia maszynowego takich jak regresja liniowa i logistyczna,
289 drzewa decyzyjne i ich wariacje, maszyny wektorów wspierających, sztuczne sieci neuronowe
290 oraz wiele innych [26], [27]. Uczenie polega na znalezieniu pewnej funkcji dopasowującej do
291 przyjmowanego na wejściu zestawu (wektora) cech (zmiennych, kolumn) pewną odpowiedź
292 (predykcję), która minimalizuje zadaną funkcję straty. Jej rolę w przypadku regresji często
293 pełni błąd średniokwadratowy a w przypadku klasyfikacji np. entropia krzyżowa (ang. *cross*
294 *entropy*)². Różne algorytmy szukają przy tym funkcji dopasowującej należącej do różnych klas
295 funkcji: przykładowo klasyczne drzewa decyzyjne przeszukują tylko przestrzeń funkcji dających
296 się opisać skończonym zbiorem reguł "jeśli – to" (ang. *if – else*).

297 W pracy wykorzystane zostały dwa rodzaje algorytmów: wzmacniane drzewa decyzyjne oraz
298 sieci neuronowe.

299 2.1 Wzmacniane drzewa decyzyjne

300 Wzmacniane drzewa decyzyjne są jednym z rozwinięć klasycznego algorytmu drzewa decyzyjnego. Pojedyncze drzewo decyzyjne dzieli przestrzeń cech uczących przy pomocy prostopadłych
301 cięć, na mniejsze/większe niż zadana wartość w przypadku zmiennej ciągłej lub na należące/nie
302 należące do danej klasy w przypadku zmiennej kategorycznej. Każdy podział, nazywany węzłem,
303 daje dwie gałęzie, które można dalej niezależnie dzielić aż do ostatniego poziomu (liści). Kolejne podziały wybierane są tak, aby zbiory przykładów wpadające do poszczególnych gałęzi
304 były jak najbardziej jednorodne. Stosuje się różne miary nieporządku takie tak: indeks Gini G_L
305 lub entropia S_L ³.

306 Drzewa decyzyjne są często łączone w komitety klasyfikatorów (ang. *ensemble methods*). Wiele „słabych” klasyfikatorów jest łączonych w jeden „silny” na dwa sposoby: workowanie
307 (ang. *bagging*) [28] oraz wzmacniania (ang. *boosting*) [29], które są często ze sobą porównywane.

308 *Bagging* – w zastosowaniu dla drzew decyzyjnych nazywany algorymem lasów losowych
309 (ang. *random forest*) - polega na wytrenowaniu wielu drzew, każdego na podstawie N przykładów wylosowanych z powtórzeniami spośród N -licznego zbioru treningowego. Dodatkowo, do

² $J = -\frac{1}{N} \sum_{i=1}^N [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$, gdzie y_i to prawidłowa klasa i -tego przykładu a \hat{y}_i to predykcja algorytmu

³ $G_L = 1 - \sum_k p_k^2$ oraz $S_L = \sum_k -p_k \log p_k$, gdzie p_k to stosunek liczby przypadków klasy k do liczby wszystkich przypadków w liściu L

314 uczenia każdego drzewa używa się tylko podzbioru wszystkich cech uczących. Końcową predy-
315 cję algorytmu otrzymuje się poprzez „głosowanie” wszystkich drzew z odpowiednimi wagami.

316 *Boosting* – wzmacniane drzewa decyzyjne (ang. *boosted decision trees*) – jest metodą po-
317 dobną do *baggingu*. Główną różnicą jest zwiększenie wag przykładom uczącym, które przez po-
318 przednie drzewo zostały źle zaklasyfikowane – każde kolejne drzewo koncentruje się bardziej na
319 poprawie błędów poprzednich drzew. Widać tu kolejną ważną cechę odróżniającą obie metody:
320 *boosting* jest algorytmem sekwencyjnym podczas gdy *bagging* daje się trywialnie zrównoleglić
321 (każde drzewo trenowane jest w osobnym wątku).

322 Parametry i sposób trenowania drzew decyzyjnych na analizowanych danych

323 W niniejszej pracy wykorzystano wzmacniane drzewa decyzyjne zaimplementowane w wydaj-
324 nej bibliotece **XGBoost** [30]. Szybkość obliczeń jest bardzo ważna, gdyż oprócz komfortu pracy
325 z algorytmem, przekłada się na jakość otrzymanych wyników – krótszy czas obliczeń oznacza
326 możliwość przeprowadzenia większej ilości eksperymentów i lepsze dobranie parametrów oraz
327 danych. Implementacja wzmacnianych drzew decyzyjnych w **XGBoost** wykorzystuje wszystkie
328 rdzenie procesora, pomimo że sam algorytm ma charakter sekwencyjny – jest to możliwe dzięki
329 paralelizacji procesu tworzenia każdego drzewa (przed każdym podziałem konieczne jest spraw-
330 dzenie pewnej ilości możliwych zmiennych i wartości progowych i ten proces jest wykonywany
331 równolegle).

332 Dzięki szybkiemu uczeniu się algorytmu, możliwe było użycie kosztownego obliczeniowo au-
333 tomatycznego przeszukiwania przestrzeni parametrów przy pomocy przeszukiwania losowego
334 (ang. *random search*), które jest zwykle preferowane nad przeszukiwanie sieciowe [31]. W tym
335 celu cały zbiór danych dzielony był na dwie części: trenującą oraz testową (80/20%). Następ-
336 nie algorytm był trenowany i oceniany z użyciem trzy- lub pięciokrotnej walidacji krzyżowej
337 (ang. *cross-validation*) na zbiorze trenującym dla różnych zestawów parametrów. Model z naj-
338 lepszym wynikiem uzyskanym w walidacji krzyżowej był sprawdzany na zbiorze testowym.

339 Parametry optymalizowane w opisanym procesie to:

- 340 • *max_depth* – maksymalna głębokość każdego drzewa (niekoniecznie osiągana)
- 341 • *n_estimators* – liczba drzew
- 342 • *learning_rate* – parametr szybkości uczenia, komplementarny do *n_estimators*, w praktyce
343 można ustalić liczbę drzew i szukać optymalnej szybkości uczenia
- 344 • *subsample*, *colsample_bytree*, *colsample_bylevel* – parametry regularyzacyjne określające
345 ułamek kolejno: wierszy użytych do trenowania każdego drzewa, kolumn użytych w ka-
346 żdym drzewie (cechy losowane raz dla danego drzewa), kolumn użytych przy każdym po-
347 dziale (cechy losowane przy każdym podziale)
- 348 • γ – minimalny zysk w postaci zmniejszenia wartości funkcji straty konieczny do wykonania
349 podziału

350 2.2 Sieci neuronowe

351 Sieci neuronowe (ang. *neural networks* – *NN*) są szczególnym algorytmem uczenia maszyno-
352 wego. Występują w bardzo wielu odmianach i są wykorzystywane w rozwiązywaniu szerokiej
353 gamy problemów. Nawet bardzo pobiczny opis sieci neuronowych wymaga dużo więcej miejsca
354 niż może być temu poświęcone w tej pracy. Wprowadzenia do sieci neuronowych od podstaw
355 można znaleźć m.in. w [32] lub [33]. Tu przedstawione zostaną wyłącznie wybrane zagadnienia

356 mające ścisłejšzy związek z pracą. Używane mogą być terminy, których znaczenie wyjaśniane
357 jest w podanych źródłach.

358 W niniejszej pracy, wykorzystane zostały dwa rodzaje sieci neuronowych: sieci w pełni połącz-
359 czone (ang. *fully connected NN – FC NN*), nazywane także wielowarstwowymi perceptronami
360 (ang. *multi-layer perceptron – MLP*) oraz sieci konwolucyjne (ang. *convolutional NN – Co-*
361 *nvNets, CNN*).

362 Sieci w pełni połączone

363 W nierekurencyjnych sieciach neuronowych (tylko takie są używane w tej pracy), informacja jest
364 przekazywana kolejno od warstw wejściowych, poprzez warstwy ukryte aż do wyjściowej. W sie-
365 ciach typu *FC* wszystkie warstwy składają się z identycznych neuronów – każdy neuron dostaje
366 na wejściu wektor, natomiast zwraca skalar – wartość pewnej zadanej, nielinowej funkcji, jako
367 argument podając średnią ważoną z elementów wektora wejściowego. Wartości zwracane przez
368 neurony w danej warstwie składają się na wektor wejściowy dla neuronów kolejnej warstwy.
369 Wejściem dla pierwszej warstwy są natomiast kolejne przykłady ze zbioru uczącego. Trenowa-
370 nie sieci neuronowych polega na zmienianiu wag (parametrów) w liczonej w każdym neuronie
371 średniej, każdy neuron posiada własny, niezależny zestaw wag.

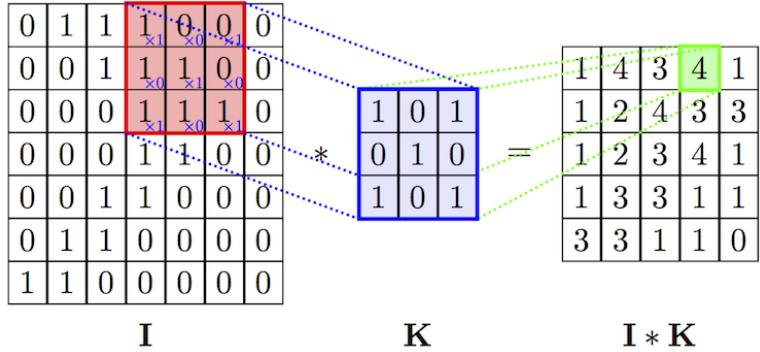
372 Istnieje twierdzenie o sieciach neuronowych jako uniwersalnych aproksymatorach funkcji
373 (ang. *universal approximation theorem*) [34], mówiące, że już sieć neuronowa o jednej warstwie
374 ukrytej jest zdolna do przybliżenia dowolnej funkcji z dowolną dokładnością. Twierdzenie to
375 nie podaje niestety liczby potrzebnych neuronów a przede wszystkim – sposobu ich trenowania.
376 Trenowanie jest prostsze w przypadku zastosowania wielu warstw, które odpowiadają kolejnym
377 poziomom abstrakcji jednak nadal jest dużym wyzwaniem ze względu na fakt, że nawet stosun-
378 kowo niewielka sieć może posiadać bardzo dużą liczbę parametrów, przykładowo sieć o czterech
379 warstwach, w każdej po 128 neuronów ma ich ponad 65 tysięcy.

380 Sieci konwolucyjne

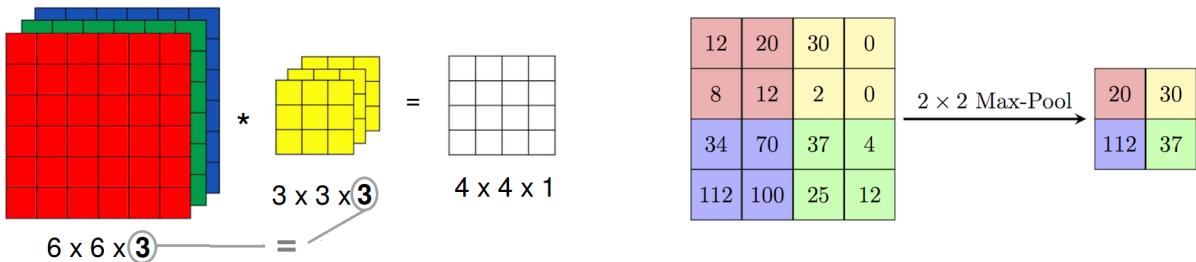
381 Jednym ze sposobów na ograniczenie liczby trenowanych parametrów jest użycie konwolucyj-
382 nych sieci neuronowych [35] (bardziej poprawną choć rzadko używaną nazwą w języku polskim
383 jest sieć splotowa). Są one inspirowane połączeniami w korze wzrokowej zwierząt i wywodzą
384 się z badań w obszarze widzenia komputerowego, gdzie liczby parametrów są szczególnie duże
385 (wektor wejściowy ma wymiar równy liczbie pikseli w obrazie), na takim przykładzie również
386 najłatwiej zrozumieć ich działanie.

387 Sieci konwolucyjne różnią się od sieci typu *FC* tym, że część wag połączeń między warstwami
388 jest dzielona. Występuje w nich nowy rodzaj warstwy, nazywany warstwą konwolucyjną. Każda
389 jednostka w warstwie konwolucyjnej (filtr) ma pewną stałą (niewielką) liczbę wag. Połączenie z
390 dużym wejściem realizowane jest przez powielanie tych samych wag w połączeniach z kolejnymi
391 fragmentami wektora wejściowego (Rys. 4). Rezultatem działania filtra na macierz jest wynik
392 operacji splotu. Liczba parametrów przypadająca na każdy filtr jest równa jego rozmiarowi i
393 nie zależy od wielkości wektora wejściowego.

394 W przypadku gdy zamiast wejścia dwuwymiarowego (jak np. obraz czarno-biały), mamy do
395 czynienia z wejściem trójwymiarowym (np. trzeci wymiar to kolejne kolory w kodowaniu RGB),
396 filtry również muszą mieć trzy wymiary, przy czym rozmiar w ostatnim wymiarze musi być
397 równy rozmiarowi w tym kierunku wektora wejściowego. Wynik operacji splotu jest ponownie
398 dwuwymiarowy, gdyż filtr przesuwany jest tylko w dwóch pierwszych wymiarach. Trzeci wymiar
399 powstaje przez składanie kolejnych filtrów. Widać zatem, że również w przypadku gdy na
400 wejścia podawany jest obraz czarno-biały, filtry w kolejnych warstwach konwolucyjnych (oprócz
401 pierwszej) mają po trzy wymiary.



Rysunek 4: Schemat działania pojedynczego filtra z warstwy konwolucyjnej (operacja splotu).
Źródło: [36].



Rysunek 5: Działanie pojedynczego filtra (3D) na wejście o trzech wymiarach.
Źródło: [37].

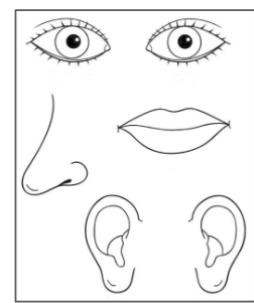
Rysunek 6: Działanie warstwy typu *max-pool*. Źródło: [36].

oprócz warstw konwolucyjnych, w sieciach tego typu stosowane są także tzw. warstwy typu *max-pooling*. Zasada jej działania jest bardzo prosta: wykonuje funkcję *maksimum* na zadanym fragmencie obrazu (Rys. 6). Ich rolą jest zmniejszanie rozmiaru przekazywanej w sieci informacji.

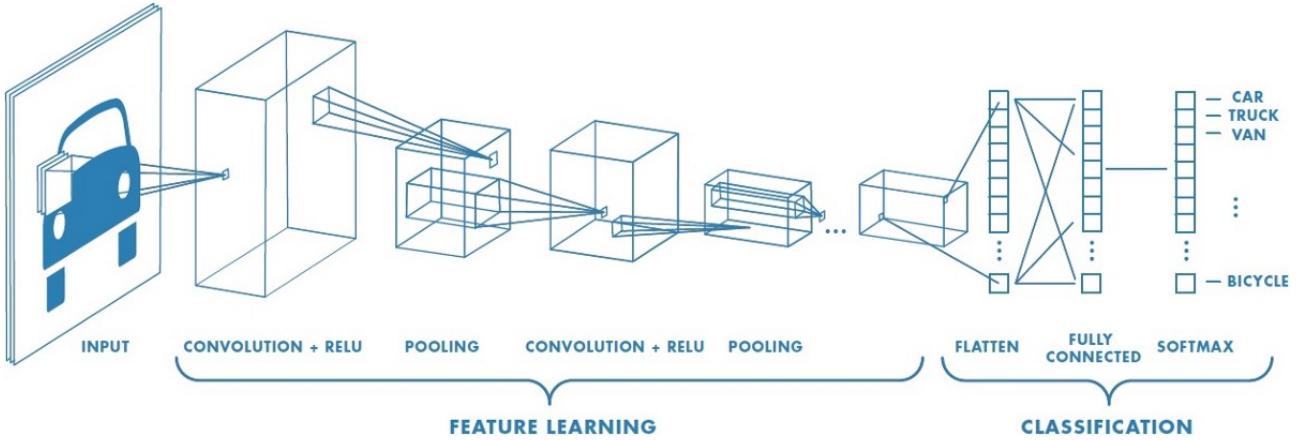
Typowa architektura stosowana w przypadku sieci konwolucyjnych jest następująca: najpierw warstwy konwolucyjne (pomiędzy nimi czasem warstwy typu *max-pool*), następnie wszystkie filtry są rozwijane i składane w długi jednowymiarowy wektor, który przekazywany jest do warstw typu *FC*. W przypadku problemu klasyfikacji, na końcu znajduje się jeszcze warstwa typu *softmax* normalizująca wyjście z sieci do jedynki (Rys. 8).

Sieci konwolucyjne posiadają dwie właściwości odróżniające je od *MLP*:

- niezmienniczość względem przesunięcia (ang. *translation invariance*) – głównie za sprawą dzielenia wag oraz obecności warstw typu *max-pool*, położenie danej cechy na obrazie jest niemal bez znaczenia (obraz po prawej stronie byłby rozpoznany jako twarz)
- lokalność połączeń – filtry obejmują tylko kilka sąsiednich pikseli (tam zwykle występują najsilniejsze zależności) nie są w stanie dostrzec cechy rozciągniętej na obszar większy od rozmiaru filtra



Rysunek 7: Źródło: [38].

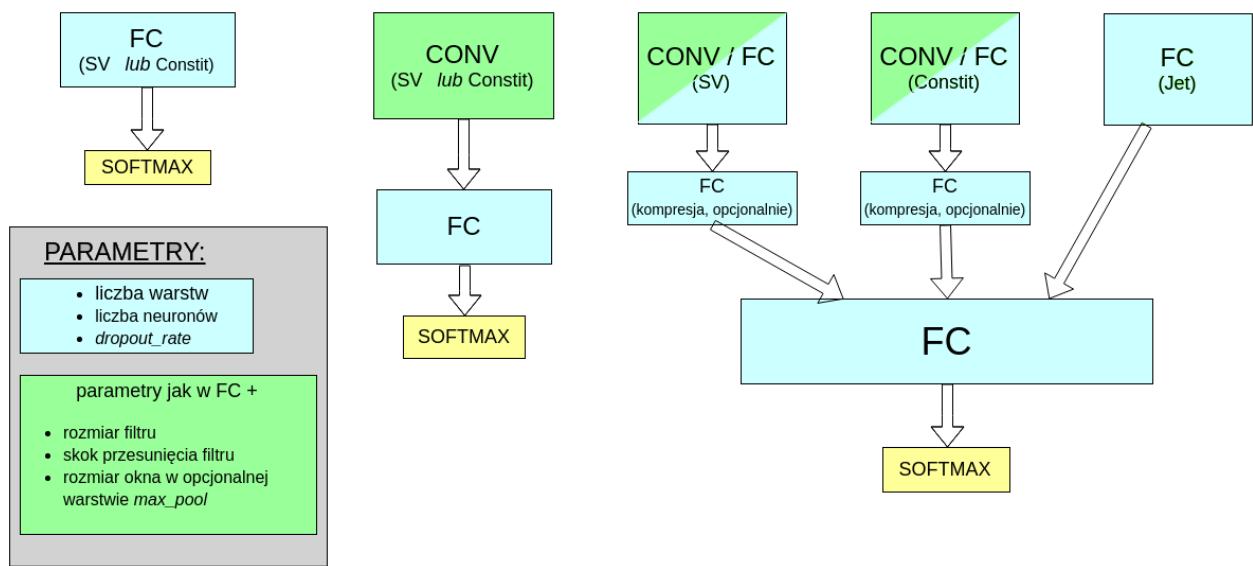


Rysunek 8: Typowa struktura stosowana w sieciach konwolucyjnych. Warstwy konwolucyjne mają za zadanie wydobywać cechy, na podstawie których późniejsze warstwy dokonują klasyfikacji. Widoczna jest charakterystyczna stopniowa zmiana rozmiaru przekazywanej macierzy: rozmiar poprzeczny maleje kosztem głębokości, co odpowiada rosnącej liczbie filtrów i malejącym rozmiarowi obszaru po jakim są one przesuwane. Źródło: [39].

422 Hiperparametry i trenowanie sieci neuronowych na analizowanych danych

423 Parametry sieci, których wartości są określane przez projektanta sieci, takie jak liczba warstw
424 ukrytych są nazywane hiperparametrami (dla odróżnienia od parametrów - wag połączeń).

425 Testowane były trzy architektury: sieci w pełni połączone, sieci konwolucyjne oraz sieć
426 złożona z dwóch gałęzi, osobnych dla wtórnych wierzchołków i częstek tworzących dżet (Rozdz.
427 3) przedstawione schematycznie na Rys. 9. Do trenowania sieci wykorzystano wysokopoziomową
428 bibliotekę Keras [40] korzystającą z silnika obliczeniowego zaimplementowanego w TensorFlow
429 [41].



Rysunek 9: Schematyczne przedstawienie trzech testowanych rodzin architektur sieci. Każdy blok odpowiada kilku warstwom danego typu. Bloki warstw typu **FC** i opisane jako „kompresja” składały się z kilku neuronów i miały za zadanie zredukować całą informację z danej gałęzi do wektora kilku liczb.

430 Zestaw hiperparametrów definiujący działanie sieci w pełni połączonej:

- liczba warstw ukrytych

- liczba neuronów w każdej warstwie

433 • funkcja aktywacji – nielinowa funkcja aplikowana przed zwróceniem wartości w każdym
434 neuronie, najpopularniejsze to *tanh*, *ReLU* ($f(x) = \max(0, x)$) oraz funkcja sigmoidalna
435 ($f(x) = \frac{1}{1+\exp(-x)}$)

- algorytm optymalizacyjny – spadek gradientowy lub jego wariacje

- parametr szybkości uczenia i jego modyfikacje w trakcie uczenia

438 • liczba przykładów trenujących przetwarzanych w jednym kroku uczenia (ang. *batch_size*)
439 – im większy tym szybsze jest trenowanie sieci (dzięki wydajnym operacjom macierzo-
440 wym), natomiast może się to odbywać kosztem precyzji

- liczba epok uczenia – ile razy będzie pokazywany sieci każdy przykład

442 • opcjonalnie: warunki stopu (ang. *early stopping*), czyli przerwanie procesu uczenia mające
443 na celu uniknięcie przetrenowania sieci, w momencie gdy błąd popełniany na zbiorze
444 walidacyjnym przestaje maleć

- opcjonalnie: regularyzacja przy pomocy różnych technik (zwykle konieczna)

446 Ponadto dla sieci konwolucyjnych:

- liczba warstw konwolucyjnych i liczba filtrów w każdej warstwie

- obecność lub brak warstw *max-pool* i rozmiar ich okna

- rozmiar filtrów i długość skoku przy ich przesuwaniu

450 Same dwie pierwsze wielkości dają nieograniczoną liczbę konfiguracji. Czas trenowania sieci
451 neuronowych jest rzędu wielkości większej niż drzew decyzyjnych, dlatego przyjęto szereg kro-
452 ków mających na celu zmniejszenie przeszukiwanej przestrzeni hiperparametrów. Na podstawie
453 wstępnych testów oraz różnych wskazówek dostępnych w literaturze przyjęto:

- *batch_size* zawsze równy 64 (inne testowane wartości: 16, 32, 128)

455 • za algorytm optymalizacyjny przyjęto algorytm o nazwie *Nadam* [42], tj. rozwinięcie al-
456 gorytmu *Adam* [43] o parametr Nesterova (inne testowane to zwykły spadek gradientowy
457 oraz *Adam*)

- funkcję aktywacji: *ReLU*

- liczba epok równa 50, 100 lub 200, zrezygnowano z *early stopping*

- stałe w trakcie treningu wartości parametru szybkości uczenia

461 • spośród technik regularizacyjnych testowano wyłącznie *dropout* [44] z prawdopodobień-
462 stwem odrzucenia równym 0.1, 0.2 lub 0.5

- kilka wybranych kombinacji dla zestawu parametrów: rozmiar filtra, długość skoku i roz-
463 miar okna w warstwach *max-pool* – takie same w kolejnych warstwach

- 465 ● liczby neuronów/filtrów w warstwach będące zawsze potęgami dwójki oraz stałą liczbę w
466 kolejnych warstwach lub zmieniającą się o stały czynnik, np. 256-128-64, 128-128-128 lub
467 16-32-64
- 468 ● liczba warstw *FC*: 2-8, konwolucyjnych 2-6

469 Nawet po przyjęciu powyższych uproszczeń nie sposób sprawdzić wszystkich możliwych
470 zestawów hiperparametrów, dlatego sposób ich dobierania w kolejnych testach był mocno em-
471 piryczny. Dostępne dane dzielone były na trzy zbiory: trenujący, walidacyjny i testowy. Wobec
472 braku warunków stopu, zbiór walidacyjny użyty był wyłącznie do porównywania różnych ze-
473 stawów parametrów, tak aby wynik testowy pozostał nieobciążony.

474 Zgodnie z zasadą ortogonalizacji działań, proces doboru hiperparametrów dzielono na dwie
475 części: najpierw starano się uzyskać jak najlepsze wyniki na zbiorze uczącym, a dopiero później
476 zmusić algorytm do lepszej generalizacji na zbiorze testowym przez zwiększoną regularyzację i
477 modyfikację parametru szybkości uczenia.

478 **2.3 Dyskusja użycia dwóch algorytmów**

479 Użycie więcej niż jednego algorytmu ma wiele zalet. Po pierwsze daje możliwość porównania
480 wyników. Pozwala to na oszacowanie błędu *bayesowskiego* (najniższego możliwego do osiągnię-
481 cia na danym zestawie danych przez jakikolwiek algorytm błędu). Jest to bardzo ważne w
482 sytuacji, gdy nie dysponuje się innym oszacowaniem tego błędu (w wielu problemach natural-
483 nych dla człowieka jak rozpoznawanie obiektów na obrazkach jest nim błąd ludzki lub też błąd
484 popełniany przez zespół ekspertów w bardziej zaawansowanych zastosowaniach).

485 Po drugie, wykorzystane zostały dwa algorytmy mocno różniące się w swojej naturze, co po-
486 zwala wykorzystać cechy każdego z nich w analizie: przykładowo sieci neuronowe dobrze radzą
487 sobie z nieustrukturyzowanymi danymi – potrafią tworzyć wysokopoziomowe cechy na podsta-
488 wie niskopoziomowego wejścia (np. położenia oka na zdjęciu twarzy na podstawie pixeli). Są
489 natomiast trudne w interpretacji i często traktowane są jako tzw. „czarne skrzynki” (ang. *black*
490 *box*). Oprócz tego, liczba możliwych konfiguracji sieci jest ogromna i przez to niemożliwe jest
491 stwierdzenie czy wykorzystane zostały pełne ich możliwości.

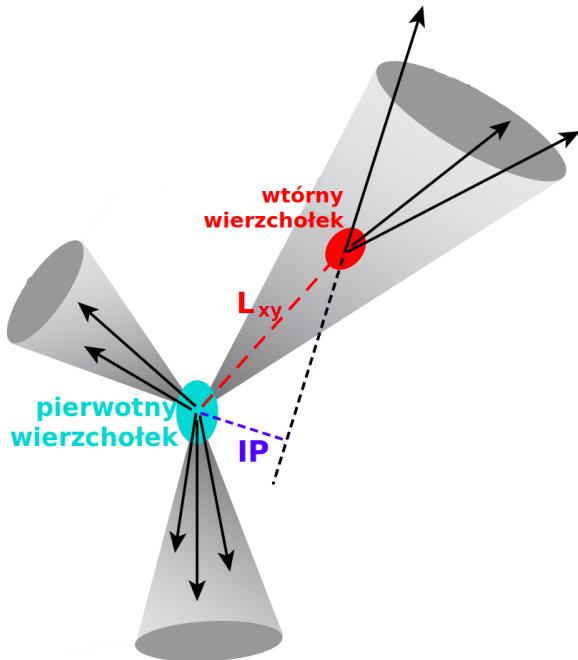
492 Z kolei drzewa decyzyjne posiadają stosunkowo niewielką liczbę parametrów, a ich trenowa-
493 nie jest bardzo szybkie co pozwala na ich ekstensywne przeszukiwanie i otrzymanie wyników,
494 które można uznać za optymalne dla tego algorytmu. Ponadto, w przypadku drzew istnieją
495 niewymagające dodatkowych obliczeń miary użyteczności poszczególnych zmiennych, co daje
496 wgłębne w działanie algorytmu i poprawia intuicyjne zrozumienie jego predykcji.

497 3 Dane

498 Dane użyte w analizie pochodzą z symulacji Monte Carlo zderzeń proton-proton przy energii w
 499 układzie środka masy równej $\sqrt{s} = 13$ TeV dostępnych na serwerach eksperymentu ALICE. Są
 500 to pełne symulacje detektora ALICE, wykorzystujące generator zderzeń Pythia8 [45] (wersja:
 501 Monash2013 [46]) oraz pakiet Geant3 [47] do transportu cząstek przez materiał detektora.
 502 Korzystano ze specjalnych symulacji w tzw. *binach p_T*, które zapewniają lepszą statystykę dla
 503 wysokich pędów.

504 Do rekonstrukcji dżetów wykorzystany został algorytm *anti-kt* z parametrem $R = 0.4$ zaim-
 505 plementowany w pakiecie FASTJET [48]. Dżetów poszukiwano wyłącznie wśród cząstek nałado-
 506 wanych (ang. *charged jets*) ze względu na słabe pokrycie przestrzeni fazowej przez kalorymetry
 507 w eksperymencie ALICE.

508 Do analizy wybrano dżety o p_T większym niż 15 GeV i mieszczące się w całości w akcep-
 509 tancji detektora TPC, tj. $|\eta| < 0.9$, co przy użytym parametrze rozmiaru dżetu $R = 0.4$, daje
 510 ograniczenie na pseudopospieszność $|\eta| < 0.5$ dla osi dżetu.



Rysunek 10: Rysunek ilustrujący znaczenie używanych wielkości: L_{xy} oraz IP . Źródło: [49].

511 Dla każdego dżetu obliczony został szereg wielkości, które można podzielić na zmienne
 512 charakteryzujące dżet, opisujące wtórne wierzchołki lub cząstki tworzące dżet. Za potencjalne
 513 wtórne wierzchołki uznaje się kombinacje trzech cząstek spełniających pewne dosyć luźne kry-
 514 teria jak $p_T > 0.15$ GeV (rozważane są wyłącznie trzy-cząstkowe wtórne wierzchołki), stąd ich
 515 liczba może być dużo większa od liczby cząstek tworzących dżet.

516 Lista używanych zmiennych:

- 517 • Zmienne charakteryzujące dżet:

518 – η_{jet}, ϕ_{jet} – pseudopospieszność i kąt azymutalny osi dżetu

519 – $p_{T,jet}$ – pęd poprzeczny dżetu

520 – $M_{jet} = \sqrt{(\sum_i E_i)^2 - (\sum_i \vec{p}_i)^2}$ – masa dżetu, gdzie E_i i \vec{p}_i to energia i pęd kolejnych
 521 cząstek tworzących dżet

522 – A_{jet} – powierzchnia dżetu wyliczana przez algorytm kt w płaszczyźnie (η, ϕ) . Do po-
 523 wierzchni dżetu zaliczany jest każdy element powierzchni, w którym dodanie części-
 524 o nieskończenie małym pędzie poprzecznym sprawi, że zostanie ona zaliczona do tego
 525 dżetu [50]

526 – ρ_{bckg} – gęstość tła w danym zdarzeniu

527 • Zmienne opisujące cząstki tworzące dżet (składniki dżetu):

528 – η, ϕ – pseudopospieszność i kąt azymutalny cząstki względem osi dżetu

529 – p_T – pęd poprzeczny cząstki

530 – IP_D – rzut na kierunek poprzeczny wektora IP , tj. odległość najbliższego zbliżenia

531 – IP_Z – rzut na oś z wektora IP

532 – $N_{Constit}$ – liczba cząstek tworzących dżet

533 • Zmienne opisujące wtórne wierzchołki:

534 – L_{xy} – odległość między pierwotnym a wtórnym wierzchołkiem (ang. *decay length*)
 535 w płaszczyźnie $x - y$

536 – $\sigma_{L_{xy}}$ – niepewność wyznaczenia L_{xy}

537 – $\sigma_{vertex} = \sqrt{d_1^2 + d_2^2 + d_3^2}$ – rozrzut śladów (ang. *tracks*) wokół wtórnego wierzchołka,
 538 gdzie d_i to najmniejsza odległość pomiędzy śladem i -tej cząstki a wtórnym wierz-
 539 chołkiem

540 – $M_{inv} = \sqrt{(E_1 + E_2 + E_3)^2 - (\vec{p}_1 + \vec{p}_2 + \vec{p}_3)^2}$ – masa niezmiennica wierzchołka, gdzie
 541 E_i, \vec{p}_i to energia i pęd i -tej cząstki tworzącej wierzchołek

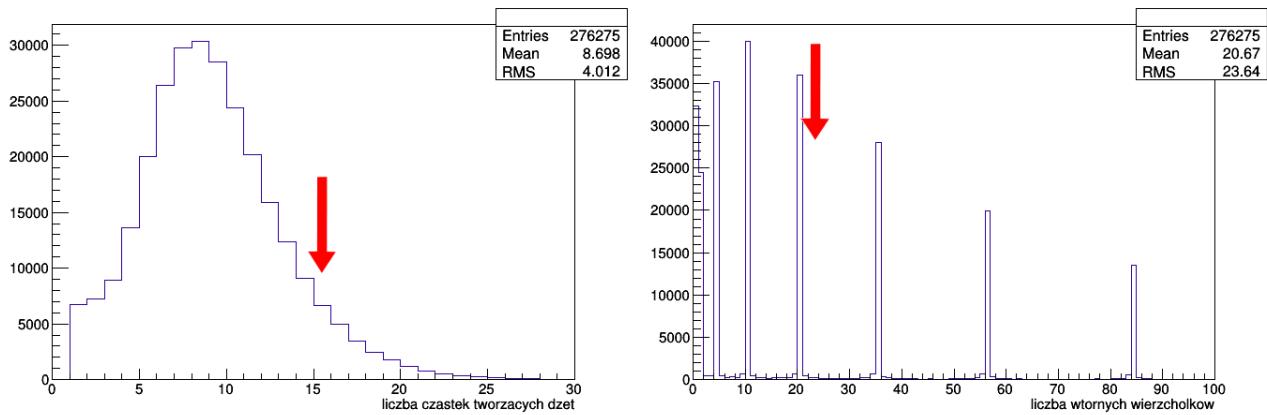
542 – χ^2/Ndf – jakość dopasowania wtórnego wierzchołka

543 – N_{SV} – liczba wtórych wierzchołków

544 Dżety różnią się liczbą cząstek je tworzących oraz liczbą wtórych wierzchołków. Większość
 545 algorytmów uczenia maszynowego wymaga natomiast dostarczenia danych w postaci tabelo-
 546 rycznej (macierzowej), ze stałą liczbą kolumn (wiersze stanowią kolejne dżety). Aby spełnić to
 547 wymaganie konieczne jest przyjęcie pewnej ustalonej liczby wtórych wierzchołków oraz cząstek
 548 tworzących dżet – w przypadku gdy dżet ma więcej elementów tego typu są one odrzucane, na-
 549 tomist puste pola są wypełniane zerami w przypadku gdy ma ich mniej. Po przeanalizowaniu
 550 rozkładów liczby wtórych wierzchołków i cząstek tworzących dżet (Rys. 11) oraz wstępny-
 551 sprawdzeniu jak dodawanie kolejnych elementów wpływa na otrzymywane wyniki (na podsta-
 552 wie wzmacnianych drzew decyzyjnych ze względu na wspomnianą w 2.3 szybkość i stabilność)
 553 przyjęto liczbę cząstek tworzących dżet równą 15 a liczbę wtórych wierzchołków równą 20.

554 Istotnym zagadnieniem jest także kolejność w jakiej ułożone będą zmienne. Dla sieci konwo-
 555 lacyjnych szukających lokalnych zależności dobrze jest pogrupować zmienne, tak aby obok siebie
 556 znajdowały się te same wielkości fizyczne, np. $L_{xy,1}, L_{xy,2}, L_{xy,3} \dots \sigma_{vertex,1}, \sigma_{vertex,2}, \sigma_{vertex,3} \dots$
 557 – tak też zrobiono. Dla sieci w pełni połączonych oraz drzew decyzyjnych kolejność zmiennych
 558 nie ma znaczenia, ale ważne jest aby położenie zmiennych było ustalone dla kolejnych wierz-
 559 chołków lub cząstek tzn. żeby L_{xy} i $\sigma_{L_{xy}}$ danego wtórnego wierzchołka były w tych samych
 560 miejscach, tak aby możliwe było szukanie zależności między nimi.

561 Nastecną kwestią jest wybór wielkości decydującej o kolejności ułożenia elementów, tj. która
 562 cząstka będzie cząstką nr 1 a która nr 5. Losowe ułożenie elementów sprawioby, że bezpośrednie
 563 porównywanie wartości w danej kolumnach (co ma miejsce bezpośrednio w drzewach decyzyj-
 564 nych a pośrednio w sieciach neuronowych) straciłoby sens. Z kolei dobór tej kolejności



Rysunek 11: Rozkłady liczby cząstek tworzących dżet i liczby wtórnych wierzchołków (pokazana jest tylko część, rozkład sięga $N_{SV} = 200$) wraz wartościami cięć.

pozwala na łatwe odtworzenie przez algorytm uczenia maszynowego motywów fizycznie algorytmów omówionych w sekcji 1.4.2. Przykładowo cięcie na wartość IP drugiej lub trzeciej cząstki (gdy są one posortowane wg malejących wartości IP) jest istotą algorytmu nazywanego *Track Counting – TC* stosowanego w CMS i ALICE. Kolejność w jakiej ułożone będą elementy, wpływa także na to, które z nich będą odrzucone w przypadku gdy dżet zawiera więcej niż 15 cząstek lub 20 wierzchołków. Ponownie posiliłowano się testami z użyciem drzew decyzyjnych. Ostatecznie wtórne wierzchołki ułożono według malejącego L_{xy} a cząstki – malejącego p_T .

Zbiór danych użytych w analizie liczył 46 000 dżetów, z czego ponad 20 tysięcy stanowiły dżety b (trening przeprowadzano przy zrównoważonej liczbie przykładów dla sygnału i tła). Z tego zbioru zbudowano trzy zestawy danych (każdy zawierał wszystkie 46 tys. wierszy, różniły się kolumnami): SV , który zawierał tylko zmienne związane z wtórnymi wierzchołkami, $constit$ – zmienne opisujące cząstki tworzące dżet (ang. *constituents*) oraz $merged$ – wszystkie zmienne.

577 4 Analiza

578 4.1 Dobór metryki

579 Bardzo ważnym elementem w trenowaniu algorytmów uczenia maszynowego jest dobór odpowiednich metryk. Kilka najczęściej używanych metryk wymieniono w Tab. A1. Klasycznym 580 złym przykładem jest używanie dokładności (ang. *accuracy*) do oceniania klasyfikacji binarnej 581 w przypadku dużego niezrównoważenia klas – algorytm przewidujący zawsze klasę większą 582 składową może osiągnąć dużą wartość dokładności będąc jednocześnie bardzo słabym modelem.

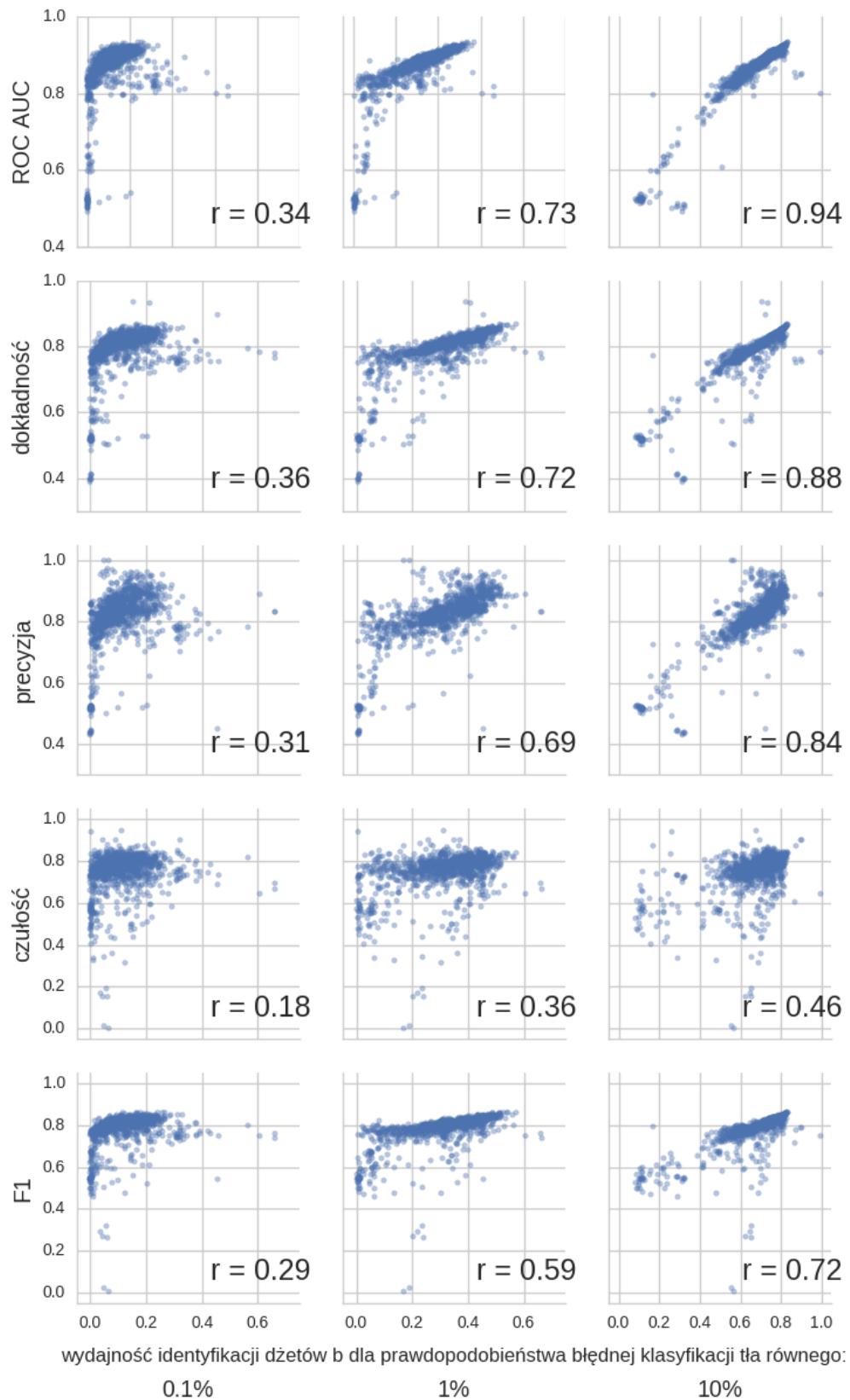
583 Używanie i porównywanie kilku miar efektywności jest często niepraktyczne dlatego dobrze 584 jest wybrać jedną metrykę. Przy jej wyborze należy kierować się potencjalnymi zastosowaniami 585 modelu. W tym przypadku są to analizy fizyczne, które mogą mieć różne wymagania dotyczące 586 czystości i liczebności otrzymywanych próbek, a co za tym idzie, preferować inne punkty 587 pracy zdefiniowane jako pary liczb: wydajność poprawnej klasyfikacji dżetów b (ang. *tagging* 588 *efficiency = true positive rate = recall*), i ułamek niepoprawnie zaklasyfikowanych przypadków 589 tła (ang. *mistagging rate = false positive rate*).

590 Naturalnym wyborem wydaje się pole pod powierzchnią krzywej *ROC* (ang. *ROC Area Under Curve – ROC AUC*) [51]. Potencjalną przeszkodą może być zakres rozsądnego wartości 591 prawdopodobieństwa błędnej klasyfikacji przypadków tła: dżety b stanowią tylko kilka procent 592 liczby wszystkich dżetów, zatem z punktu widzenia analizy dopuszczalne będą punkty pracy 593 zapewniające wydajność identyfikacji dżetów b ok. 10 – 100 razy większą niż częstość niepoprawnego 594 zaklasyfikowania przypadków tła. Oznacza to, że zdecydowana większość punktów 595 pracy znajdujących się na krzywej *ROC* jest nie do zaakceptowania – interesujące są tylko te 596 o najniższych częstościach błędnej klasyfikacji.

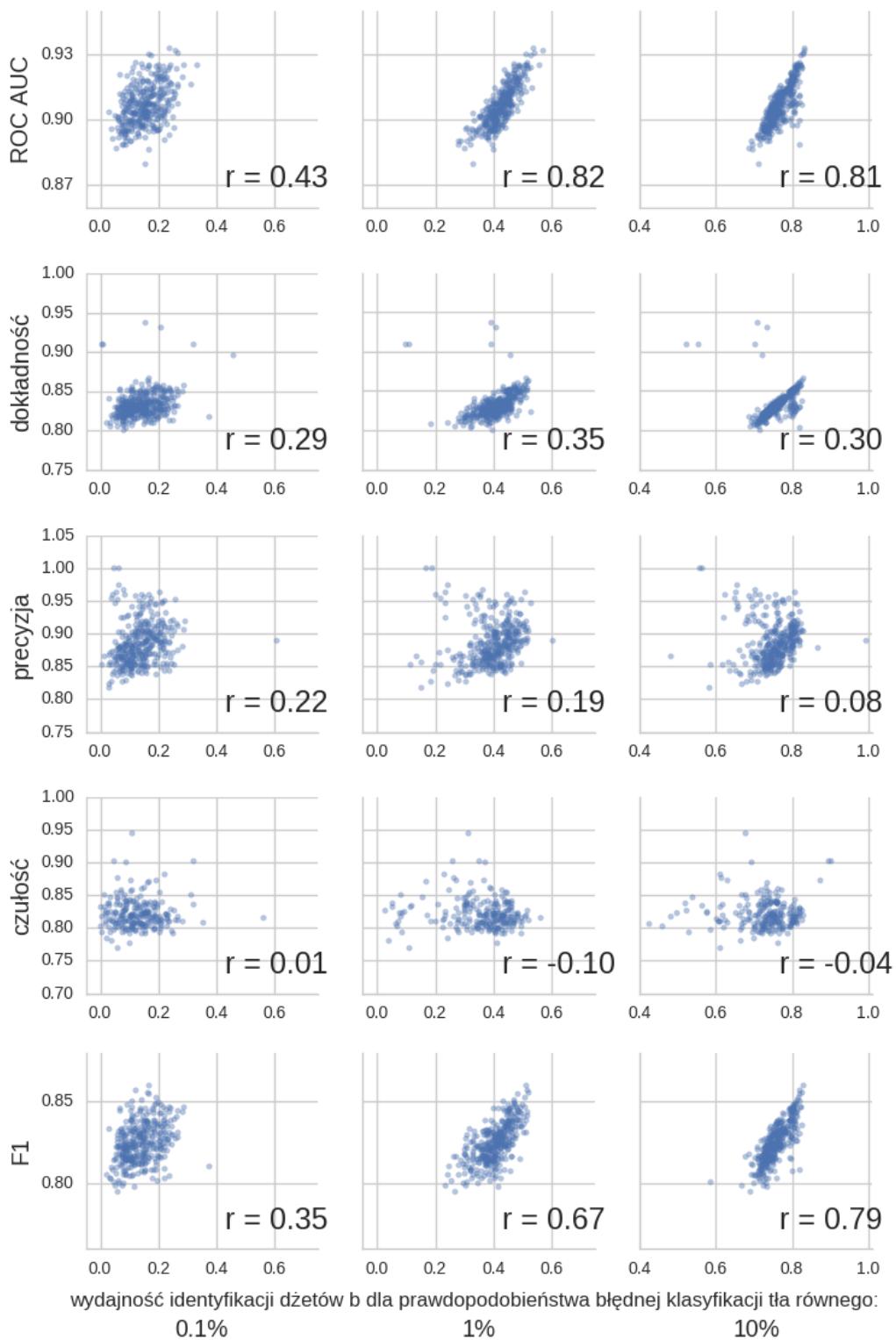
597 Aby ilościowo porównywać różne algorytmy wprowadzone zostaną trzy punkty pracy: o 598 prawdopodobieństwie błędnej klasyfikacji tła równej 0.1%, 1% oraz 10%. Na Rys. 12 przed- 599 stawione zostały zależności poszczególnych metryk od wydajności identyfikacji dżetów b w 600 tych trzech punktach pracy. Każdy punkt odpowiada jednemu eksperymentowi (dla dowolnego 601 algorytmu) przeprowadzonemu w trakcie przygotowywania analizy. Daje to pogląd, na to war- 602 tościami której metryki należy się kierować przy wyborze algorytmu i hiperparametrów, aby 603 zapewnić sobie jednocześnie wysokie wartości wydajności na identyfikację dżetów b w wybra- 604 nych punktach pracy. Najwyższe koreacje występują dla punktu pracy o najwyższym prawdo- 605 podobieństwie błędnej klasyfikacji tła – jak będzie to pokazane później wyniki dla tego punktu 606 pracy są najbardziej stabilne. Spośród analizowanych metryk najwyższe wartości współczyn- 607nika Pearsona otrzymano dla pola pod powierzchnią krzywej *ROC*, dosyć wysokie również dla 608 dokładności i precyzji. Widać, że wartości czułości są najsłabiej skorelowane z wydajnością iden- 609 tyfikacji dżetów b . Jest rzeczą zrozumiałą, że korelacja jest słabsza niż dla precyzji, ponieważ 610 wartości czułości maleją za każdym razem gdy błędnie klasyfikowane są dżety b stanowiące 611 sygnał, podczas gdy wartości precyzji maleją, gdy błędnie klasyfikowane są przypadki tła. Z 612 tych dwóch błędów, drugi jest bardziej kosztowny, gdyż błędna klasyfikacja nawet niewielkiej 613 części tła znacznie pogarsza czystość otrzymywanej próbki.

614 Na Rys. 13 przedstawiono te same wielkości co na Rys. 12, ale tym razem wybrano tylko 615 25% najwyższych (najlepszych) wartości dla każdej metryki – te punkty są bardziej znaczące, 616 gdyż ostatecznie modele z eksperymentów dających najlepsze wyniki będą używane. Dla takiej 617 selekcji otrzymano zdecydowaną dominację *ROC AUC* – wybór modeli dających najwyższe 618 pole pod powierzchnią krzywej *ROC* zapewnia jednocześnie otrzymanie wysokich wartości wy- 619 dajności identyfikacji dżetów b dla wybranych punktów pracy.

620 Pole pod powierzchnią krzywej *ROC* zostało wybrane jako główna metryka używana w 621 procesie dobierania parametrów modeli oraz przy prezentacji wyników.



Rysunek 12: Zależność podstawowych metryk od wydajności identyfikacji dżetów b dla punktów pracy o prawdopodobieństwie błędnej klasyfikacji tła równej 0.1%, 1% oraz 10%. Dla każdego wykresu przedstawiono współczynnik korelacji r Pearsona.



Rysunek 13: Rysunek podobny do Rys. 12, ale przedstawione zostały tylko punkty odpowiadające eksperymentom o wartościach metryki będących w górnym kwartylu wartości danej metryki dla wszystkich eksperymentów.

624 4.2 Wyniki dla poszczególnych modeli

625 W następnych podrozdziałach przedstawione zostały wyniki uzyskane dla poszczególnych mo-
626 deli. Jako model rozumiana jest para: algorytm (wraz z jego hiperparametrami) oraz zestaw
627 danych użyty do jego trenowania. Oznaczenia używane w prezentacji wyników zebrane zostały
628 w Dodatku B.

629 Każdy przedstawiony wynik jest rezultatem uśrednienia pięciu powtórzeń procedury, na
630 którą składa się: losowy podział zbioru danych na dane treningowe, walidacyjne i testowe oraz
631 uczenie algorytmu (randomizacji podlega także inicjalizacja wag połączeń w przypadku sieci
632 neuronowych).

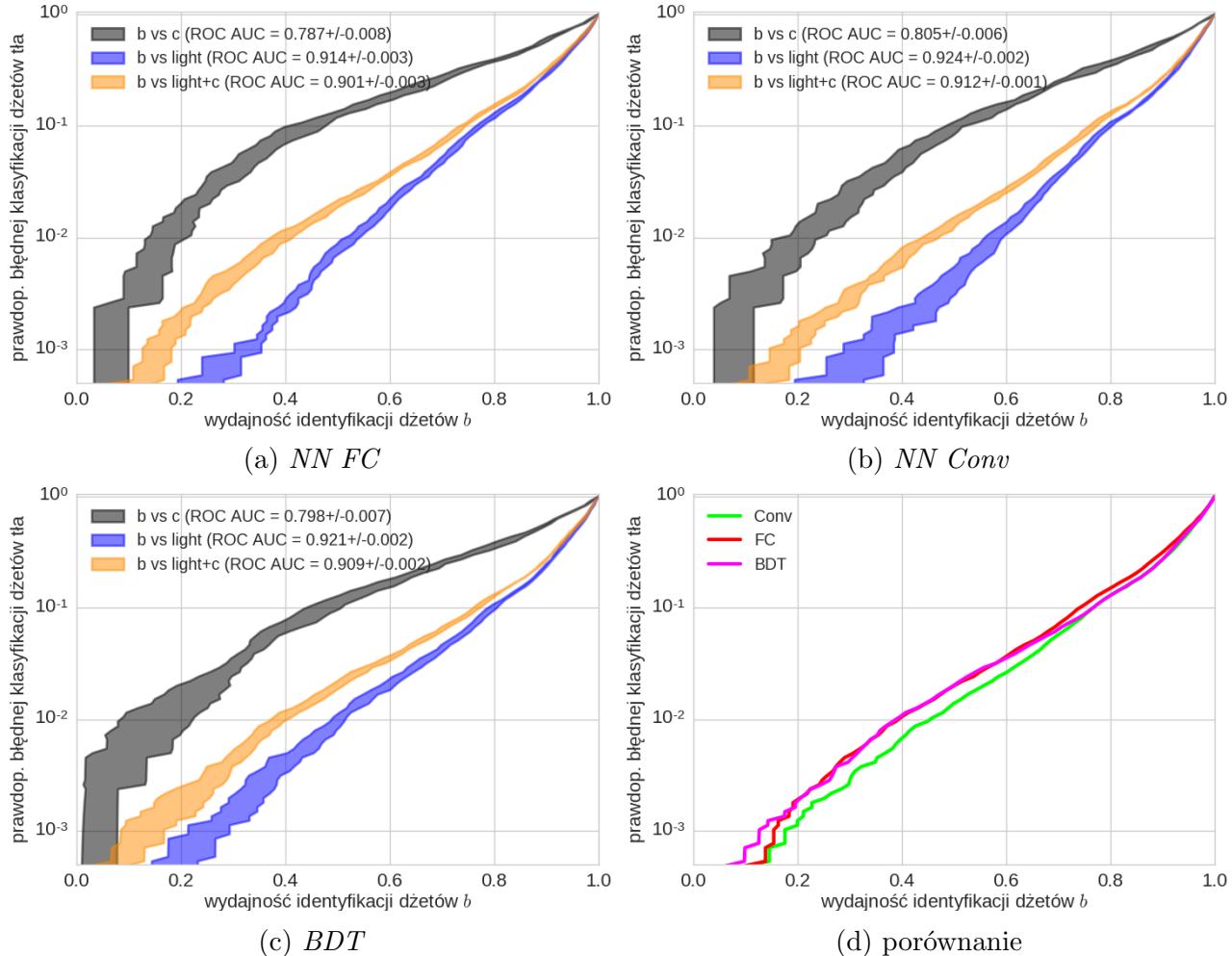
633 Na Rys. 14, 15, 16 przedstawiono odmianę krzywej *ROC* – wykresy przedstawiające zależ-
634 ności wydajności identyfikacji dżetów b z niepewnościami (na osi poziomej) dla danego praw-
635 dopodobieństwa błędnej klasyfikacji dżetów tła (na osi pionowej).

636 Trzy krzywe na każdym wykresie odpowiadają separacji dżetów b od dżetów: lekkich, c
637 oraz mieszanej próbki złożonej w 90% z dżetów lekkich i w 10% z dżetów powabnych. Cztery
638 wykresy na każdym rysunku odpowiadają trzem użytym algorytmom: sieciom neuronowym
639 typu *FC* (na górze po lewej), konwolucyjnym sieciom neuronowych (na górze po prawej) i
640 wzmacnianym drzewom decyzyjnym (na dole po lewej) oraz porównaniu wszystkich trzech
641 (na dole po prawej). Porównane zostały tylko krzywe odpowiadające separacji dżetów b od
642 mieszanego tła (bez niepewności).

643 Jeśli nie zaznaczono inaczej, prezentowane wyniki dotyczą separacji dżetów b od mieszanego
644 tła (90% dżetów lekkich + 10% c).

645 4.2.1 Wyniki dla zmiennych związanych z wtórnymi wierzchołkami

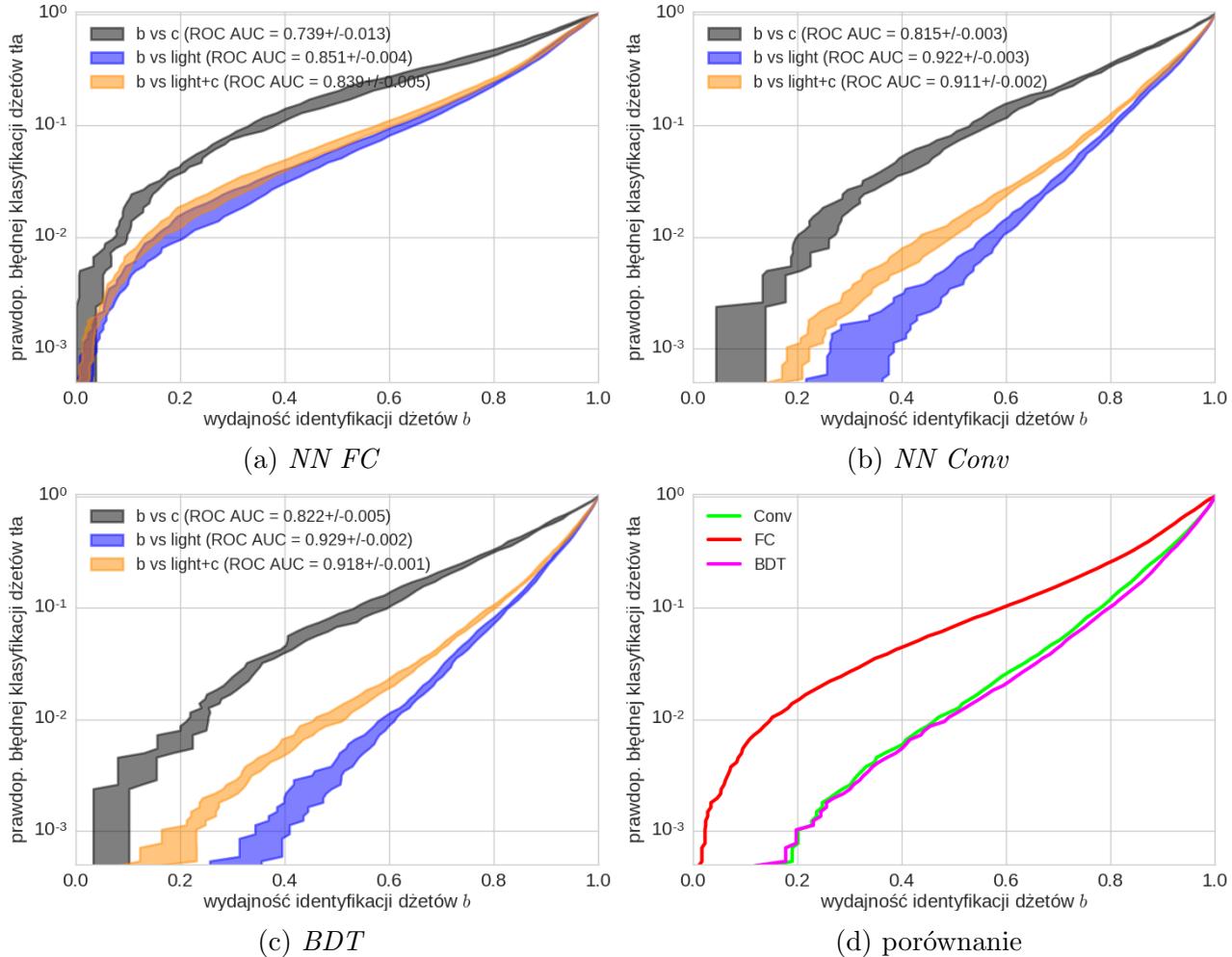
646 Na Rys. 14 przedstawiono rezultaty uzyskane przy trenowaniu na zbiorze danych *SV*. Uzyskano
 647 bardzo zbliżone wyniki dla wszystkich trzech algorytmów, sieci konwolucyjne były nieznacznie
 648 lepsze od pozostałych przy wydajnościach na identyfikację dżetów b poniżej 70%. Dla tych
 649 samych prawdopodobieństw błędnej klasyfikacji tła uzyskiwały wydajności identyfikacji b lepsze
 650 o ok. 5%.



Rysunek 14: Zależności wydajności identyfikacji dżetów b od prawdopodobieństwa błędnej klasyfikacji dżetów tła dla poszczególnych algorytmów oraz ich porównanie. Algorytmy wytrenowane na zmiennych *SV*.

651 4.2.2 Wyniki dla zmiennych związanych z częstками tworzącymi dżet

652 Na Rys. 15 przedstawiono rezultaty uzyskane przy trenowaniu na zbiorze danych *constit*. Wyniki
 653 dla *BDT* oraz *Conv* są niemal identyczne, dla *BDT* trochę lepsze niż w przypadku zestawu da-
 654nych *SV*, natomiast te uzyskane dla sieci neuronowych typu *FC* są znacznie gorsze. Stosunkowo
 655 najmniejsze różnice pomiędzy zestawami danych *SV* i *constit* dla algorytmu *FC* otrzymano w
 656 przypadku separacji dżetów *b* od *c*.

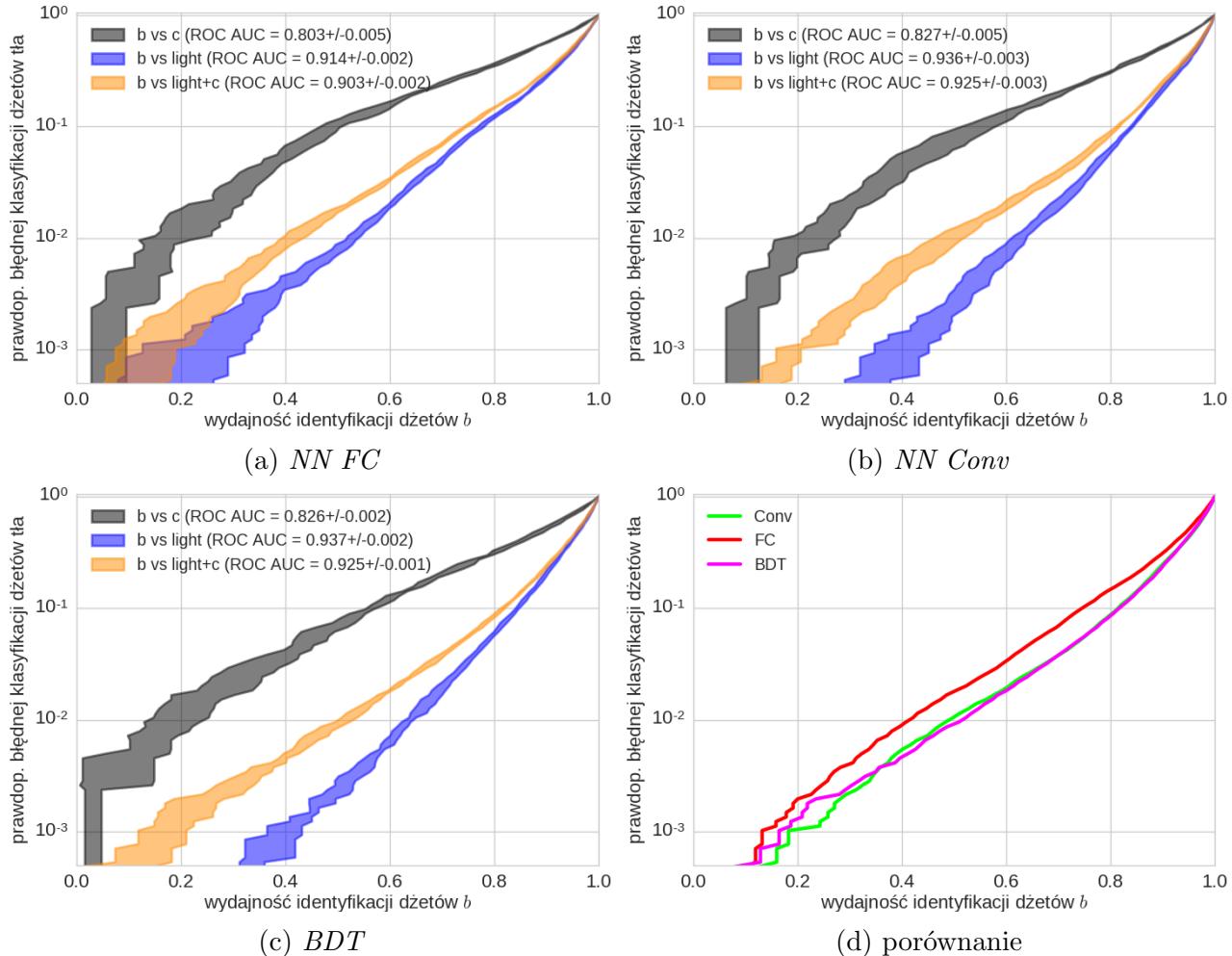


Rysunek 15: Zależności wydajności identyfikacji dżetów *b* od prawdopodobieństwa błędnej klasifikacji dżetów tła dla poszczególnych algorytmów oraz ich porównanie. Algorytmy wytreno-
wane na zmiennych *constit*.

657 4.2.3 Wyniki dla wszystkich zmiennych

658 Na Rys. 16 przedstawiono rezultaty uzyskane przy trenowaniu na zbiorze danych *merged*. Po-
 659 nownie krzywe *ROC* wzmacnianych drzew decyzyjnych i sieci konwolucyjnych prawie się nie
 660 różnią, natomiast dla sieci w pełni połączonych krzywa jest przesunięta o ok. 10% w stronę
 661 niższych wydajności.

662 W porównaniu do poprzednich zbiorów danych wszystkie algorytmy poprawiły swoje pre-
 663 dykce, najmniej *FC*, którego zdolności separacyjne są prawie takie same jak w przypadku *SV*
 664 (trochę lepsza separacja *b* od *c*).



Rysunek 16: Zależności wydajności identyfikacji dżetów *b* od prawdopodobieństwa błędnej klasyfikacji dżetów tła dla poszczególnych algorytmów oraz ich porównanie. Algorytmy wytrenowane na zmiennych *merged*.

665 4.2.4 Podsumowanie - wyniki poszczególnych modeli

666 Uzyskane wartości $ROC AUC$ oraz wydajności identyfikacji dżetów b dla trzech punktów pracy
 667 zebrane w Tab. 1.

668 Tak jak było to widać na Rys. 13, wartości $ROC AUC$ są najbardziej czułe na wartości
 669 wydajności osiągane dla punktu pracy o najwyższych wydajnościach. Wynika to z faktu, że
 670 pierwsze dwa punkty pracy leżą na samym początku krzywej ROC (dają niewielki wkład do
 671 pola pod krzywą).

672 Algorytm wytrenowany na połączonym zbiorze danych daje trochę lepsze wyniki niż trenowa-
 673 wany na zbiorach SV i $constit$, co jest oczywiście oczekiwane. Można było natomiast spodziewać
 674 się większej poprawy, co pokazuje, że predykcje algorytmów trenowanych na SV oraz $constit$
 675 muszą być skorelowane (przy założeniu, że model trenowany na dwóch połączonych zbiorach
 676 danych daje wyniki niegorsze niż trywialne połączenie dwóch modeli trenowanych na osobnych
 677 zbiorach danych).

678 W każdym przypadku najgorsze wyniki uzyskano dla sieci w pełni połączonych. Z tabeli
 679 wynika, że duże znaczenie mają zarówno użyty algorytm jak i zestaw danych. Podobne wyniki
 680 uzyskiwane dla BDT oraz $Conv$ pozwalają przypuszczać, że uzyskiwane przez nie poziom błędu
 681 jest zbliżony do minimum osiągalnego przy tych zestawach danych (błędu *bayesowskiego*).

model	$ROC AUC$	wydajność identyfikacji dżetów b [%]		
		dla prawd. błędnej klas. tła równej	0.1%	1%
$SV-FC$	0.901 ± 0.003	15 ± 3	39 ± 2	73.7 ± 0.6
$SV-Conv$	0.912 ± 0.001	18 ± 3	45 ± 2	76.4 ± 0.8
$SV-BDT$	0.909 ± 0.002	13 ± 4	38 ± 1	76.3 ± 0.6
$constit-FC$	0.839 ± 0.005	2 ± 1	15 ± 2	58.6 ± 1.5
$constit-Conv$	0.911 ± 0.002	20 ± 2	46 ± 3	77.8 ± 0.6
$constit-BDT$	0.918 ± 0.001	20 ± 3	48 ± 3	79.4 ± 0.7
$merged-FC$	0.903 ± 0.002	13 ± 6	41 ± 2	74.2 ± 0.3
$merged-Conv$	0.925 ± 0.003	18 ± 2	48 ± 3	81.3 ± 0.4
$merged-BDT$	0.925 ± 0.001	16 ± 5	51 ± 1	81.4 ± 0.4

Tablica 1: Tabela podsumowująca wyniki uzyskane przez poszczególne modele.

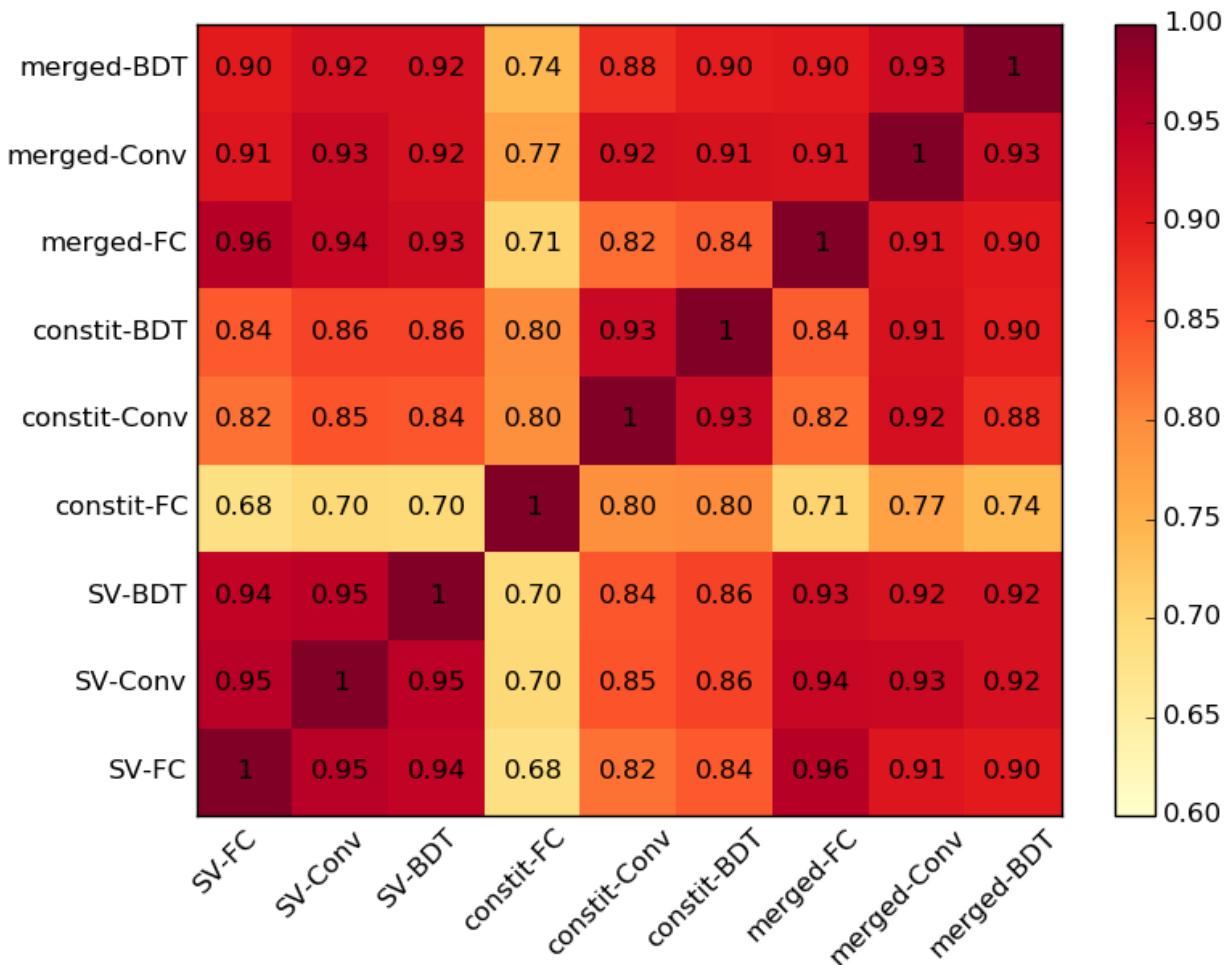
682 4.3 Korelacje predykcji modeli

683 Na Rys. 17 przedstawiono korelacje pomiędzy poszczególnymi modelami. Korelacje obliczano
 684 pomiędzy wynikami zwracanym bezpośrednio przez algorytmy (liczba rzeczywista $\in [0, 1]$), nie
 685 na podstawie odpowiadających im klas ($\{0, 1\}$).

686 Pierwszy wniosek płynący z wykresu korelacji to fakt, że wszystkie modele są ze sobą silnie
 687 skorelowane. Tłumaczy to niewielki zysk płynący z połączenia zestawów danych *SV* i *constit*
 688 co wspomniano w podrozdziale 4.2.4.

689 Najmniejszymi korelacjami z innymi wyróżnia się model *constit-FC*, który dawał zdecydowanie
 690 najsłabsze wyniki.

691 Widać wyraźny wzrost korelacji w przypadku modeli trenowanych na tych samych zbiorach
 692 danych (szczególnie dla modeli trenowanych na zmiennych związanych z wtórnymi wierzchołkami). Modele trenowane na połączonym zbiorze danych są nieco silniej skorelowane z modelami
 694 *SV-X* niż *constit-X*. Nie obserwuje się natomiast istotnie większych korelacji między modelami
 695 wykorzystującymi ten sam algorytm.



Rysunek 17: Średnie współczynniki korelacji Pearsona pomiędzy predykcjami poszczególnych modeli. Odchylenia standardowe nie zostały pokazane na wykresie, ich wartości wynoszą 0.001 – 0.009.

696 4.4 Analiza istotności zmiennych w *BDT*

697 Jedną z zalet drzew decyzyjnych jest możliwość dokładnego prześledzenia procesu decyzyjnego
698 poprzedzającego klasyfikację danego przypadku, co pozwala na lepsze zrozumienie działania
699 modelu. Nie jest to możliwe dla klasyfikatora złożonego z dużej liczby drzew jak *BDT*, jednak
700 nadal istnieją dosyć proste sposoby na zrozumienie, które cechy miały największy wpływ na
701 predykcję.

702 Wykorzystano trzy miary istotności zmiennych zaimplementowane w bibliotece **XGBoost**:⁴

- 703 • *weight* informuje ile razy dana zmienna została wykorzystana w podziałach drzew.
- 704 • *total_gain* jest sumą po spadkach funkcji straty uzyskiwanych dzięki podziałom wykorzy-
705 stującym daną zmienną
- 706 • *total_cover* to suma liczb przykładów trenujących, na które miały wpływ podziały wyko-
707 rzystujące daną zmienną.

708 Zasadniczo najważniejszą miarą jest *total_gain*, gdyż uwzględnia ona zarówno częstość wyko-
709 rzystania danej zmiennej (*weight*) jak i średni zysk w postaci spadku funkcji straty uzyskiwany
710 w poszczególnych podziałach. Zmienna binarna wykorzystana już w jednym podziale drzewa,
711 nie może zostać użyta ponownie w żadnej gałęzi będącej kontynuacją danego podziału, stąd
712 zmienne binarne zwykle osiągają mniejsze wartości miary *weight*. *total_cover* jest z kolei zwią-
713 zane z głębokościami drzewa na jakiej wykorzystywana była zmienna – pierwszy podział w
714 danym drzewie wpływa zawsze na wszystkie przykłady trenujące, natomiast kolejne – na stop-
715 niowo coraz mniejszy ułamek. Zatem niskie wartości *total_cover* sugerują, że dana zmienna
716 wykorzystywana była raczej na dużych głębokościach (na już mocno podzielonym drzewie).

717 W Tab. 2 przedstawiono wartości miar istotności zmiennych dla 15 najważniejszych cech
718 (wg *total_gain*) dla modeli *SV-BDT* oraz *constit-BDT*. Z kolei w Tab. 3 przedstawiono wartości
719 tych miar pogrupowane wg wielkości fizycznych. Wartości każdej miary zostały unormowane
720 do 100 (suma dla wszystkich elementów, nie tylko tych przedstawionych w tabelach jest równa
721 100). System oznaczeń opisany jest w Dodatku B.

722 Zdecydowanie najważniejszymi zmiennymi okazały się: L_{xy} w przypadku wtórnych wierz-
723 chołków oraz IP_D w przypadku częstek składowych dżetu, czyli zmienne, które zależą od czasu
724 życia hadronów. Wyraźna jest także tendencja, że im wyższa pozycja elementu na liście, tym
725 istotniejsze są jego właściwości (najistotniejsze są elementy nr 0).

726 O ile miary *total_gain* i *total_cover* są dosyć silnie skorelowane (co widać w obu tabelach)
727 to *weight*, czyli częstość ich użycia wydaje się być niezależna i przyjmować zbliżone wartości
728 dla wszystkich zmiennych. Okazuje się, że nie ma zmiennych, które byłyby wyraźnie rzadziej
729 wykorzystywane od innych.

730 Pojawiają się także niespodziewane wyniki, przykładowo zmienna σ_{Lxy} pojawia się częściej
731 niż L_{xy} , której niepewność opisuje, co jest zaskakujące tym bardziej, że σ_{Lxy} jest tylko jednym z
732 trzech rodzajów zmiennych związanych z jakością wtórnego wierzchołka (σ_{Lxy} , σ_{vertex} i χ^2/Ndf).

733 Wyniki uzyskane dla modelu *merged-BDT* (prawa strona Tab. 3) pokazują, że przy treно-
734 waniu na pełnym zestawie danych zmienne ze zbioru *SV* są bardziej znaczące niż te ze zbioru
735 *constit*, co jest zgodne z odnotowaną silniejszą korelacją modeli *merged-X* z modelami *SV-X*
736 niż z modelami *constit-X*.

⁴nazwy miar pochodzą z wersji biblioteki dla języka Python, w wersji dla innych języków, np. R nazwy miar
oraz ich znaczenie są różne

	<i>weight</i>	trening osobno <i>total_gain</i>	<i>total_cover</i>
L_{xy} - SV0	2.76	13.37	4.68
L_{xy} - SV2	1.88	9.83	3.80
L_{xy} - SV3	1.66	7.68	3.51
L_{xy} - SV1	1.34	6.79	2.84
L_{xy} - SV5	0.77	3.79	1.78
L_{xy} - SV4	1.03	2.79	2.06
χ^2/Ndf - SV0	3.30	2.30	2.56
L_{xy} - SV6	0.96	2.21	1.75
L_{xy} - SV8	0.96	2.14	1.81
L_{xy} - SV7	0.89	1.99	1.59
N_{SV}	0.73	1.73	1.33
σ_{vertex} - SV0	2.70	1.40	2.08
σ_{Lxy} - SV9	0.91	1.36	1.38
χ^2/Ndf - SV3	1.91	1.33	2.14
M_{inv} - SV9	0.77	1.31	1.24
IP_D - C0	2.73	19.29	9.68
IP_D - C1	2.76	15.06	8.74
IP_D - C2	3.39	11.37	8.69
IP_D - C3	2.84	6.56	6.16
IP_Z - C0	3.01	5.26	5.86
IP_D - C4	3.02	3.90	5.37
IP_Z - C1	2.50	3.44	4.63
IP_Z - C2	2.29	2.28	3.55
IP_D - C5	1.95	1.67	3.15
IP_Z - C3	2.14	1.50	2.65
IP_D - C6	2.07	1.33	2.96
IP_Z - C4	1.95	1.10	2.16
p_T - C0	2.22	1.00	1.79
p_T - C2	2.14	0.98	2.04
ϕ - C0	2.31	0.98	1.27

Tablica 2: Tabela zawierająca wartości miar istotności 15 najważniejszych (wg *total_gain*) cech, osobno dla zmiennych związanych z wtórnymi wierzchołkami (górsza połowa) oraz z częstotliwościami składowymi (dolna połowa).

	trening osobno			trening razem		
	<i>weight</i>	<i>total_gain</i>	<i>total_cover</i>	<i>weight</i>	<i>total_gain</i>	<i>total_cover</i>
L_{xy}	17.2	54.5	28.9	6.9	21.1	12.9
σ_{Lxy}	19.4	13.1	18.1	7.2	8.5	7.9
M_{inv}	18.3	8.0	14.1	7.6	6.8	6.8
σ_{vertex}	17.0	8.2	14.2	6.6	6.0	6.9
χ^2/Ndf	19.3	12.8	19.5	7.1	6.6	7.9
IP_D	22.4	60.6	47.5	11.3	12.3	16.1
IP_Z	18.3	16.1	23.3	9.9	7.7	9.3
p_T	14.0	7.1	14.1	7.0	5.7	6.2
ϕ	23.2	8.7	8.2	12.4	8.4	8.2
η	21.7	7.3	6.4	13.0	8.8	8.9

Tablica 3: Tabela zawierająca wartości miar istotności cech, posumowane według rodzaju zmiennej (sumy po wszystkich wtórnych wierzchołkach / wszystkich cząstkach). Wartości w lewej części odpowiadają modelom *SV-BDT* i *constit-BDT* natomiast z prawej – *merged-BDT*.

737 4.5 Analiza wpływu zmiennych na predykcje modeli

738 Dalszą analizę wpływu poszczególnych zmiennych przeprowadzono przy użyciu wykresów zależ-
739 ności cząstkowych (ang. *partial dependence plots*). Ze względu na wysoki koszt obliczeniowy tej
740 metody, przedstawione wyniki wyjątkowo nie są uśrednieniem kilka powtórzeń całej procedury.

741 Metoda wykresów zależności cząstkowych to metoda polegająca na sprawdzeniu zachowania
742 się predykcji już wytrenowanego modelu na skutek zmiany wartości wybranej zmiennej. W tym
743 celu przeprowadzane są predykcje na normalnym zbiorze danych, ze zmienioną jedną kolumną,
744 zawierającą wartości analizowanej zmiennej. Wszystkie wartości w tej kolumnie ustawiane są na
745 tę samą wartość, która jest stopniowo zmieniana od minimum do maksimum wartości przyjmowa-
746 nych przez daną zmienną, za każdym razem przeprowadzana jest predykcja. Współrzędnymi
747 punktów na wykresie zależności cząstkowej są: kolejne wartości ustawiane w analizowanej ko-
748 lumnie (oś pozioma) i średnia wartość predykcji modelu (oś pionowa), czyli odsetek wszystkich
749 przykładów jaki zostałby zaliczony przez algorytm do klasy pozytywnej (uznana za dżet b).

750 Taka analiza służy kilku celom. Po pierwsze, pozwala na weryfikację, czy modele działają
751 zgodnie z naszą wiedzą, w przypadkach gdy wiadomo jak powinien zachować się algorytm. Po
752 drugie, w przypadku braku wiedzy na temat pożądanego zachowania pozwala na wyciągnięcie
753 wniosków na podstawie sposobu działania algorytmu co wzbogaca intuicję i zrozumienie anali-
754 zowanych danych. Po trzecie, jest to jeden ze sposobów na zrozumienie działania algorytmów,
755 innych niż drzewa decyzyjne czy regresja liniowa, w których nie istnieją proste metryki pozwalan-
756 jące oszacować wpływ poszczególnych zmiennych na predykcję. W przypadku sieci neuronowych
757 często jest to dużym wyzwaniem.

758 Na Rys. 18 i 19 przedstawiono wykresy zależności cząstkowych dla wybranych zmiennych.
759 Przy wyborze zmiennych posługiwano się wartościami miar istotności cech z sekcji 4.4, kiero-
760 wano się także wartością ilustracyjną dla omawianych zagadnień. Skala osi pionowej jest inna
761 dla każdej zmiennej, im większy zakres, tym większy bezpośredni wpływ tej zmiennej na predy-
762 cje, tym istotniejsza jest ta zmienna. Poniżej każdego wykresu przedstawiono znormalizowane
763 rozkłady prawdopodobieństwa wartości analizowanej zmiennej dla dżetów b oraz dżetów tła.
764 Warto zaznaczyć na wstępie, że najbardziej istotne są obszary wykresów w okolicach maksimów
765 rozkładów prawdopodobieństwa zmiennych.

766 Wykresy na Rys. 18 a) i b) oraz 19 a), b) i c) pokazują, że znane podstawowe cechy dże-
767 tów b pozwalające odróżniać je od tła, tj. duże wartości L_{xy} których wierzchołków oraz duże
768 wartości bezwzględne odległości częstek od pierwotnego wierzchołka (IP_D , IP_Z) są intensywnie
769 wykorzystywane przez wszystkie algorytmy. Także wartości przy jakich najsilniej zmieniają się
770 predykcje modeli znajdują odzwierciedlenie w rozkładach poniżej.

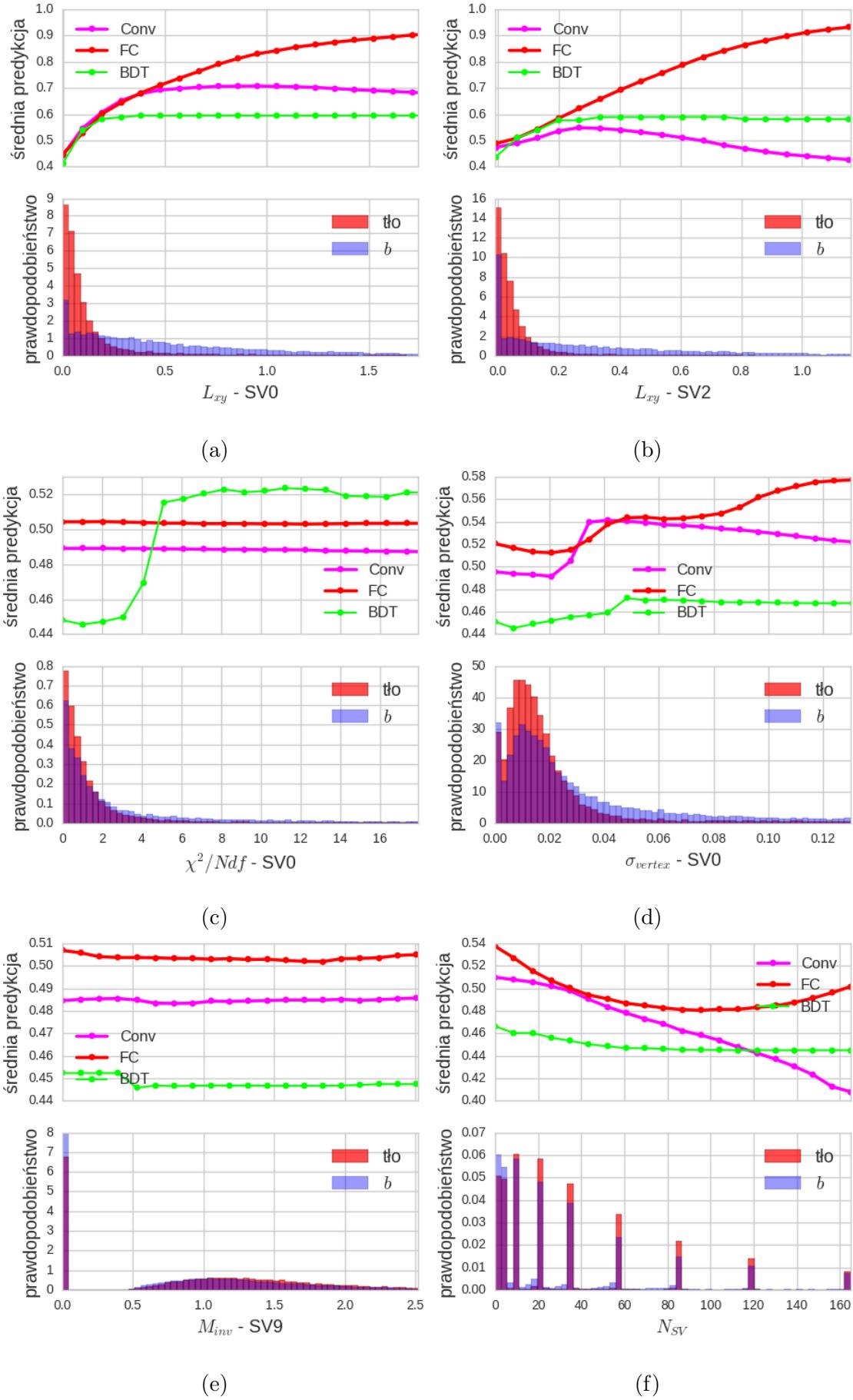
771 Analiza tych pięciu wykresów pozwala na zaobserwowanie ciekawych różnic między algoryt-
772 mami. Pierwsza z nich: zależności cząstkowe wydają się mieć mniejszą amplitudę w przypadku
773 BDT niż w przypadku sieci neuronowych. Może wynikać to z kształtu funkcji – w przypadku
774 BDT jest to złożenie kilku funkcji schodkowych, podczas gdy sieci neuronowe dają bardziej
775 gładkie zależności. Efektem tego jest inne zachowanie w obszarze ekstrapolacji (obszary gdzie
776 rozkłady prawdopodobieństwa mają niską gęstość). Drzewa decyzyjne nie mają potrzeby dalszych
777 podziałów np. w zakresie $L_{xy} - SV0 > 0.3$ lub $|IP_D| > 5$ dlatego ich predykcje są w tych
778 zakresach niemal stałe. Z kolei w przypadku sieci neuronowych, z powodu braku wystarczają-
779 cej liczby przykładów o takich wartościach zmiennych, wykres jest w przybliżeniu ekstrapolacją
780 zachowania z zagościętego zakresu, co jest bardzo wyraźnie w przypadku sieci w pełni połączonych.

782 Druga obserwacja to to asymetria w predykcji BDT na wykresie $IP_Z - C1$. Pomimo wyją-
783 kowo symetrycznego rozłożenia wartości tej zmiennej wokół zera, predykcje BDT są wyraźnie
784 przesunięte, z minimum zlokalizowanym wokół wartości -1. Przyczyną takiego zachowania może

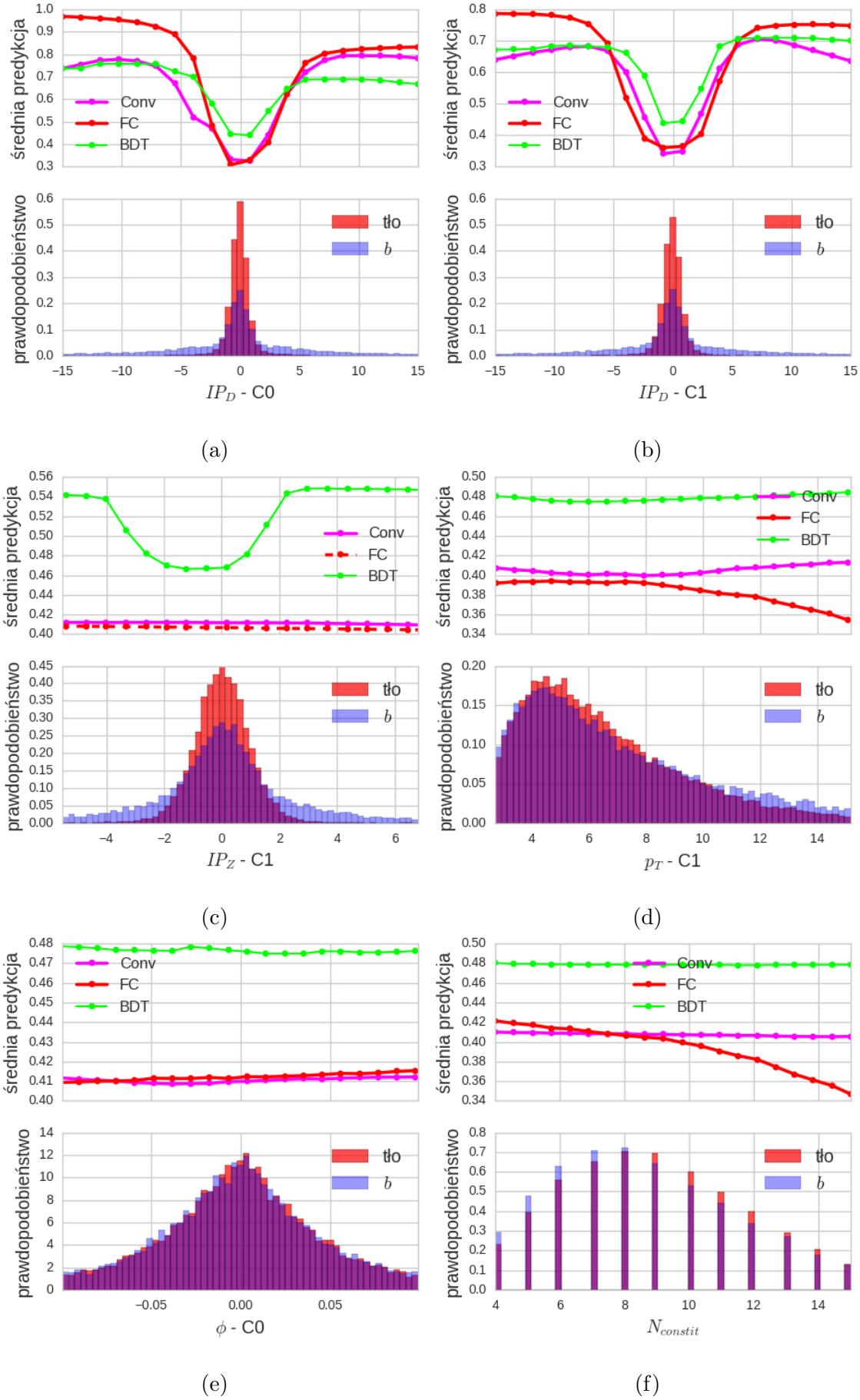
785 być specyficzny sposób trenowania drzew decyzyjnych, które w przypadku takiego rozkładu nie
786 mogą podzielić przestrzeń fazowej na 3 części w jednym kroku, co byłoby najbardziej korzystne,
787 ale zmuszone są do dwóch kolejnych podziałów, np. wzduż wartości -2 i 2, z czego pierwszy
788 podział musi niejako złamać symetrię rozkładu. Konieczność wykonania dwóch podziałów, z
789 których pierwszy nie zapewnia dobrej separacji klas może być przyczyną, dla której L_{xy} było
790 chętniej wykorzystywane przez model *merged-BDT* (Tab. 3) – w przypadku tej zmiennej dobrą
791 separację klas daje już pierwszy podział.

792 Wykresy takie jak te na Rys. 18 c) i f) lub 19 c), d) i f) pokazują, że predykcje algorytmów
793 mogą wykazywać nie tylko ilościowe, ale także jakościowe różnice w reakcji na wariacje wartości
794 jednej zmiennej.

795 Wreszcie wykresy przedstawione na Rys. 18 e) oraz Rys. 19 e) pokazują, że nawet w przy-
796 padku zmiennych intensywnie wykorzystywanych przy podziałach drzewa (o czym świadczy
797 Tab. 2) zmiana wyłącznie ich wartości ma często znikomy wpływ na predykcje całego modelu.
798 Są to zmienne, których rozkłady dla sygnału i tła prawie się nie różnią. Ich wpływ na predykcje
799 algorytmu ujawnia się dopiero w kombinacji z innymi zmiennymi. Wykorzystanie tych zmien-
800 nych jest dużo trudniejsze dla ludzi i między innymi na tym polega przewaga podejścia analizy
801 wielowymiarowej i uczenia maszynowego. Wykresy te sygnalizują jednocześnie ograniczenie me-
802 tody zależności cząstkowych, która jak każda metoda graficzna nie radzi sobie z przestrzeniami
803 zmiennych o liczbie wymiarów większej niż 3-4.



Rysunek 18: Zależności cząstkowe dla wybranych cech przedstawione dla modeli: *SV-Conv*, *SV-FC* i *SV-BDT*.



Rysunek 19: Zależności cząstkowe dla wybranych cech przedstawione dla modeli: *constit-Conv*, *constit-FC* i *constit-BDT*.

5 Podsumowanie i wnioski

W pracy przeanalizowano możliwości użycia algorytmów uczenia maszynowego w problemie identyfikacji dżetów b . Bazowano na danych z symulacji Monte Carlo przeprowadzonych dla detektora ALICE. Sprawdzono działanie trzech rodzajów algorytmów: wzmacnianych drzew decyzyjnych, sieci neuronowych w pełni połączonych oraz konwolucyjnych. Procedurę uczenia przeprowadzono osobno na trzech zbiorach zmiennych, uzyskując tym samym $3 \times 3 = 9$ modeli.

Zaprezentowano wyniki poszczególnych modeli (w postaci krzywych ROC oraz wydajności na identyfikację dżetów b w trzech punktach pracy) oraz obliczono korelacje między ich predykcjami. Sieci konwolucyjne oraz wzmacniane drzewa decyzyjne okazały się dawać lepsze wyniki od sieci w pełni połączonych. Dla najlepszych modeli uzyskano wartości $ROCAUC = 0.925(1)$. Dla prawdopodobieństwa błędnej klasyfikacji tła równego 1% i 10% uzyskano wydajności na identyfikację dżetów b na poziomie kolejno 50% oraz 81%. Predykcje wszystkich modeli okazały się być dosyć silnie skorelowane, przy czym czynnikiem wzmacniającym korelacje okazało się użycie do treningu tych samych zmiennych, natomiast nie zaobserwowano wzmacnienia korelacji dla algorytmów tego samego rodzaju.

W celu lepszego zrozumienia działania modeli przeprowadzono analizę wpływu poszczególnych zmiennych na ich predykce. W tym celu wykorzystano miary istotności zmiennych dostępne dla drzew decyzyjnych oraz wykresy zależności cząstkowych dla wszystkich trzech algorytmów. Analiza pokazuje (tylko dla drzew decyzyjnych), że wszystkie używane wielkości okazały się mieć niepomijalny wpływ na predykcje ($total_gain \gtrsim 6\%$). Ponadto wszystkie modele działają zgodnie z podstawową wiedzą nt. właściwości dżetów b , tzn. zmienne wykorzystywane w klasycznych metodach cięć (takie jak odległość wtórnego wierzchołka L_{xy} czy odległości śladów od pierwotnego wierzchołka $IP_{D,Z}$) mają największy wpływ na predykcje algorytmów.

Istnieje kilka sposobów na potencjalną poprawę uzyskanych wyników. Pierwszym z nich mogłyby być wykorzystanie dodatkowych wielkości, związanych z półleptonowym kanałem rozpadu ciężkich hadronów jak np. pęd poprzeczny leptonu względem osi dżetu lub z wewnętrzną strukturą dżetów, jak np. tzw. *pull* [52]. Kolejnym mogłyby być użycie innych algorytmów lub architektur sieci neuronowych, co jednak mojej ocenie ma niewielki potencjał bez zmiany w danych treningowych. Istotniejsze mogłyby okazać się użycie większej ilości danych, co niemal na pewno poprawiłyby wyniki głębokich sieci neuronowych, które znane są ze świetnego skalowania się w obszarze ogromnych zbiorów danych.

Kolejnym etapem pracy powinna być próba użycia modeli na danych eksperymentalnych. Wiąże się to z szeregiem nowych trudności, głównie związanych z brakiem możliwości bezpośredniej oceny wyników uzyskiwanych przez algorytm, jest jednak niezbędne do wyjścia poza obszar czysto akademickich rozważań. Innym kierunkiem rozwoju byłoby przeprowadzenie podobnej analizy dla dużo bardziej trudnych danych ze zderzeń ciężkich jonów.

840 Dodatek A Metryki

841 W poniżej tabeli zebrano stosowane najczęściej miary jakości klasyfikatorów. We wzorach defi-
 842 niujących metryki wykorzystano następujące wielkości:

843 TP (ang. *true positives*) – liczba poprawnie zaklasyfikowanych przypadków klasy pozytywnej

844 TN (ang. *true negatives*) – liczba poprawnie zaklasyfikowanych przypadków klasy negatywnej

845 FP (ang. *false positives*) – liczba błędnie zaklasyfikowanych przypadków klasy negatywnej

846 FN (ang. *false negatives*) – liczba błędnie zaklasyfikowanych przypadków klasy pozytywnej

847

nazwa metryki	nazwa angielska	wzór
dokładność	accuracy	$(TP+TN) / (TP+TN+FP+FN)$
precyzja	precision	$TP / (TP+FP)$
czułość, wydajność id. sygnału	recall, sensitivity, TP Rate	$TP / (TP+FN)$
swoistość	specificity, TN Rate	$TN / (TN+FP)$
F1	F1	$2 \frac{precision \cdot recall}{precision + recall}$
prawd. błędnej klas. tła	mistagging rate, FP Rate	$FP / (FP+TN)$

Tablica A1: Tabela zawierająca nazwy i definicje popularnych metryk.

848 Podanie tylko jednej metryki jest zwykle niewystarczające i stosuje się pary metryk, np.
 849 precyzja - czułość. Jeśli predykcja klasyfikatora ma charakter ciągły, to poprzez zmienianie
 850 wartości progowej otrzymuje się różne punkty pracy, scharakteryzowane przez wartości obu
 851 metryk. Kolejne punkty pracy wykreślone np. na wykresie $TPR(FPR)$ dają krzywą nazywaną
 852 krzywą ROC. Wykres ten w przypadku klasyfikatora losowego jest prostą łączącą punkty o
 853 współrzędnych $(0,0)$ i $(1,1)$. Im lepszy klasyfikator tym bardziej wykres wygięty jest w stronę
 854 punktu $(0,1)$. Pole pod tą krzywą (ang. *ROC Area Under Curve – ROC AUC*) jest kolejną
 855 metryką, bardzo często wykorzystywaną w praktyce, gdyż łączy w sobie wszystkie możliwe
 856 punkty pracy, jest także odporne na niezrównoważenie klas. Pole pod krzywą *ROC* przyjmuje
 857 wartość 0.5 dla klasyfikatora losowego oraz 1 dla idealnego.

858 W literaturze dot. klasyfikacji dżetów wyniki zwykle przedstawia się z użyciem dwóch po-
 859 wszechnie znanych metryk, ale o zmienionych nazwach: *True Positive Rate*, nazywanej *b jet*
 860 *tagging efficiency* – wydajności na identyfikację dżetów *b* oraz *False Positive Rate*, nazywanym
 861 *mistagging rate* – prawdopodobieństwa błędnej klasyfikacji dżetów tła jako dżety *b*.

862 Dodatek B Skróty i oznaczenia

863 W pracy używane są następujące skróty i oznaczenia:

- 864 • algorytmy:
865 FC – sieci neuronowe w pełni połączone,
866 $Conv$ – sieci neuronowe konwolucyjne,
867 BDT – wzmacniane drzewa decyzyjne
- 868 • zbiory danych:
869 SV – zestaw zawierający tylko zmienne związane z wtórnymi wierzchołkami,
870 $constit$ – zestaw zawierający tylko zmienne związane z częstками tworzącymi dżet,
871 $merged$ – zestaw zawierający wszystkie zmienne (tj. $SV + Constit +$ zmienne charaktery-
872 zujące dżet jako całość)
- 873 • modele (zestaw danych + algorytm) nazywane są wg wzoru:
874 (oznaczenie_zbioru_danych)-(oznaczenie_algorytmu),
875 np. $SV-Conv$ oznacza konwolucyjną sieć neuronową wytrenowaną na zmiennych zwi-
876 ązanych z wtórnymi wierzchołkami a $merged-BDT$ - wzmacniane drzewa decyzyjne, do
877 treningu których użyte zostały wszystkie zmienne. Zapis $merged-X$ stanowi zbiorcze ozna-
878 czenie modeli $merged-SV$, $merged-Conv$ i $merged-BDT$.
- 879 • poszczególne zmienne (kolumny w zbiorze danych) oznaczane są według wzoru:
880 (nazwa_zmiennej) – (numer_obiektu),
881 np. $\sigma_{L_{xy}}$ – $SV2$ oznacza niepewność wyznaczenia L_{xy} dla wtórnego wierzchołka nr 2, a
882 IP_Z-C5 – rzut odległości najbliższego zbliżenia cząstki na oś wiązki cząstki nr 5 – numery
883 na posortowanych listach (Rozdz. 3, gdzie znajdują się także opisy wielkości fizycznych).

884 Bibliografia

- 885 [1] Donald H. Perkins. "Oddziaływanie międzykarkowe i chromodynamika kwantowa". W:
886 *Wstęp do Fizyki Wysokich Energii*. 2 wydr. PWN, 2005, 171–193.
- 887 [2] David J. Gross i Frank Wilczek. "Ultraviolet Behavior of Nonabelian Gauge Theories".
888 W: *Phys. Rev. Lett.* 30 (1973). [,271(1973)], s. 1343–1346. DOI: 10.1103/PhysRevLett.
889 30.1343.
- 890 [3] H. David Politzer. "Reliable Perturbative Results for Strong Interactions?" W: *Phys. Rev.*
891 *Lett.* 30 (1973). [,274(1973)], s. 1346–1349. DOI: 10.1103/PhysRevLett.30.1346.
- 892 [4] C. Patrignani i in. "Review of Particle Physics". W: *Chin. Phys.* C40.10 (2016), s. 100001.
893 DOI: 10.1088/1674-1137/40/10/100001.
- 894 [5] John C. Collins i M. J. Perry. "Superdense Matter: Neutrons Or Asymptotically Free
895 Quarks?" W: *Phys. Rev. Lett.* 34 (1975), s. 1353. DOI: 10.1103/PhysRevLett.34.1353.
- 896 [6] N. Cabibbo i G. Parisi. "Exponential Hadronic Spectrum and Quark Liberation". W:
897 *Phys. Lett.* 59B (1975), s. 67–69. DOI: 10.1016/0370-2693(75)90158-6.
- 898 [7] D. Boyanovsky, H. J. de Vega i D. J. Schwarz. "Phase transitions in the early and the
899 present universe". W: *Ann. Rev. Nucl. Part. Sci.* 56 (2006), s. 441–500. DOI: 10.1146/
900 annurev.nucl.56.080805.140539. arXiv: hep-ph/0602002 [hep-ph].
- 901 [8] Mark G. Alford i Kai Schwenzer. "What the Timing of Millisecond Pulsars Can Teach
902 us about Their Interior". W: *Phys. Rev. Lett.* 113.25 (2014), s. 251102. DOI: 10.1103/
903 PhysRevLett.113.251102. arXiv: 1310.3524 [astro-ph.HE].
- 904 [9] Vardan Khachatryan i in. "Evidence for collectivity in pp collisions at the LHC". W:
905 *Phys. Lett.* B765 (2017), s. 193–220. DOI: 10.1016/j.physletb.2016.12.009. arXiv:
906 1606.06198 [nucl-ex].
- 907 [10] Jaroslav Adam i in. "Enhanced production of multi-strange hadrons in high-multiplicity
908 proton-proton collisions". W: *Nature Phys.* 13 (2017), s. 535–539. DOI: 10.1038/nphys4111.
909 arXiv: 1606.07424 [nucl-ex].
- 910 [11] Matteo Cacciari, Gavin P. Salam i Gregory Soyez. "The Anti-k(t) jet clustering algori-
911 thm". W: *JHEP* 04 (2008), s. 063. DOI: 10.1088/1126-6708/2008/04/063. arXiv:
912 0802.1189 [hep-ph].
- 913 [12] Vardan Khachatryan i in. "Charged-particle nuclear modification factors in PbPb and
914 pPb collisions at $\sqrt{s_{NN}} = 5.02$ TeV". W: *JHEP* 04 (2017), s. 039. DOI: 10.1007/
915 JHEP04(2017)039. arXiv: 1611.01664 [nucl-ex].
- 916 [13] Betty Abelev i in. "Centrality Dependence of Charged Particle Production at Large Trans-
917 verse Momentum in Pb–Pb Collisions at $\sqrt{s_{NN}} = 2.76$ TeV". W: *Phys. Lett.* B720 (2013),
918 s. 52–62. DOI: 10.1016/j.physletb.2013.01.051. arXiv: 1208.2711 [hep-ex].
- 919 [14] Carlos A. Salgado i Urs Achim Wiedemann. "Calculating quenching weights". W: *Phys.*
920 *Rev.* D68 (2003), s. 014008. DOI: 10.1103/PhysRevD.68.014008. arXiv: hep-ph/0302184
921 [hep-ph].
- 922 [15] Yuri L. Dokshitzer i D. E. Kharzeev. "Heavy quark colorimetry of QCD matter". W:
923 *Phys. Lett.* B519 (2001), s. 199–206. DOI: 10.1016/S0370-2693(01)01130-3. arXiv:
924 hep-ph/0106202 [hep-ph].
- 925 [16] Georges Aad i in. "Performance of *b*-Jet Identification in the ATLAS Experiment". W:
926 *JINST* 11.04 (2016), P04008. DOI: 10.1088/1748-0221/11/04/P04008. arXiv: 1512.
927 01094 [hep-ex].

- [17] M. Aaboud i in. “Measurements of b-jet tagging efficiency with the ATLAS detector using $t\bar{t}$ events at $\sqrt{s} = 13$ TeV”. W: *JHEP* 08 (2018), s. 089. DOI: 10.1007/JHEP08(2018)089. arXiv: 1805.01845 [hep-ex].
- [18] Serguei Chatrchyan i in. “Identification of b-quark jets with the CMS experiment”. W: *JINST* 8 (2013), P04013. DOI: 10.1088/1748-0221/8/04/P04013. arXiv: 1211.4462 [hep-ex].
- [19] A. M. Sirunyan i in. “Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV”. W: *JINST* 13.05 (2018), P05011. DOI: 10.1088/1748-0221/13/05/P05011. arXiv: 1712.07158 [physics.ins-det].
- [20] Linus Feldkamp. “Study of b-jet tagging performance in ALICE”. W: *J. Phys. Conf. Ser.* 509 (2014), s. 012061. DOI: 10.1088/1742-6596/509/1/012061. arXiv: 1310.2817 [hep-ex].
- [21] Rüdiger Haake. “Machine and deep learning techniques in heavy-ion collisions with ALICE”. W: *Proceedings, 2017 European Physical Society Conference on High Energy Physics (EPS-HEP 2017): Venice, Italy, July 5-12, 2017*. T. EPS-HEP2017. 2017. DOI: 10.22323/1.314.0498. arXiv: 1709.08497 [physics.data-an]. URL: <https://pos.sissa.it/314/498/pdf>.
- [22] Roel Aaij i in. “Identification of beauty and charm quark jets at LHCb”. W: *JINST* 10.06 (2015), P06013. DOI: 10.1088/1748-0221/10/06/P06013. arXiv: 1504.07670 [hep-ex].
- [23] K. Aamodt i in. “The ALICE experiment at the CERN LHC”. W: *JINST* 3 (2008), S08002. DOI: 10.1088/1748-0221/3/08/S08002.
- [24] Betty Bezverkhny Abelev i in. “Performance of the ALICE Experiment at the CERN LHC”. W: *Int. J. Mod. Phys.* A29 (2014), s. 1430044. DOI: 10.1142/S0217751X14300440. arXiv: 1402.4476 [nucl-ex].
- [25] Wikimedia Commons. *Schematics of the ALICE subdetectors*. 2014. URL: [https://commons.wikimedia.org/wiki/File:2012-Aug-02-ALICE_3D_v0_with_Text_\(1\)_2.jpg](https://commons.wikimedia.org/wiki/File:2012-Aug-02-ALICE_3D_v0_with_Text_(1)_2.jpg).
- [26] Sotiris B Kotsiantis, I Zaharakis i P Pintelas. “Supervised machine learning: A review of classification techniques”. W: *Emerging artificial intelligence applications in computer engineering* 160 (2007), s. 3–24.
- [27] Marcin Wolter. “Metody analizy wielu zmiennych w fizyce wysokich energii”. Prac. dokt. IFJ PAN, 2012.
- [28] Leo Breiman. “Bagging predictors”. W: *Machine learning* 24.2 (1996), s. 123–140.
- [29] Yoav Freund i Robert E Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. W: *Journal of computer and system sciences* 55.1 (1997), s. 119–139.
- [30] Tianqi Chen i Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. W: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: ACM, 2016, s. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: <http://doi.acm.org/10.1145/2939672.2939785>.
- [31] James Bergstra i Yoshua Bengio. “Random Search for Hyper-parameter Optimization”. W: *J. Mach. Learn. Res.* 13 (lut. 2012), s. 281–305. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=2188385.2188395>.

- [32] Sandhya Samarasinghe. *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. Auerbach publications, 2016.
- [33] Ian Goodfellow, Yoshua Bengio i Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [34] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. W: *Neural Networks* 4.2 (1991), s. 251 –257. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T). URL: <http://www.sciencedirect.com/science/article/pii/089360809190009T>.
- [35] Alex Krizhevsky, Ilya Sutskever i Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. W: *Advances in neural information processing systems*. 2012, s. 1097–1105.
- [36] Petar Veličković. *Deep learning for complete beginners: convolutional neural networks with keras*. 2017. URL: <https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html> (term. wiz. 15.07.2018).
- [37] Andrew Ng. *Convolutional Neural Networks*. 2017. URL: <https://www.coursera.org/learn/convolutional-neural-networks> (term. wiz. 15.07.2018).
- [38] Kendrick Tan. *Capsule Networks Explained*. 2017. URL: https://kndrck.co/posts/capsule_networks_explained/ (term. wiz. 15.07.2018).
- [39] MathWorks. *Convolutional Neural Network*. URL: <https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html> (term. wiz. 15.07.2018).
- [40] François Chollet i in. *Keras*. <https://keras.io>. 2015.
- [41] Martín Abadi i in. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [42] Timothy Dozat. *Incorporating Nesterov Momentum into Adam*. 2015. URL: {http://cs229.stanford.edu/proj2015/054_report.pdf}.
- [43] Diederik P. Kingma i Jimmy Ba. “Adam: A Method for Stochastic Optimization”. W: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: <http://arxiv.org/abs/1412.6980>.
- [44] Nitish Srivastava i in. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. W: *J. Mach. Learn. Res.* 15.1 (sty. 2014), s. 1929–1958. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=2627435.2670313>.
- [45] Torbjorn Sjostrand, Stephen Mrenna i Peter Z. Skands. “A Brief Introduction to PYTHIA 8.1”. W: *Comput. Phys. Commun.* 178 (2008), s. 852–867. DOI: 10.1016/j.cpc.2008.01.036. arXiv: 0710.3820 [hep-ph].
- [46] Peter Skands, Stefano Carrazza i Juan Rojo. “Tuning PYTHIA 8.1: the Monash 2013 Tune”. W: *Eur. Phys. J. C*74.8 (2014), s. 3024. DOI: 10.1140/epjc/s10052-014-3024-y. arXiv: 1404.5630 [hep-ph].
- [47] René Brun i in. “GEANT Detector Description and Simulation Tool”. W: (1994). DOI: 10.17181/CERN.MUHF.DMJ1.
- [48] Matteo Cacciari, Gavin P. Salam i Gregory Soyez. “FastJet User Manual”. W: *Eur. Phys. J.* C72 (2012), s. 1896. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].
- [49] D0 Collaboration. *Observation of Single Top Quark Production*. 2009. URL: https://www-d0.fnal.gov/Run2Physics/top/singletop_observation/singletop_observation_updated.html (term. wiz. 15.07.2018).

- 1017 [50] Matteo Cacciari i Gavin P. Salam. “Pileup subtraction using jet areas”. W: *Phys. Lett.*
1018 B659 (2008), s. 119–126. DOI: 10.1016/j.physletb.2007.09.077. arXiv: 0707.1378
1019 [[hep-ph](#)].
- 1020 [51] Andrew P Bradley. “The use of the area under the ROC curve in the evaluation of
1021 machine learning algorithms”. W: *Pattern recognition* 30.7 (1997), s. 1145–1159.
- 1022 [52] Jason Gallicchio i Matthew D. Schwartz. “Seeing in Color: Jet Superstructure”. W: *Phys.*
1023 *Rev. Lett.* 105 (2010), s. 022001. DOI: 10.1103/PhysRevLett.105.022001. arXiv: 1001.
1024 5027 [[hep-ph](#)].