

# Spis treści

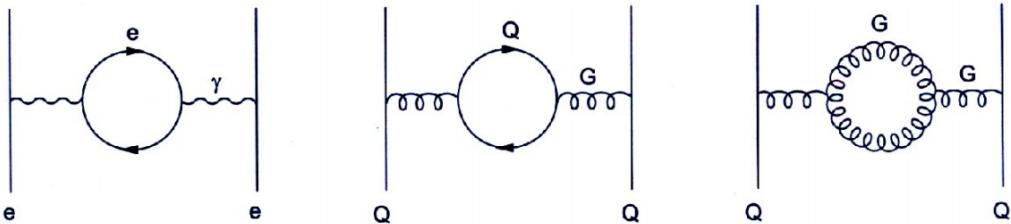
<b>1 Fizyka dżetów cząstek</b>	<b>2</b>
1.1 Chromodynamika kwantowa . . . . .	2
1.2 Plazma kwarkowo-gluonowa . . . . .	3
1.3 Dżety . . . . .	4
1.4 Dżety $b$ . . . . .	5
1.5 Eksperyment ALICE . . . . .	6
<b>2 Uczenie maszynowe</b>	<b>9</b>
2.1 Wzmacniane drzewa decyzyjne . . . . .	9
2.2 Sieci neuronowe . . . . .	10
2.3 Dyskusja użycia dwóch algorytmów . . . . .	15
<b>3 Dane</b>	<b>16</b>
<b>4 Analiza</b>	<b>19</b>
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
<b>5 Podsumowanie i wnioski</b>	<b>35</b>
<b>Dodatki</b>	<b>36</b>
<b>Dodatek A Metryki</b>	<b>36</b>
<b>Dodatek B Skróty i oznaczenia</b>	<b>37</b>

# 1 Fizyka dżetów cząstek

## 1.1 Chromodynamika kwantowa

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

Próżnia, w rozumieniu klasycznym będąca zupełnie pusta, w teoriach kwantowych wypełniona jest pojawiającymi i znikającymi wirtualnymi cząstками. Cząstki te ekranują ładunek próbkowy umieszczony w kwantowej próżni, wywołując zjawisko polaryzacji próżni (analogiczne do polaryzacji dielektryków), które efektywnie zmniejsza pole wytwarzane przez ten ładunek. Siła tego efektu zależy od liczby ekranujących cząstek, czyli pośrednio od skali odległości. Skala ta wyznaczona jest przez długości fali próbującej cząstki, zatem także jej energię (im większa energia, tym mniejsza długość fali i mniejsza ilość ekranujących cząstek obserwowanych w pobliżu rzeczywistego ładunku, zatem tym słabszy efekt ekranowania i większy efektywny ładunek). Prowadzi to do zależnej od energii stałej sprężenia  $\alpha$ , którą nazywamy efektywną lub bieżącą stałą sprężenia (ang. *running coupling constant*).



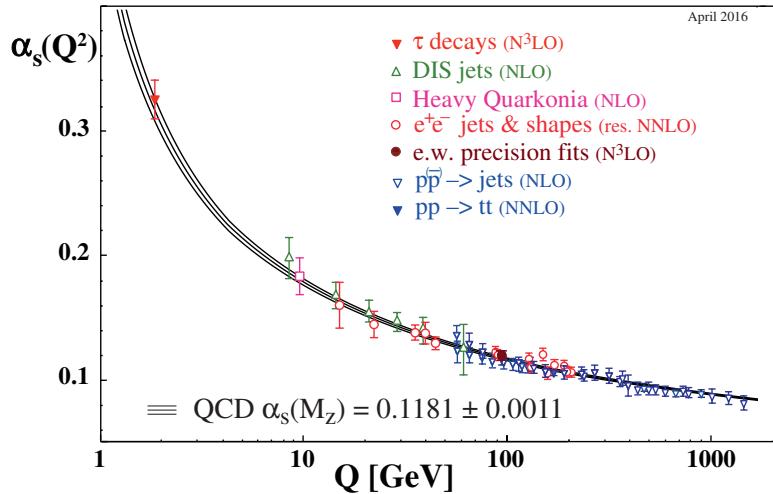
Rysunek 1: Diagramy Feynmana opisujące polaryzację próżni w *QED* (lewy) i *QCD* (środkowy i prawy). Rysunki lewy i środkowy są swoimi odpowiednikami w tych dwóch teoriach, natomiast prawy, w którym oddziałują jedynie bozony pośredniczące nie ma swojego odpowiednika w *QED*. Źródło: [1]

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 jest także samooddziałanie gluonów, przez co dopuszczalne są diagramy Feynmana jak ten przedstawiony na Rys. 1 po prawej. Pętle gluonowe działają anty-ekranującą – zwiększą efektywną wartość silnej stałej sprężenia, ponadto jest to efekt dominujący nad przyczynkiem od par kwark-antykwark, co sprawia że zależność bieżącej stałej sprężenia w *QCD* jest odwrotna i dużo silniejsza niż w przypadku *QED*. Wartość  $\alpha_{em}$  maleje od wartości  $\frac{1}{128}$  przy energiach ok. 90 GeV do  $\frac{1}{137}$  przy energii bliskiej zeru, co oznacza zmianę o kilka procent. Tymczasem  $\alpha_S$  rośnie w miarę zbliżania się do niskich energii od wartości  $\alpha_S \lesssim 0.1$  dla  $E \gtrsim 100$  GeV do  $\alpha_S > 1$  dla energii poniżej 200 MeV (Rys. 2). Prowadzi to do dwóch zjawisk charakterystycznych dla chromodynamiki kwantowej:

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

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

- uwięzienie koloru (ang. *colour confinement*) – przy zwiększaniu odległości między partonami siła oddziaływanego rośnie do nieskończoności, dlatego nigdy nie obserwuje się swobodnych cząstek obdarzonych ładunkiem kolorowym a jedynie w postaci związanej w kolorowo obojętnych hadronach.



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

## 1.2 Plazma kwarkowo-gluonowa

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

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

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

### 1.3 Dżety

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

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

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

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

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

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

---

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

## 1.4 Dżety $b$

### 1.4.1 Właściwości

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

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

### 1.4.2 Przegląd algorytmów używanych do identyfikacji dżetów $b$ na LHC

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

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

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

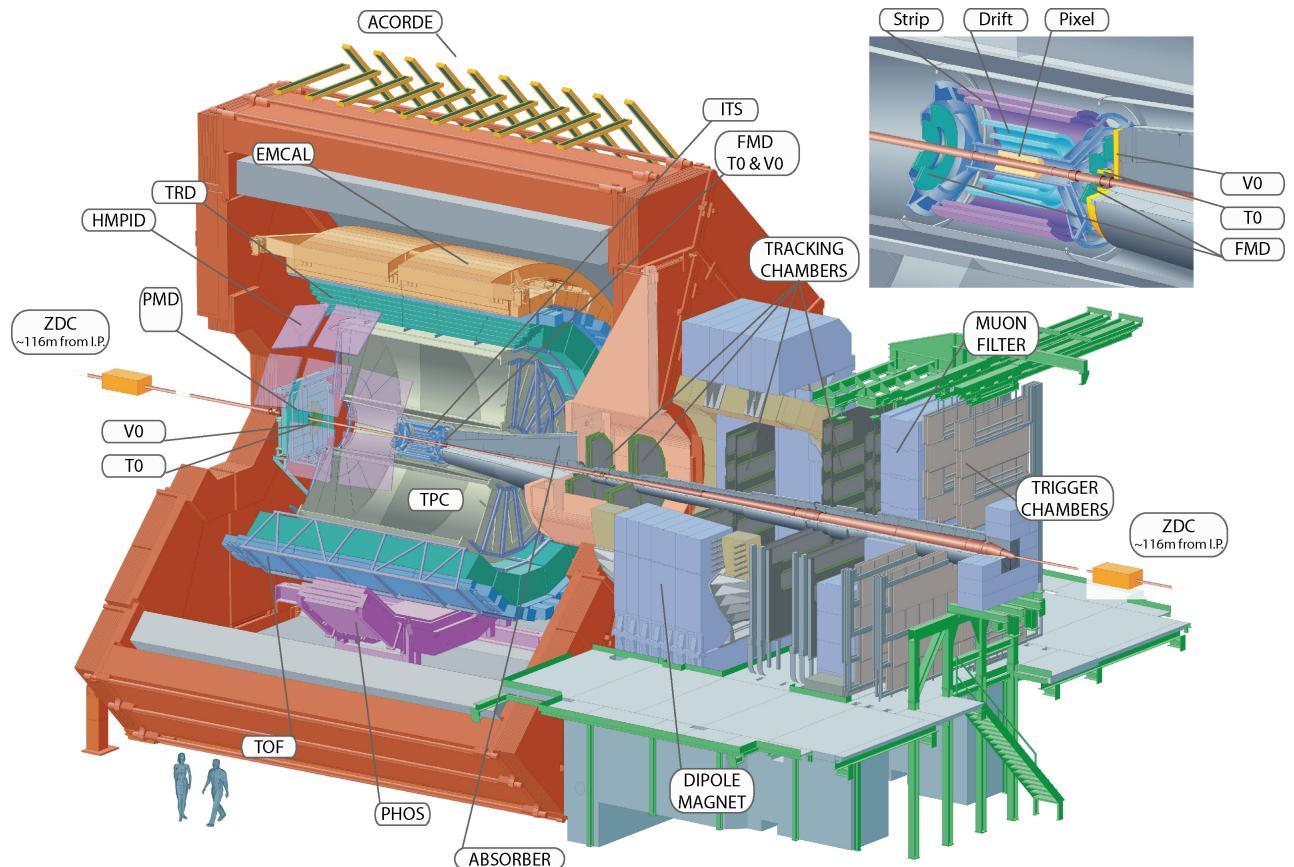
Do wykorzystania półleptonowego kanału rozpadu ciężkich hadronów do identyfikacji dżetów  $b$  w eksperymentach ATLAS (SMT) i CMS (SE, SM) użyto wzmacnianych drzew decyzyjnych trenowanych na kilku ręcznie zdefiniowanych w tym celu zmiennych takich jak pęd leptonu transwersalny względem osi dżetu.

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

Przykład innego podejścia zaprezentowała współpraca przy eksperymencie LHCb, gdzie wykorzystano dwa zestawy wzmacnianych drzew decyzyjnych operujących na zmiennych związanych z wtórnymi wierzchołkami. Pierwszy zapewnia separację dżetów lekkich od ciężkich a drugi odróżnia dżety  $b$  od  $c$ . Do wyboru punktu pracy, zamiast jednowymiarowego rozkładu 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  $\eta$ . 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.

- Time Projection Chamber (TPC) – długa na 5 m i o takiej średnicy komora projekcji czasowej. Jest to główny detektor śladowy ALICE, wraz z ITS służy do wyznaczania trajektorii cząstek i na ich podstawie również wierzchołków zderzenia. Elektryny uwolnione ze zjonizowanego przez poruszające się w nim naładowane cząstki gazu dryfują wzdłuż kierunku wiązki w stronę końcowych elektrod. Następnie są tam zbierane dostarczając informacji o dwóch współrzędnych toru cząstki: odległości od wiązki i kącie azymutalnym. Trzecia składowa trajektorii jest otrzymywana na podstawie czasu dotarcia elektronów do elektrod. TPC jest najwolniejszym detektorem ALICE (ze względu na ograniczający czas dryfu elektronów wynoszący  $\sim 90 \mu\text{s}$ ), użycie detektora tego typu podyktowane jest jego zdolnością do mierzenia śladów tysięcy cząstek spodziewanych w centralnych zderzeniach PbPb.

Znajomość toru ruchu cząstki pozwala na wyznaczenie jej pędu. Oprócz dokładnej trajektorii każdej cząstki próbowanej do 159 razy, TPC mierzy straty energii cząstek  $dE/dx$ . Pozwala to na ich identyfikację na podstawie zależności Bethego-Blocha, najwyższą zdolność rozdzielczą TPC osiąga dla cząstek o  $p_T < 1 \text{ GeV}$ .

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

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

- kalorymetry

- Photon Spectrometer (PHOS) – elektromagnetyczny kalorymetr o wysokiej rozdzielcości energetycznej i przestrzennej (podzielony na kryształy o rozmiarze poprzecznym  $2.2 \times 2.2 \text{ cm}$ , co odpowiada przestrzeni fazowej  $\Delta\eta \times \Delta\phi = 0.004 \times 0.004$ ). Pokrywa zakres pseudopospieszności  $|\eta| < 0.12$  i kąta azymutalnego równy  $100^\circ$ . PHOS ma za zadanie identyfikację i pomiar czteropędów fotonów, w szczególności tych niepochodzących z rozpadu innych cząstek (ang. *direct photons*) oraz lekkich mezonów neutralnych (np.  $\pi^0$ ) przez dwufotonowy kanał rozpadu.
- Electromagnetic Calorimeter (EMCal) – drugi elektromagnetyczny kalorymetr ALICE o mniejszej ziarnistości ( $\Delta\eta \times \Delta\phi = 0.014 \times 0.014$ ), ale dużo większej akceptancji ( $|\eta| < 0.7$ ,  $\Delta\phi = 107^\circ$ ). EMCal poprawia możliwości ALICE w zakresie pomiarów 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 ( $\omega$ ,  $\phi$ ,  $J/\Psi$ ,  $\Upsilon$ ) 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.

## 2 Uczenie maszynowe

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

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*).

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

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

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

### 2.1 Wzmacniane drzewa decyzyjne

Wzmacniane drzewa decyzyjne są jednym z rozwinięć klasycznego algorytmu drzewa decyzyjnego. Pojedyncze drzewo decyzyjne dzieli przestrzeń cech uczących przy pomocy prostopadłych cięć, na mniejsze/większe niż zadana wartość w przypadku zmiennej ciągłej lub na należące/nie należące do danej klasy w przypadku zmiennej kategorycznej. Każdy podział, nazywany węzłem, 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 były jak najbardziej jednorodne. Stosuje się różne miary nieporządku takie tak: indeks Gini  $G_L$  lub entropia  $S_L$ <sup>3</sup>.

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

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

---

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

<sup>3</sup> $G_L = 1 - \sum_k p_k^2$  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  $L$

uczenia każdego drzewa używa się tylko podzbioru wszystkich cech uczących. Końcową predykcję algorytmu otrzymuje się poprzez „głosowanie” wszystkich drzew z odpowiednimi wagami.

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

## Parametry i sposób trenowania drzew decyzyjnych na analizowanych danych

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

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

Parametry optymalizowane w opisany procesie to:

- *max\_depth* – maksymalna głębokość każdego drzewa (niekoniecznie osiągana)
- *n\_estimators* – liczba drzew
- *learning\_rate* – parametr szybkości uczenia, komplementarny do *n\_estimators*, w praktyce można ustalić liczbę drzew i szukać optymalnej szybkości uczenia
- *subsample*, *colsample\_bytree*, *colsample\_bylevel* – parametry regularyzacyjne określające ułamek kolejno: wierszy użytych do trenowania każdego drzewa, kolumn użytych w każdym drzewie (cechy losowane raz dla danego drzewa), kolumn użytych przy każdym podziale (cechy losowane przy każdym podziale)
- $\gamma$  – minimalny zysk w postaci zmniejszenia wartości funkcji straty konieczny do wykonania podziału

## 2.2 Sieci neuronowe

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

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

W niniejszej pracy, wykorzystane zostały dwa rodzaje sieci neuronowych: sieci w pełni połączone (ang. *fully connected NN – FC NN*), nazywane także wielowarstwowymi perceptronami (ang. *multi-layer perceptron – MLP*) oraz sieci konwolucyjne (ang. *convolutional NN – ConvNets, CNN*).

## Sieci w pełni połączone

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

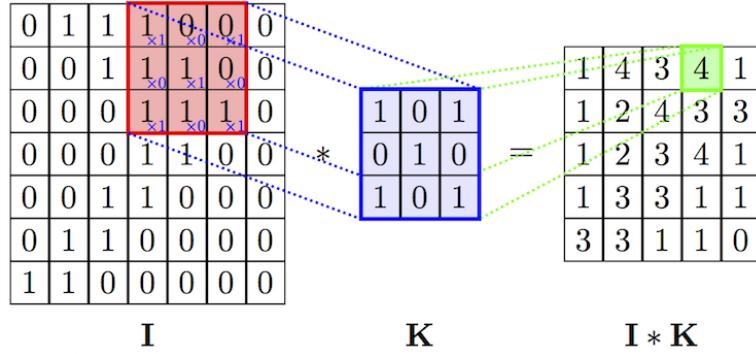
Istnieje twierdzenie o sieciach neuronowych jako uniwersalnych aproksymatorach funkcji (ang. *universal approximation theorem*) [34], mówiące, że już sieć neuronowa o jednej warstwie ukrytej jest zdolna do przybliżenia dowolnej funkcji z dowolną dokładnością. Twierdzenie to nie podaje niestety liczby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

## Sieci konwolucyjne

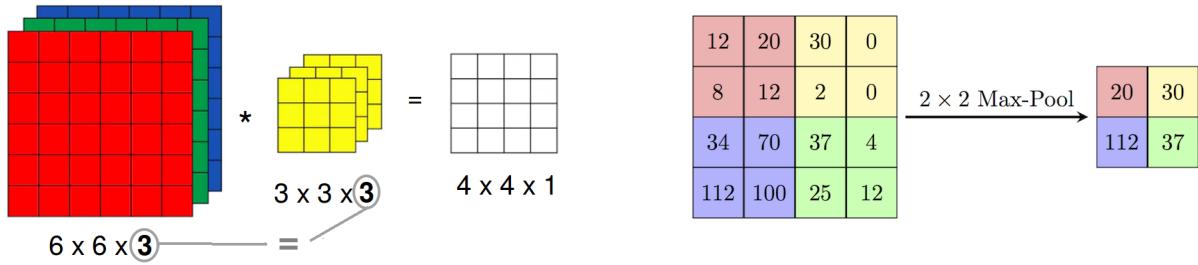
Jednym ze sposobów na ograniczenie liczby trenowanych parametrów jest użycie konwolucyjnych sieci neuronowych [35] (bardziej poprawną choć rzadko używaną nazwą w języku polskim jest sieć splotowa). Są one inspirowane połączonymi w korze wzrokowej zwierząt i wywodzą się z badań w obszarze widzenia komputerowego,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

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

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

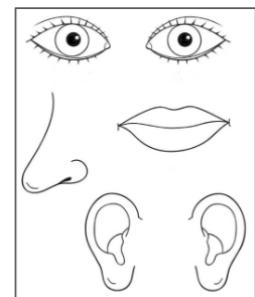
Rysunek 6: Działanie warstwy typu *max-pool*. Źródło: [36].

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.

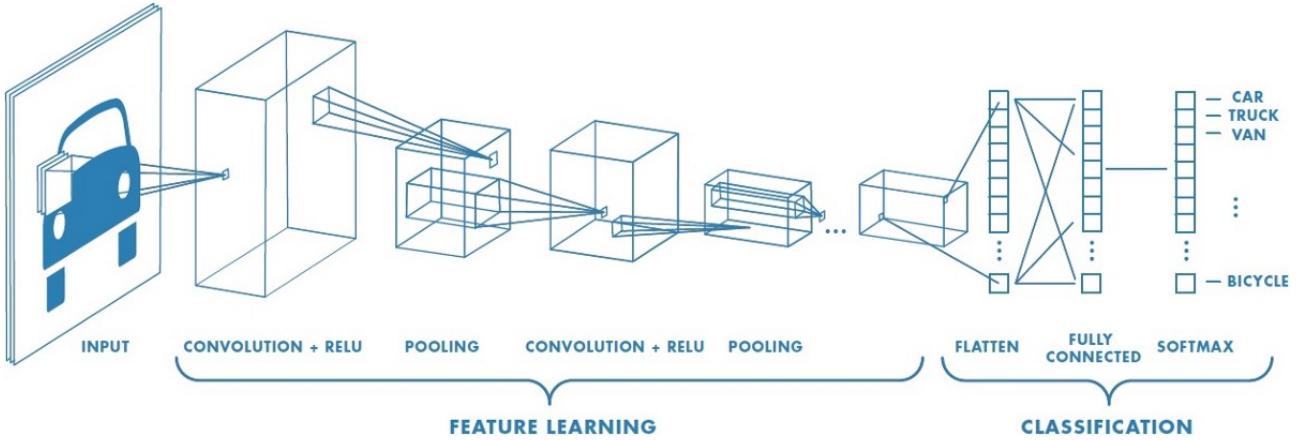
Typowa architektura stosowana w przypadku sieci konwolucyjnych jest następująca: najpierw warstwy konwolucyjne (pomiędzy nimi czasem warstwy typu *max-pool*), następnie wszystkie filtry są rozwijane i składane w długi jednowymiarowy wektor, który przekazywany jest do warstw typu *FC*. W przypadku problemu 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*:

- niezmienność 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].

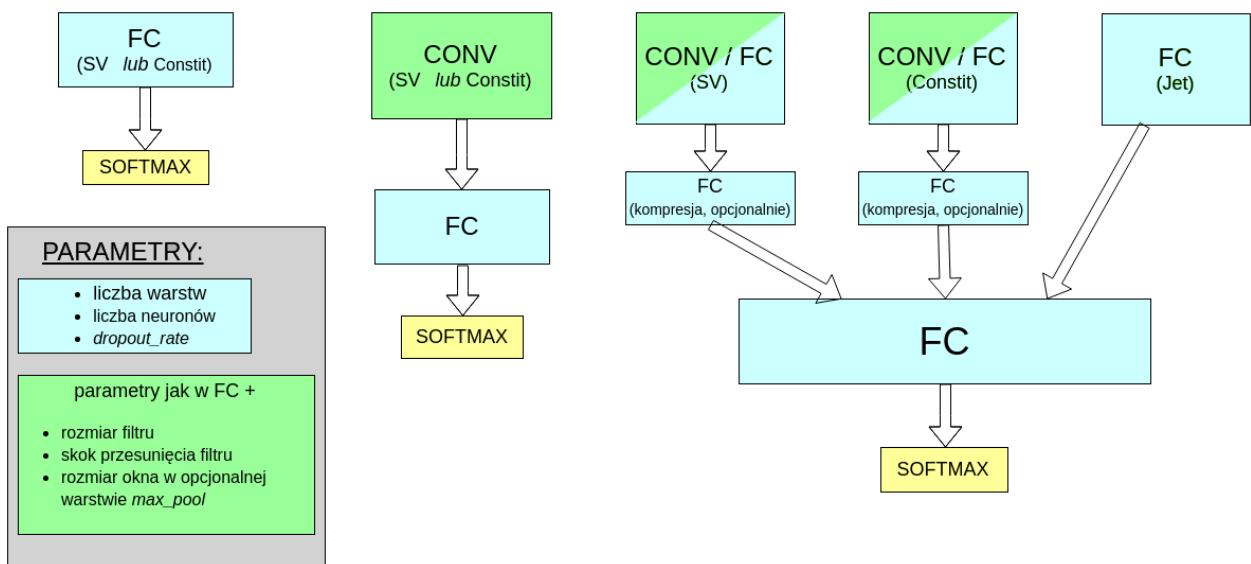


Rysunek 8: Typowa struktura stosowana w sieciach konwolucyjnych. Warstwy konwolucyjne mają za zadanie wydobywać cechy, na podstawie których późniejsze warstwy dokonują klasyfikacji. Widoczna jest charakterystyczna stopniowa zmiana rozmiaru przekazywanej macierzy: rozmiar poprzeczny maleje kosztem głębokości,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

### Hiperparametry i trenowanie sieci neuronowych na analizowanych danych

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

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

Zestaw hiperparametrów definiujący działanie sieci w pełni połączonej:

- liczba warstw ukrytych
- liczba neuronów w każdej warstwie
- funkcja aktywacji – nielinowa funkcja aplikowana przed zwróceniem wartości w każdym neuronie, najpopularniejsze to  $tanh$ ,  $ReLU$  ( $f(x) = \max(0, x)$ ) oraz funkcja sigmoidalna ( $f(x) = \frac{1}{1+exp(x)}$ )
- algorytm optymalizacyjny – spadek gradientowy lub jego wariacje
- parametr szybkości uczenia i jego modyfikacje w trakcie uczenia
- liczba przykładów trenujących przetwarzanych w jednym kroku uczenia (ang. *batch\_size*) – im większy tym szybsze jest trenowanie sieci (dzięki wydajnym operacjom macierzowym), natomiast może się to odbywać kosztem precyzji
- liczba epok uczenia – ile razy będzie pokazywany sieci każdy przykład
- opcjonalnie: warunki stopu (ang. *early stopping*), czyli przerwanie procesu uczenia mające na celu uniknięcie przetrenowania sieci, w momencie gdy błąd popełniany na zbiorze walidacyjnym przestaje maleć
- opcjonalnie: regularyzacja przy pomocy różnych technik (zwykle konieczna)

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

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

- *batch\_size* zawsze równy 64 (inne testowane wartości: 16, 32, 128)
- za algorytm optymalizacyjny przyjęto algorytm o nazwie *Nadam* [42], tj. rozwinięcie algorytmu *Adam* [43] o parametr Nesterova (inne testowane to zwykły spadek gradientowy oraz *Adam*)
- funkcję aktywacji: *ReLU*
- liczbę epok równą 50, 100 lub 200, zrezygnowano z *early stopping*
- stałe w trakcie treningu wartości parametru szybkości uczenia
- spośród technik regularizacyjnych testowano wyłącznie *dropout* [44] z prawdopodobieństwem odrzucenia równym 0.1, 0.2 lub 0.5
- kilka wybranych kombinacji dla zestawu parametrów: rozmiar filtra, długość skoku i rozmiar okna w warstwach *max-pool* – takie same w kolejnych warstwach

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

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

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

### 2.3 Dyskusja użycia dwóch algorytmów

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

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

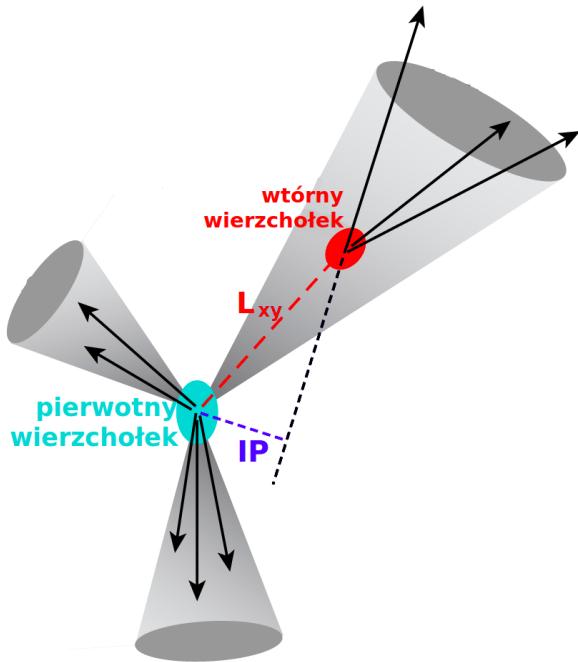
Z kolei drzewa decyzyjne posiadają stosunkowo niewielką liczbę parametrów, a ich trenowanie jest bardzo szybkie co pozwala na ich ekstensywne przeszukiwanie i otrzymanie wyników, które można uznać za optymalne dla tego algorytmu. Ponadto, w przypadku drzew istnieją niewymagające dodatkowych obliczeń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

### 3 Dane

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

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

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



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

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

Lista używanych zmiennych:

- Zmienne charakteryzujące dżet:

- $\eta_{jet}, \phi_{jet}$  – pseudopospieszność i kąt azymutalny osi dżetu
- $p_{T,jet}$  – pęd poprzeczny dżetu
- $M_{jet} = \sqrt{(\sum_i E_i)^2 - (\sum_i \vec{p}_i)^2}$  – masa dżetu, gdzie  $E_i$  i  $\vec{p}_i$  to energia i pęd kolejnych cząstek tworzących dżet

- $A_{jet}$  – powierzchnia dżetu wyliczana przez algorytm  $kt$  w płaszczyźnie  $(\eta, \phi)$ . Do powierzchni dżetu zaliczany jest każdy element powierzchni, w którym dodanie części o nieskończonym małym pędzie poprzecznym sprawi, że zostanie ona zaliczona do tego dżetu [50]

- $\rho_{bckg}$  – gęstość tła w danym zdarzeniu

- Zmienne opisujące cząstki tworzące dżet (składniki dżetu):

- $\eta, \phi$  – pseudopospieszność i kąt azymutalny cząstki względem osi dżetu
- $p_T$  – pęd poprzeczny cząstki
- $IP_D$  – rzut na kierunek poprzeczny wektora  $IP$ , tj. odległość najbliższego zbliżenia
- $IP_Z$  – rzut na oś  $z$  wektora  $IP$
- $N_{Constit}$  – liczba cząstek tworzących dżet

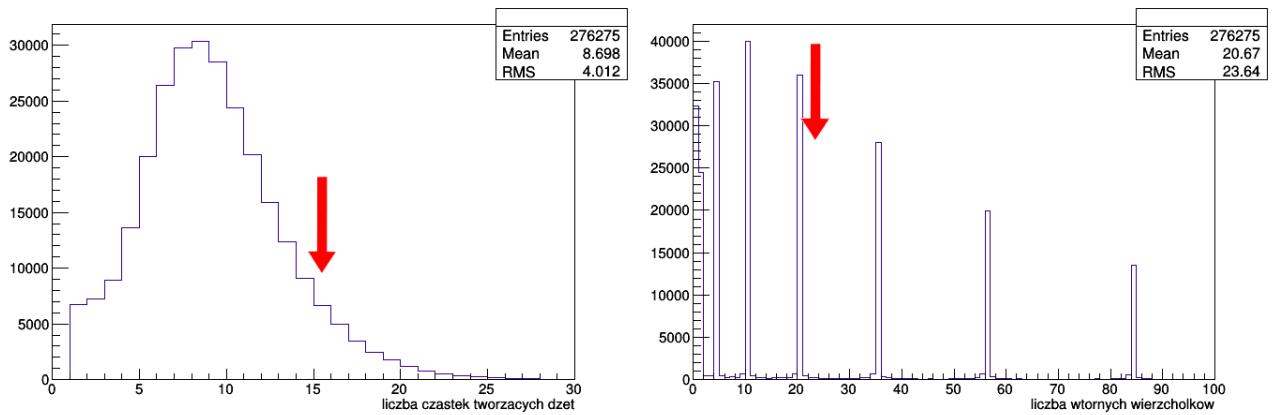
- Zmienne opisujące wtórne wierzchołki:

- $L_{xy}$  – odległość między pierwotnym a wtórnym wierzchołkiem (ang. *decay length*) w płaszczyźnie  $x - y$
- $\sigma_{L_{xy}}$  – niepewność wyznaczenia  $L_{xy}$
- $\sigma_{vertex} = \sqrt{d_1^2 + d_2^2 + d_3^2}$  – rozrzut śladów (ang. *tracks*) wokół wtórnego wierzchołka, gdzie  $d_i$  to najmniejsza odległość pomiędzy śladem  $i$ -tej cząstki a wtórnym wierzchołkiem
- $M_{inv} = \sqrt{(E_1 + E_2 + E_3)^2 - (\vec{p}_1 + \vec{p}_2 + \vec{p}_3)^2}$  – masa niezmiennica wierzchołka, gdzie  $E_i, \vec{p}_i$  to energia i pęd  $i$ -tej cząstki tworzącej wierzchołek
- $\chi^2/Ndf$  – jakość dopasowania wtórnego wierzchołka
- $N_{SV}$  – liczba wtórych wierzchołków

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

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

Nastecną kwestią jest wybór wielkości decydującej o kolejności ułożenia elementów, tj. która cząstka będzie cząstką nr 1 a która nr 5. Losowe ułożenie elementów sprawioby, że bezpośrednie porównywanie wartości w danej kolumnach (co ma miejsce bezpośrednio w drzewach decyzyjnych a pośrednio w sieciach neuronowych) stracioby 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 posiliwano 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.

## 4 Analiza

### 4.1 Dobór metryki

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

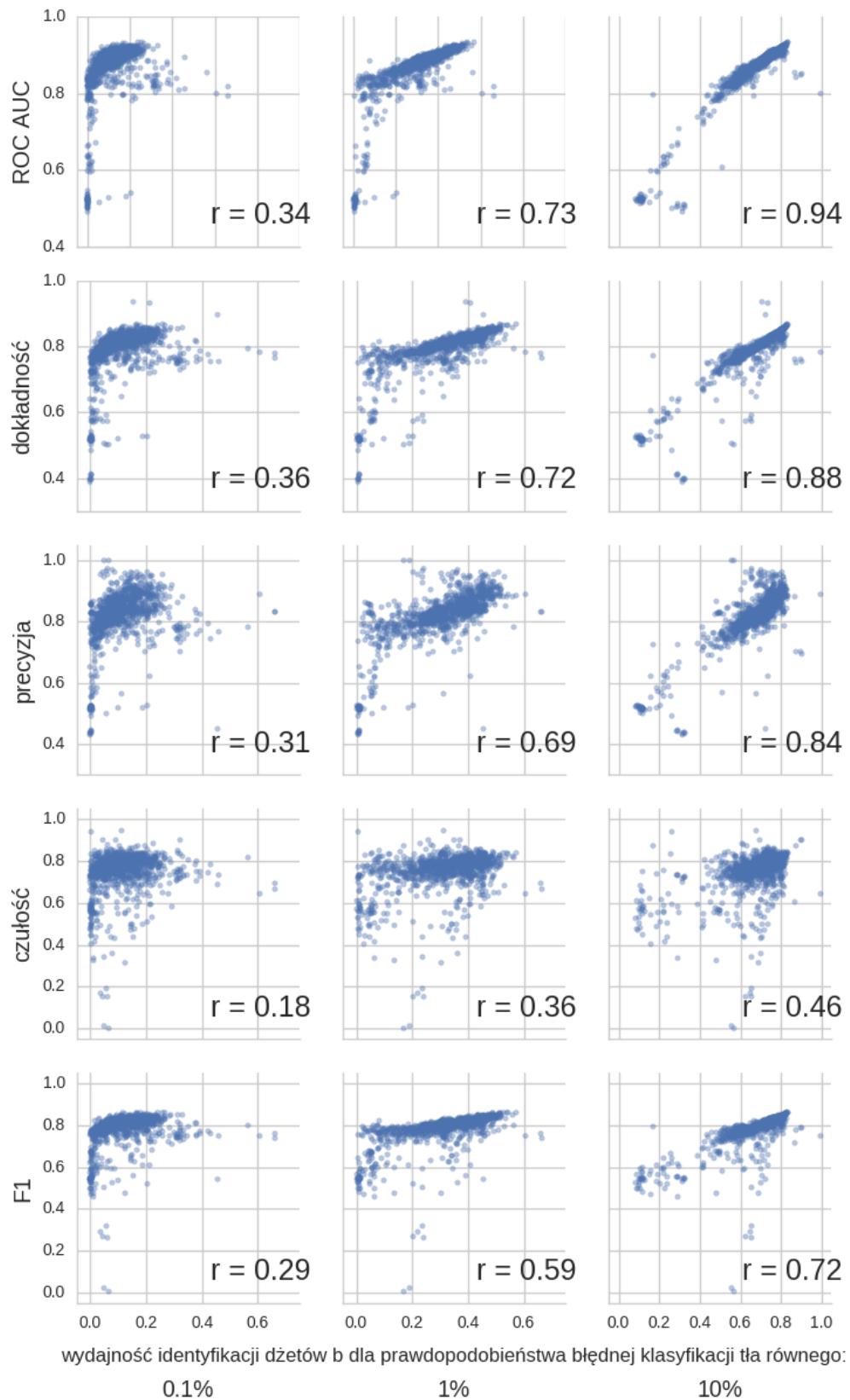
Używanie i porównywanie kilku miar efektywności jest często niepraktyczne dlatego dobrze jest wybrać jedną metrykę. Przy jej wyborze należy kierować się potencjalnymi zastosowaniami modelu. W tym przypadku są to analizy fizyczne, które mogą mieć różne wymagania dotyczące czystości i liczebności otrzymywanych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

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

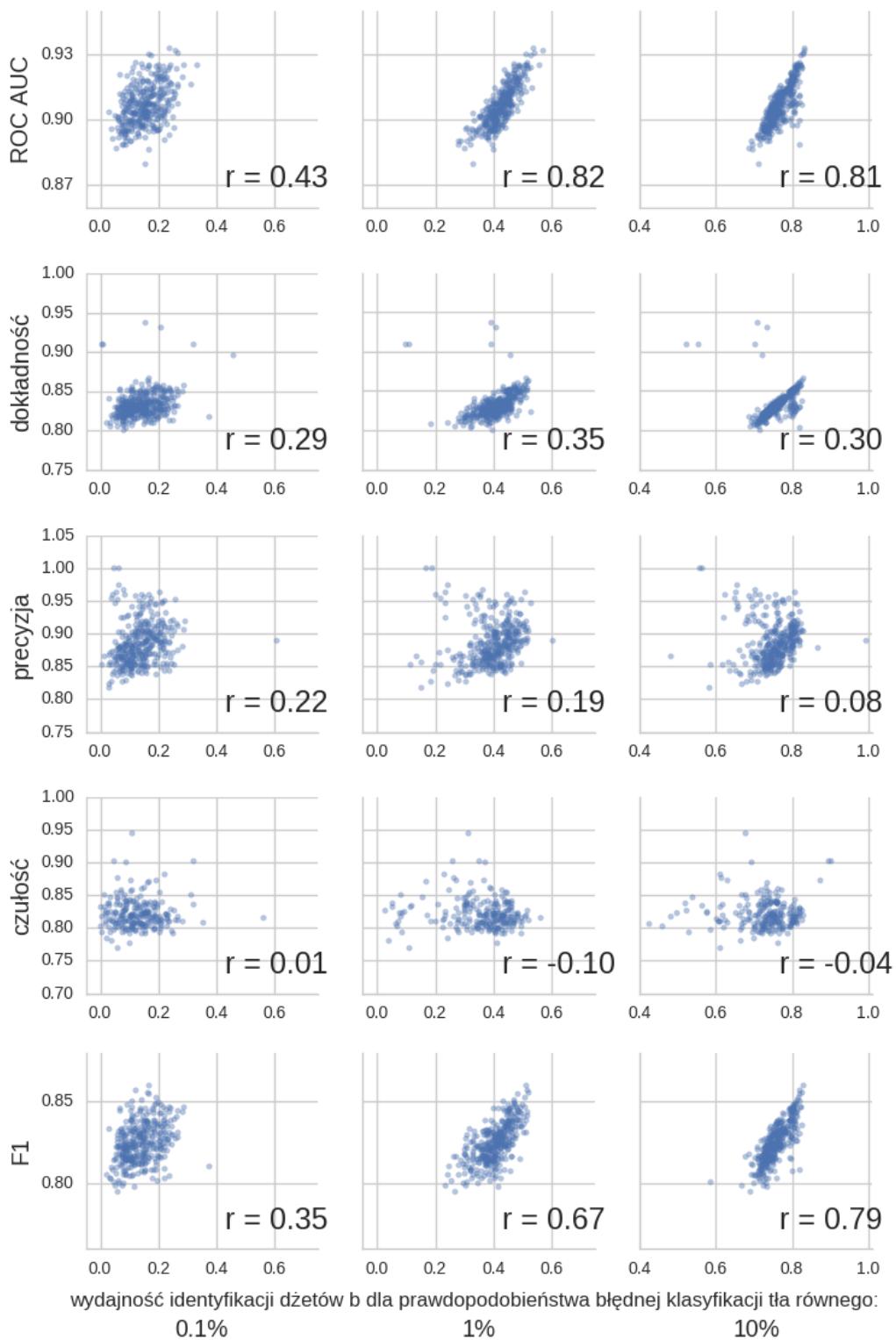
Aby ilościowo porównywać różne algorytmy wprowadzone zostaną trzy punkty pracy: o prawdopodobieństwie błędnej klasyfikacji tła równej 0.1%, 1% oraz 10%. Na Rys. 12 przedstawione zostały zależności poszczególnych metryk od wydajności identyfikacji dżetów  $b$  w tych trzech punktach pracy. Każdy punkt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

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

Pole pod powierzchnią krzywej *ROC* zostało wybrane jako główna metryka używana w 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.

## 4.2 Wyniki dla poszczególnych modeli

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

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

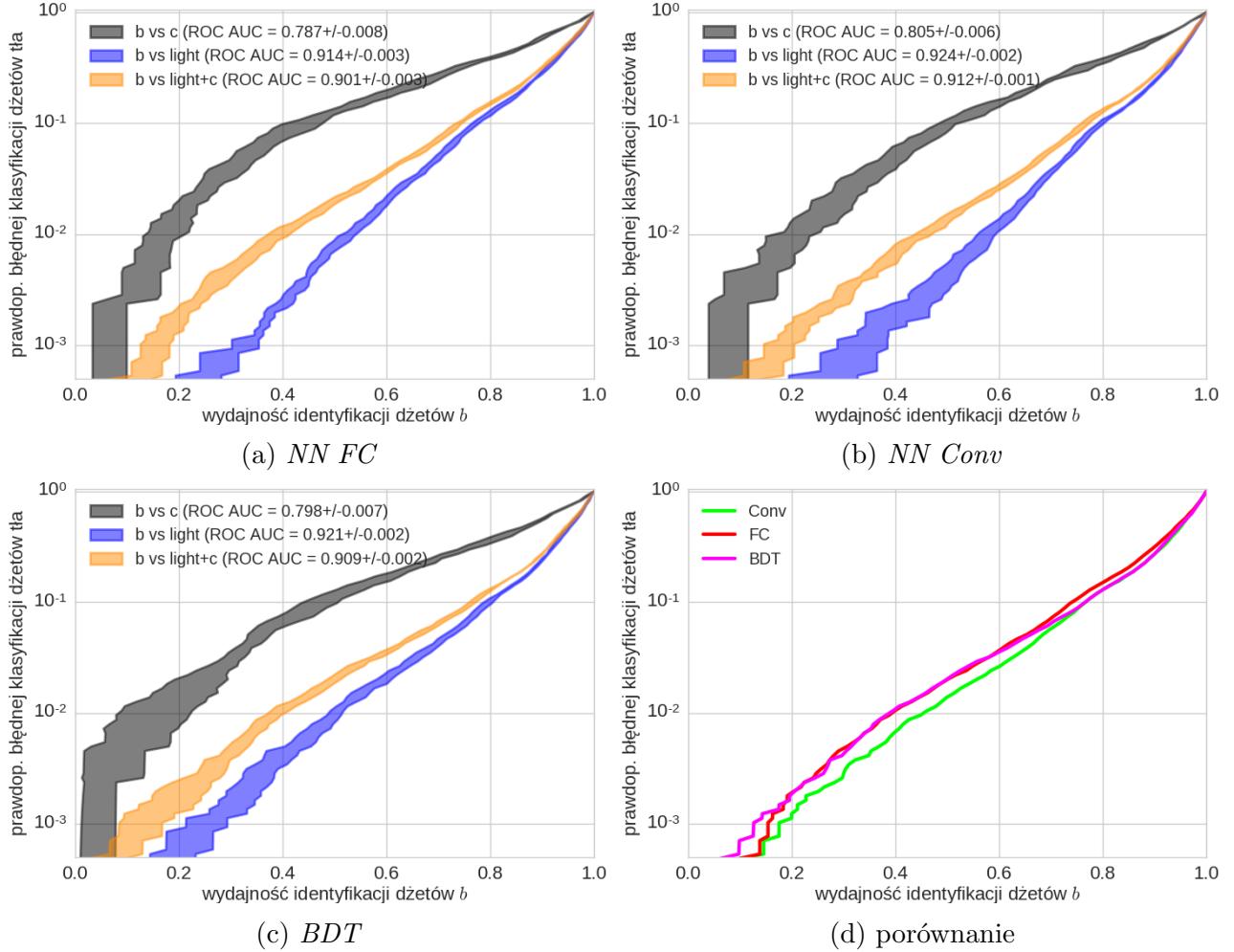
Na Rys. 14, 15, 16 przedstawiono odmianę krzywej *ROC* – wykresy przedstawiające zależności wydajności identyfikacji dżetów  $b$  z niepewnościami (na osi poziomej) dla danego prawdopodobieństwa błędnej klasyfikacji dżetów tła (na osi pionowej).

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

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

#### 4.2.1 Wyniki dla zmiennych związanych z wtórnymi wierzchołkami

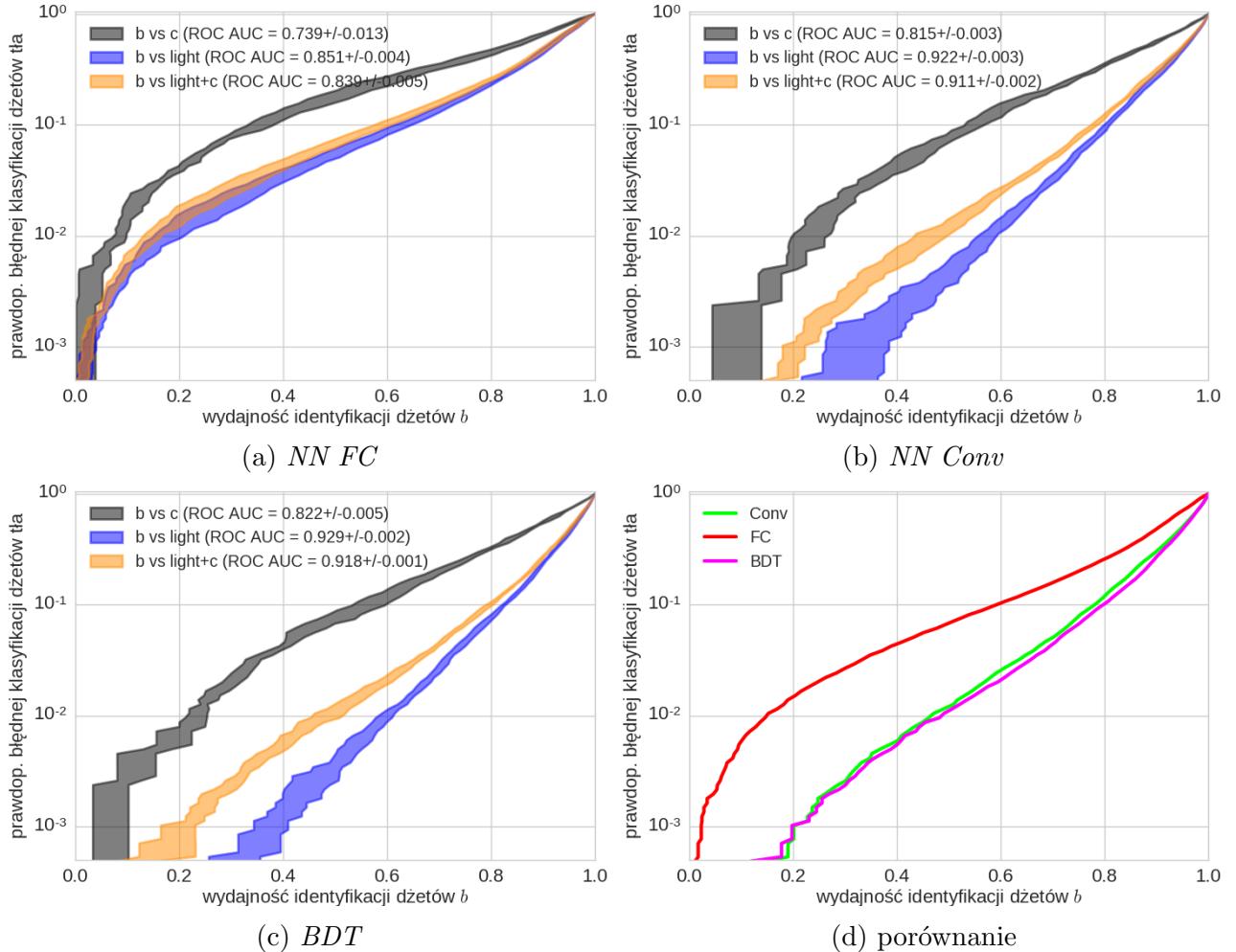
Na Rys. 14 przedstawiono rezultaty uzyskane przy trenowaniu na zbiorze danych *SV*. Uzyskano bardzo zbliżone wyniki dla wszystkich trzech algorytmów, sieci konwolucyjne były nieznacznie lepsze od pozostałych przy wydajnościach na identyfikację dżetów  $b$  poniżej 70%. Dla tych samych prawdopodobieństw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

#### 4.2.2 Wyniki dla zmiennych związanych z częstkomami tworzącymi dżet

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

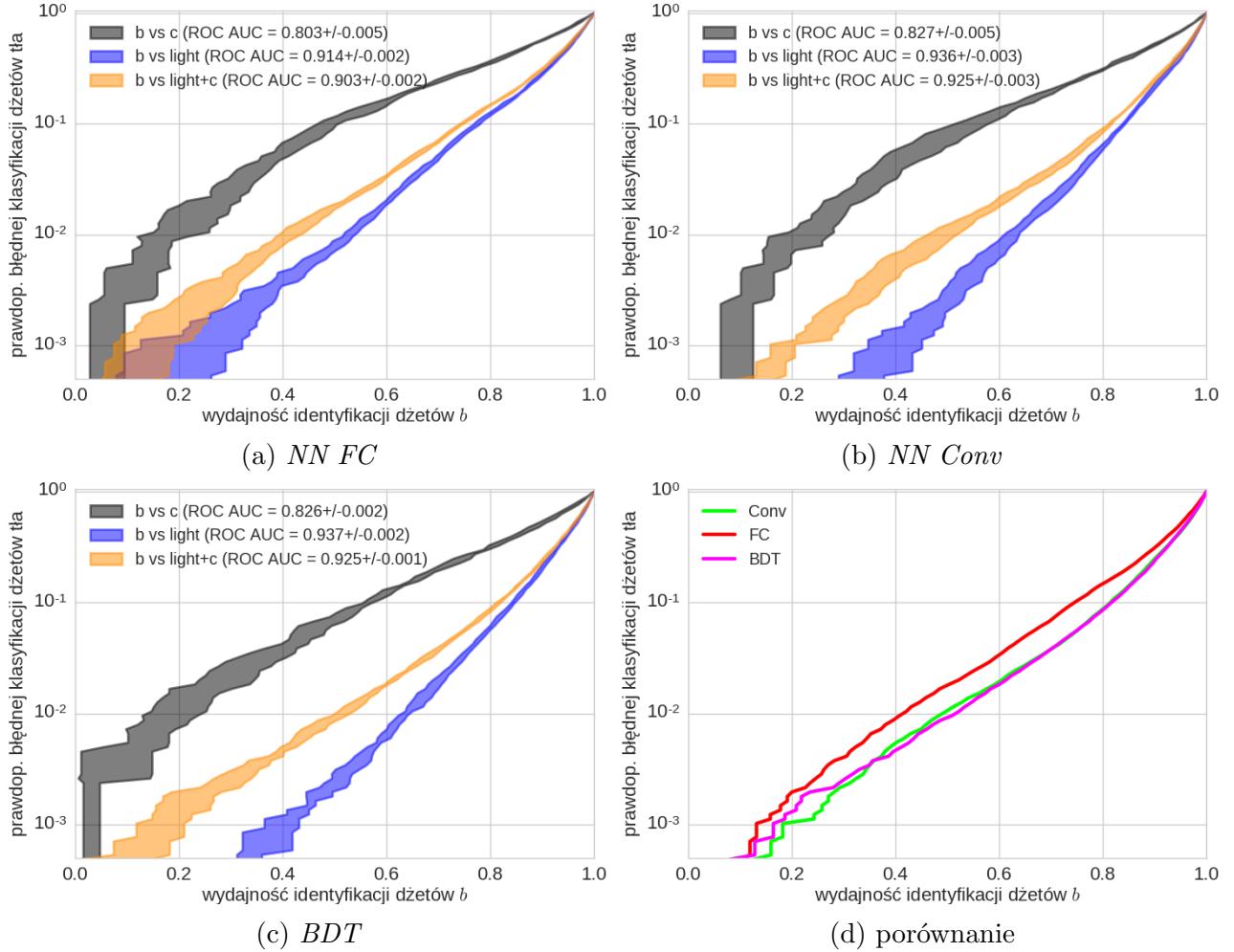


Rysunek 15: Zależności wydajności identyfikacji dżetów *b* od prawdopodobieństwa błędnej klasyfikacji dżetów tła dla poszczególnych algorytmów oraz ich porównanie. Algorytmy wytrenowane na zmiennych *constit*.

### 4.2.3 Wyniki dla wszystkich zmiennych

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

W porównaniu do poprzednich zbiorów danych wszystkie algorytmy poprawiły swoje predykcje, najmniej *FC*, którego zdolności separacyjne są prawie takie same jak w przypadku *SV* (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*.

#### 4.2.4 Podsumowanie - wyniki poszczególnych modeli

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

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

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

W każdym przypadku najgorsze wyniki uzyskano dla sieci w pełni połączonych. Z tabeli wynika, że duże znaczenie mają zarówno użyty algorytm jak i zestaw danych. Podobne wyniki uzyskiwane dla  $BDT$  oraz  $Conv$  pozwalają przypuszczać, że uzyskiwane przez nie poziom błędu 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%	10%
$SV-FC$	$0.901 \pm 0.003$	$15 \pm 3$	$39 \pm 2$	$73.7 \pm 0.6$
$SV-Conv$	$0.912 \pm 0.001$	$18 \pm 3$	$45 \pm 2$	$76.4 \pm 0.8$
$SV-BDT$	$0.909 \pm 0.002$	$13 \pm 4$	$38 \pm 1$	$76.3 \pm 0.6$
$constit-FC$	$0.839 \pm 0.005$	$2 \pm 1$	$15 \pm 2$	$58.6 \pm 1.5$
$constit-Conv$	$0.911 \pm 0.002$	$20 \pm 2$	$46 \pm 3$	$77.8 \pm 0.6$
$constit-BDT$	$0.918 \pm 0.001$	$20 \pm 3$	$48 \pm 3$	$79.4 \pm 0.7$
$merged-FC$	$0.903 \pm 0.002$	$13 \pm 6$	$41 \pm 2$	$74.2 \pm 0.3$
$merged-Conv$	$0.925 \pm 0.003$	$18 \pm 2$	$48 \pm 3$	$81.3 \pm 0.4$
$merged-BDT$	$0.925 \pm 0.001$	$16 \pm 5$	$51 \pm 1$	$81.4 \pm 0.4$

Tablica 1: Tabela podsumowująca wyniki uzyskane przez poszczególne modele.

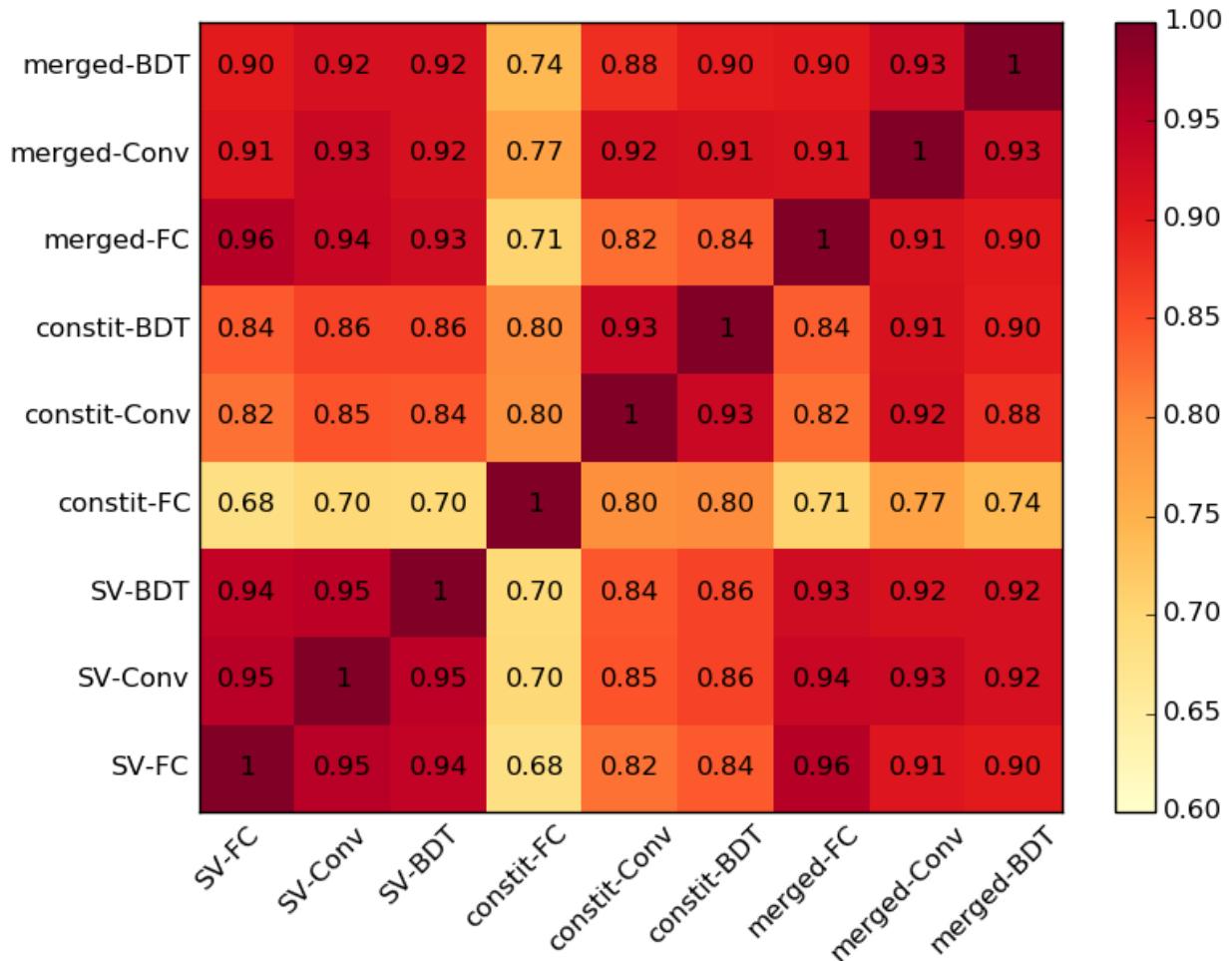
### 4.3 Korelacje predykcji modeli

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

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

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

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

## 4.4 Analiza istotności zmiennych w *BDT*

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

Wykorzystano trzy miary istotności zmiennych zaimplementowane w bibliotece **XGBoost**:<sup>4</sup>

- *weight* informuje ile razy dana zmienna została wykorzystana w podziałach drzew.
- *total\_gain* jest sumą po spadkach funkcji straty uzyskiwanych dzięki podziałom wykorzystującym daną zmienną
- *total\_cover* to suma liczb przykładów trenujących, na które miały wpływ podziały wykorzystujące daną zmienną.

Zasadniczo najważniejszą miarą jest *total\_gain*, gdyż uwzględnia ona zarówno częstość wykorzystania danej zmiennej (*weight*) jak i średni zysk w postaci spadku funkcji straty uzyskiwany w poszczególnych podziałach. Zmienna binarna wykorzystana już w jednym podziale drzewa, nie może zostać użyta ponownie w żadnej gałęzi będącej kontynuacją danego podziału, stąd zmienne binarne zwykle osiągają mniejsze wartości miary *weight*. *total\_cover* jest z kolei związane z głębokościami drzewa na jakiej wykorzystywana była zmienna – pierwszy podział w danym drzewie wpływa zawsze na wszystkie przykłady trenujące, natomiast kolejne – na stopniowo coraz mniejszy ułamek. Zatem niskie wartości *total\_cover* sugerują, że dana zmienna wykorzystywana była raczej na dużych głębokościach (na już mocno podzielonym drzewie).

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

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

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

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

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

<sup>4</sup>nazwy 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>	osobno <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
$\chi^2/Ndf$ - 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
$\sigma_{vertex}$ - SV0	2.70	1.40	2.08
$\sigma_{Lxy}$ - SV9	0.91	1.36	1.38
$\chi^2/Ndf$ - 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
$\phi$ - 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
$\sigma_{Lxy}$	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
$\sigma_{vertex}$	17.0	8.2	14.2	6.6	6.0	6.9
$\chi^2/Ndf$	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
$\phi$	23.2	8.7	8.2	12.4	8.4	8.2
$\eta$	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*.

## 4.5 Analiza wpływu zmiennych na predykcje modeli

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

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

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

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

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

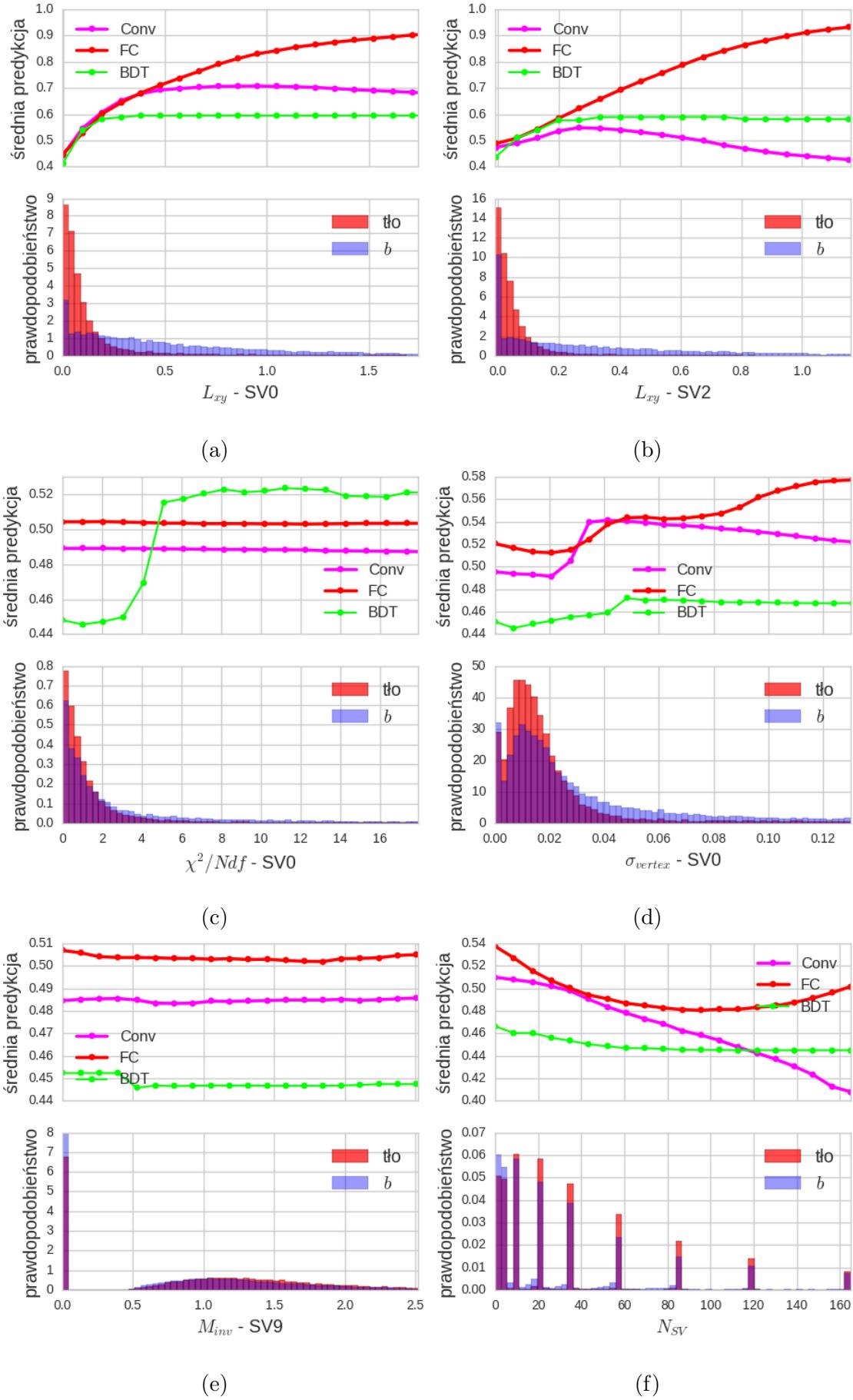
Analiza tych pięciu wykresów pozwala na zaobserwowanie ciekawych różnic między algorytmami. Pierwsza z nich: zależności cząstkowe wydają się mieć mniejszą amplitudę w przypadku  $BDT$  niż w przypadku sieci neuronowych. Może wynikać to z kształtu funkcji – w przypadku  $BDT$  jest to złożenie kilku funkcji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

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

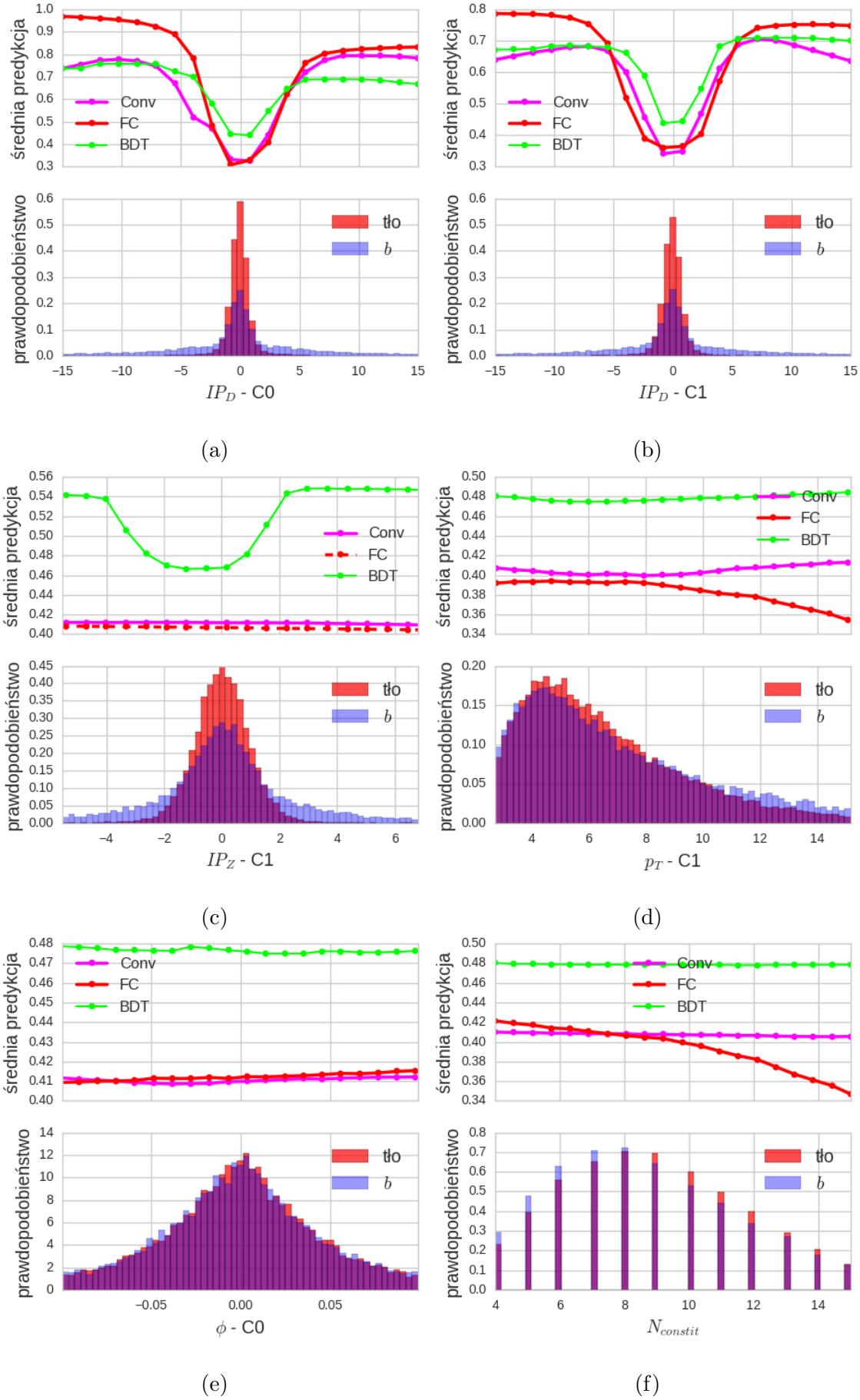
być specyficzny sposób trenowania drzew decyzyjnych, które w przypadku takiego rozkładu nie mogą podzielić przestrzeni fazowej na 3 części w jednym kroku, co byłoby najbardziej korzystne, ale zmuszone są do dwóch kolejnych podziałów, np. wzduż wartości -2 i 2, z czego pierwszy podział musi niejako złamać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

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

Wreszcie wykresy przedstawione na Rys. 18 e) oraz Rys. 19 e) pokazują, że nawet w przypadku zmiennych intensywnie wykorzystywanych przy podziałach drzewa (o czym świadczy Tab. 2) zmiana wyłącznie ich wartości ma często znikomy wpływ na predykcje całego modelu. Są to zmienne, których rozkłady dla sygnału i tła prawie się nie różnią. Ich wpływ na predykcje algorytmu ujawnia się dopiero w kombinacji z innymi zmiennymi. Wykorzystanie tych zmiennych jest dużo bardziej trudne dla ludzi i między innymi na tym polega przewaga podejścia analizy wielowymiarowej i uczenia maszynowego. Wykresy te sygnalizują jednocześnie ograniczenie metody zależności cząstkowych, która jak każda metoda graficzna nie radzi sobie z przestrzeniami 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 niepotrzebny wpływ na predykcje ( $total\_gain < 6\%$ ). 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 wewnętrzna 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.

## Dodatek A Metryki

W poniżej tabeli zebrano stosowane najczęściej miary jakości klasyfikatorów. We wzorach definiujących metryki wykorzystano następujące wielkości:

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

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

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

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

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.

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

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

## Dodatek B Skróty i oznaczenia

W pracy używane są następujące skróty i oznaczenia:

- algorytmy:  
 $FC$  – sieci neuronowe w pełni połączone,  
 $Conv$  - sieci neuronowe konwolucyjne,  
 $BDT$  - wzmacniane drzewa decyzyjne
- zbiory danych:  
 $SV$  – zestaw zawierający tylko zmienne związane z wtórnymi wierzchołkami,  
 $constit$  – zestaw zawierający tylko zmienne związane z częstками tworzącymi dżet,  
 $merged$  – zestaw zawierający wszystkie zmienne (tj.  $SV + Constit +$ zmienne charakteryzujące dżet jako całość)
- modele (zestaw danych + algorytm) nazywane są wg wzoru:  
(oznaczenie\_zbioru\_danych)-(oznaczenie\_algorytmu),  
np.  $SV\text{-}Conv$  oznacza konwolucyjną sieć neuronową wytrenowaną na zmiennych związanych z wtórnymi wierzchołkami a  $merged\text{-}BDT$  - wzmacniane drzewa decyzyjne, do treningu których użyte zostały wszystkie zmienne. Zapis  $merged\text{-}X$  stanowi zbiorcze oznaczenie modeli  $merged\text{-}SV$ ,  $merged\text{-}Conv$  i  $merged\text{-}BDT$ .
- poszczególne zmienne (kolumny w zbiorze danych) oznaczane są według wzoru:  
(nazwa\_zmiennej) – (numer\_obiektu),  
np.  $\sigma_{L_{xy}}$  –  $SV2$  oznacza niepewność wyznaczenia  $L_{xy}$  dla wtórnego wierzchołka nr 2, a  $IP_Z\text{-}C5$  – rzut odległości najbliższego zbliżenia cząstki na oś wiązki cząstki nr 5 – numery na posortowanych listach (Rozdz. 3, gdzie znajdują się także opisy wielkości fizycznych).

# Bibliografia

- [1] Donald H. Perkins. "Oddziaływanie międzykarkowe i chromodynamika kwantowa". W: *Wstęp do Fizyki Wysokich Energii*. 2 wydr. PWN, 2005, 171–193.
- [2] David J. Gross i Frank Wilczek. "Ultraviolet Behavior of Nonabelian Gauge Theories". W: *Phys. Rev. Lett.* 30 (1973). [,271(1973)], s. 1343–1346. DOI: [10.1103/PhysRevLett.30.1343](https://doi.org/10.1103/PhysRevLett.30.1343).
- [3] H. David Politzer. "Reliable Perturbative Results for Strong Interactions?" W: *Phys. Rev. Lett.* 30 (1973). [,274(1973)], s. 1346–1349. DOI: [10.1103/PhysRevLett.30.1346](https://doi.org/10.1103/PhysRevLett.30.1346).
- [4] C. Patrignani i in. "Review of Particle Physics". W: *Chin. Phys. C*40.10 (2016), s. 100001. DOI: [10.1088/1674-1137/40/10/100001](https://doi.org/10.1088/1674-1137/40/10/100001).
- [5] John C. Collins i M. J. Perry. "Superdense Matter: Neutrons Or Asymptotically Free Quarks?" W: *Phys. Rev. Lett.* 34 (1975), s. 1353. DOI: [10.1103/PhysRevLett.34.1353](https://doi.org/10.1103/PhysRevLett.34.1353).
- [6] N. Cabibbo i G. Parisi. "Exponential Hadronic Spectrum and Quark Liberation". W: *Phys. Lett.* 59B (1975), s. 67–69. DOI: [10.1016/0370-2693\(75\)90158-6](https://doi.org/10.1016/0370-2693(75)90158-6).
- [7] D. Boyanovsky, H. J. de Vega i D. J. Schwarz. "Phase transitions in the early and the present universe". W: *Ann. Rev. Nucl. Part. Sci.* 56 (2006), s. 441–500. DOI: [10.1146/annurev.nucl.56.080805.140539](https://doi.org/10.1146/annurev.nucl.56.080805.140539). arXiv: [hep-ph/0602002 \[hep-ph\]](https://arxiv.org/abs/hep-ph/0602002).
- [8] Mark G. Alford i Kai Schwenzer. "What the Timing of Millisecond Pulsars Can Teach us about Their Interior". W: *Phys. Rev. Lett.* 113.25 (2014), s. 251102. DOI: [10.1103/PhysRevLett.113.251102](https://doi.org/10.1103/PhysRevLett.113.251102). arXiv: [1310.3524 \[astro-ph.HE\]](https://arxiv.org/abs/1310.3524).
- [9] Vardan Khachatryan i in. "Evidence for collectivity in pp collisions at the LHC". W: *Phys. Lett.* B765 (2017), s. 193–220. DOI: [10.1016/j.physletb.2016.12.009](https://doi.org/10.1016/j.physletb.2016.12.009). arXiv: [1606.06198 \[nucl-ex\]](https://arxiv.org/abs/1606.06198).
- [10] Jaroslav Adam i in. "Enhanced production of multi-strange hadrons in high-multiplicity proton-proton collisions". W: *Nature Phys.* 13 (2017), s. 535–539. DOI: [10.1038/nphys4111](https://doi.org/10.1038/nphys4111). arXiv: [1606.07424 \[nucl-ex\]](https://arxiv.org/abs/1606.07424).
- [11] Matteo Cacciari, Gavin P. Salam i Gregory Soyez. "The Anti-k(t) jet clustering algorithm". W: *JHEP* 04 (2008), s. 063. DOI: [10.1088/1126-6708/2008/04/063](https://doi.org/10.1088/1126-6708/2008/04/063). arXiv: [0802.1189 \[hep-ph\]](https://arxiv.org/abs/0802.1189).
- [12] Vardan Khachatryan i in. "Charged-particle nuclear modification factors in PbPb and pPb collisions at  $\sqrt{s_{NN}} = 5.02$  TeV". W: *JHEP* 04 (2017), s. 039. DOI: [10.1007/JHEP04\(2017\)039](https://doi.org/10.1007/JHEP04(2017)039). arXiv: [1611.01664 \[nucl-ex\]](https://arxiv.org/abs/1611.01664).
- [13] Betty Abelev i in. "Centrality Dependence of Charged Particle Production at Large Transverse Momentum in Pb–Pb Collisions at  $\sqrt{s_{NN}} = 2.76$  TeV". W: *Phys. Lett.* B720 (2013), s. 52–62. DOI: [10.1016/j.physletb.2013.01.051](https://doi.org/10.1016/j.physletb.2013.01.051). arXiv: [1208.2711 \[hep-ex\]](https://arxiv.org/abs/1208.2711).
- [14] Carlos A. Salgado i Urs Achim Wiedemann. "Calculating quenching weights". W: *Phys. Rev.* D68 (2003), s. 014008. DOI: [10.1103/PhysRevD.68.014008](https://doi.org/10.1103/PhysRevD.68.014008). arXiv: [hep-ph/0302184 \[hep-ph\]](https://arxiv.org/abs/hep-ph/0302184).
- [15] Yuri L. Dokshitzer i D. E. Kharzeev. "Heavy quark colorimetry of QCD matter". W: *Phys. Lett.* B519 (2001), s. 199–206. DOI: [10.1016/S0370-2693\(01\)01130-3](https://doi.org/10.1016/S0370-2693(01)01130-3). arXiv: [hep-ph/0106202 \[hep-ph\]](https://arxiv.org/abs/hep-ph/0106202).
- [16] Georges Aad i in. "Performance of b-Jet Identification in the ATLAS Experiment". W: *JINST* 11.04 (2016), P04008. DOI: [10.1088/1748-0221/11/04/P04008](https://doi.org/10.1088/1748-0221/11/04/P04008). arXiv: [1512.01094 \[hep-ex\]](https://arxiv.org/abs/1512.01094).

- [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/](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](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.* C74.8 (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](https://www-d0.fnal.gov/Run2Physics/top/singletop_observation/singletop_observation_updated.html) (term. wiz. 15.07.2018).

- [50] Matteo Cacciari i Gavin P. Salam. “Pileup subtraction using jet areas”. W: *Phys. Lett.* B659 (2008), s. 119–126. DOI: 10.1016/j.physletb.2007.09.077. arXiv: 0707.1378 [hep-ph].
- [51] Andrew P Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. W: *Pattern recognition* 30.7 (1997), s. 1145–1159.
- [52] Jason Gallicchio i Matthew D. Schwartz. “Seeing in Color: Jet Superstructure”. W: *Phys. Rev. Lett.* 105 (2010), s. 022001. DOI: 10.1103/PhysRevLett.105.022001. arXiv: 1001.5027 [hep-ph].