Społeczna Akademia Nauk w Łodzi

ROZPRAWA DOKTORSKA

mgr inż. Stanisław Lota

Analiza wyjaśnialności algorytmów uczenia maszynowego w zadaniach detekcji ataków w programowalnych sieciach komputerowych

Rozprawa doktorska napisana pod kierunkiem
prof. dr hab. inż. Leszka Rutkowskiego

ŁÓDŹ, 2025 r.

Tytuł w języku angielskim / Title in English

“Analysis of Explainability of Machine Learning Algorithms in Attack Detection Tasks in Software-Defined Networks”

Słowa kluczowe w j. polskim

sieci definiowane programowo, rozproszone ataki odmowy usługi, wyjaśnialna sztuczna inteligencja, uczenie maszynowe zorientowane na dane, przeciwstawne uczenie maszynowe, zatruwanie danych, kontroler sdn, ngboost, shap, metryki wyjaśnialności, sanityzacja etykiet, stabilność modeli, monotoniczność modeli, wierność modeli, niewierność modeli, niekompletność modeli, perturbacje danych wejściowych, selekcja cech

Słowa kluczowe w j. angielskim / Keywords in English

software-defined networking, distributed denial of service, explainable artificial intelligence, data-centric artificial intelligence, adversarial machine learning, data poisoning, sdn controller, ngboost, shap, explainability metrics, label sanitization, model stability, model monotonicity, model fidelity, model infidelity, model incompleteness, input data perturbations, feature selection

„Trudne czasy tworzą silnych ludzi…”

Pragnę serdecznie podziękować Panu Prof. dr hab. inż. Leszkowi Rutkowskiemu za nieocenioną pomoc, cierpliwość, zaangażowanie, cenne wskazówki inspirację i wsparcie na każdym etapie mojej pracy badawczej. Pańska życzliwość i profesjonalizm były dla mnie bezcenną motywacją.

Rozprawę doktorską dedykuję mojej rodzinie, żonie Katarzynie i córeczkom Wiktorii, Antoninie i Zofii.
8.03.2025

Spis treści

1. Wstęp

Sieci definiowane programowo (SDN, ang. Software-Defined Network) w ostatnich latach zyskują na popularności, będąc coraz szerzej wykorzystywane w różnorodnych środowiskach, takich jak zwirtualizowane centra danych, sieci rozległe WAN, czy sieci dostępowe. Głównym ich celem jest zastąpienie tradycyjnego modelu sieci komputerowych opartych o urządzenia fizyczne, ich logicznymi odpowiednikami, co pozwala na znaczne ograniczenie kosztów finansowych oraz zredukowanie potrzeby stosowania sprzętu fizycznego.

Sieci definiowane programowo wykorzystują kontrolery programowe lub interfejsy programowania aplikacji tzw. API do scentralizowanej komunikacji z infrastrukturą sieciową i kierowania ruchem zastępując fizyczne routery oraz przełączniki. Skutkiem tego działania jest to, że dzięki kontrolerom sieci SDN administratorzy mają możliwość automatycznego i dynamicznego wykrywania oraz konfiguracji sieci, a także przełączników bez konieczności ręcznego wprowadzania zmian [1] [2]. W branży sieciowej często spotykamy dwa terminy: "sieć programowalna" (ang. network programmability) oraz "otwarta sieć" (ang. open networking). Oba te pojęcia odnoszą się do koncepcji, która umożliwia programowanie sieci za pomocą popularnych języków programowania, takich jak Python, niezależnie od systemów operacyjnych czy sprzętu dostarczonego przez konkretnych producentów (np. Cisco). Otwiera to drogę do większej elastyczności i kontroli nad infrastrukturą sieciową przez dostosowanie jej do potrzeb organizacji bez ograniczeń wynikających z zamkniętych rozwiązań sprzętowych.

Z technicznego punktu widzenia nie ma problemu w połączeniu fizycznej sieci komputerowej z siecią opartą o rozwiązania definiowane programowo [3], gdyż obie technologie płynnie integrują się, a rosnąca popularność sieci SDN znajduje odzwierciedlenie w statystykach prezentowanych w materiałach prasowych. Według danych opublikowanych przez firmę Statista ruch z sieci definiowanych programowo i wykorzystywanych w tradycyjnych sieciach komputerowych zwirtualizowanych funkcji sieciowych NFV umiejscowionych w centrach danych na całym świecie, tylko w samym 2021 r. oszacowano na 7,4 zettabajtów [4]. Zaś w 2023 roku globalny rynek SDN przyniósł przychód w wysokości ponad 24 miliardów dolarów. Statista przewiduje, że do 2028 roku rynek sieci SDN wzrośnie o ponad 60 miliardów dolarów amerykańskich [5]. Jak widać, rosnąca popularność rozwiązań sieciowych opartych na oprogramowaniu w różnych branżach pokazuje, jak ważna stała się ta

technologia dla globalnych firm, szczególnie uwzględniając wzrost popularności chmur obliczeniowych w których usługi sieciowe w modelu subskrypcyjnym są uruchamiane i wyłączane przez klientów na żądanie.

Jednym z głównych problemów związanych z bezpieczeństwem sieci programowalnych jest rozproszony atak odmowy usługi DDoS (ang. Distributed Denial of Service), który ma na celu zakłócenie działania usług sieciowych poprzez wyczerpanie przepustowości i zasobów sieciowych [6]. Według raportu firmy StormWall podsumowującego 2023 rok [7] całkowita liczba ataków DDoS na całym świecie wzrosła o 63% w porównaniu do roku poprzedzającego. Zaobserwowano wtedy gwałtowny wzrost różnorodnych, hybrydowych ataków DDoS, w tym ataków wielowektorowych, ataków na protokół DNS oraz ataków wykorzystujących techniki maskowania i botnety oparte na wirtualnych maszynach. Eskalacja ta była w dużej mierze napędzana czynnikami geopolitycznymi, które nasiliły się w ciągu roku, zwłaszcza po wybuchu konfliktu między Izraelem a Palestyną. Pod względem branż najbardziej dotkniętymi tymi atakami była branża finansowa, w której odnotowano 26% ataków, a także usługi rządowe z 21% i handel detaliczny z 14%. Istotny jest również wzrost liczby ataków w krajach takich jak USA, Rosja, Ukraina, Izrael, Niemcy, Francja, Polska i Zjednoczone Emiraty Arabskie, wynikający głównie z powodu wzmożonej aktywności i bezprecedensowym zaangażowaniu grup (cybergangów czy też specjalnych jednostek rządowych/wojskowych) sponsorowanych przez państwa. Jednak wiele krajów i firm wciąż nadrabia i będzie zmuszona w najbliższym czasie nadrabiać zaległości we wdrażaniu skutecznych środków ochrony przed atakami DDoS.

Samo wykrycie ataków DDoS rzeczywiście stanowi wyzwanie nie tylko z uwagi na ich złożoność, ale przede wszystkim hybrydowość. W ciągu ostatnich lat opracowano wiele rozwiązań mających na celu wykrywanie i łagodzenie ataków DDoS na kontrolery sieci programowalnych w środowisku SDN [8] [9]. Niemniej jednak istnieją pewne charakterystyczne wskaźniki, które mogą wskazywać na to zagrożenie. Przykładowo, nagły i znaczący wzrost ruchu sieciowego z tego samego adresu IP lub zakresu adresów może być alarmującym sygnałem. W celu szybkiej reakcji i umożliwienia zachowania ciągłości pracy infrastruktury i usług w przypadku identyfikacji potencjalnego ataku DDoS warto zwrócić uwagę na nieregularności w wydajności sieci lub obserwowane spowolnienia, a także przypadki, gdy usługa nagle przechodzi w tryb offline [6].

Wykorzystanie sztucznej inteligencji do wykrywania ataków DDoS jest powszechnie uznawane za skuteczną strategię. Badacze na całym świecie podkreślają zalety algorytmów uczenia maszynowego w cyberbezpieczeństwie, szczególnie ze względu na ich zdolność do

dokładnego rozróżniania ruchu szkodliwego od normalnego, a nawet przewidywania potencjalnych ataków [10] [11]. Jeśli włączymy do tego procesu implementację metod wyjaśnialnej sztucznej inteligencji, to jesteśmy w stanie sprawdzić, przeanalizować i zweryfikować podejmowane przez algorytmy decyzje. Wśród algorytmów stosowanych do wykrywania ataków DDoS znajdują się zarówno klasyczne metody, takie jak Naive Bayes, Decision Tree, Random Forest czy Support Vector Machine (SVM), jak i coraz szerzej wykorzystywane, nowsze algorytmy wzmacniania gradientowego, takie jak XGBoost oraz LightGBM, które charakteryzują się wysoką skutecznością w identyfikowaniu tego rodzaju zagrożeń. Sztuczne sieci neuronowe i algorytmy uczenia głębokiego, takie jak konwolucyjne sieci neuronowe (CNN) oraz sieci rekurencyjne (RNN, LSTM, GRU), są również szeroko badane w kontekście detekcji ataków DDoS. Jednak ich zastosowanie wiąże się z pewnymi wyzwaniami. Głównym problemem jest wysoka złożoność obliczeniowa tych modeli, co przekłada się na zwiększone wymagania sprzętowe, takie jak większa moc obliczeniowa i pamięć. Wdrożenie modeli głębokiego uczenia w środowiskach zwirtualizowanych może prowadzić do dodatkowego obciążenia systemu i obniżenia wydajności sieci. Aby skutecznie wykorzystać potencjał modeli głębokiego uczenia, wymagane jest często zastosowanie wydajnych procesorów graficznych (GPU), które przyspieszają proces uczenia i wnioskowania.

W niniejszej rozprawie doktorskiej skupiono się na praktycznych metodach wykrywania i reagowania na ataki DDoS, które można skutecznie wdrożyć w rzeczywistych środowiskach sieciowych. Z tego powodu wybrano algorytmy klasyfikacyjne, które nie wymagają nadmiernej mocy obliczeniowej, w przeciwieństwie do bardziej złożonych sieci głębokich. Zaproponowane algorytmy cechują się skalowalnością i łatwością implementacji w różnych środowiskach, szczególnie w tych zwirtualizowanych.

Pomimo znaczącego postępu w badaniach nad klasyfikacją ataków DDoS przy użyciu uczenia maszynowego, wiele publikacji naukowych nie skupia się na wyjaśnialności modeli. Wynika to z faktu, że wiele współczesnych modeli jest tzw. "modelami czarno-skrzynkowymi", co oznacza, że trudno jest zrozumieć, jak dokładnie model dochodzi do swoich decyzji. Badacze często skupiają się na innych aspektach, takich jak wydajność, skuteczność czy też optymalizacja parametrów modelu. W obliczu rosnącego wdrażania sztucznej inteligencji w różnych sektorach, w tym w obszarach o krytycznym znaczeniu, wzrasta zapotrzebowanie na transparentność i zrozumiałość modeli decyzyjnych. Sztuczna inteligencja zorientowana na wytłumaczalność (ang. Explainable Artificial Intelligence - XAI) stanowi odpowiedź na te potrzeby, oferując ramy dla tworzenia modeli AI, które nie tylko zapewniają wysoką wydajność, ale także umożliwiają zrozumienie procesów decyzyjnych leżących u ich podstaw.

XAI zyskuje na znaczeniu jako dynamicznie rozwijająca się dziedzina badawcza, szczególnie tam gdzie bezpieczeństwo i niezawodność są priorytetem, np. w cyberbezpieczeństwie [12]. Niewątpliwie zauważalna staje się potrzeba weryfikacji podjętych decyzji przez modele uczenia maszynowego w zadaniu wykrywania i reagowania na cyberzagrożenia.

Bezpieczeństwo systemów opartych na sztucznej inteligencji jest ściśle związane z zaufaniem do prawidłowego działania modeli uczenia maszynowego. Jednak modele te są podatne na ataki typu "zatrucie danych" (ang. data poisoning), które polegają na celowym wprowadzeniu błędnych informacji do zbioru treningowego, co może prowadzić do obniżenia dokładności modelu lub błędnych klasyfikacji [13]. Zatrucie danych jest obecnie jednym z najpoważniejszych zagrożeń dla technologii opartych na danych, szczególnie w przypadku danych pochodzących ze źródeł niezaufanych, takich jak sieci sensorowe czy aplikacje zbierające dane użytkowników. Problem ten wynika nie tylko z potencjalnych błędów w algorytmach klasyfikacji, ale również z procesu pozyskiwania i etykietowania danych. Błędy w etykietowaniu mogą pojawić się już na etapie zbierania danych, prowadząc do fałszywych lub zmanipulowanych etykiet w procesie uczenia maszynowego. Bez odpowiednich metod walidacji i filtracji danych, systemy uczące mogą stać się podatne na manipulacje i dezinformację. Aczkolwiek, jeśli dane treningowe są skażone błędami, outlierami czyli wartościami, które znacząco odbiegają od reszty danych w zestawie lub nieścisłościami, model może nauczyć się nieprawidłowych wzorców, co także wpłynie negatywnie na jego zdolność do dokładnej klasyfikacji. Ataki na modele uczenia maszynowego wpisują się w badawczy obszar uczenia maszynowego tzw. Adversarial Machine Learning który skupia się na analizie wpływu szkodliwych ataków i opracowywaniem technik, strategii mających na celu zabezpieczenie modeli uczenia maszynowego przed nimi, wszędzie tam, gdzie użyte algorytmy podejmują istotne decyzje np. systemy bezpieczeństwa, diagnostyka medyczna czy samochody autonomiczne [15]. Większą popularność zyskuje również podejście Data Centric AI (pol. sztuczna inteligencja skoncentrowana na danych), które kładzie nacisk na jakość danych, a nie tylko na rozwój algorytmów. Zamiast skupiać się wyłącznie na modelach Data Centric AI koncentruje się na odpowiednim zarządzaniu danymi, ich oczyszczaniu, identyfikacji i korekcie błędów, a także na dogłębnym zrozumieniu kontekstu danych [14].

W niniejszej rozprawie doktorskiej teza brzmi:

"Autorskie propozycje i implementacje kryteriów wyjaśnialnej sztucznej inteligencji umożliwiają skuteczne wyjaśnienie podejmowanych decyzji przez modele uczenia maszynowego w procesie detekcji rozproszonych ataków odmowy usługi w sieciach definiowanych programowo."

Rozprawa poprzez tą tezę ma na celu wykazanie, iż odpowiednie połączenie algorytmów klasyfikacji, technik selekcji cech oraz metod wyjaśnialności może nie tylko efektywnie wykrywać ataki DDoS, ale również zwiększać zrozumienie, które elementy autorskiego systemu bezpieczeństwa sieci definiowanych programowo są kluczowe. Z racji tego, że system uczący bazuje na zgromadzonych danych, ważnym aspektem staje się identyfikacja cech, które mają największy wpływ na skuteczną detekcję ataków DDoS.

Istotnym wkładem badawczym przedstawionym w niniejszej rozprawie doktorskiej są autorskie propozycje implementacji metryk znanych z literatury służących wyjaśnialności modeli detekcji ataków DDoS. Pomimo dostępności opisu metodologicznego lub matematycznego oraz odniesień w literaturze światowej, opisywane metryki nie mają precyzyjnych instrukcji dotyczących sposobu ich obliczania. Stanowi to interesujący obszar badawczy, gdyż autorskie propozycje implementacji tych metryk wskazują, że badacze mogą stosować różne metody do ich obliczania. Nie eliminuje to niestety problemu subiektywności i braku możliwości porównywania wyników tych metryk z innymi badaniami. Pomimo to, jakość proponowanych metryk została potwierdzona eksperymentalnie na autorskim zbiorze ruchu sieciowego, który został zmapowany w symulowanej sieci definiowanej programowo. Zawarty w nim ruch normalny i szkodliwy DDoS pozwolił w praktyce przetestować autorską metodę selekcji cech i porównać jej wyniki z popularnymi metodami selekcji.

Należy podkreślić, że w opublikowanych dotychczas publikacjach naukowych, badania nad implementacją i zastosowaniem metod wyjaśnialnej sztucznej inteligencji w zadaniu wykrywania ataków DDoS są stosunkowo rzadkie. Pomimo istnienia pewnych analiz wykorzystujących narzędzia takie jak SHAP, LIME czy ważności cech, brakuje kompleksowych badań skupiających się na wyjaśnianiu decyzji algorytmów klasyfikacji w obszarze bezpieczeństwa sieciowego. Koncentrując się nie tylko na ocenie efektywności algorytmów uczenia maszynowego, ale również na dogłębnym zrozumieniu mechanizmów decyzyjnych tych algorytmów zbadano, jak modele opisane w rozprawie funkcjonują w dwóch fundamentalnych kontekstach: pełnej przejrzystości decyzji (model białoskrzynkowy) oraz ograniczonej wiedzy o działaniu modelu (model czarnoskrzynkowy). Szczególną uwagę poświęcono algorytmowi Naturalnego Wzmacniania Gradientowego – NGBoost, opracowanemu w 2019 roku na Uniwersytecie Stanforda, który do tej pory nie był przedmiotem szerokich badań w kontekście wykrywania ataków DDoS jak i w procesie wyjaśnialności podejmowanych przez niego decyzji. W rozprawie zaproponowano przeprowadzenie porównawczej analizy skuteczności algorytmu NGBoost w stosunku do innych popularnych algorytmów klasyfikacji binarnej, stosowanych w niedawnych badaniach dotyczących detekcji

ataków DDoS tj. XGBoost, LightGBM, Decision Tree oraz Random Forest z naciskiem na ich wyjaśnialność.

W ramach eksperymentów sprawdzono również, jak modele zachowują się w warunkach niepewności, na przykład w obliczu celowych perturbacji danych oraz jak stabilne są wyjaśnienia w przypadku ataków zatrucia danych. Celem tych analiz było zbadanie odporności modeli na manipulacje oraz ocena stabilności i niezawodności, ale także odpowiedzenie na pytanie czy wyjaśnienia pozostają wiarygodne nawet w przypadku wystąpienia ataków.

Rozprawa doktorska została podzielona na 10 rozdziałów.

W rozdziale 1 przedstawiono wstęp do rozprawy doktorskiej, który przedstawia tezę, cel i zakres rozprawy doktorskiej oraz strukturę pracy. Wyszczególniono także oryginalne rozwiązania będące rezultatem badań.

W rozdziale 2 omówiono koncepcje związane z tradycyjnymi sieciami komputerowymi oraz sieciami definiowanymi programowo, podkreślając różnice w architekturze, funkcjonalności i implikacjach dla bezpieczeństwa. Przedstawiono również wprowadzenie do problematyki ataków DoS i DDoS, opisując ich mechanizmy i potencjalne skutki. Rozdział ten zawiera także omówienie roli wyjaśnialnej sztucznej inteligencji w zapewnieniu bezpieczeństwa sieci, koncepcji Data-Centric Artificial Intelligence z naciskiem na jakość danych oraz wpływu Adversarial Machine Learning i ataków typu data poisoning na modele uczenia maszynowego stosowane w wykrywaniu ataków DDoS.

W rozdziale 3 przedstawiono przegląd literaturowy, który zawiera przegląd istniejących prac naukowych i badań dotyczących:

- Detekcji ataków DDoS za pomocą algorytmów uczenia maszynowego, w których to omówiono różnorodne techniki wykorzystywane do identyfikacji ataków DDoS, w tym klasyczne algorytmy uczenia maszynowego oraz głębokie sieci neuronowe, podkreślając ich skuteczność i ograniczenia w różnych środowiskach, takich jak sieci definiowane programowo i Internet Rzeczy (IoT).
- Wykorzystania metod wyjaśnialnej sztucznej inteligencji w detekcji ataków DDoS w szczególności takich jak SHAP i LIME, w celu zwiększenia przejrzystości i zaufania do modeli wykrywających ataki DDoS, a także ich zastosowanie w innych dziedzinach, takich jak cyberbezpieczeństwo i medycyna.
- Przykładów ataków zatruwania danych na dokładność i stabilność modeli uczenia maszynowego wykrywających ataki DDoS oraz przedstawienia mechanizmu obronnego,

takiego jak sanityzacja etykiet, która ma na celu zwiększenie odporności systemu na tego typu ataki.

W rozdziale 4 przedstawiono szczegółowy opis oryginalnego środowiska badawczego, które zostało zbudowane w oparciu o narzędzie Mininet (do symulacji sieci SDN), kontroler Ryu (do zarządzania siecią), Scikit-learn (do uczenia maszynowego) oraz inne niezbędne narzędzia. Środowisko to posłużyło do przeprowadzenia eksperymentów i symulacji hybrydowych ataków DDoS.

Zaprezentowano i omówiono proponowaną architekturę systemu detekcji ataków DDoS, który składa się z trzech warstw:

- Warstwa sieciowa SDN, która odpowiada za przechwytywanie ruchu sieciowego w środowisku SDN.
- Warstwa kontroli danych, która przetwarza przechwycony ruch sieciowy, dokonuje selekcji bądź ekstrakcji cech i przygotowuje dane do analizy.
- Warstwa aplikacji uczenia maszynowego, która wykorzystuje zaawansowane algorytmy NGBoost, XGBoost, LightGBM, Random Forest, Decision Tree do klasyfikacji ruchu sieciowego jako normalny lub atak DDoS. Warstwa ta dodatkowo posiada zaimplementowane metryki oceny skuteczności i wyjaśnialności modelu.

W rozdziale omówiono proces zbierania danych, w którym wykorzystano narzędzie CICFlowMeter do przechwytywania ruchu sieciowego w symulowanej sieci SDN w czasie rzeczywistym podczas przeprowadzania symulowanych hybrydowych ataków DDoS. Zebrane dane posłużyły do stworzenia autorskiego zbioru danych o nazwie DVCDDoS2022.

W rozdziale 5 przedstawiono i omówiono autorski zbiór danych DVCDDoS2022, który zawiera zmapowany ruch sieciowy (normalny i szkodliwy) w sieci SDN. Zbiór ten składa się z 85 cech opisujących atrybuty ruchu sieciowego, który został stworzony specjalnie na potrzeby tej rozprawy doktorskiej, aby uwzględnić specyfikę sieci SDN i umożliwić analizę wyjaśnialności modeli detekcji ataków DDoS. Cechy te obejmują m.in. statystyki dotyczące pakietów, przepływów, czasów i innych charakterystyk ruchu sieciowego. W rozdziale wyjaśniono, dlaczego istniejące i publicznie dostępne zbiory danych (w tym najpopularniejszy w literaturze zbiór CICDDoS2019) nie były odpowiednie do celów tej pracy. Oprócz tego omówiono krótko zastosowane w pracy różne metody selekcji i redukcji cech, takie jak metoda SelectKBest z zastosowaniem testu chi-kwadrat (chi2) i testu Anova, algorytm Recursive Feature Elimination (RFE), Principal Component Analysis (PCA) i narzędzie AutoML o nazwie FeatureWiz, które zostały zastosowane do wyodrębnienia istotnych cech w zbiorze

danych. Zaprezentowano również autorską metodę selekcji cech SelectDVC wraz z pseudokodem, która wykorzystuje ranking ważności cech. Przeprowadzono także analizę korelacji między cechami, aby zidentyfikować ewentualne zależności między nimi. Wyniki analizy korelacji zostały przedstawione w postaci macierzy korelacji obliczonych na podstawie współczynnika korelacji Pearsona.

W rozdziale 6 dokonano kompleksowej oceny algorytmów detekcji ataków DDoS w sieci SDN, skupiając się na algorytmie NGBoost oraz jego porównaniu z innymi metodami klasyfikacji (Decision Tree, XGBoost, LightGBM, Random Forest). Celem było nie tylko znalezienie najskuteczniejszego modelu, ale także zrozumienie, jak różne metody selekcji cech wpływają na wydajność detekcji oraz jak zrównoważyć precyzję modeli z ich wyjaśnialnością. Na potrzeby eksperymentów zaadoptowano metrykę D zaproponowaną przez A. Rosenfelda, która ocenia wyjaśnialność modeli przez analizę ich wydajności np. dokładności(ang. accuracy), precyzji(ang. precision), czułości(ang. recall) lub wartości wskaźnika F1(ang. F1 score). Analiza wyjaśnialności skupiła się na porównaniu modeli "czarnej skrzynki" (NGBoost, LightGBM, XGBoost, Random Forest) z modelem "białej skrzynki" (Decision Tree), aby zrozumieć, czy zwiększona złożoność modeli "czarnej skrzynki" przekłada się na lepszą wydajność detekcji ataków. Oceniono modele również pod kątem ich złożoności obliczeniowej, wykorzystując do tego celu kryteria informacyjne Akaike (AIC) i Bayesowskie (BIC). Kluczowa okazała się modyfikacja metryki D o dodanie tych dwóch kryteriów i analiza wyników tej metryki w celu znalezienia kompromisu między wydajnością a wyjaśnialnością modeli. Wykorzystując metrykę D, określono, kiedy warto wybrać model bardziej złożony, ale potencjalnie mniej zrozumiały, a kiedy prostszy model, który łatwiej zinterpretować.

W rozdziale 7 rozprawa doktorska zagłębia się w wyjaśnialność modeli klasyfikacji ataków DDoS, wykorzystując metodę SHAP (SHapley Additive exPlanations) oraz autorską implementację metryki F zaproponowanej przez A. Rosenfelda. Kluczem było nie tylko zrozumienie, które cechy ruchu sieciowego są najważniejsze dla detekcji ataków, ale także jak te cechy wpływają na decyzje podejmowane przez modele oraz jak zapewnić zwięzłe, ale precyzyjne wyjaśnienia tych decyzji. Metoda SHAP, która oparta jest na teorii gier, została wykorzystana do szczegółowej analizy wpływu poszczególnych cech na predykcje modeli. W wyniku analizy SHAP wykazano, które cechy ruchu sieciowego mają największy wpływ na detekcję ataków DDoS, zarówno pod względem kierunku (czy zwiększają, czy zmniejszają prawdopodobieństwo ataku), jak i siły tego wpływu. Zidentyfikowano kluczowe cechy, które decydują o klasyfikacji ruchu sieciowego jako normalny lub stanowiący atak DDoS. W dalszej części badań, analizowano wpływ stopniowego usuwania cech o niskiej ważności (zgodnie z

wynikami analizy SHAP) na wydajność modeli. Wykorzystano autorski algorytm eliminacji cech, który pozwolił na znalezienie optymalnego zestawu cech, zapewniającego wysoką dokładność detekcji przy jednoczesnym ograniczeniu złożoności modelu. Następnie zaproponowano autorską implementację metryki F, która mierzy kompromis między dokładnością modelu a zwięzłością jego wyjaśnienia. Szczegółowo przeanalizowano wpływ cech pozytywnych (wskazujących na obecność ataku) i negatywnych (wskazujących na brak ataku) na decyzje podejmowane przez modele. Zbadano różnice w liczbie i rodzaju tych cech między różnymi klasyfikatorami, a także ich wpływ na ogólną skuteczność i wiarygodność detekcji ataków DDoS. W konkluzji wskazano cechy ruchu sieciowego, które w największym stopniu wpływają na detekcję ataków DDoS.

W rozdziale 8 zaproponowano badanie wyjaśnialności klasyfikacji ataków DDoS w oparciu o celowe perturbacje danych wejściowych za pomocą autorskich implementacji metryk, takich jak metryka monotoniczności, wierności, niewierności oraz niekompletności. Wyniki pokazały, że zaproponowane modele uczenia maszynowego wykazują wysoką odporność na niewielkie zaburzenia danych oraz mogą być skuteczne i stabilne w detekcji ataków DDoS, nawet w obliczu niewielkich zmian w danych wejściowych. Wysoka monotoniczność, wierność oraz niska niewierność i niekompletność modeli świadczą o ich niezawodności i przewidywalności.

Rozdział 9 skupia się na badaniu kluczowego aspektu bezpieczeństwa modeli uczenia maszynowego w zadaniu detekcji ataków DDoS – ich odporności na ataki zatrucia danych (data poisoning) oraz wpływu celowych manipulacji etykietami w danych treningowych na wydajność i wyjaśnialność modeli klasyfikacyjnych. Zaproponowano autorskie modyfikacje algorytmów ataków Flip-Label, metody label sanitization (służącej do identyfikacji i oczyszczenia zainfekowanych etykiet) oraz metryki S do oceny stabilności modeli wobec zmian etykiet. Zbadano, jak trzy celowe ataki zmiany etykiet (przeciwstawny Flip-Label, atak zmiany etykiety z 0 na1 oraz atak zmiany etykiety z 1 na 0) wpływają na skuteczność działania modeli uczenia maszynowego w detekcji ataków DDoS. Wprowadzono kolejno w różnym stopniu (5%, 10%, 20%, 30%) celowe zmiany etykiet w danych treningowych. Przeprowadzono eksperymenty z modelami NGBoost, XGBoost i LightGBM, analizując zmiany wartości SHAP (ranking ważności cech) po atakach Flip-Label i po oczyszczeniu etykiet. Wyniki pokazały, że metoda SHAP umożliwia śledzenie wpływu manipulacji etykietami. Zbadano także stabilność modeli wobec zmian etykiet, wykorzystując adaptację metryki S zaproponowanej przez A. Rosenfelda, która uwzględniła miary podobieństwa Jaccarda między oczyszczonym a zainfekowanym zbiorem. Eksperymenty wykazały, iż

modele XGBoost i LightGBM są bardziej odporne na ataki Flip-Label niż Decision Tree i Random Forest, podczas gdy NGBoost wykazuje kompromis pomiędzy tymi czterema modelami.

W ostatnim 10 rozdziale podsumowano najważniejsze wyniki i wnioski płynące z przeprowadzonych badań, omówiono ich implikacje dla przyszłych badań i praktycznych zastosowań w detekcji ataków DDoS oraz przedstawiono potencjalne kierunki dalszych badań. Wskazano na skuteczność proponowanych metod detekcji ataków DDoS w sieciach SDN oraz na znaczenie analizy wyjaśnialności modeli dla zrozumienia ich działania i zwiększenia zaufania do nich.

Oryginalne rozwiązania przedstawione w tej rozprawie to :

- Opracowanie autorskiej metody selekcji cech SelectDVC oraz opracowanie autorskiego zbioru danych DVCDDoS2022.
- Stworzenie autorskiego modelu systemu detekcji ataków DDoS w sieci SDN w autorskim środowisku badawczym.
- Przeprowadzenie eksperymentów z hybrydowymi atakami zalewania pakietami (ICMP Flood, TCP-SYN Flood oraz UDP Flood) DDoS w symulowanej sieci SDN.
- Analiza porównawcza algorytmów klasyfikacji zaadaptowanych do zadania detekcji ataków DDoS w sieciach definiowanych programowo oraz ocena wpływu redukcji cech na predykcje modeli.
- Analiza porównawcza wpływu cech sieciowych oraz ich redukcji na predykcje modeli ze szczególnym uwzględnieniem jak stopniowa redukcja liczby cech wpływa na metryki oceny modelu, takie jak dokładność, precyzja, czułość oraz miara F1.
- Opracowanie autorskich implementacji metryk D, F, S zdefiniowanych przez A.Rosenfelda oraz autorskich propozycji ich modyfikacji.
- Opracowanie autorskich implementacji metryk monotoniczności, wierności, niewierności oraz niekompletności.
- Opracowanie autorskich modyfikacji algorytmów ataków Flip-Label oraz oczyszczania zainfekowanych etykiet label-sanitization.
- Implementacja metody SHAP do analizy wpływu ataków Flip-Label oraz procesu sanitization label na modele klasyfikacyjne.

Niniejsza rozprawa doktorska w związku z powyższym prezentuje aktualne spojrzenie na przedstawioną problematykę badawczą, uwzględniając najnowsze trendy w uczeniu maszynowym – wyjaśnialną sztuczną inteligencję, sztuczną inteligencję skoncentrowaną na

danych oraz uczenie maszynowe odporne na ataki – z praktycznymi zastosowaniami i wynikami eksperymentalnymi w detekcji DDoS. Przedstawione eksperymenty w wyżej wymienionych rozdziałach pozwalają na potwierdzenie tezy, iż autorskie propozycje i implementacje kryteriów wyjaśnialnej sztucznej inteligencji umożliwiają skuteczne wyjaśnienie decyzji podejmowanych przez modele uczenia maszynowego w procesie detekcji rozproszonych ataków odmowy usługi (DDoS) w sieciach definiowanych programowo.

2. Wprowadzenie

W niniejszym rozdziale przedstawione zostaną podstawy związane z kluczowymi aspektami współczesnych sieci komputerowych oraz sztucznej inteligencji, które stanowią fundament dla dalszych badań i analiz przeprowadzonych w ramach tej rozprawy doktorskiej.

Rozdział rozpoczyna się od porównania tradycyjnych sieci komputerowych z sieciami definiowanymi programowo, podkreślając różnice w architekturze, funkcjonalności oraz wynikające z nich implikacje dla bezpieczeństwa. Następnie omówione zostaną ataki typu Denial of Service (DoS) oraz Distributed Denial of Service (DDoS) z analizą ich mechanizmów, rodzajów oraz potencjalnych skutków dla infrastruktury sieciowej sieci definiowanych programowo. W dalszej części rozdziału uwaga zostanie skupiona na wyjaśnialnej sztucznej inteligencji podkreślając jej znaczenie w procesie zapewnienia bezpieczeństwa sieci oraz opisując kluczowe metody i techniki wyjaśniania decyzji modeli. Następnie omówiona zostanie koncepcja Data-Centric Artificial Intelligence, z naciskiem na rolę jakości danych w procesie uczenia maszynowego oraz prezentacją technik poprawy jakości danych, które mogą przyczynić się do zwiększenia skuteczności modeli wykrywających ataki DDoS. Rozdział zamyka analiza zagadnień związanych z Adversarial Machine Learning oraz data poisoning, ze szczególnym uwzględnieniem ich wpływu na modele uczenia maszynowego.

2.1 Porównanie tradycyjnych sieci komputerowych z sieciami definiowanymi programowo

Sieci definiowane programowo stanowią stosunkowo nową technologię w porównaniu do tradycyjnych modeli sieci komputerowych, co wiąże się z pewnymi wspólnymi cechami, ale przede wszystkim z istotnymi różnicami. Szczególnie wartościowym obszarem, w którym technologia SDN znalazła szerokie zastosowanie, są centra danych, środowiska chmurowe oraz rozległe sieci szerokopasmowe (WAN).

Podstawowym elementem w sieciach SDN jest protokół OpenFlow, który służy jako standardowy protokół komunikacyjny. OpenFlow umożliwia wykorzystanie kontrolerów programowych, które decydują o przepływie danych w sieci oraz umożliwiają interakcję z siecią za pomocą różnorodnych aplikacji. Dzięki niemu możliwe jest wyraźne oddzielenie płaszczyzny sterowania, czyli oprogramowania, od płaszczyzny danych, odpowiadającej za sprzęt. W praktyce oznacza to, że płaszczyzna sterowania definiuje działanie sieci, podejmując

decyzje o kierowaniu ruchem sieciowym np. wyboru trasy dla przesyłanych pakietów, podczas gdy płaszczyzna danych jest odpowiedzialna za fizyczne przesyłanie pakietów od nadawcy do odbiorcy zgodnie z instrukcjami warstwy sterowania. Elastyczność ta sprzyja implementacji wirtualnych funkcji sieciowych (NFV), które można łatwo implementować, zarządzać i skalować [16]. W kontraście, tradycyjne sieci komputerowe opierają się na konwencjonalnym modelu, w którym to dedykowane urządzenia sprzętowe, takie jak routery czy przełączniki, są odpowiedzialne za kontrolę ruchu i realizację funkcji sieciowych. Główną różnicą między tymi dwoma podejściami jest fakt, że tradycyjne sieci zależą od stałego sprzętu fizycznego, podczas gdy sieci SDN opierają się głównie na oprogramowaniu, co zapewnia znacznie większą elastyczność i adaptacyjność do zmieniających się potrzeb i warunków operacyjnych.

Architektura sieci SDN zbudowana jest z trzech kluczowych komponentów, które współpracują, tworząc zintegrowane środowisko:

- Kontroler sieci SDN to centralny element architektury SDN, który zarządza i koordynuje działanie pozostałych składników sieci, głównie przełączników. Kontroler komunikuje się z przełącznikami SDN za pomocą protokołów takich jak OpenFlow, odbierając od nich informacje o stanie sieci i na ich podstawie podejmuje decyzje dotyczące kierowania ruchem sieciowym. Popularne kontrolery sieci definiowanych programowo takie jak np. Ryu oraz Faucet mogą być programowane za pomocą języka Python.
- Przełączniki SDN są urządzeniami realizującymi instrukcje otrzymane od kontrolera. W odróżnieniu od tradycyjnych, fizycznych przełączników, które działają niezależnie, przełączniki SDN są sterowane centralnie przez kontroler, co zwiększa elastyczność i zwrotność sieci, umożliwiając bardziej złożone zarządzanie ruchem.
- Interfejsy programowania aplikacji (API) umożliwiają komunikację między kontrolerem a aplikacjami zarządzającymi siecią. Dzięki API, programiści mają możliwość tworzenia zaawansowanych aplikacji wykorzystujących potencjał sieci SDN, co obejmuje optymalizację ruchu, efektywne zarządzanie przepływem danych i monitorowanie działania całej sieci.

Migracja do architektury sieci definiowanych programowo z istniejących fizycznych sieci, które działają na zastrzeżonych protokołach i sprzęcie, takim jak rozwiązania firmy Cisco, mogą być procesem złożonym i kosztownym. Wiele firm stoi przed wyzwaniem łączenia nowoczesnych technologii SDN z rozwiązaniami chmurowymi oraz z fizycznymi rozwiązaniami działającymi w ramach lokalnych serwerowni, które często opierają się na dedykowanych rozwiązaniach typu routery, przełączniki, firewalle oraz serwery. Aktualnie

dostępne są liczne rozwiązania open source, które umożliwiają wdrożenie sieci SDN w centrach danych, oferując przy tym elastyczność i niższe koszty początkowe. Jednocześnie, wiodący dostawcy sprzętu sieciowego, tacy jak Cisco [17] i Juniper [18], a także lider technologii wirtualizacji VMware [19] w swoim portfolio posiadają płatne rozwiązania SDN zapewniając dodatkowe funkcje i wsparcie, które mogą być wartościowym atutem dla przedsiębiorstw wymagających stabilnych i zaawansowanych funkcjonalności [3].

W architekturze sieci SDN wyróżnić można 4 modele, które administratorzy sieci mogą wdrożyć, dostosowując je do swoich specyficznych potrzeb. Wyróżnić można:

- Model Open SDN, który wykorzystuje otwarte protokoły, takie jak OpenFlow do zarządzania ruchem na poziomie płaszczyzny danych.
- Model Hybrydowy SDN, który łączy elementy sieci SDN z tradycyjnymi protokołami sieciowymi w jednym środowisku. Jest to często stosowane rozwiązanie podczas migracji z tradycyjnych architektur sieciowych do SDN. Pozwala to na stopniową modernizację sieci, zapewniając jednocześnie kompatybilność i współpracę pomiędzy starymi a nowymi urządzeniami.
- Sieć SDN za pośrednictwem API, w którym kontrola ruchu sieciowego odbywa się przez interfejs programowania aplikacji, a nie standardowe protokoły sieciowe. Pozwala to na tworzenie zaawansowanych aplikacji, które mogą łączyć różne usługi sieciowe i dostosowywać funkcjonowanie sieci do bieżących potrzeb biznesowych.
- Nakładkowa sieć SDN (Overlay SDN), która tworzy wirtualną sieć nałożoną na istniejącą infrastrukturę sprzętową, przydzielając zasoby w różnych segmentach sieci. Pozwala to na utrzymanie fizycznej infrastruktury sieciowej w niezmienionej formie, jednocześnie umożliwiając tworzenie elastycznych i skalowalnych sieci wirtualnych, które można dostosować do zmieniających się wymagań biznesowych. Model ten znacząco wspiera integrację centrów danych, sieci, operatorów bezprzewodowych oraz dostawców usług internetowych (ISP), przy wykorzystaniu technologii chmury obliczeniowej i wirtualizacji, w połączeniu z SDN i NFV [20].

Zastosowanie innowacyjnych strategii takich jak SDN i NFV w centrach danych umożliwia elastyczne wdrażanie, zarządzanie i skalowanie funkcji sieciowych, eliminując jednocześnie potrzebę inwestycji w dodatkowy sprzęt fizyczny. Taka elastyczność pozwala znacząco obniżyć koszty operacyjne. W tradycyjnej sieci komputerowej, implementacja wielu funkcji sieciowych wymagała wdrożenia w serwerowni dedykowanych urządzeń sprzętowych, co generowało znaczne koszty związane z ich zakupem, licencją, pomocą techniczną,

konserwacją i wymianą. Dzięki technologiom SDN i NFV, funkcje sieciowe mogą być realizowane jako oprogramowanie działające na standardowym sprzęcie komputerowym wyposażonym w narzędzia wirtualizacyjne.

W sieciach definiowanych, kontrolery sieci SDN wykorzystują interfejs mostka północnego (ang. Northbound) do komunikacji z interfejsami API. Dzięki temu połączeniu zamiast stosować protokoły wymagane w tradycyjnych sieciach, programiści sieci programowalnych mogą bezpośrednio zarządzać siecią, wykorzystując programy, skrypty i algorytmy [21]. Tradycyjna sieć komputerowa wykorzystuje rozproszone podejście do płaszczyzny sterowania, a także protokoły takie jak OSPF, EIGRP, BGP, MPLS i inne, które funkcjonują oddzielnie dla każdego urządzenia sieciowego, często w oparciu o sprzęt jednego dostawcy sieciowego np. Cisco. Mimo fizycznego połączenia urządzeń sieciowych przez łącza, to routery indywidualnie podejmują decyzje dotyczące trasowania pakietów. Co ważne, tradycyjne fizyczne sieci komputerowe opierają się na ręcznej, statycznej konfiguracji, dość często terminalowej, ręcznym monitoringu i zarządzaniu zasobami sieciowymi.

Kolejną istotną kwestią jest to, że w tradycyjnych sieciach komputerowych wdrażanie nowych usług i urządzeń, a nawet aktualizacja tych urządzeń może wymagać wymiany lub dodania nowego sprzętu sieciowego, co jest czasochłonne i kosztowne np. powodując przerwy konserwacyjne. Oczywiście istnieją również narzędzia umożliwiające automatyzację tradycyjnych sieci komputerowych. W sieciach programowalnych wdrożenie nowych usług może odbywać się za pomocą programowej konfiguracji, co znacznie przyspiesza proces wdrożenia.

Warto zwrócić uwagę na to, iż sieci programowalne pozwalają na bardziej efektywne wykorzystanie zasobów sieciowych niż w przypadku tradycyjnych sieci, gdzie duża część przepustowości łącza czy nadmiarowość urządzeń sieciowych może nie być w pełni wykorzystywana. Ta cecha sieci programowalnych doskonale wpisuje się w ideę green computing, co w głównej mierze opiera się na minimalizacji negatywnego wpływu technologii IT na środowisko poprzez optymalizację zużycia energii i zasobów.

Kontroler SDN wykorzystując interfejs API w kierunku mostka południowego (ang. Southbound) umożliwia programową konfigurację urządzeń sieciowych, co pozwala na dynamiczne modyfikacje, np. takie jak aktualizacja tablic przekierowań pakietów. Jedną z kluczowych zalet tego rozwiązania jest zdolność do kompleksowego gromadzenia danych sieciowych. Kontroler zbiera szeroki zakres informacji, w tym dane o wykorzystaniu sieci, przepustowości, obciążeniu urządzeń oraz inne istotne parametry. Te szczegółowe informacje stanowią fundament dla podejmowania precyzyjnych decyzji dotyczących konfiguracji sieci i

dynamicznego zarządzania ruchem. Efektem takiego podejścia jest znacząca poprawa wydajności całej infrastruktury sieciowej. Co więcej, efektywne zarządzanie zasobami przekłada się na wymierne korzyści finansowe, przyczyniając się do redukcji kosztów operacyjnych. Dzięki temu kontroler SDN staje się nie tylko narzędziem do zarządzania siecią, ale także instrumentem optymalizacji ekonomicznej.

W przypadku wystąpienia awarii w części sieci, kontroler może natychmiastowo wykryć ten problem poprzez monitorowanie danych z różnych punktów w sieci. Następnie może zainicjować automatyczne działania zaradcze takie jak dynamiczne przekierowanie ruchu lub restrukturyzacja topologii sieci, aby ominąć uszkodzone, wyłączone obszary sieci definiowanej programowo. W ten sposób, kontroler jest w stanie reagować na dynamiczne zmiany w sieci, takie jak awarie lub przeciążenia, umożliwiając szybsze przywrócenie normalnego funkcjonowania sieci [21].

Ważny element architektury SDN stanowi warstwa aplikacji, umożliwiająca tworzenie różnorodnych aplikacji do zarządzania siecią. Przykłady obejmują aplikacje monitorujące sieć, które dostarczają informacji o jej stanie i wydajności, aplikacje zarządzające przepustowością, które regulują wykorzystanie pasma w zależności od potrzeb, aplikacje blokujące szkodliwy ruch sieciowy w celu zwiększenia bezpieczeństwa oraz aplikacje do wirtualizacji funkcji sieciowych [21]. W sieciach programowalnych niezbędna jest również możliwość scentralizowanego i programowego wdrażania polityk bezpieczeństwa, ponieważ kontrolery SDN wykorzystują dynamiczne tworzenie i modyfikowanie reguł bezpieczeństwa na podstawie różnych czynników, takich jak rodzaj ruchu, stan aplikacji czy adres źródłowy.

Niewątpliwie technologia sieci definiowanych programowo prezentuje się jako rozwiązanie oferujące wyższą efektywność operacyjną, lepszą kontrolę nad zasobami sieciowymi oraz zwiększony poziom bezpieczeństwa w porównaniu z tradycyjnymi sieciami komputerowymi, umożliwiając szybką reakcję na pojawiające się zagrożenia oraz skuteczne zabezpieczanie sieci przed atakami.

2.2 Ataki DoS oraz DDoS

Ataki odmowy usługi (DoS) oraz rozproszone ataki odmowy usługi (DDoS) stanowią poważne zagrożenie dla firm i organizacji, często powodując straty finansowe oraz osłabienie reputacji. Zwiększenie liczby urządzeń podłączonych do Internetu oraz rozwój narzędzi wykorzystywanych do przeprowadzania ataków spowodowały, że współczesne ataki DDoS stały się coraz bardziej hybrydowe, ale przede wszystkim wyrafinowane i dość skuteczne.

Rozproszony atak typu odmowa usługi to celowe i złośliwe działanie mające na celu zakłócenie normalnego ruchu sieciowego kierowanego do docelowych urządzeń fizycznych, serwerów lub całych sieci. Działanie to polega na przytłoczeniu ich infrastruktury sieciowej dużą ilością ruchu internetowego. Ataki DDoS mogą również pełnić funkcję strategiczną, ponieważ pozwalają atakującym na przeprowadzenie innych rodzajów cyberataków podczas lub po zrealizowaniu ataku DDoS. W 2023 roku [7] zaobserwowano 54% wzrost liczby ataków DDoS użytych jako zasłony dymnej do ukrycia szkodliwych działań wykonywanych w tle tj. kradzież danych, filtracja danych lub penetracja sieci komputerowych. Wiele z ataków w ostatnich latach było wielokrotnie nagłaśniane, prowadząc do irytujących przerw w świadczeniu usług lub nawet uchwalenia praw dotyczących przestępstw internetowych. Oto kilka z najbardziej znanych ataków DDoS.

Amazon Web Service w lutym 2020 roku padł ofiarą ataku DDoS z maksymalną przepustowością szkodliwego ruchu wynoszącą aż 2,3 terabitów na sekundę [22]. Inny duży atak przeprowadzono w lutym 2018 roku na platformę GitHub, gdzie hakerzy podczas 20 minut wykorzystali słabości pamięci podręcznej Memcached, aby ręcznie wysłać 1,3 terabitów na sekundę szkodliwego ruchu i zablokować dostęp do repozytoriów społeczności programistów [23]. W październiku 2016 roku firma Dyn, światowy dostawca usług DNS i rejestrator domen internetowych został zaatakowany przez botnet Mirai. Atak dotknął klientów takich firm jak Netflix, PayPal, Amazon, Visa czy The New York Times [24]. W 2000 roku 15-letni haker o pseudonimie Mafiaboy zorganizował szereg ataków DoS i DDoS, które wyłączyły strony takich firm jak Dell, E-Trade, eBay czy Yahoo [25]. W 2017 roku hakerzy zaatakowali blisko 180 000 serwerów internetowych usługi Google Cloud, co przełożyło się na szkodliwy ruch wynoszący 2,54 terabitów na sekundę. Google Cloud atakowany był w ten sposób, aż przez sześć miesięcy, zaś Google ujawniło informacje o tych incydentach 3 lata później [26]. W 2007 Estonia została zaatakowana atakami DDoS, skierowanymi w instytucje finansowe, rządowe i media. Atak ten uważany jest za pierwszy akt cyberwojny, gdyż stał się odpowiedzią na konflikt polityczny z Rosją [27]. Wiele informacji o nowych, spektakularnych atakach DDoS możemy odnaleźć co jakiś czas w licznych raportach czy informacjach prasowych, przy czym na pewno, nikogo ze specjalistów od cyberbezpieczeństwa nie dziwi skala rosnącej z roku na rok liczby ataków. Tendencja ta będzie się utrzymywać w najbliższych latach w związku z toczącymi się konfliktami na świecie, co podkreśla pilną potrzebę powszechnego przyjęcia solidnych środków bezpieczeństwa przez firmy i państwa z całego świata.

Na przestrzeni kilku ostatnich lat ataki DDoS ewoluowały, wykorzystując zaawansowane technologie do zwiększenia swojej skuteczności. Kluczowym elementem tych

ataków stały się botnety - rozległe sieci zainfekowanych urządzeń, zwanych botami, kontrolowanych zdalnie przez cyberprzestępców. Współczesne botnety nie ograniczają się już tylko do komputerów osobistych. W ich skład wchodzą coraz częściej urządzenia Internetu Rzeczy (IoT), smartfony, tablety oraz inne inteligentne urządzenia. Te zainfekowane urządzenia, działające jak "zombie", są gotowe do wykonywania zdalnych poleceń atakującego w dowolnym momencie. Botnety, zwłaszcza te wykorzystujące chmurę obliczeniową, były zaangażowane w 43% ataków DDoS w 2023 roku [7]. Gdy botnet zostaje skierowany na określony cel (serwer lub sieć), każde z zainfekowanych urządzeń rozpoczyna masowe wysyłanie żądań. Skala tego ataku może być ogromna, biorąc pod uwagę liczbę urządzeń w sieci botnet. Niezwykle trudnym zadaniem staje się oddzielenie szkodliwego ruchu od normalnego, ponieważ każdy bot wydaje się być legalnym urządzeniem działającym w sieci komputerowej czy też w Internecie. To sprawia, że wykrywanie i obrona przed atakami DDoS wymaga zaawansowanych technik analizy ruchu sieciowego oraz stosowania skomplikowanych algorytmów, by efektywnie chronić infrastrukturę sieciową przed zakłóceniami i zagrożeniami wynikającymi z tych ataków.

Popularność ataków DDoS bierze się z ich względnej łatwości realizacji i problemów z ich efektywnym zwalczaniem. Dostępnych jest wiele narzędzi umożliwiających przeprowadzenie takiego ataku, w tym trinoo, Low Orbit Ion Cannon, Trinity, mstream, hping3, czy Tribe Flood Network. Kilka z nich znajduje się w specjalnych dystrybucjach systemów operacyjnych, takich jak Kali Linux, które przeznaczone są do prowadzenia testów penetracyjnych. Łatwy dostęp do tych narzędzi oraz do instrukcji ich obsługi, które są szeroko dostępne w Internecie, sprawia, że teoretycznie każdy użytkownik sieci mógłby przeprowadzić atak DDoS. Niestety częstotliwość ataków DDoS i problem w identyfikacji źródła ataku częściowo wynikający z manipulacji adresami IP dodatkowo komplikuje ich eliminację. Atakujący mogą synchronizować fałszywy ruch lub żądania w celu wyczerpania zasobów systemu lub przeciążenia infrastruktury sieciowej ofiary, blokady połączeń sieciowych i utrudnienia dostępu dla normalnego ruchu. Warto zaznaczyć, że ataki DDoS zazwyczaj nie skutkują utratą poufnych informacji. W 2023 roku cyberprzestępcy często stosowali takie metody jak "Carpet Bombing" i "Bits and Pieces" polegające na rozproszeniu małych, szkodliwych pakietów danych na szeroki zakres adresów IP [7].

Ataki DDoS stanowią szczególne zagrożenie dla firm korzystających z chmury obliczeniowej, gdyż atak na jeden z serwerów może zakłócić całą infrastrukturę chmurową. Prowadzi to do wyłączenia usług dla wielu użytkowników jednocześnie. Zasadne jest by organizacje odpowiednio przygotowywały się na takie zagrożenia poprzez wdrażanie

skutecznych rozwiązań bezpieczeństwa sieciowego i prowadzenie ciągłego monitoringu sieci w celu wykrywania podejrzanych działań. Ponadto, warto zauważyć, że ataki DDoS i DoS nie dotyczą wyłącznie firm i organizacji. Mogą one także negatywnie wpływać na indywidualnych użytkowników Internetu, zagrażając ich codziennemu dostępowi do usług cyfrowych. Do najbardziej oczywistych objawów ataków DDoS należą :

- Nagły wzrost podejrzanego ruchu pochodzącego z jednego adresu IP lub zakresu adresów IP.
- Nietypowy wzrost liczby żądań kierowanych do pojedynczej usługi lub punktu końcowego.
- Dziwne wzorce ruchu, takie jak niezwykłe wzrosty w nieparzystych porach dnia lub nienaturalne wzorce, np. skoki ruchu co 10 minut [28].

Należy jednak pamiętać, że problemy z dostępnością usług mogą wynikać nie tylko z ataków DDoS, ale także z innych przyczyn, takich jak awarie fizycznego sprzętu czy problemy po stronie dostawców usług sieciowych. Dlatego niezbędne jest przeprowadzenie dokładnej i szczegółowej analizy za pomocą narzędzi do monitorowania ruchu, aby ustalić rzeczywiste przyczyny tych zakłóceń.

Ataki DDoS można sklasyfikować na trzy podstawowe typy:

- Atak na warstwy aplikacji, który celuje w konkretne funkcje lub funkcjonalności stron internetowych, dążąc do ich spowolnienia lub unieruchomienia.
- Atak protokołowy, wykorzystujący słabości protokołów sieciowych, w celu przeciążenia docelowego systemu.
- Atak wolumetryczny, który polega na zalewaniu celu ogromną ilością danych, co obciąża dostępną przepustowość i zakłóca normalny ruch sieciowy [28].

Wyróżniamy także specyficzne typy ataków, takie jak:

- TCP SYN Flood, polegający na wysyłaniu wielu żądań TCP SYN do kontrolera SDN, co wyczerpuje jego zasoby i destabilizuje działanie sieci [29] [30].
- ICMP Flood, w którym atakujący wysyła liczne pakiety ICMP do kontrolera, przeciążając go i potencjalnie powodując jego wyłączenie [31].
- UDP Flood, gdzie atakujący generuje ogromne ilości pakietów UDP, które mogą imitować różne aplikacje i znacząco obciążać zasoby kontrolera [32].

Współcześnie rzadziej obserwuje się te ataki samodzielne, gdyż zostały one zastąpione poprzez hybrydowe ataki DDoS. To nic innego jak połączenie różnych metod ataków DDoS

w jedną, hybrydową formę, która jednocześnie zwiększa destrukcyjny potencjał i utrudnia obronę przed nimi.

W architekturze SDN, interfejsy mostków północnego i południowego umożliwiają interakcję między różnymi płaszczyznami, przez co są wyjątkowo narażone na ataki zakłócające działanie przełączników lub ich komunikację z kontrolerem SDN, co w efekcie może prowadzić do zablokowania całej sieci. Efektywnymi strategiami obronnymi w takich przypadkach mogą być techniki filtrowania ruchu, które umożliwia selektywne blokowanie lub przepuszczanie pakietów, kształtowanie ruchu, które ogranicza przepustowość dla podejrzanych lub niepożądanych danych, a także przekierowywanie ruchu, które izoluje ataki od normalnego ruchu sieciowego. Rozwój i implementacja rozwiązań bazujących na sztucznej inteligencji oraz uczeniu maszynowym, także pozwala na szybkie i efektywne wykrywanie nietypowych wzorców zachowań. Dzięki temu możliwa jest ochrona przed atakami DDoS i DoS w infrastrukturze sieciowej sieci definiowanych programowo co pokazują liczne publikacje naukowe.

2.3 Wyjaśnialna Sztuczna Inteligencja (XAI)

Rozwój i implementacja technik wyjaśnialnej sztucznej inteligencji (XAI) pozwala użytkownikom zrozumieć, dlaczego algorytmy uczenia maszynowego dochodzą do konkretnych wyników i wniosków. W konsekwencji może to prowadzić do wzmacniania zaufania użytkowników do modeli uczenia maszynowego, choć ich obecna niedoskonałość czasami może podważać to zaufanie [12]. Oczywiście, dzięki włączaniu XAI do procesu tworzenia modeli uczenia maszynowego możemy starać się zrozumieć i interpretować ich działanie. W pewnym sensie problem wytłumaczalności w sztucznej inteligencji wynika z sukcesu samej dziedziny. Początkowe metody uczenia maszynowego opierały się na logicznym i symbolicznym rozumowaniu tworząc tzw. systemy ekspertowe, które potrafiły rozumować, wykonując logiczne wnioskowania na zrozumiałych dla ludzi symbolach. Niestety, te wczesne podejścia do sztucznej inteligencji okazały się nieskuteczne i kosztowne w budowie. Nie sprostały także wymaganiom nowych technik uczenia maszynowego takich jak algorytmy boostingowe, sztuczne sieci neuronowe oraz uczenie głębokie, które choć nowoczesne i znacznie skuteczniejsze, to niestety mniej przejrzyste i trudniejsze do wytłumaczenia [33]. Szczególnie zauważalne jest to w obszarze klasyfikacji problemów medycznych, takich jak choroby serca czy wykrywanie nowotworów, gdzie wyjaśnienia decyzji modelu są niezwykle cenne [34]. W przypadku diagnozowania chorób, błędne lub

niejasne decyzje mogą prowadzić do poważnych konsekwencji dla pacjentów np. do utraty zdrowia albo życia. Wdrożenie dobrze znanych i udokumentowanych w literaturze technik wyjaśnialnej sztucznej inteligencji jak LIME (ang. Local Interpretable Model-agnostic Explanations) [35] czy SHAP (ang. SHapley Additive exPlanations) [36] umożliwia analizę wpływu poszczególnych cech na decyzje modelu oraz pomaga zrozumieć, dlaczego dany przypadek został sklasyfikowany w określony sposób.

Algorytmy uczenia maszynowego można podzielić na modele czarnoskrzynkowe i białoskrzynkowe. Modele czarnoskrzynkowe charakteryzują się tym, że ich wewnętrzne mechanizmy działania i procesy decyzyjne są trudne do zrozumienia dla ludzi. Są one projektowane bezpośrednio na podstawie danych treningowych i funkcjonują na zasadzie generowania prognoz, nie dostarczając pełnych i zrozumiałych wyjaśnień dla swoich decyzji. Na przykład, sieci neuronowe stosowane w głębokim uczeniu są przykładem modeli, które często są nietransparentne. Dodatkowo skuteczność modeli opartych na sztucznej inteligencji może się różnić, jeśli dane produkcyjne odbiegają od danych treningowych np. jeżeli dane wejściowe są zniekształcone lub zmodyfikowane [37]. Istnieje także ogromne ryzyko ataków na zbiory treningowe poprzez szkodliwe modyfikacje etykiet. Modele białoskrzynkowe w przeciwieństwie do czarnoskrzynkowych charakteryzują się wysokim poziomem wyjaśnialności, która jest wbudowana bezpośrednio w model. Przykładami takich modeli są algorytmy oparte na drzewach decyzyjnych, gdzie ścieżki decyzyjne są jawne i można łatwo zrozumieć, na podstawie jakich cech czy reguł został osiągnięty konkretny wynik. Metody XAI dzielą się również na lokalne, które dostarczają wyjaśnień dla poszczególnych przypadków w zbiorze danych, oraz globalne, które objaśniają działanie modelu w odniesieniu do całego zbioru danych [38].

Jedną z trudności jest to, że wyjaśnienia sztucznej inteligencji są wykorzystywane do różnych celów, dla różnych odbiorców, co oznacza także uwzględnienie aspektów etycznych i moralnych w procesie tworzenia i użytkowania tych technologii. Wdrażanie zrozumiałej dla człowieka sztucznej inteligencji wymaga przestrzegania czterech kluczowych zasad określonych przez amerykański Narodowy Instytut Standardów i Technologii (NIST) [39]:

- Wyjaśnienie - Systemy AI powinny dostarczać dowodów lub uzasadniać swoje wyniki.
- Znaczenie - Wyjaśnienia powinny być zrozumiałe dla poszczególnych użytkowników.
- Dokładność wyjaśnień - Wyjaśnienia powinny dokładnie odzwierciedlać procesy, które doprowadziły do wyników sztucznej inteligencji.

- Granice wiedzy - System sztucznej inteligencji powinien działać w zaprojektowanych warunkach lub gdy jego dane wyjściowe osiągną wystarczający poziom pewności.

Oczywistym staje się fakt, iż budowanie systemów sztucznej inteligencji opartych na zaufaniu i przejrzystości jest niezbędne do osiągnięcia pełnego potencjału tych technologii. Wyjaśnienia mogą przyjąć różnorodne formy, począwszy od prostych i zwięzłych, aż po bardziej złożone i szczegółowe. Niektóre rodzaje wyjaśnień będą na przykład nieodpowiednie dla odbiorców nietechnicznych. Ważnym aspektem w budowaniu zaufania do systemów AI jest opracowanie kryteriów oceny wyjaśnialności, które powinny być oparte przede wszystkim na metrykach stosowanych w danym systemie i dostosowane do profilu odbiorcy. Jednakże, wyzwanie to wymaga interdyscyplinarnego podejścia, angażującego ekspertów z różnych dziedzin, takich jak informatyka, uczenie maszynowe, sztuczna inteligencja, psychologia, a nawet biznes i ekonomia. Taka współpraca może być trudna, gdyż badacze często specjalizują się w jednej dziedzinie, a konieczność pracy w interdyscyplinarnym zespole wymaga wysiłku i otwartości na nowe obszary badawcze. W szczególności, warto zwrócić uwagę na problemy związane z korzyściami dla użytkowników, akceptacją społeczną, zgodnością z regulacjami prawnymi, ewolucją systemów oraz korzyściami dla właścicieli technologii. Rozwiązanie tych problemów wymaga holistycznego podejścia, uwzględniającego zarówno aspekty techniczne, jak i społeczne, moralne, etyczne, a także prawne i ekonomiczne.

Do oceny metod wyjaśnialnej sztucznej inteligencji można zastosować zróżnicowane metryki z takich dziedzin jak matematyka, informatyka czy socjologia. Standardowo, w procesie tworzenia modeli, priorytetem jest wysoka dokładność przewidywań, aby spełnić główne założenia modelu. Jednocześnie, kluczowe jest monitorowanie procesu decyzyjnego modelu, co pozwoli użytkownikom i ekspertom zrozumieć, na jakiej podstawie model podejmuje decyzje, co jest niezbędne by zbudować ludzkie zaufanie do narzędzi działających w oparciu o sztuczną inteligencję.

W literaturze światowej możemy znaleźć różnego rodzaju proponowane metryki pozwalające na wprowadzenie wyjaśnialnej sztucznej inteligencji do uczenia maszynowego.

W artykule [40] autorzy skupiają się na ocenie metod wyjaśniania, podkreślając znaczenie związane z doświadczeniem użytkownika. Autorzy proponują cztery wymagania dla oceny metod, takie jak:

- Koncentracja na dokładności - Metoda powinna dążyć do zbliżonej do rzeczywistej dokładności.

- Koncentracja na funkcjonalności - Metoda powinna skupić się na funkcjonalności (cechach) modelu.
- Stabilność dystrybucyjna - Metoda powinna być wierna dystrybucji danych.
- Prowadzenie analizy instancji - Metoda powinna być zdolna do prowadzenia analizy na poziomie pojedynczych przypadków.

Dodatkowo przedstawiono przegląd 100 różnych metod wyjaśniania kontrfaktycznego opisanych w literaturze dotyczących zakresu, w jakim metody te zostały odpowiednio ocenione, zarówno pod względem psychologicznym, jak i obliczeniowym. Autorzy także określili ilościowo występujące niedobory w procesie testowania przez użytkowników.

W artykule [41] Avi Rosenfeld przedstawia metryki określające, w jaki sposób można określić ilościowo wytłumaczalną sztuczną inteligencję. Propozycja metryk obejmuje zaimplementowanie i określenie:

- Różnic w wydajności – będącą oceną różnicy w wynikach między modelami transparentnymi i nietransparentnymi,
- Liczby reguł - będącą liczbą reguł w wyjaśnieniach modelu (dotyczy modeli opartych na regułach).
- Liczby cech – będącą liczbą cech użytych w wyjaśnieniach modelu(dotyczy modeli opartych na cechach).
- Stabilności – będącą miarą stabilności wyjaśnień modelu.

W innym artykule [42] i dołączonym benchmarku badacze proponują użycie poniższych metryk:

- Monotoniczność – będącą odniesieniem do zdolności modelu do przewidywalnej zmiany predykcji w odpowiedzi na wzrost lub spadek ważności cech modelu (wagi).
- Niewierność - określającą różnicę między przewidywaniem oryginalnej próbki a przewidywaniem próbki zaburzonej.
- Wierność - określającą, jak dobrze wyjaśnienia odzwierciedlają faktyczne zachowanie modelu.
- Niekompletność - określającą ilościowo, jak wrażliwe są predykcje modelu na małe perturbacje cech wejściowych.

Dość ciekawym zagadnieniem wpisującym się w wyjaśnialną sztuczną inteligencję jest innowacyjne podejście do zarządzania systemami sztucznej inteligencji o nazwie AI TRiSM pochodzące od angielskiego terminu "AI Trust, Risk, and Security Management". Jej głównym

celem jest zapewnienie, iż mechanizmy sztucznej inteligencji implementowane w oprogramowaniu komercyjnym i biznesowym są nie tylko wydajne, ale przede wszystkim godne zaufania i bezpieczne w użytkowaniu. Według raportów rynkowych od Gartnera, Emergen Research i Allied Market Research globalny rynek rozwiązań AI TRiSM ma osiągnąć wartość 7,74 miliarda dolarów do końca 2032 roku [43]. W ramach AI TRiSM rozwijane są produkty i usługi, w tym narzędzia do audytu i monitorowania działania sztucznej inteligencji, a także ramy zarządcze obejmujące aspekty takie jak transparentność, zarządzanie danymi oraz wymogi bezpieczeństwa [44]. Szczególnie istotnym elementem AI TRiSM jest koncentracja na aspekcie ludzkim. Framework ten kładzie nacisk na to, by systemy AI były nie tylko technicznie sprawne, ale również etyczne i sprawiedliwe w podejmowaniu decyzji. Wymaga to ścisłej współpracy między programistami a ekspertami ds. etyki, wykorzystując specjalistyczne narzędzia do oceny i poprawy uczciwości systemów AI. Do znanych narzędzi w tej dziedzinie należą między innymi TRiSM framework od Gartnera, NIST AI Risk Management Framework, Microsoft Responsible AI Framework, Google AI Principles, oraz World Economic Forum Principles of Responsible AI.

Rosnące zainteresowanie sztuczną inteligencją, szczególnie widoczne jest po upowszechnieniu narzędzi generatywnych takich jak ChatGPT, Gemini, Perplexity, Midjourne. Akcentuje to potrzebę skupienia się na kwestiach związanych z ryzykiem i zaufaniem do tych technologii. Dodatkowo, rozwijające się, ale często nienadążające za technologią regulacje prawne i wzrost świadomości na temat potencjalnych zagrożeń związanych z AI (np. wykorzystanie deep fake'ów) podkreślają konieczność stosowania odpowiednich ram zarządzania. W odpowiedzi na te potrzeby, organizacje mogą dodatkowo korzystać z zaawansowanych metod oceny i walidacji algorytmów AI, które pozwalają na głębszą analizę i lepsze zrozumienie decyzji podejmowanych przez systemy sztucznej inteligencji. Przykładowo, można wdrożyć regularne przeglądy etyczne by oceniać, jak technologie sztucznej inteligencji wpływają na użytkowników i społeczeństwo oraz rozwijać mechanizmy odpowiedzialności, które umożliwiają szybką reakcję na ewentualne problemy, co pozwoli sprostać zarówno współczesnym wymaganiom rynkowym, jak i etycznym.

Wyjaśnialna sztuczna inteligencja może odgrywać istotną rolę, szczególnie w zadaniu detekcji ataków DDoS przy użyciu algorytmów uczenia maszynowego. Techniki te pozwalają na uzyskanie cennych informacji o dokładności i precyzji modelu oraz dostarczenie wyjaśnień dotyczących podejmowanych decyzji. Pozyskane informacje mogą stać się pomocne nie tylko dla specjalistów zajmujących się bezpieczeństwem sieciowym, ale również dla użytkowników końcowych, którzy dzięki temu mogą zrozumieć, dlaczego określone działania zostały

zakwalifikowane jako zagrożenie i następnie zablokowane. W konsekwencji modele uczenia maszynowego stosowane do detekcji DDoS powinny być transparentne zarówno dla poprawy ich skuteczności, jak i dla budowania zaufania wśród użytkowników [45].

2.4 Data-Centric Artificial Intelligence

W dziedzinie uczenia maszynowego istnieją dwa główne podejścia do tworzenia modeli: skoncentrowane na modelu (Model-Centric) i skoncentrowane na danych (Data-Centric AI). Pierwsze podejście kładzie nacisk na rozwój i optymalizację samego algorytmu, podczas gdy drugie, promowane przez Andrew Ng z Uniwersytetu Stanforda, podkreśla znaczenie jakości zbiorów danych, kosztem ilości rekordów, wprowadzając cykl ciągłej poprawy jakości danych równolegle do cyklu rozwoju modelu. Sukces algorytmów uczenia maszynowego w dużej mierze zależy od jakości danych wykorzystywanych do ich trenowania, ponieważ umożliwiają one modelom wykrywanie najważniejszych wzorców i dokładne przewidywanie. Zamiast jedynie zwiększać rozmiar zbiorów danych, konieczne staje się zatem ich staranne gromadzenie i przetwarzanie, co znacząco usprawnia proces uczenia maszynowego [46] [47]. Co istotne, współczesne modele uczenia maszynowego wymagają dużych ilości danych do uczenia się i tworzenia dokładnych prognoz. Jednak sam proces gromadzenia, przetwarzania i szkolenia może być kosztowny i czasochłonny co obrazuje dobrze przykład organizacji OpenAI, gdzie koszt wytrenowania modelu GPT-4 wyniósł 100 mln dolarów amerykańskich [48].

W podejściu Data-Centric AI, inżynierowie danych i eksperci domenowi współpracują, aby zapewnić, że dane są odpowiednio zebrane, przetworzone i wykorzystane. Ogromny nacisk położony jest na rozumienie jakości danych oraz skupieniem się na poprawie danych [93]. Niezbędne stają się zabiegi typu eliminacja szumów, niespójności i brakujących wartości, co osiąga się przez takie techniki jak czyszczenie danych, normalizacja i transformacja. Wymagany jest także proces ręcznej bądź automatycznej identyfikacji i eliminacji błędnie oznaczonych danych np. za pomocą metod statystycznych, technik przetwarzania języka naturalnego czy widzenia komputerowego[14] [46] [47]. By zwiększyć jakość i ilość danych stosowane są również techniki rozszerzania danych, takie jak syntezowanie nowych danych, perturbacje i interpolacje, które generują dodatkowe przykłady na podstawie istniejących danych.

Co więcej, kwestie etyczne stanowią przedmiot szczególnej uwagi. Konieczne jest zapewnienie, iż dane wykorzystywane do trenowania i zasilania modeli uczenia maszynowego

pozyskane zostały legalnie. W praktyce oznacza to przede wszystkim zapewnienie uczciwości w procesach gromadzenia i przetwarzania danych, by zapobiec stronniczości i dyskryminacji w wynikach działania algorytmów. Zatem, zasadne jest również przyjęcie odpowiedzialności za konsekwencje decyzji podejmowanych na podstawie danych, co obejmuje gotowość do naprawiania błędów i ciągłego doskonalenia systemów sztucznej inteligencji [14] [46].

2.5 Adversarial Machine Learning oraz Data Poisoning

W dziedzinie sztucznej inteligencji wyróżnić można tzw. Adversarial Machine Learning czyli specjalny obszar skupiający się na badaniu wpływu szkodliwych ataków i opracowywaniem technik, strategii mających na celu zabezpieczenie modeli maszynowego uczenia się przed nimi [15]. Systemy detekcji ataków wykorzystujące uczenie maszynowe w programowalnych sieciach mogą być narażone na tak zwane ataki data poisoning, czyli zatrucia danych. Jest to specyficzne zagrożenie dla systemów AI, gdzie atakujący celowo manipulują danymi, na przykład przez dodawanie złośliwych lub mylących rekordów do zestawu danych treningowych, a także modyfikację etykiet. Wiele publikacji naukowych na ten temat pokazuje, że nawet mała ilość zatrutych danych może wpływać negatywnie na model, a sam atak staje się trudnym do wykrycia. Dzieje się tak, ponieważ tego rodzaju ataki są wprowadzane stopniowo, przez co trudno jest ustalić moment, w którym dokładność modelu została zakłócona [13]. Jeśli model oparty został nawet na częściowo zatrutych danych, może być to trudne do rozróżnienia. Podobnie słabe lub nieoczekiwane wyniki mogą efektem celowego działania, ale również innych czynników, takich jak nadmierne dopasowanie, niedopasowanie czy naturalne zakłócenia w danych.

Ataki zatrucia danych są powszechnym i poważnym zagrożeniem dla niezawodności, wiarygodności i bezpieczeństwa systemów sztucznej inteligencji, szczególnie w zastosowaniach krytycznych, takich jak pojazdy autonomiczne, diagnostyka medyczna czy systemy finansowe [13]. Zaobserwowano je w praktyce przeciwko silnikom antywirusowym, filtrom antyspamowym oraz systemom wykrywania fałszywych profili czy fałszywych wiadomości w sieciach społecznościowych. Interesującym przypadkiem ataków na algorytmy uczenia maszynowego były działania wymierzone w filtry antyspamowe Google w latach 2017-2018. System ochrony przed spamem Google uczy się na podstawie zestawu danych wejściowych, które stanowią wiadomości e-mail lub SMS oraz etykiet określających, czy wiadomość jest spamem. Atakujący manipulując etykietami, mogli sprawić, że wiadomości

spamowe były klasyfikowane jako bezpieczne. Taka manipulacja obniżyła skuteczność modelu uczenia maszynowego, co z kolei dla Google i użytkowników oznaczało, iż spam mógł być rozsyłany bez obawy o zablokowanie go przez filtr antyspamowy. Warto zauważyć, że wiele systemów uczenia maszynowego korzysta z publicznych repozytoriów, zestawów danych i innych zasobów dostępnych publicznie w sieci Internet. Niestety takie otwarte podejście do dzielenia się danymi i modelami uczenia maszynowego, choć korzystne dla wspólnoty badawczej, stwarza jednocześnie potencjalne ryzyko dla bezpieczeństwa modeli. Atakujący mogą manipulować danymi dostępnymi w publicznych repozytoriach, co prowadzi do skażenia procesu treningowego i wprowadza błędy czy zakłócenia w samym modelu.

Ataki na dane i modele uczenia maszynowego mogą mieć poważne konsekwencje, szczególnie dla firm, które intensywnie wykorzystują algorytmy do podejmowania ważnych decyzji biznesowych. Na przykład, w sektorze finansowym mogą one prowadzić do błędnych ocen ryzyka kredytowego lub fałszywych transakcji. W medycynie, wyzwania związane z modelami uczenia maszynowego stają się szczególnie widoczne. Gdy algorytmy są narażone na błędne lub niejednoznaczne cechy w danych treningowych, mogą spowodować nieprawidłowe diagnozy lub zastosowanie niewłaściwego leczenia [15]. Przykładem może być błędne uznanie oznaczeń linijki na zdjęciach skóry za potencjalne oznaki czerniaka przez model wykrywania raka skóry. W większości przypadków zmiany złośliwe zawierają pewne charakterystyczne cechy, takie jak nierówności, nierównomierne kolory czy nieprawidłowe kształty. Jednakże, jeśli modele uczą się na danych, gdzie oznaczenia linijkowe są powszechne w przypadkach chorób skórnych, mogą one zaczynać mylnie utożsamiać linie z czynnikiem ryzyka.

W zadaniu wykrywania ataków DDoS, modele uczenia maszynowego są trenowane na danych dotyczących ruchu sieciowego. Ich zadaniem jest rozpoznawanie typowych wzorców ruchu oraz potencjalnych ataków DDoS. Zaimplementowanie w nich ataków typu data poisoning może obejmować modyfikację cech ruchu sieciowego, takich jak rozkład źródeł, docelowych portów czy typów pakietów. Można wykorzystać także ataki typu Flip-label, polegające na zamianie etykiet normalnego ruchu na etykiety ataków DDoS w danych treningowych, by zakłócić efektywność i niezawodność modelu w klasyfikacji prawdziwych zagrożeń [49]. Można również wykorzystać inny sposób manipulacji np. poprzez symulowanie fałszywych ataków DDoS, gdzie atakujący celowo dodaje do danych treningowych nieistniejące ataki. To w konsekwencji może wprowadzić model w błąd, ucząc go identyfikacji ataków, które faktycznie nie występują. Po zakończeniu fazy trenowania modelu bardzo trudno jest poprawić model uczenia maszynowego. Wymagałoby to niestety długiej analizy

wszystkich danych wejściowych, które wytrenowały model, w celu wykrycia fałszywych danych wejściowych i ich usunięcia. Jeśli zbiór danych jest zbyt duży, taka analiza jest po prostu niemożliwa, co w praktyce prowadzi do dezinformacji i tworzenia modelu źle dostosowanego do realnych warunków.

3. Przegląd literaturowy

Postępy w uczeniu maszynowym znacznie poprawiły zdolność do wykrywania ataków DDoS i DoS w sieciach definiowanych programowo, jednakże pełna i niezawodna identyfikacja wszystkich ich form wciąż stanowi wyzwanie. W niniejszym rozdziale, przegląd literaturowy dotyczyć będzie poniższych obszarów:

- Wykrywanie ataków DDoS z wykorzystaniem algorytmów uczenia maszynowego.
- Zastosowanie metod XAI (Explainable Artificial Intelligence) w detekcji ataków DDoS oraz w innych dziedzinach naukowych.
- Wykorzystanie ataków typu data poisoning na algorytmy uczenia maszynowego w kontekście detekcji ataków DDoS i innych dziedzinach naukowych.

W celu zapoznania się z badaniami i artykułami naukowymi z ostatnich lat wykorzystano wiele bibliotek cyfrowych (w tym IEEE, ACM, MDPI, Springer) oraz jedną wyszukiwarkę akademicką (Google Scholar). Przegląd literaturowy wskazuje stan wiedzy na koniec 2023 roku.

3.1 Przykłady detekcji ataków DDoS za pomocą algorytmów uczenia maszynowego

W literaturze przedmiotu powszechnie wykorzystuje się publicznie dostępne zbiory danych, takie jak KDD Cup99, NSL-KDD czy CICDDoS2019 do trenowania algorytmów uczenia maszynowego w zadaniach wykrywania anomalii sieciowych. Analizy porównawcze często uwzględniają także zbiory Kyoto 2006+, UNSW-NB15, CIC-IDS2017, CSE-CIC-IDS2018 czy SCX2012. Należy jednak zauważyć, że dominująca część badań opiera się na analizie danych offline, co może prowadzić do rozbieżności w skuteczności modeli w warunkach rzeczywistych. Istnieje wyraźny brak publikacji z wykorzystaniem autorskich zbiorów danych, które odzwierciedlałyby ruch sieciowy z działających, produkcyjnych środowisk sieci definiowanych programowo.

W ostatnim czasie badania w większości przypadków skupiały się na adaptacji i porównaniu klasycznych algorytmów uczenia maszynowego takich jak drzewa decyzyjne [50], [51], [52], lasy losowe [51], [52], [53] oraz metody boostingowe [51], [53] do specyfiki detekcji ataków DDoS. Przykładowo w pracy [50] drzewa decyzyjne osiągają dokładność 99,99% w

klasyfikacji ataków DDoS. W artykule [52] autorzy porównują sześć różnych algorytmów (SVM, k-NN, DT, NB, RF, LR) na zbiorze CICDDoS2019, wskazując na wysoką skuteczność drzew decyzyjnych (DT) i losowych lasów (RF). Z kolei w innym artykule [10] autorzy wykorzystują zbiór CSE-CIC-IDS2018 i porównują także pięć różnych algorytmów (RF, SVM, k-NN, DT i XGBoost), z których to najlepsze wyniki po ekstrakcji cech osiągnął algorytm RF, który następnie został wdrożony w kontrolerze SDN. W publikacji [132] zaproponowano użycie czterech klasycznych technik uczenia maszynowego (RF, k-NN, LR, NB). Oprócz tego zaimplementowano technikę łagodzenia ataków, by zminimalizować wpływ ataku na sieci SDN. W artykule [53] autorzy zastosowali algorytm CatBoost do detekcji i łagodzenia ataków DDoS w środowisku SDN, osiągając wysoką skuteczność i szybkość detekcji. Z kolei XGBoost został wykorzystany do detekcji ataków DDoS w chmurze opartej na SDN [54].

Równolegle, choć nie co mniej badacze eksplorują zastosowanie sztucznych sieci neuronowych i głębokich sieci neuronowych w procesie wykrywania ataków DDoS. W publikacji [55] zaproponowano system Deep-Discovery będący autorskim systemem wykrywania włamań (IDS) służącym do wykrywania anomalii w sieciach SDN przy użyciu sztucznych sieci neuronowych - Multi-Layer Perceptron i SSN Feedforward (FF). System charakteryzował się dokładnością wynoszącą 98,81% do sześciu klas ataków i minimalnym współczynnikiem fałszywych alarmów (FAR) wynoszącym 0,002. W innym artykule [56] zaprezentowano projekt hybrydowej struktury sieci neuronowej DDosTC, która łączy sieci transformer z splotową siecią neuronową (CNN), aby wykrywać ataki DDoS w środowisku sieci definiowanych programowo. Testy przeprowadzono na zbiorze danych CICDDoS2019, a wyniki porównano z różnymi algorytmami i kombinacjami (Transformer, Transformer+LSTM, Transformer+CNN+LSTM, a także GRU, CNN, B-GRU, RNN, LSTM-GRU oraz LSTM). W artykule [57] zaproponowano hybrydowe połączenie CuDNNLSTM + CuDNNGRU, które osiągnęły dobre wyniki w sieci SDN wdrożonej w środowisku Internetu Rzeczy. Podobnie w publikacji [58] zastosowano połączony model CNN-LSTM w sieci SDN w 5G, co pozwoliło na szybkie blokowanie szkodliwego ruchu. Natomiast w pracy [59] zaproponowano model SSAE-BiLSTM, łączący Stacked Sparse Autoencoder (SSAE) z Bidirectional Long Short-Term Memory (BiLSTM). Model ten charakteryzował się wysoką dokładnością i niskim wskaźnikiem fałszywych alarmów na zbiorze UNSW-NB15. Z kolei w artykule [60] przedstawiono model OptMLP-CNN, który łączy wielowarstwową sieć neuronową z siecią splotową, osiągając przy tym wysoką skuteczność na zbiorach InSDN oraz CICDDoS-2019. Warto również wspomnieć o innych innowacyjnych podejściach. W artykule [61] zastosowano połączenie dwóch algorytmów K-means++ z Random Forest, osiągając 100% we wszystkich

metrykach wydajności (dokładność, precyzja, czułość, miara F1). Zaś w artykule [62] wykorzystano autorski model Decision Tree Detection (DTD), obejmujący algorytm Greedy Feature Selection (GFS) i algorytm Decision Tree Algorithm (DTA), osiągając dokładność 98,42% na zbiorze gureKddcup. Nowym potencjalnym, ale jeszcze szeroko niestosowanym w dziedzinie detekcji ataków DDoS sposobem jest wykorzystanie modeli LLM w tym ostatnio popularnych modeli GPT. W artykule [63] porównano zdolności detekcji GPT-3.5, GPT-4 i Ada z tradycyjnymi sieciami neuronowymi za pomocą dwóch zestawów danych: CICIDS 2017 i bardziej skomplikowanego zestawu danych Urban IoT.

W literaturze naukowej istnieje również wiele prac dotyczących wykrywania ataków DDoS w innych systemach niż SDN. W publikacji [64] autorzy przedstawiają kompleksową analizę porównawczą efektywności różnych algorytmów (DT,RF, LR, XGBoost, AdaBoost SVM, NB i k-NN) w wykrywaniu ataków DoS w nowym zjawisku technologicznym Internetu Pojazdów (IoV). Badania wykazały, iż algorytmy XGBoost, Random Forest i SVM osiągają najwyższą precyzję detekcji. Ponadto w publikacjach [65] [66] omówiono wyniki klasyfikacji ataków DDoS z wykorzystaniem różnorodnych algorytmów uczenia maszynowego w środowiskach Internetu Rzeczy.

3.2 Przykłady wykorzystywania metod XAI w detekcji ataków DDoS oraz w pozostałych wątkach naukowych

Rozwój zaawansowanych algorytmów uczenia maszynowego przyniósł znaczące postępy w detekcji ataków DDoS, choć często kosztem przejrzystości. Obecnie, obserwujemy zmianę paradygmatu, w ramach której badacze coraz częściej poszukują sposobów na połączenie wysokiej efektywności modeli zespołowych z transparentnością. W rezultacie, liczne publikacje prezentują coraz śmielej zastosowania metod wyjaśnialnej sztucznej inteligencji takich jak SHAP w szerokim spektrum zastosowań naukowych w tym również w zadaniu detekcji ataków DDoS.

W artykule [67] przeanalizowano szerokie zastosowanie metod XAI w różnych dziedzinach cyberbezpieczeństwa, takich jak wykrywanie złośliwego oprogramowania, phishingu, botnetów, oszustw, zagrożeń zero-day, cryptojacking itp. Autorzy wskazują, iż metody takie jak SHAP i LIME, są niezbędne dla zwiększenia zaufania do modeli klasyfikujących w tych obszarach. W artykule [68] autorzy wybrali 20 najważniejszych cech,

a następnie wykorzystali wielowarstwową sieć neuronową, która osiągnęła dokładność powyżej 99%. Do wyjaśnienia modelu wykorzystano metodę SHAP z globalnymi i lokalnymi wyjaśnieniami. Podobnie w artykule [69] wykorzystano technikę SHAP do zrozumienia istotności cech wpływających na detekcję ataków w systemie wykrywania intruzów w Przemyśle 5.0. System oparty został na sieciach rekurencyjnych BiLSTM i Bi-GRU. Podobnie w publikacji [133] badacze zastosowali model LSTM do klasyfikacji ataków DDoS, a następnie przeanalizowali jego wyjaśnienia za pomocą metod LIME, SHAP, Anchor i LORE. Badacze porównują te metody za pomocą metryk dokładności opisowej (DA) i rzadkości opisowej (DS), a także podkreślają, iż wyjaśniania mogą objaśnić klasyfikację ataków typu DDoS poprzez uchwycenie albo dominującego wkładu cech wejściowych(51 cech wewnętrznych potrzebnych do klasyfikacji ataków) w predykcję klasyfikatora albo zestawu cech o wysokim znaczeniu.

Wyjaśnialna sztuczna inteligencja wykorzystywana jest również w innych wątkach naukowych związanych z cyberbezpieczeństwem. W artykule [70] zaproponowano nowatorskie podejście do wykrywania ataków DDoS w środowiskach Internetu Rzeczy za pomocą autoenkodera z wykorzystaniem SHAP. Proponowana metoda została oceniona na zbiorze danych USB-IDS. Z kolei w innym artykule [66] przedstawiono analizę porównawczą modeli uczenia maszynowego w zadaniu klasyfikacji złośliwych domen DNS z zastosowaniem technik SHAP oraz LIME. W innej publikacji [71] użyto również techniki SHAP oraz LIME do wyjaśnienia decyzji podejmowanych przez splotowe sieci neuronowe w procesie wykrywania spamu w zbiorze zawierającym 6636 obrazów. W zadaniu klasyfikacji plików PDF jako bezpiecznych i zarażonych przez szkodliwe oprogramowanie typu malware autorzy artykułu [72] przedstawili autorski koncept XAI-PDF wykorzystujący kilka algorytmów uczenia maszynowego oraz SHAP. Metody XAI swoje zastosowanie szczególnie znalazły jednak w problemach klasyfikacji medycznej co widać po liczbie publikacji naukowych. Przykładowo artykuł [73] opisuje wykorzystanie metod SHAP oraz LIME w modelu opartym na algorytmie Random Forest do klasyfikacji pacjentów z objawami cukrzycy. W artykule [74] zastosowano algorytmy Random Forest, XGBoost i AdaBoost do przewidywania choroby wieńcowej (AHD) na podstawie danych z elektronicznych kart zdrowia. W pierwszym przypadku metoda SHAP zidentyfikowała glukozę, natomiast w drugim przypadku poziom hemoglobiny jako najistotniejsze cechy które mają wpływ na predykcję choroby.

3.3 Przykłady wykorzystania ataków zatruwania danych w detekcji ataków DDoS oraz pozostałych wątkach naukowych

Ataki zatruwania danych, które manipulują danymi treningowymi, stwarzają poważne problemy dla systemów uczenia maszynowego. W literaturze naukowej szeroko dyskutowane są konsekwencje tych ataków oraz metody ich przeciwdziałania. Co więcej, dowody z innych dziedzin nauki wskazują, że ataki zatruwania danych są realnym narzędziem do manipulacji modelami uczenia maszynowego, co negatywnie wpływa na ich wydajność i wiarygodność.

W artykule [75] skupiono się na zagrożeniach związanych z atakami zatruwania danych, które mogą zagrażać sieciom SDN obejmując zarówno rodzaje ataków, jak i mechanizmy obronne. Autorzy podkreślają rolę kontrolera w monitorowaniu stanu sieci i wymianie informacji, co jednocześnie stwarza podatność na ataki. W ramach analizy porównawczej przeprowadzonej przez autorów w publikacji [76] zbadano odporność algorytmów NB, RF, SVM, DT, LR oraz modelami uczenia głębokiego AlexNet i LeNet na dwa rodzaje ataków modyfikacji etykiet w zadaniu klasyfikacji spamu na trzech zbiorach danych tj. Spambase, TREC 2006c oraz TREC 2007. Analiza wykazała, że nawet umiarkowany poziom modyfikacji etykiet (20%) może znacząco zwiększyć współczynnik fałszywie negatywnych wyników dla wszystkich badanych modeli. W artykule [77] autorzy zaproponowali analizę ataków Flip-Label w systemach detekcji szkodliwego oprogramowania typu malware w oparciu o algorytmy Stochastic Gradient Descent (SGD), LR, DT, RF, k-NN, SVM, Perceptron oraz MLP. Zaobserwowano spadek wydajności wszystkich modeli podczas procesu zatruwania etykiet już na poziomie 10%. W innym artykule [78] autorzy zaproponowali algorytm służący do przeprowadzania ataków Flip-Label oraz mechanizm oparty na algorytmie k-NN do wykrywania i oczyszczenia podejrzanych etykiet label-sanitization. Badania wykazały, że zaproponowany atak znacząco obniża wydajność klasyfikatorów, a także potwierdziły skuteczność mechanizmu sanityzacji etykiet. Z kolei w innym artykule [79] przeprowadzono eksperymenty z systemami rozpoznawania aktywności ludzkiej (ang. HAR), które oparte są na danych sensorowych pochodzących z urządzeń noszonych(ubieralnych). Zaproponowano autorski algorytm ataku Flip-Label w którym etykieta odczytu sensora jest złośliwie zmieniana podczas fazy zbierania danych w klasyfikacji wieloklasowej. W badaniu wykorzystano klasyfikatory MLP, DT, RF, XGBoost. Zaobserwowano, iż przy 15% zatruwanych danych, dokładność wszystkich modeli spada do 50% lub mniej, co czyni je praktycznie bezużytecznymi. Na koniec przetestowano i oceniono skuteczność mechanizmu label-sanitization opartego na algorytmie k-NN przeciwko proponowanemu atakowi. Na przykład,

dokładność modelu MLP, która spadła do 39% przy 15% zatruwanych danych, wzrasta do 91% po zastosowaniu sanityzacji. W innej publikacji [80] zaproponowano nową metodę ataku Flip-label opartą na aglomeracyjnym grupowaniu hierarchicznym do identyfikacji podatnych próbek w danych treningowych, a następnie dokonania zmiany ich etykiet. W odpowiedzi na ten atak zaproponowano metodę obronną korygującą etykiety opartą na algorytmie TrAdaBoost. Metoda ta używa algorytmu TrAdaBoost do aktualizacji wag zanieczyszczonych danych, a następnie wykorzystuje zaktualizowane wartości wag do oceny i ponownego oznaczenia zanieczyszczonych próbek treningowych. Oczyszczone próbki są wykorzystywane do ponownego wytrenowania klasyfikatora. Zaproponowany algorytm ataku potrafi skutecznie obniżyć dokładność modelu, a zaproponowana metoda obrony lepiej chronić model przed skutkami tego ataku. W artykule [81] wykazano że modele uczenia maszynowego stosowane w systemach wykrywania złośliwego oprogramowania są podatne na ataki typu "clean-label poisoning". W tego typu atakach przeciwnik celowo modyfikuje próbki treningowe, aby model nauczył się backdoora (ukrytego wzorca), który później może być wykorzystany do omijania detekcji. Badacze proponują autorską technikę obrony nazwaną „Nested Training", która wykorzystuje modele zespołowe do usunięcia większości zatruwających próbek z zanieczyszczonego zbioru treningowego. Podobnie w artykule [82] przedstawiano nowatorskie podejście do obrony przed atakami Flip-Label, gdzie zaproponowano algorytm label-sanitization o nazwie AdaSSL (AdaBoost-based Semi-Supervised Learning). Badania przeprowadzono na pięciu rzeczywistych zbiorach danych z repozytorium UCI, wykorzystując sześć klasycznych algorytmów uczenia maszynowego (NB, LR, SVM, DT, k-NN i MLP). W artykule [83] autorzy przedstawiają wyniki badań nowego algorytmu ALFA (Adversarial Label Flips Attack) do atakowania modeli opartych na SVM. Eksperymenty przeprowadzone na danych syntetycznych i rzeczywistych wykazują, że ALFA skutecznie obniżyła wydajność modeli SVM.

3.4 Podsumowanie przeglądu literaturowego

W niniejszym rozdziale dokonano przeglądu literatury dotyczącej wykorzystania algorytmów uczenia maszynowego w detekcji ataków DDoS wskazując na ich duży potencjał Szczególnie obiecujące wyniki badacze przedstawili w środowiskach sieci SDN, ale również w innych technologiach, takich jak IoV czy IoT. Niemniej jednak, większość badań opierała się na analizie danych offline z gotowych zbiorów, co może ograniczać ich przydatność w

warunkach rzeczywistych. Przeanalizowano, także publikacje naukowe dotyczące ataków zatruwania danych. Badania wykazały, że ataki polegające na modyfikacjach etykiet czy wprowadzaniu celowych backdoorów, mogą znacząco obniżyć skuteczność modeli uczenia maszynowego, a nawet uczynić je bezużytecznymi. W odpowiedzi na te zagrożenia, badacze proponowali różne mechanizmy obronne, takie jak sanityzacja etykiet czy wykorzystanie modeli zespołowych, które mają na celu zwiększenie odporności systemów na tego typu ataki. Mimo że metody wyjaśnialnej sztucznej inteligencji, takie jak SHAP i LIME, zyskują na popularności w zwiększaniu zaufania do modeli uczenia maszynowego poprzez identyfikację kluczowych cech wpływających na ich decyzje, ich zastosowanie w detekcji ataków DDoS pozostaje nadal ograniczone. Obecnie większość publikacji naukowych wykorzystujących XAI skupia się na problematyce medycznej, co wskazuje na potencjał tych metod w rozwoju bardziej transparentnych i odpowiedzialnych systemów sztucznej inteligencji w różnych dziedzinach, niekoniecznie związanych z IT. W związku z powyższym zidentyfikowane luki badawcze w zadaniu detekcji ataków DDoS w sieciach SDN mogą zostać wypełnione przez eksperymenty wykonane i omówione w ramach tejże rozprawy doktorskiej.

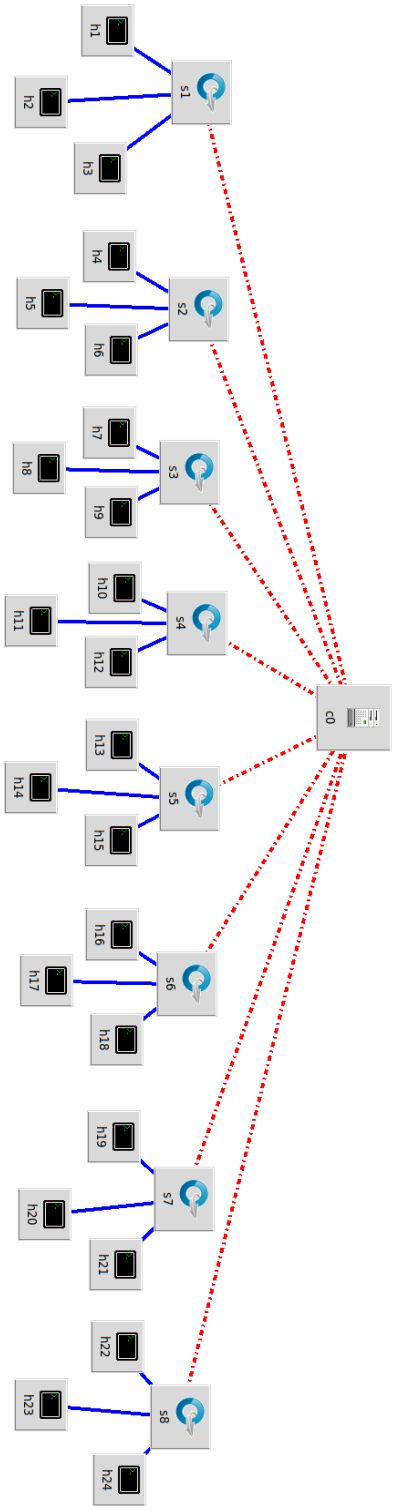
4. Model systemu detekcji ataków DDoS w oparciu o algorytmy uczenia maszynowego

W ramach niniejszego rozdziału przedstawiono projekt systemu detekcji ataków DDoS w sieciach definiowanych programowo, wykorzystujący zaawansowane techniki uczenia maszynowego i selekcji cech. Szczegółowo opisano założenia projektowe, środowisko badawcze, zastosowane metody, techniki oraz narzędzia niezbędne do zbierania i przetwarzania danych.

4.1 Środowisko badawcze i narzędzia

Autorskie środowisko sieci definiowanej programowo dla przeprowadzonych w niniejszej rozprawie doktorskiej eksperymentów zbudowano w oparciu o narzędzie Mininet [84] oraz kontroler SDN Ryu [85], zainstalowanych w systemie operacyjnym Ubuntu Linux 20.04. Kontroler Ryu dzięki wykorzystaniu standardu OpenFlow, oferuje programistom możliwość tworzenia własnych aplikacji do zarządzania siecią SDN za pomocą języka Python i dedykowanego interfejsu API. Może też pełnić funkcje monitora ruchu, firewalla oraz przełącznika. Jego wybór został podyktowany zgodnością z językiem Python, używanym do programowania algorytmów uczenia maszynowego, oraz dostępnością rozbudowanej dokumentacji technicznej. Mininet w którym stworzono symulacyjne środowisko sieciowe, pozwala na konfigurację wirtualnej sieci złożonej z hostów, przełączników, routerów i różnych połączeń za pomocą pojedynczego jądra systemu Linux. Dzięki tej technologii można efektywnie symulować złożone topologie sieciowe, testując przepustowość, łączność oraz szybkość przepływów między węzłami [86]. Do przeprowadzenia skutecznych ataków DDoS wykorzystano oprogramowanie open source - hping3, które oferuje generowanie pakietów o zmiennej długości, testowanie reguł firewalli, skanowanie portów oraz ocenę wydajności sieci opartej na różnych protokołach [87]. Za pomocą specjalnie przygotowanego skryptu przeprowadzono hybrydowe ataki zalewania pakietami, w tym ICMP Flood, TCP-SYN Flood oraz UDP Flood, realizowane losowo z dwóch i więcej hostów. Do symulacji normalnego ruchu sieciowego zastosowano polecenie ping, które zostało zaimplementowane w skrypcie wykorzystującym losowo wybrane hosty do wysyłania standardowych pakietów do losowo wybranych celów.

W ramach eksperymentu zaproponowano drzewiastą topologię sieci komputerowej definiowanej programowo wykorzystującej kontroler Ryu(c0). Sieć zawiera 24 hosty(h1-h24) i 8 przełączników(s1-s8). Na rysunku 4.1 przedstawiono w postaci graficznej topologię wykorzystanej w badaniu sieci SDN.



Rys. 4.1. Autorska topologia sieci SDN w środowisku badawczym.

Zaproponowana topologia sieci została wybrana ze względu na jej reprezentatywność dla rzeczywistych środowisk korporacyjnych oraz możliwość efektywnego testowania różnych scenariuszy ataków DDoS. Struktura zawierająca 24 hosty i 8 przełączników zapewnia wystarczającą złożoność do przeprowadzenia wiarygodnych eksperymentów, jednocześnie pozostając zarządzalną w środowisku symulacyjnym. Dane pozyskiwane w środowisku symulacyjnym w trakcie wykonywania eksperymentów w czasie rzeczywistym zapisywane były za pomocą narzędzia open source o nazwie CICFlowMeter, Jest to aplikacja rozwijana przez Canadian Institute for Cybersecurity przy Uniwersytecie w Nowym Brunszwiku, dedykowana do generowania i analizy przepływów ruchu sieciowego [88]. Pozwala ona na rejestracje, aż 84 różnych cech sieciowych (m.in. czas trwania, liczba pakietów, liczba przesłanych bajtów, długość pakietów, odnotowywane oddzielnie dla kierunku przepływu w przód i w tył) z symulowanej sieci SDN. Wyniki następnie zapisywane były w pliku CSV.

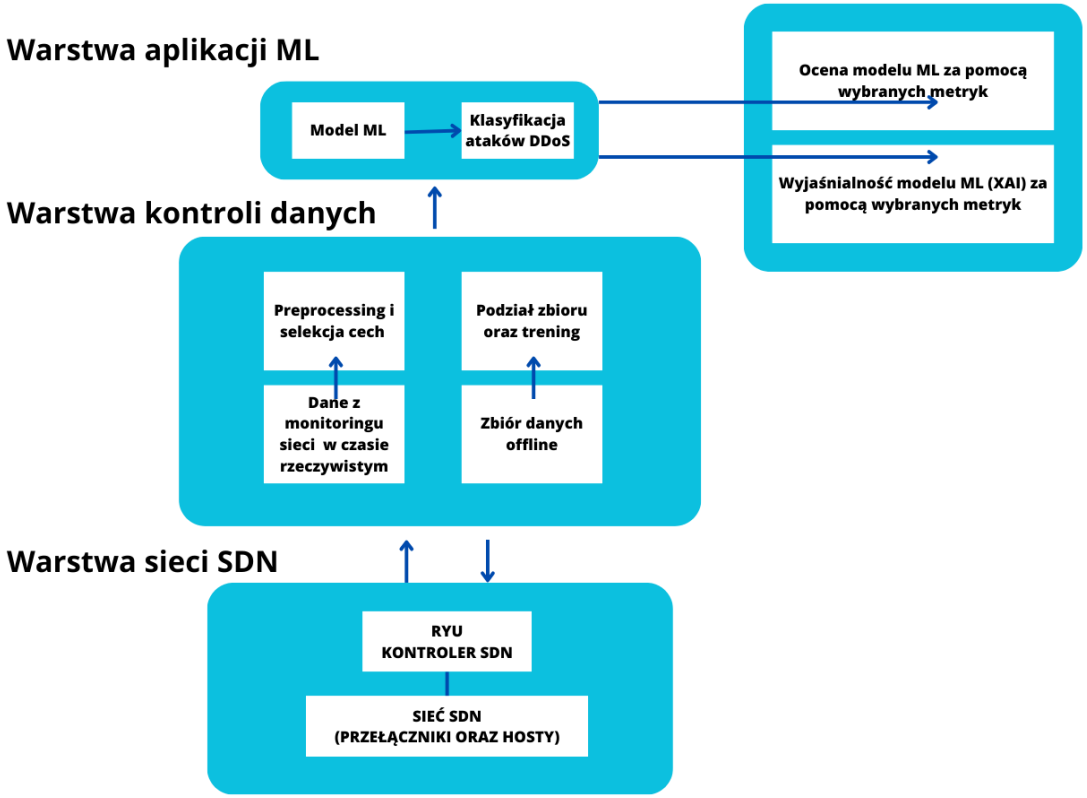
Przepływ ruchu sieciowego definiuje się jako serię pakietów przesyłanych między dwoma urządzeniami w sieci, charakteryzujących się wspólnymi cechami, które są analizowane jako jedna całość. Identyfikuje się je na podstawie różnorodnych parametrów, takich jak adresy IP lub MAC źródłowe i docelowe, numery portów, używany protokół, czy interfejs sieciowy. Przepływy mogą być jednokierunkowe (od źródła do celu) lub dwukierunkowe (wymiana danych odbywa się w obie strony). W przypadku przepływów dwukierunkowych, pierwszy pakiet wyznacza kierunek "w przód" (od źródła do celu), a drugi określa kierunek "wstecz" (od celu do źródła). Warto również zauważyć, że przepływy TCP zwykle są zamykane poprzez wysłanie pakietu FIN kończącego połączenie, podczas gdy przepływy UDP kończą się po osiągnięciu limitu czasowego przypisanego do przepływu, który może być ustawiony przez użytkownika, na przykład na 600 sekund dla obu protokołów, TCP i UDP [89].

4.2 Proponowany system detekcyjny

W niniejszej rozprawie doktorskiej skupiono się na analizie przepływów ruchu sieciowego za pomocą algorytmów uczenia maszynowego w celu wykrycia ataków DDoS. Użyte w badaniu pliki CSV, po przeprowadzeniu odpowiedniego preprocessingu danych posłużyły jako materiał do treningu i testowania modeli. Proces oczyszczania danych był

niezbędny, aby wyeliminować wszelkie niedoskonałości w zbiorze danych, co jest zgodne z ideą Data-Centric AI.

Proponowany system detekcyjny, zaprezentowany na rysunku 4.2, zintegrowany został z eksperymentalnym środowiskiem i obejmuje trzy warstwy: warstwę sieciową SDN, warstwę kontroli danych oraz warstwę aplikacji uczenia maszynowego.



Rys. 4.2. Schemat autorskiego systemu detekcji ataków DDoS.

Najniższa warstwa modelu systemu detekcji ataków DDoS to warstwa sieciowa SDN. Stanowi ona fundament systemu, zawierający infrastrukturę zarządzania siecią. W warstwie tej, kontroler sieciowy SDN Ryu odgrywa centralną rolę w koordynowaniu działania przełączników i hostów. Emulator Mininet używany jest do tworzenia wirtualnych przełączników OpenFlow i hostów, co umożliwia elastyczne modelowanie różnorodnych scenariuszy sieciowych.

Warstwa kontroli danych pełni funkcję monitoringu ruchu sieciowego w czasie rzeczywistym. Dane są nie tylko zbierane i przetwarzane, ale również poddawane

zaawansowanym technikom preprocessingu, takim jak normalizacja i eliminacja szumów, co przygotowuje je do dalszej analizy. Dane z symulowanych ataków DDoS są przekształcane w zbiór treningowy, podczas gdy dane z bieżącego ruchu sieciowego stanowią zbiór testowy. Na potrzeby tej pracy finalnym etapem przygotowania danych był ich podział na zbiory treningowe i testowe z wykorzystaniem techniki 5-krotnej crosswalidacji StratifiedKFold. Zasada działania tej metody polega na podziale zbioru danych na pięć równych części (fold), z których cztery są używane do trenowania modelu, a jedna do jego walidacji. Proces ten powtarza się pięć razy, przy czym w każdej iteracji inny fold jest używany jako zbiór walidacyjny. Ostateczna ocena modelu jest uzyskiwana poprzez uśrednienie wyników uzyskanych w poszczególnych iteracjach [135][136]. Unikamy sytuacji, w której jedna klasa mogłaby dominować nad innymi, co mogłoby prowadzić do przeuczenia i błędnych wniosków dotyczących jakości modelu. Szkodliwy ruch ataku DDoS oraz ruch normalny są identyfikowane i analizowane przez wytrenowany model na podstawie danych zbieranych przez kontroler z przełączników OpenFlow i monitora ruchu CIC-Flowmeter. Implementacja zasad Data-Centric AI na tym poziomie gwarantuje, że dane wykorzystywane do treningu modeli są wolne od anomalii i błędów, które mogłyby negatywnie wpłynąć na skuteczność detekcji.

Warstwa aplikacji ML stanowi najwyższą warstwę systemu detekcji, w której dochodzi do klasyfikacji i wyjaśnialności zachowań sieciowych i potencjalnych ataków DDoS. Modele uczenia maszynowego, wytrenowane na bazie odpowiednio przygotowanego zbioru treningowego, są wykorzystywane do analizy przepływu ruchu sieciowego w celu identyfikacji anomaliów. Skuteczność tych modeli oceniana jest za pomocą takich metryk jak dokładność, precyzja, czułość oraz wskaźnik F1, które dostarczają informacji na temat zdolności modeli do efektywnego rozpoznawania ataków DDoS.

Implementacja metod wyjaśnialności modeli, takich jak SHAP (SHapley Additive exPlanations) oraz autorskich metryk opisanych w dalszych rozdziałach pozwala na wykorzystanie informacji z nich pozyskanych do odpowiedzi na zadane pytania badawcze oraz głębokie zrozumienie, które cechy ruchu sieciowego są kluczowe dla decyzji predykcyjnych modeli. System detekcji został zaprojektowany z myślą o płynnej integracji wszystkich komponentów, zapewniając efektywny przepływ informacji między warstwami. Modułowa architektura systemu umożliwia łatwą rozbudowę i modyfikację poszczególnych komponentów np. warstwa sieciowa SDN może być łatwo rozbudowywana o dodatkowe przełączniki i hosty, warstwa kontroli danych jest przystosowana do obsługi zwiększonego wolumenu danych, a warstwa aplikacji ML może być wzbogacana o nowe modele i metody wyjaśnialności. Komunikacja między warstwami odbywa się w czasie rzeczywistym, co jest niezbędne dla

szybkiej detekcji potencjalnych zagrożeń. Zastosowanie jednolitego środowiska programistycznego Python w zaproponowanym systemie, zarówno dla kontrolera jak i algorytmów uczenia maszynowego znacząco ułatwiło integrację poszczególnych komponentów systemu.

5. Analiza zbioru danych i użytych algorytmów selekcji cech

W ramach rozprawy doktorskiej przeprowadzono symulacje w środowisku badawczym Mininet, tworząc autorski zbiór danych (zebrany w 2022 roku) nazwany na potrzeby rozprawy doktorskiej DVCDDoS2022. Zawarte w nim są przepływy generowane wyłącznie w sieci definiowanej programowo, typowe dla hybrydowych ataków DDoS oraz normalnego ruchu sieciowego. W procesie etykietowania zastosowano jednolite oznaczenie rekordów: wartość 1 w kolumnie ”label” oznacza szkodliwy ruch reprezentujący atak DDoS, a wartość 0 – normalny ruch sieciowy. Całkowita liczba rekordów w zbiorze wyniosła 107220, z czego 51,99% stanowią przypadki ataków DDoS, a 48,01% to normalny ruch sieciowy. Zrównoważony rozkład klas został osiągnięty w sposób naturalny, bez stosowania sztucznych technik balansowania. Wykorzystano również narzędzie CleanLab do identyfikacji i korekty danych w zbiorze DVCDDoS2022, które zostały błędnie oznaczone lub zawierają nieścisłości, co wpisuje się w koncepcję Data Centric AI i Confident Learning (umożliwia identyfikację i korekcję błędnych etykiet)[134]. Oznacza to, że proces oczyszczania danych odbywał się w sposób inteligentny, na podstawie analizy modelu z wykorzystaniem algorytmu XGBoost, co pozwoliło na automatyczne wykrywanie problematycznych etykiet. W przypadku wykrycia przez CleanLab błędnej etykiety, etykieta ta zmieniana była na przeciwstawną. Dzięki temu zabiegowi, przyszłe modele detekcji nie tylko mogą zyskać na stabilności, ale także stać się mniej podatne na błędy wynikające z szumów w danych. Co więcej, autorski zbiór danych pod względem liczby cech będących z mapowanymi atrybutami przepływów sieciowych nawiązuje do popularnego w literaturze naukowej zbioru ataków DDoS znanego jako CICDDoS2019, stworzonego przez Instytut Cyberbezpieczeństwa Uniwersytetu w Nowym Brunszwiku [90]. Użycie w niniejszej rozprawie doktorskiej w eksperymentach zbioru CICDDoS2019, mimo jego popularności i częstego wykorzystania w literaturze naukowej, nie byłoby odpowiednie z kilku kluczowych powodów. Po pierwsze, główna różnica polega na tym, że dane umieszczone w zbiorze CICDDoS2019 nie są z mapowane z sieci SDN. Uwidocznia się tutaj różnica w architekturze, co wpływa na zachowanie i charakterystykę przepływów sieciowych, a to z kolei mogłoby prowadzić do błędnych wniosków dotyczących ataków i obrony w architekturze SDN. Po drugie w zbiorze CICDDoS2019 zauważalny jest problem nierównomierności klas (ok. 98% stanowią ataki DDoS), co może prowadzić do nadmiernego dopasowania modelu. Techniki balansowania klas takie jak oversampling, undersampling czy SMOTE (Synthetic Minority Over-sampling Technique) choć pomocne, mogłyby wprowadzić sztuczność do danych, negatywnie wpływając na model [91]. Zgodnie z ideą Data Centric AI ogromną rolę w

tworzeniu wiarygodnych modeli odgrywa jakość i zrównoważenie danych, co wpływa na przeprowadzenie bardziej miarodajnej i rzetelnej analizy. Po trzecie powszechne wykorzystanie zbiorów danych, takich jak CICDDoS2019, CIC-IDS2017 i CSE-CIC-IDS2018, niesie ryzyko ataków typu data poisoning. Eksperymentalne manipulacje tymi zbiorami mogą ujawnić luki w systemach detekcji DDoS, co w konsekwencji obniży ich skuteczność w rzeczywistych warunkach. W związku z tym nowy, autorski zbiór danych dostosowany do specyfiki sieci SDN i celów badawczych rozprawy doktorskiej minimalizuje te zagrożenia oraz zapewni wiarygodność analiz i precyzyjną weryfikację wyników dla przyszłego wdrożenia systemu detekcji w środowisku produkcyjnym.

5.1 Autorski zbiór danych i wyodrębnione cechy

Autorski zbiór danych DVCDDoS2022 stworzony na potrzeby tej rozprawy doktorskiej zawiera 85 cech opisujących atrybuty ruchu sieciowego oraz etykietę. Cechy te obejmują zarówno podstawowe informacje o przepływach (czas trwania, liczba i rozmiar pakietów), jak i dane dotyczące adresów IP oraz protokołów sieciowych. Każdy przepływ został oznaczony etykietą wskazującą, czy jest to ruch normalny, czy szkodliwy ruch DDoS. Szczegółowy opis każdej z cech przedstawiono w poniższej tabeli na podstawie dokumentacji narzędzia CICFlowMeter [88][138][139].

Tabela 5.1.: Lista cech w zbiorze wraz z opisem.

Lp.	Nazwa cechy: Opis
1.	Flow Duration: Czas trwania przepływu danych.
2.	Total Fwd Packet: Całkowita liczba pakietów w przód.
3.	Total Bwd Packets: Całkowita liczba pakietów wstecz.
4.	Total Length of Fwd Packet: Całkowita wielkość pakietów w przód.
5.	Fwd Packet Length Max: Maksymalna wielkość pakietów w przód.
6.	Fwd Packet Length Min: Minimalna wielkość pakietów w przód.
7.	Fwd Packet Length Mean: Średnia wielkość pakietów w przód.
8.	Fwd Packet Length Std: Odchylenie standardowe wielkości pakietów w przód.
9.	Bwd Packet Length Max: Maksymalna wielkość pakietów wstecz.
10.	Bwd Packet Length Min: Minimalna wielkość pakietów wstecz.
11.	Bwd Packet Length Mean: Średnia wielkość pakietów wstecz.

12. Bwd Packet Length Std: Odchylenie standardowe wielkości pakietów wstecz.
13. Flow Bytes/s: Szybkość przepływu danych w bajtach na sekundę.
14. Flow Packets/s: Szybkość przepływu pakietów w pakietach na sekundę.
15. Flow IAT Mean: Średni czas między dwoma przepływami.
16. Flow IAT Std: Odchylenie standardowe czasu między dwoma przepływami.
17. Flow IAT Max: Maksymalny czas między dwoma przepływami.
18. Flow IAT Min: Minimalny czas między dwoma przepływami.
19. Fwd IAT Total: Całkowity czas między dwoma pakietami wysłanymi w przód.
20. Fwd IAT Mean: Średni czas między dwoma pakietami wysłanymi w przód.
21. Fwd IAT Std: Odchylenie standardowe czasu między dwoma pakietami wysłanymi w w przód.
22. Fwd IAT Max: Maksymalny czas między dwoma pakietami wysłanymi w przód.
23. Fwd IAT Min: Minimalny czas między dwoma pakietami wysłanymi w przód.
24. Bwd IAT Total: Całkowity czas między dwoma pakietami wysłanymi wstecz.
25. Bwd IAT Mean: Średni czas między dwoma pakietami wysłanymi wstecz.
26. Bwd IAT Std: Odchylenie standardowe czasu między dwoma pakietami wysłanymi wstecz.
27. Bwd IAT Max: Maksymalny czas między dwoma pakietami wysłanymi wstecz.
28. Bwd IAT Min: Minimalny czas między dwoma pakietami wysłanymi wstecz.
29. Fwd PSH Flags: Liczba wystąpień, gdy flaga PSH była ustawiona w pakietach przesyłanych w przód (0 dla UDP).
30. Bwd PSH Flags: Liczba wystąpień, gdy flaga PSH była ustawiona w pakietach przesyłanych wstecz (0 dla UDP).
31. Fwd URG Flags: Liczba wystąpień, gdy flaga URG była ustawiona w pakietach przesyłanych w przód (0 dla UDP).
32. Bwd URG Flags: Liczba wystąpień, gdy flaga URG była ustawiona w pakietach przesyłanych wstecz (0 dla UDP).
33. Fwd Header Length: Całkowita liczba bajtów używanych na nagłówki w przód.
34. Bwd Header Length: Całkowita liczba bajtów używanych na nagłówki wstecz.
35. Fwd Packets/s: Liczba pakietów przesyłanych na sekundę w przód.
36. Bwd Packets/s: Liczba pakietów przesyłanych na sekundę wstecz.
37. Packet Length Min: Minimalna długość przepływu pakietów.
38. Packet Length Max: Maksymalna długość przepływu pakietów.

39. Packet Length Mean: Średnia długość przepływu pakietów.
40. Packet Length Std: Odchylenie standardowe długości przepływu pakietów.
41. Packet Length Variance: Wariancja długości przepływu pakietów.
42. FIN Flag Count: Liczba pakietów z flagą FIN.
43. SYN Flag Count: Liczba pakietów z flagą SYN.
44. RST Flag Count: Liczba pakietów z flagą RST.
45. PSH Flag Count: Liczba pakietów z flagą PSH.
46. ACK Flag Count: Liczba pakietów z flagą ACK.
47. URG Flag Count: Liczba pakietów z flagą URG.
48. CWR Flag Count: Liczba pakietów z flagą CWR.
49. ECE Flag Count: Liczba pakietów z flagą ECE.
50. Down/Up Ratio: Współczynnik pobierania i wysyłania.
51. Average Packet Size: Średni rozmiar pakietu.
52. Fwd Segment Size Avg: Średni rozmiar segmentów obserwowany w przód.
53. Bwd Segment Size Avg: Średni rozmiar segmentów obserwowany wstecz.
54. Fwd Bytes/Bulk Avg: Średnia liczba bajtów w trybie pakietowym w przód.
55. Fwd Packet/Bulk Avg: Średnia liczba pakietów w trybie pakietowym w przód.
56. Fwd Bulk Rate Avg: Średnia liczba trybu pakietowego w przód.
57. Bwd Bytes/Bulk Avg: Średnia liczba bajtów w trybie pakietowym wstecz.
58. Bwd Packet/Bulk Avg: Średnia liczba pakietów w trybie pakietowym wstecz.
59. Bwd Bulk Rate Avg: Średnia liczba trybu pakietowego wstecz.
60. Subflow Fwd Packets: Średnia liczba pakietów w podprzepływie w przód.
61. Subflow Fwd Bytes: Średnia liczba bajtów w podprzepływie w przód.
62. Subflow Bwd Packets: Średnia liczba pakietów w podprzepływie wstecz.
63. Subflow Bwd Bytes: Średnia liczba bajtów w podprzepływie wstecz.
64. FWD Init Win Bytes: Liczba bajtów wysłanych w oknie inicjacyjnym w przód.
65. Bwd Init Win Bytes: Liczba bajtów wysłanych w oknie inicjacyjnym wstecz.
66. Fwd Act Data Pkts: Liczba pakietów z co najmniej 1 bajtem danych TCP w przód.
67. Fwd Seg Size Min: Minimalny rozmiar segmentu obserwowany w przód.
68. Active Mean: Średni czas aktywności przepływu przed przejściem w stan bezczynności.
69. Active Std: Odchylenie standardowe czasu aktywności przepływu przed przejściem w stan bezczynności.

70. Active Max: Maksymalny czas aktywności przepływu przed przejściem w stan bezczynności.
71. Active Min: Minimalny czas aktywności przepływu przed przejściem w stan bezczynności.
72. Idle Mean: Średni czas bezczynności przepływu przed powrotem do stanu aktywności.
73. Idle Std: Odchylenie standardowe czasu bezczynności przepływu przed powrotem do stanu aktywności.
74. Idle Max: Maksymalny czas bezczynności przepływu przed powrotem do stanu aktywności.
75. Idle Min: Minimalny czas bezczynności przepływu przed powrotem do stanu aktywności.
76. Fwd Act Data Pkts: Liczba pakietów z co najmniej 1 bajtem danych TCP w przód.
77. Label: etykieta przypisana do rekordu danych w celu oznaczenia go jako część określonej grupy.
78. Src_ip: Źródłowy adres IP komputera lub urządzenia, które jest źródłem przesyłanych danych.
79. Dst_ip: Docelowy adres IP komputera lub urządzenia, które jest celem przesyłanych danych.
80. Src_port: Numer portu używany przez źródłowe urządzenie do komunikacji.
81. Dst_port: Numer portu używany przez docelowe urządzenie do komunikacji.
82. Src_mac: Źródłowy adres MAC źródłowego urządzenia używany na niższym poziomie warstwy łącza danych w sieci.
83. Dst_mac: Docelowy adres MAC docelowego urządzenia używany na niższym poziomie warstwy łącza danych w sieci.
84. Protocol: Protokół komunikacyjny używany w przesyłanych danych, na przykład TCP, UDP, HTTP, itp.
85. Timestamp: Data i godzina, w której dane zostały zarejestrowane lub przesłane, używane do analizy czasu przepływu danych.

W procesie przygotowania danych do analizy i modelowania w ramach tej rozprawy doktorskiej zastosowano szereg technik preprocessingu, mających na celu optymalizację i ujednolicenie danych. W pierwszym kroku, wszelkie wartości nieskończone, ujemne oraz NaN (Not a Number) zastąpiono zerami, co miało na celu eliminację potencjalnych błędów, które

mogłyby zakłócić proces modelowania. Następnie, w celu zapewnienia spójności i optymalizacji procesu uczenia maszynowego, wszystkie wartości liczbowe zostały przekształcone do formatu float64. Konieczny również był proces standaryzacji danych, przeprowadzony za pomocą metody StandardScaler, ponieważ wiele algorytmów uczenia maszynowego jest wrażliwych na różnice w skalach cech, a dzięki niej dane stają się porównywalne i możliwe do efektywnego przetwarzania [92]. Pozwoliło to na przeskalowanie cech do rozkładu o średniej 0 i odchyleniu standardowym 1. Podjęto, także decyzję o usunięciu ośmiu cech takich jak :

- Adresy IP nadawcy i odbiorcy (src_ip, dst_ip), ponieważ mogą być unikalne dla każdego przypadku oraz podatne na fałszowanie i manipulacje przez co nie dostarczają potrzebnych informacji analitycznych ruchu sieciowego.
- Porty źródłowe i docelowe (src_port, dst_port) pomimo, że są unikalne dla poszczególnych sesji to jednak nie dostarczają istotnych informacji potrzebnych do analizy ogólnych wzorców ruchu DDoS.
- Adresy MAC nadawcy i odbiorcy (src_mac, dst_mac) mogą być także podatne na fałszowanie i manipulację i również nie dostarczają kluczowych informacji analitycznych ruchu sieciowego.
- Protokół komunikacyjny (protocol) zwykle ogranicza się do informacji o ruchu TCP bądź UDP co w przypadku analizy wzorców hybrydowych ataków DDoS nie jest konieczne.
- Znacznik czasowy (timestamp) nie zawiera istotnych danych, gdyż analiza nie skupia się na specyficznych ramach czasowych.

5.2 Zastosowane metody selekcji cech

W celu optymalizacji procesu klasyfikacji binarnej kluczowe jest wyodrębnienie najbardziej istotnych cech charakterystycznych dla danego zbioru próbek. W podejściu Data-Centric AI, iteracyjne doskonalenie zbioru danych jest priorytetem, ponieważ nawet zaawansowane algorytmy wymagają dobrze przygotowanych danych. Praktyczne zastosowanie idei Data-Centric AI odnaleźć możemy w inżynierii cech, będącej procesem w którym przekształca się surowe dane na odpowiednią reprezentację lub określony wektor cech, przez co klasyfikator jest w stanie znaleźć i wyodrębnić wzorce, które posłużą do późniejszej klasyfikacji [93] [95]. Oczywiście, użycie całego zbioru cech może znacząco obniżyć ich efektywność ze względu na tzw. "przekleństwo wymiarowości" czyli sytuacji, w której nadmiar cech w stosunku do liczby próbek utrudnia skuteczną klasyfikację [94].

Selekcja cech różni się od ekstrakcji cech, która polega na tworzeniu nowych atrybutów przez kombinację istniejących, natomiast selekcja cech skupia się na wyborze już istniejących atrybutów. W literaturze wyróżnia się trzy główne kategorie algorytmów selekcji cech: metody rankingowe (filtry), wrappery i metody wbudowane.

- Metody rankingowe (filtry) oceniają każdą cechę indywidualnie na podstawie określonych kryteriów, takich jak współczynnik korelacji Pearsona, współczynnik korelacji Spearmana, miara chi-kwadrat, nie uwzględniając przy tym interakcji między cechami [96] [97].
- Metody typu wrapper są technikami bardziej zaawansowanymi, które analizują zależności między cechami, korzystając z modelu klasyfikacyjnego do oceny jakości poszczególnych podzbiorów cech. Proces ten odbywa się iteracyjnie przez dodawanie lub usuwanie cech w oparciu o wyniki klasyfikacji. Przykładowe techniki to Forward Selection, Backward Selection oraz algorytmy genetyczne [98] [99].
- Metody wbudowane łączą proces selekcji cech z uczeniem klasyfikatora, wykorzystując wewnętrzne mechanizmy wybranych modeli do oceny przydatności cech. Są cenione za ich efektywność i zdolność do integracji selekcji z procesem uczenia. Przykłady obejmują liniowe modele z regularyzacją (L1, L2, L1+L2), sieci neuronowe, algorytmy SVM, RF czy też Recursive Feature Elimination (RFE) [100] [101].

Na podstawie przeglądu literatury do niniejszych badań wybrano kilka metod selekcji cech, które są zgodne z podejściem Data-Centric AI. Są to:

- **Metoda SelectKBest** jest popularnym podejściem opierającym się na analizie statystycznej, umożliwiającym wyselekcjonowanie najlepszych cech na podstawie określonego kryterium. W badaniu zastosowano dwa warianty tej metody: pierwszy wykorzystuje **test chi-kwadrat (chi2)** do oceny zależności między cechami a zmienną docelową, drugi zaś – **test Anova (f_classif),** który analizuje znaczenie cech w związku z ich różnicami między klasami [102] [103].
- **Recursive Feature Elimination (RFE)** opiera się na iteracyjnym procesie, który zaczyna się od kompletu cech, następnie ocenia się wagę każdej z nich za pomocą współczynników zwracanych przez model i sukcesywnie eliminuje te o najniższej wadze, powtarzając proces, aż do osiągnięcia pożądanego rozmiaru podzbioru [101] [104]. RFE wymaga zastosowania klasyfikatora do oceny istotności cech – w tym przypadku wybrano model XGBoost.

W trakcie przeglądu literatury naukowej zwrócono uwagę na metodę **Principal Component Analysis (PCA)**, która, choć jest techniką redukcji wymiarowości, a nie selekcji

cech w ścisłym znaczeniu, może być stosowana do obu tych celów. W związku z tym, PCA często występuje w badaniach porównawczych obok klasycznych metod selekcji cech, w kontekście różnych eksperymentów klasyfikacyjnych [94]. Poprzez przekształcenie oryginalnego zestawu cech w nowy zestaw nieskorelowanych głównych składowych, PCA umożliwia zachowanie maksymalnej ilości informacji zawartej w danych przy jednoczesnym zmniejszeniu ich wymiarowości. W związku z tym podjęto również decyzję o wykorzystaniu w niniejszej rozprawie doktorskiej metody PCA [105] [106].

Jako ostatnią technikę pozwalającą na wybór optymalnego podzbioru cech wybrano zautomatyzowane narzędzie AutoML o nazwie **FeatureWiz** wpisujące się we współczesny trend stosowania zautomatyzowanych bibliotek, które ułatwiają analizę danych, w tym wstępną obróbkę, trening modeli i testowanie ich wydajności. FeatureWiz wykorzystuje metodę selekcji cech znaną jako SULOV (Searching for Uncorrelated List of Variables). Metoda ta polega na identyfikacji i eliminacji par zmiennych o wysokiej korelacji. Dzięki wykorzystaniu techniki Mutual Information Score (MIS), która ocenia ilość informacji przekazywanych przez jedną zmienną o drugiej, SULOV wybiera pary o najniższej korelacji i najwyższym MIS. Wyselekcjonowane w ten sposób zmienne są następnie analizowane przez algorytm XGBoost, który wybiera najbardziej informatywne cechy względem zmiennej docelowej, tworząc uproszczony zestaw danych z pełnego zbioru. Ta metoda jest porównywalna z techniką minimalnej redundancji (mRMR), która jest szeroko stosowana w naukowych badaniach selekcji cech [107].

Wstępne badania z użyciem narzędzia FeatureWiz z pośród 77 rozważanych cech wyłoniły 19 kluczowych cech, co stało się punktem wyjścia do ustalenia tej samej liczby cech dla wszystkich stosowanych metod selekcji w celu zapewnienia spójności i porównywalności wyników. Metody takie jak SelectKBest z testem chi-kwadrat (chi2) i testem Anova, RFE oraz PCA wymagają od użytkownika manualnego ustalenia liczby cech do zachowania. Dzięki temu podejściu możliwe będzie precyzyjne porównanie efektywności poszczególnych metod w identyfikacji najbardziej informatywnych cech.

Po usunięciu 8 cech, które nie dostarczały istotnych informacji, będziemy rozważać system z 77 cechami.

Poniżej przedstawiono listę 19 zaproponowanych cech dla każdej z technik.

A. **'FeatureWiz'**: ['bwd_pkt_len_min', 'pkt_size_avg', 'init_fwd_win_byts', 'subflow_fwd_pkts', 'bwd_pkt_len_mean', 'init_bwd_win_byts', 'flow_iat_max', 'flow_duration', 'bwd_pkts_s', 'tot_bwd_pkts', 'bwd_iat_min', 'fwd_pkt_len_mean', 'idle_mean',

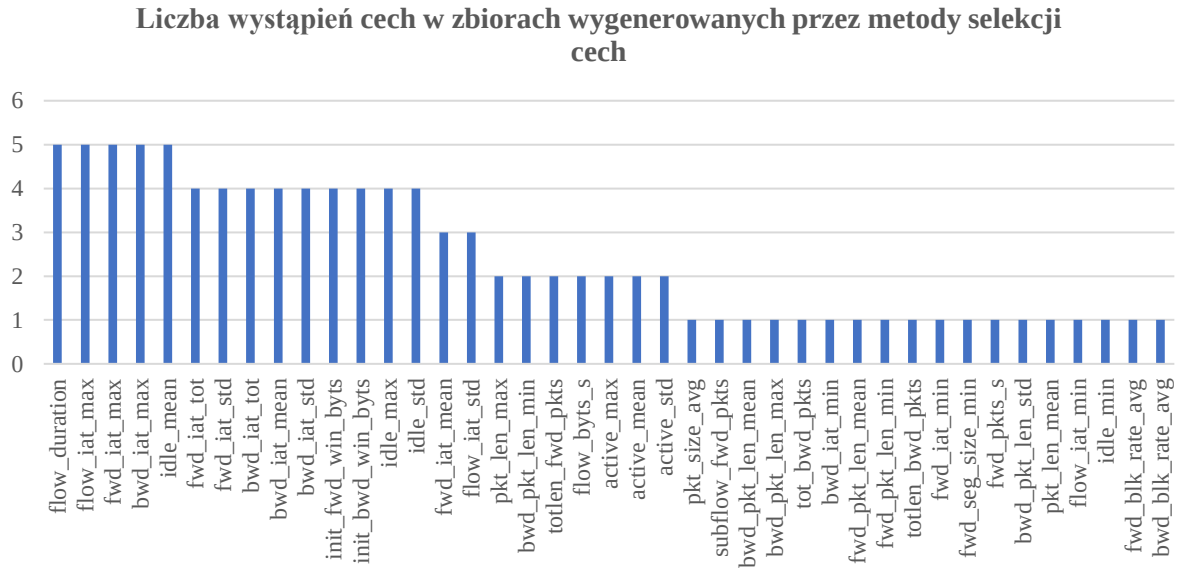
'totlen_fwd_pkts', 'bwd_pkt_len_max', 'totlen_bwd_pkts', 'fwd_iat_min', 'pkt_len_max', 'tot_fwd_pkts'],

B. **'SelectKBest_f_classif'**: ['flow_duration', 'fwd_pkt_len_min', 'bwd_pkt_len_min', 'pkt_len_min', 'fwd_seg_size_min', 'flow_iat_max', 'flow_iat_std', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_mean', 'fwd_iat_std', 'bwd_iat_tot', 'bwd_iat_max', 'bwd_iat_std', 'init_fwd_win_byts', 'init_bwd_win_byts', 'idle_max', 'idle_mean', 'idle_std'],

C. **'SelectBest_chi2'**: ['flow_duration', 'flow_iat_mean', 'flow_iat_max', 'flow_iat_std', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_mean', 'fwd_iat_std', 'bwd_iat_tot', 'bwd_iat_max', 'bwd_iat_mean', 'bwd_iat_std', 'active_max', 'active_mean', 'active_std', 'idle_max', 'idle_min', 'idle_mean', 'idle_std'],

D. **'RFE'**: ['flow_duration', 'flow_byts_s', 'fwd_pkts_s', 'totlen_fwd_pkts', 'fwd_pkt_len_max', 'bwd_pkt_len_min', 'bwd_pkt_len_std', 'pkt_len_max', 'pkt_len_mean', 'flow_iat_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_max', 'bwd_iat_min', 'bwd_iat_mean', 'init_fwd_win_byts', 'init_bwd_win_byts'],

E. **'PCA'**: ['flow_duration', 'fwd_iat_tot', 'bwd_iat_tot', 'bwd_iat_max', 'fwd_iat_max', 'flow_iat_max', 'idle_max', 'bwd_iat_std', 'bwd_iat_mean', 'idle_mean', 'flow_byts_s', 'fwd_iat_std', 'idle_std', 'active_max', 'flow_iat_std', 'fwd_blk_rate_avg', 'fwd_iat_mean', 'active_mean', 'bwd_blk_rate_avg']



Rys. 5.1 Wykres liczby wystąpień cech w każdym ze zbiorów wygenerowanym przez metody selekcji cech

Zidentyfikowane przez algorytmy cechy koncentrują się głównie na czasowych i behawioralnych aspektach przepływów danych. Analiza porównawcza (rys. 5.1) wykazała, że spośród zwróconych 41 analizowanych cech istnieje spójny zestaw pięciu cech –

flow_duration, flow_iat_max, fwd_iat_max, bwd_iat_max oraz idle_mean – które zostały wybrane co najmniej pięciokrotnie przez różne metody selekcji. Ich częsta obecność wskazuje na istotną rolę w wykrywaniu anomalii sieciowych.

5.3 Autorska metoda selekcji cech

Bazując na wynikach różnych algorytmów selekcji cech, zaprojektowano nową metodę o nazwie SelectDVC, która tworzy ranking cech, identyfikując te, które najczęściej pojawiają się w różnych zbiorach danych wygenerowanych przez inne metody selekcji. Zaproponowany autorski algorytm kieruje się założeniem, że im częściej cecha pojawia się w różnych zbiorach wygenerowanych przez różnorodne metody selekcji cech, tym większe jest prawdopodobieństwo, że jest ona istotna. Użytkownik ma możliwość zdefiniowania parametru k, który określa liczbę cech w końcowym zbiorze SelectDVC, co jest przydatne, gdy wymagane jest ograniczenie liczby cech do tych najbardziej znaczących. Również w przypadku tej metody zdecydowano się na wyselekcjonowanie 19 cech. Pseudokod algorytmu selekcji cech SelectDVC został przedstawiony poniżej.

Algorytm 1: Algorytm Selekcji Cech SelectDVC
Wejście: Wczytaj listę zbiorów cech FeatureSets wygenerowanych przez różne metody selekcji cech.

Procedura:

1. Użyj listy FeatureSets, zawierającej zbiory cech wygenerowane wcześniej przez wybrane metody selekcji cech.
2. Utwórz pustą listę FeatureCounts, która będzie przechowywać liczbę wystąpień każdej cechy we wszystkich zbiorach.
3. Dla każdego zbioru cech feature_set w liście FeatureSets:

    1. Iteruj po cechach w danym zbiorze feature_set. Jeśli cecha nie występuje jeszcze w FeatureCounts, dodaj ją z licznikiem wystąpień równym w przeciwnym razie jeśli cecha już istnieje w FeatureCounts, zwiększ jej licznik wystąpień o 1.

4. Utwórz listę SelectedFeatures, zawierającą cechy z FeatureCounts, które mają największą liczbę wystąpień.
5. Posortuj cechy w liście SelectedFeatures malejąco według liczby wystąpień.

6. Utwórz zbiór SelectDVC, wybierając k pierwszych cech (gdzie k jest liczbą określoną przez użytkownika) z posortowanej listy SelectedFeatures.
7. Zwróć zbiór SelectDVC jako wynik działania algorytmu.

Koniec procedury

Wynik: Zbiór SelectDVC, zawierający k cech, które najczęściej pojawiały się we wszystkich analizowanych zbiorach cech.

Po wykonaniu algorytmu, metoda SelectDVC zwróciła następujące cechy:

'SelectDVC': [
'flow_iat_max', 'flow_duration', 'idle_mean', 'fwd_iat_tot', 'fwd_iat_max',
'fwd_iat_std', 'bwd_iat_max', 'bwd_pkt_len_min', 'init_fwd_win_byts',
'init_bwd_win_byts', 'flow_iat_std', 'fwd_iat_mean', 'bwd_iat_tot',
'bwd_iat_std', 'idle_max', 'idle_std', 'bwd_iat_mean', 'bwd_iat_min',
'totlen_fwd_pkts']

Należy jednak szerzej wyjaśnić te cechy, które potencjalne mogą dawać największy udział w zadaniu detekcji ataków DDoS w sieciach SDN na podstawie zbioru SelectDVC. Przy czym wyróżnić możemy cztery główne kategorie cech wykorzystywanych tj. cechy czasowe, cechy między przepływami, cechy pakietowe oraz cechy protokołowe [137][138][139].

Cechy czasowe

- **Maksymalny Interwał Czasowy (Maximum Flow IAT) – flow_iat_max**
  Cecha określa maksymalny czas pomiędzy pakietami w ramach jednego przepływu. W trakcie ataków DDoS często obserwuje się krótkie przerwy między pakietami, co wskazuje na intensywny ruch atakujący.
- **Czas Trwania Przepływu (Flow Duration) – flow_duration**
  Długość trwania pojedynczego przepływu. W przypadku ataków DDoS przepływy są krótkie, ale intensywne, co wynika z dużej liczby krótkotrwałych żądań przeciążających serwery.
- **Maksymalny Czas Bezczynności (Maximum Idle Time) – idle_max**
  Najdłuższy czas bezczynności w przepływie. Podczas DDoS wartości tej cechy mogą być bardzo małe, co oznacza ciągły napływ pakietów.

- **Odchylenie Standardowe Czasu Bezczynności (Standard Deviation of Idle Time) – idle_std**

  Niskie wartości sugerują, iż przepływy są utrzymywane w sposób ciągły i powtarzalny, co może wskazywać na atak DDoS.
- **Średnia Czasu Bezczynności (Idle Mean) – idle_mean**

  Niska wartość tej cechy oznacza, że przepływy są aktywne niemal bez przerw, co może być oznaką ataku DDoS.

Cechy między przepływami

- **Całkowity Czas między Przepływami w przód (Total Forward Interarrival Time) – fwd_iat_tot**

  Mierzy czas między kolejnymi pakietami wysyłanymi w przód. Podczas ataku DDoS ten czas jest zwykle bardzo krótki, ponieważ ma miejsce intensywny ruch w sieci.
- **Maksymalny Czas między Przepływami w przód (Maximum Forward Interarrival Time) – fwd_iat_max**

  Krótkie interwały mogą wskazywać na powtarzalny, intensywny ruch, typowy dla ataków DDoS.
- **Odchylenie Standardowe Czasu między Przepływami w przód (Standard Deviation of Forward Interarrival Time) – fwd_iat_std**

  Cecha wskazująca zmienność czasu między pakietami. Niskie wartości sugerują powtarzalność, co jest charakterystyczne dla ataków DDoS.
- **Całkowity Czas między Przepływami w wstecz (Total Backward Interarrival Time) – bwd_iat_tot**

  Suma czasów między kolejnymi pakietami w wstecz. Podczas ataku DDoS wartości te mogą być znacząco niższe.
- **Maksymalny Czas między Przepływami w wstecz (Maximum Backward Interarrival Time) – bwd_iat_max**

  Krótkie wartości tej cechy mogą sugerować intensywny ruch w odpowiedzi na nadmierne żądania, typowe dla ataków DDoS.
- **Średnia Czasu między Przepływami w przód (Forward Interarrival Time Mean) – fwd_iat_mean**

  Niska wartość wskazuje na częste i regularne przesyłanie pakietów w przód, co jest typowe dla ataków DDoS.

- **Odchylenie Standardowe Czasu między Przepływami w wstecz (Standard Deviation of Backward Interarrival Time) – bwd_iat_std**

  Niskie wartości mogą wskazywać na jednolity ruch, co sugeruje automatyczny charakter ataku DDoS.
- **Średnia Czasu między Przepływami w wstecz (Backward Interarrival Time Mean) – bwd_iat_mean**

  Podczas ataku DDoS wartości tej cechy mogą być niższe niż w normalnym ruchu sieciowym, co oznacza zwiększoną intensywność ruchu.
- **Minimalny Czas między Przepływami wstecz (Minimum Backward Interarrival Time) – bwd_iat_min**

  Niskie wartości mogą świadczyć o przeciążeniu serwera bądź kontrolera SDN poprzez nadmierne żądania.

Cechy pakietowe

- **Minimalna Długość Pakietu w wstecz (Minimum Backward Packet Length) – bwd_pkt_len_min**

  Wartość ta może wskazywać na obecność dużej liczby krótkich pakietów, charakterystycznych dla ataków DDoS.
- **Całkowita Długość Pakietów Wysyłanych w Przepływie (Total Length of Forward Packets) – totlen_fwd_pkts**

  Wysokie wartości mogą wskazywać na masowy transfer danych przez sieć komputerową, co w przypadku nagłego skoku może sygnalizować atak DDoS.

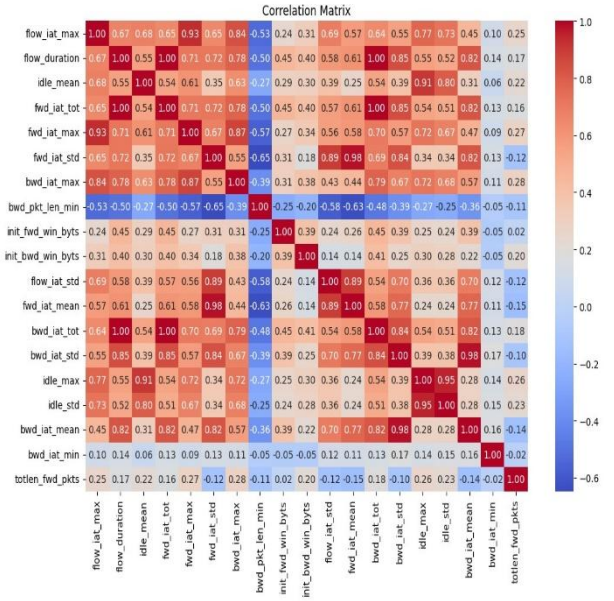
Cechy protokołowe

- **Poczatkowa Wielkość Okna TCP dla Strumienia Wysyłanego (Initial Forward Window Bytes) – init_fwd_win_byts.**

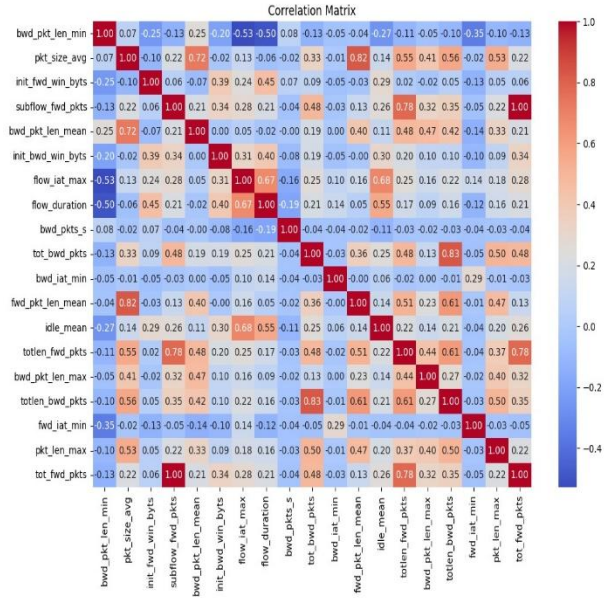
  Wielkość okna TCP może być modyfikowana przez atakującego w celu zwiększenia szybkości transmisji, co bywa wykorzystywane w trakcie ataków DDoS.
- **Poczatkowa Wielkość Okna TCP dla Strumienia Odbieranego (Initial Backward Window Bytes) – init_bwd_win_byts.**

  Analogicznie do wersji dla ruchu w przód, nietypowe wartości tej cechy mogą wskazywać na manipulację ruchem sieciowym w tym atak DDoS.

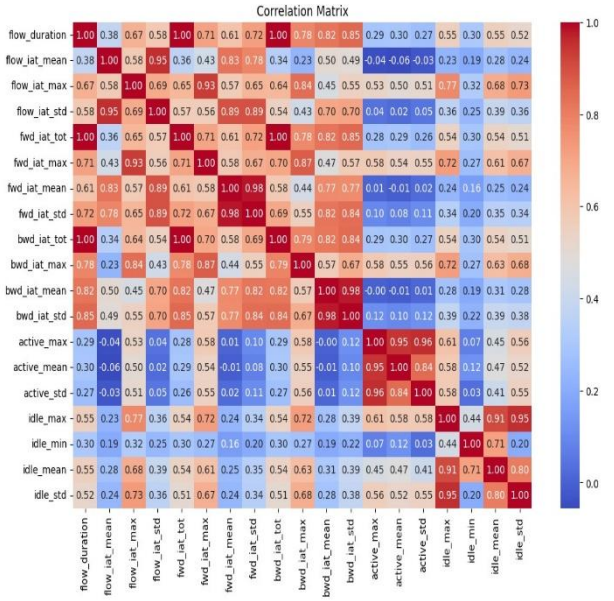
W ramach dalszych badań przeprowadzono analizę korelacji cech we wszystkich zestawach cech w celu zrozumienia, jak zmiany jednej cechy są powiązane ze zmianami w innej cesze. Uzyskane wyniki wskazują na wysoką korelację pomiędzy zmiennymi co można zaobserwować w zbiorach wygenerowanych przez metody takie jak SelectDVC (12 silnie skorelowanych cech), SelectKBest z użyciem testu Chi-kwadrat (15 silnie skorelowanych cech). Zbiór wygenerowany przez narzędzie skupiające się na eliminowaniu silnie skorelowanych cech FeatureWiz posiada tylko 3 takie cechy. Poniżej na rysunku 5.2 przedstawiono macierze korelacji obliczone na podstawie współczynnika korelacji Pearsona.



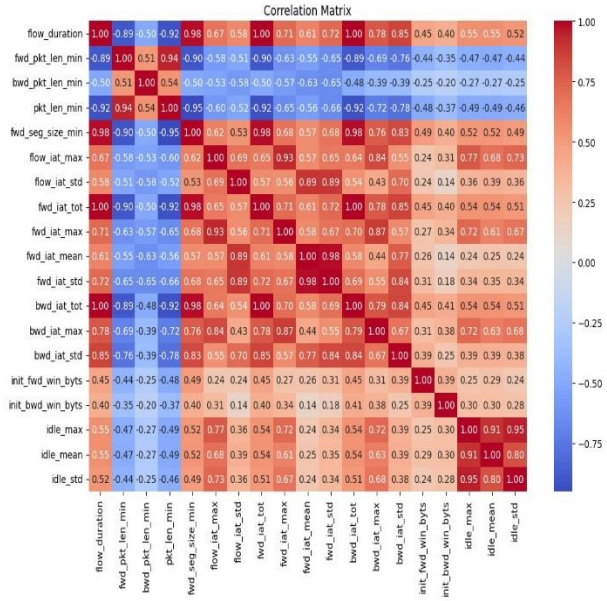
a) SelectDVC



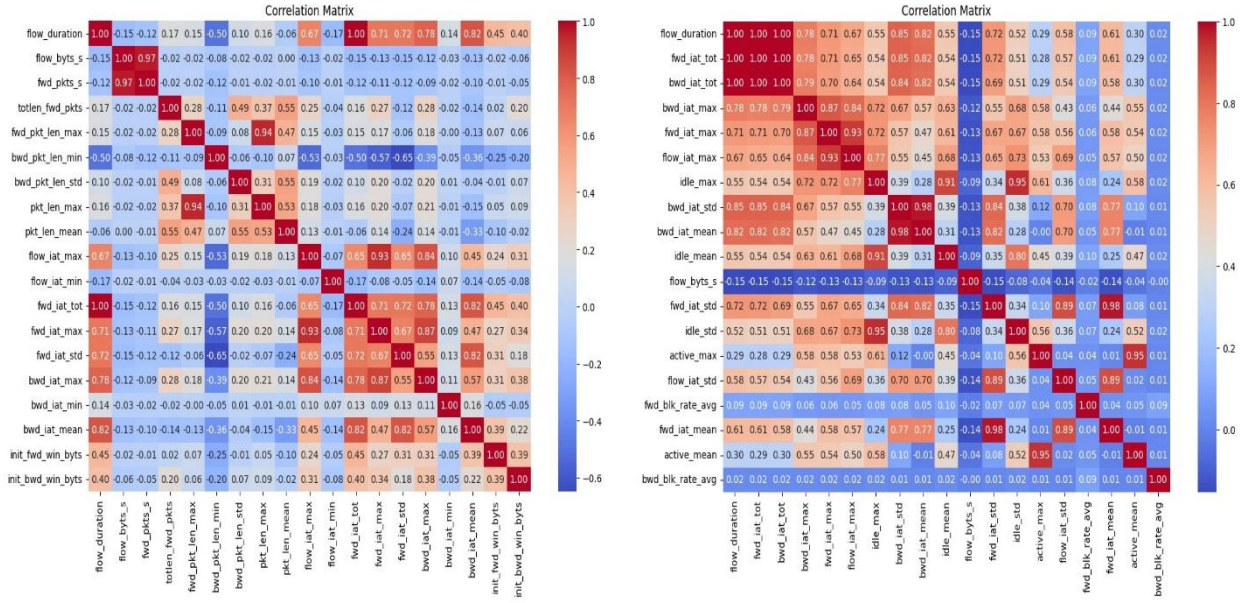
b) FeatureWiz



c) SelectKBest_chi2



d) SelectKBest_f_classif



e) RFE f) PCA

Rys. 5.2. Macierze korelacji Pearsona wygenerowanych zbiorów przez metody selekcji cech.

W macierzach tych każdy element na pozycji *i*,*j* reprezentuje współczynnik korelacji Pearsona między *i*-tą a *j*-tą zmienną. Wartości te mieszczą się w zakresie od -1 do 1, gdzie 1 oznacza doskonałą korelację dodatnią, -1 oznacza doskonałą korelację ujemną, a 0 oznacza brak liniowej zależności między zmiennymi [108]. Nie stwierdzono, aby wszystkie cechy były ze sobą skorelowane w żadnym ze zbiorów. Zidentyfikowano natomiast 5 kluczowych cech charakteryzujących się najwyższym stopniem korelacji w analizowanych zbiorach. Są to:

- czas trwania przepływu (flow_duration),
- całkowity czas między pakietami wysłanymi w przód (fwd_iat_tot),
- standardowe odchylenie czasu między pakietami w przód (fwd_iat_std),
- maksymalny czas między pakietami (flow_iat_max)
- maksymalny czas między pakietami w tył (bwd_iat_max).

Po przeprowadzonej analizie korelacji cech nasuwa się pytanie, czy w systemach detekcji ataków DDoS pożądane jest użycie zbiorów danych o wysokiej korelacji czy również dobre wyniki zarówno pod względem dokładności, jak i innych miar wydajności mogą osiągnąć modele uczenia maszynowego korzystające ze zbioru niskoskorelowanych cech FeatureWiz.

6. Analiza możliwości użycia algorytmu klasyfikacji NGBoost w detekcji ataków DDOS w oparciu o metryki wydajności, kryteria informacyjne złożoności obliczeniowej oraz wyjaśnialności

Jednym z obiecujących, ale wciąż mało zbadanych algorytmów uczenia maszynowego, który może być użyteczny w detekcji ataków DDoS, jest NGBoost (Natural Gradient Boosting). Dotychczasowe badania naukowe nie skupiały się na zastosowaniu tego algorytmu w takich scenariuszach, co stanowi motywację do dalszego testowania, szczególnie biorąc pod uwagę jego otwarty i modyfikowalny kod źródłowy.

Z uwagi na udokumentowaną w literaturze przedmiotu wysoką skuteczność algorytmów takich jak Decision Tree, XGBoost, LightGBM oraz Random Forest w identyfikacji ataków DDoS, celowe wydaje się przeprowadzenie analizy porównawczej z algorytmem NGBoost. Analiza ta uwzględniać będzie nie tylko wydajność i złożoność obliczeniową poszczególnych metod, ale również ich wyjaśnialność, którą można ocenić za pomocą metryki D zaproponowanej przez A. Rosenfelda. W niniejszym rozdziale sformułowano następujące pytania badawcze:

- Czy model oparty na algorytmie NGBoost może z powodzeniem być stosowany w systemach detekcji ataków DDoS w sieciach SDN?
- Czy modele wykorzystujące zbiór cech SelectDVC osiągają wyższą skuteczność w identyfikacji ataków DDoS w sieciach SDN w porównaniu z alternatywnymi metodami selekcji cech?
- Czy model NGBoost, jako model o charakterze "czarnej skrzynki" lub inny proponowany model zespołowy, powinien być preferowany względem transparentnego modelu Decision Tree w kontekście detekcji ataków DDoS?
- Czy modyfikacja metryki D poprzez zastosowanie kryteriów informacyjnych Akaike oraz Bayesa, w celu zastąpienia wydajności modelu przez jego złożoność obliczeniową, faworyzuje modele o charakterze "czarnej skrzynki", w tym NGBoost?

6.1 Zastosowane klasyfikatory w eksperymentach

W ramach niniejszych badań wykorzystano dwa odmienne podejścia do uczenia zespołowego, reprezentowane przez algorytmy Random Forest i Gradient Boosted Trees (GBT). Random Forest wykorzystuje technikę bagging, która redukuje wariancję modelu

poprzez równoległe trenowanie wielu niezależnych drzew decyzyjnych. Z kolei GBT, w tym XGBoost, LightGBM i NGBoost, stosują technikę boosting, polegającą na sekwencyjnym budowaniu modeli, z naciskiem na redukcję błędów popełnianych w poprzednich iteracjach [109].

W fazie eksperymentalnej dokonano ewaluacji proponowanych metod selekcji cech, wykorzystując do tego celu klasyfikatory tj. NGBoost, Decision Tree, XGBoost, LightGBM oraz Random Forest. Implementacja tych algorytmów została zrealizowana z wykorzystaniem biblioteki języka Python o nazwie scikit-learn, a także dedykowanych bibliotek dla XGBoost, LightGBM i NGBoost, które zapewniają bezproblemową integrację z ekosystemem scikit-learn. Poniżej znajduje się krótki opis tych algorytmów.

Algorytm Decision Tree rozwiązuje problemy regresji i klasyfikacji, tworząc strukturę drzewa z optymalnie wybranymi węzłami i przycinając niepotrzebne gałęzie. Jest to algorytm zachłanny, który redukuje nadmierne dopasowanie poprzez selektywne eliminowanie gałęzi na podstawie kryteriów takich jak entropia [110].

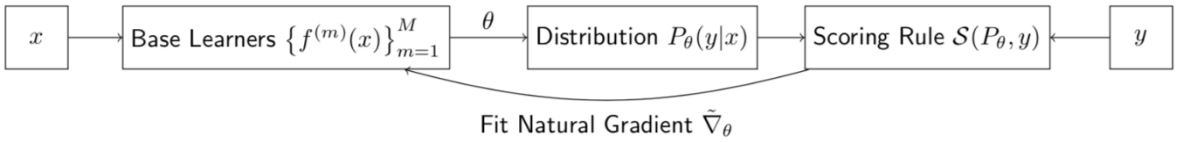
Random Forest stosuje technikę zwaną bagging. Polega ona na tworzeniu licznych podzbiorów danych treningowych poprzez bootstrapping, a następnie trenowaniu osobnego drzewa decyzyjnego na każdym z tych podzbiorów. Ostateczna prognoza jest wynikiem głosowania lub uśredniania wyników wszystkich drzew [111].

LightGBM wykorzystuje technikę dzielenia drzewa w sposób „leaf-wise" zamiast tradycyjnej metody „level-wise". Dodatkowo zastosowano w nim technikę histogramową, która przyspiesza proces obliczeniowy podczas poszukiwania optymalnych podziałów w drzewie. Optymalizuje funkcję straty za pomocą metody gradientowej, gdzie prognoza dostarczona przez grupę drzew decyzyjnych oblicza gradienty funkcji straty. Aktualizując prognozę w kierunku ujemnego gradientu, model dąży do minimalizacji straty. Dodawany jest także człon regularyzacyjny, aby zapobiec nadmiernemu dopasowaniu modelu [112].

XGBoost zdobył popularność dzięki doskonałym wynikom na dużych zbiorach danych i licznym zwycięstwom w konkursach data science. Posiada zaawansowaną regularyzację (L1 i L2), która pomaga zapobiegać nadmiernemu dopasowaniu. Algorytm wykorzystuje strukturę zespołową zwaną DMatrix, która optymalizuje zarówno przechowywanie, jak i efektywność obliczeniową. Ostatnie rozwinięcia tego algorytmu obejmują funkcje takie jak uczenie się na GPU i uczenie rozproszone [113].

NGBoost został opracowany przez Stanford ML Group, to algorytm uczenia maszynowego wykorzystujący naturalne gradienty do przewidywania rozkładów prawdopodobieństwa. Architektura NGBoost składa się z trzech głównych komponentów tj.

Base Learners, Distribution oraz Scoring Rule co przedstawiono na rysunku 6.1 korzystając z oryginalnej pracy [114].



Rys. 6.1. Schemat działania algorytmu NGBoost.

Algorytm został pierwotnie stworzony dla problemów regresji, a następnie zaadaptowany do zadań klasyfikacyjnych. Domyślnie jako parametr Base Learners algorytm przyjmuje drzewa decyzyjne, choć możliwe jest również zastosowanie regresji grzbietowej (ang. Ridge Regression). NGBoost przyjmuje dane wejściowe x, a wyjścia y służą do estymacji prawdopodobieństwa warunkowego. W przypadku klasyfikacji binarnej, domyślnym rozkładem jest rozkład Bernoulliego. Scoring Rule, czyli reguła oceniająca przewidywany rozkład prawdopodobieństwa w odniesieniu do obserwowanej wartości cechy docelowej. NGBoost domyślnie implementuje metodę Maximum Likelihood Estimation (MLE), choć dostępna jest również metoda Continuous Ranked Probability Score (CRPS) do optymalizacji prognoz. Dostępność NGBoost na licencji open-source umożliwia jego modyfikację i dostosowanie do specyficznych zastosowań, co czyni go elastycznym narzędziem w ocenie ryzyka i niepewności [114].

6.2 Środowisko badawcze

Eksperymenty przeprowadzone w ramach niniejszej rozprawy doktorskiej zostały zrealizowane przy użyciu domyślnych konfiguracji klasyfikatorów, rezygnując z procesu dostrajania hiperparametrów. Takie podejście jest zgodne z paradygmatem Data-Centric AI, kładąc nacisk na jakość i specyfikę danych jako czynników niezbędnych dla osiągnięcia wysokiej wydajności modelu. Zaproponowane w niniejszych badaniach podejście jest o tyle ważne, gdyż w przypadku ataków DDoS, nadmierne dopasowanie może prowadzić do generowania fałszywych alarmów. Model wytrenowany wyłącznie na danych treningowych może błędnie klasyfikować normalny ruch sieciowy jako atak, jeżeli dane testowe są zbyt podobne do danych treningowych, ale nie zawierają rzeczywistych ataków.

W celu przyspieszenia obliczeń oraz zwiększenia wydajności analizy, zastosowano bibliotekę Scikit-Learn Intelex opracowaną i opublikowaną przez firmę Intel [115]. Jest to rozszerzenie dla pakietu naukowego Scikit-Learn pozwalające na wykorzystanie zoptymalizowanych wersji standardowych algorytmów klasyfikacji binarnej, takich jak SVM, Random Forest czy K-NN w obliczeniach równoległych na wszystkich rdzeniach procesora. Implementacja tej biblioteki w środowisku badawczym stanowi kontrast do standardowego środowiska Scikit-Learn, które domyślnie uruchamia symulacje na pojedynczym rdzeniu. Taki sposób działania może znacząco wydłużać czas przeprowadzania eksperymentów oraz zwiększać złożoność obliczeniową całego badania.

Autorskie skrypty odpowiadające za opracowanie modeli i systemu detekcji ataków DDoS napisane zostały w języku Python, a także w badaniach wykorzystano oficjalne biblioteki NGBoost, XGBoost, LightGBM oraz SHAP.

6.3 Proponowane metryki wydajności

Celem przeprowadzonych eksperymentów jest ocena skuteczności różnych metod selekcji cech w wykrywaniu ataków DDoS w sieciach SDN oraz identyfikacja optymalnego zestawu cech pozwalającego na uzyskanie wysokiej jakości klasyfikacji przy jednoczesnym zmniejszeniu złożoności obliczeniowej modeli. Do tego celu zastosowano szereg metryk, które pozwoliły na określenie ich dokładności klasyfikacji binarnej, złożoności obliczeniowej oraz wydajności. W szczególności, analiza skupiła się na wynikach opartych na macierzy pomyłek, która umożliwia porównanie wartości przewidywanych z wartościami rzeczywistymi. Wydajność modelu oceniano używając krzywej ROC (Receiver Operating Characteristic) oraz pola pod krzywą AUC (Area Under the Curve). Krzywa ROC przedstawia zależność między wskaźnikiem fałszywie pozytywnych a wskaźnikiem prawdziwie pozytywnych wyników.

Macierz pomyłek (ang. confusion matrix) to narzędzie stosowane w uczeniu maszynowym i klasyfikacji, które umożliwia ocenę jakości klasyfikatora poprzez porównanie rzeczywistych i przewidywanych wartości [116].

Dla klasyfikatora binarnego, macierz pomyłek przedstawia się następująco:

	Klasa A	**Klasa B**
Sklasyfikowane jako A	True positive (TP)	False positive (FP)
Sklasyfikowane jako B	False negative (FN)	True negative (TN)

W zadaniu detekcji ataków DDoS, poszczególne elementy macierzy interpretuje się następująco:

- True Positive (TP) będącą liczbą przypadków poprawnie zidentyfikowanych ataków DDoS.
- False Positive (FP) będącą liczbą błędnie sklasyfikowanych przypadków jako ataki DDoS, gdy w rzeczywistości nimi nie były.
- True Negative (TN) będącą liczba przypadków poprawnie zidentyfikowanych jako brak ataku.
- False Negative (FN) będącą liczbą przypadków błędnie sklasyfikowanych jako brak ataku, gdy w rzeczywistości atak DDoS miał miejsce.

W przypadku algorytmów wykrywających ataki DDoS, prawdziwie pozytywne wyniki są zazwyczaj bardziej krytyczne niż prawdziwie negatywne, gdyż błędy polegające na niezauważeniu przez system ataku mają znacznie poważniejsze konsekwencje niż sytuacje, w których system omyłkowo zasygnalizuje zagrożenie (fałszywe alarmy). System oparty na uczeniu maszynowym powinien maksymalizować wykrywanie ataków przy jednoczesnym minimalizowaniu fałszywych alarmów.

Każdy model był oceniany na podstawie metryk wydajności wyprowadzonych macierzy pomyłek oraz formuł takich jak dokładność, precyzja, czułość, wskaźnik F1, które zostały zdefiniowane poniżej.

Accuracy mierzy ogólną skuteczność klasyfikatora. Jest to stosunek liczby poprawnie sklasyfikowanych próbek do całkowitej liczby próbek określoną poniższym wzorem:

$$Accuracy = \frac{TN+TP}{TN+FP+TP+FN} \quad [116]$$

Precision mierzy dokładność pozytywnych predykcji klasyfikatora. Jest to stosunek liczby poprawnie zaklasyfikowanych pozytywnych przypadków TP do sumy TP i fałszywie pozytywnych przypadków FP określoną poniższym wzorem:

$$Precision = \frac{TP}{TP + FP} \quad [116]$$

Recall mierzy zdolność klasyfikatora do poprawnego zidentyfikowania wszystkich pozytywnych przypadków. Jest to stosunek liczby TP do sumy TP i fałszywie negatywnych przypadków FN określoną poniższym wzorem:

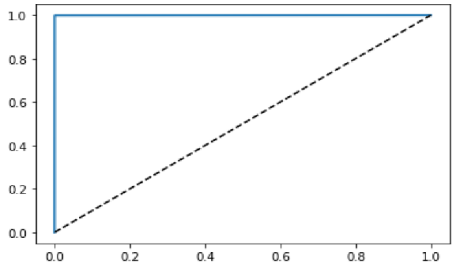
$$Recall = \frac{TP}{TP + FN} \quad [116]$$

F1-Score to miara harmoniczna precyzji i czułości, która zapewnia zrównoważoną ocenę klasyfikatora. F1-Score jest szczególnie przydatne, gdy równocześnie zależy nam na wysokiej precyzji i czułości określoną poniższym wzorem:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad [116]$$

ROC AUC (Receiver Operating Characteristics Area Under the Curve) to wskaźnik efektywności klasyfikatorów binarnych, reprezentujący obszar pod krzywą ROC. Krzywa ta ilustruje zdolność klasyfikatora do odróżniania między klasami pozytywnymi a negatywnymi, ukazując relację między wskaźnikami prawdziwie pozytywnych wyników (TP) a fałszywie pozytywnymi (FP). Wartości ROC AUC mieszczą się w przedziale od 0 do 1. Przerywana linia na rysunku 6.2 reprezentuje krzywą ROC dla klasyfikatora działającego na zasadzie losowości, co skutkuje AUC równym 0,5. Wyższa wartość ROC AUC świadczy o lepszej zdolności klasyfikacyjnej modelu. ROC AUC jest obliczana niezależnie od macierzy błędów. [116]. Dla idealnego klasyfikatora, pole pod krzywą (AUC) powinno odpowiadać 1, co jest demonstracyjnie przedstawione na rysunku 6.2.

ROC AUC = Pole powierzchni pod krzywą ROC

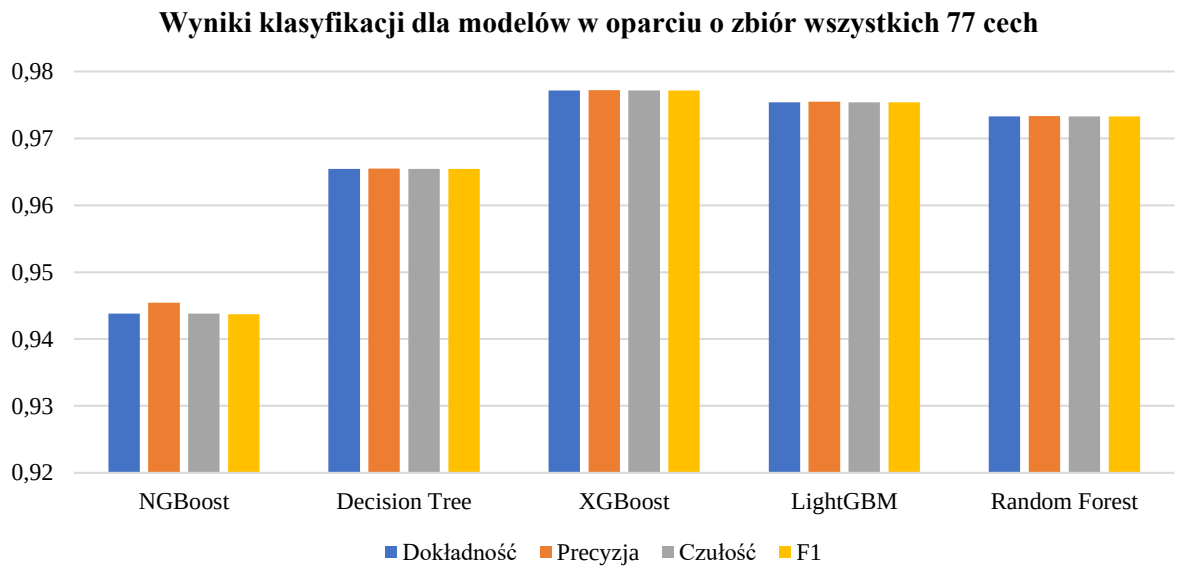


Rys. 6.2. Krzywa ROC dla idealnego klasyfikatora.

6.4 Analiza wyników eksperymentów

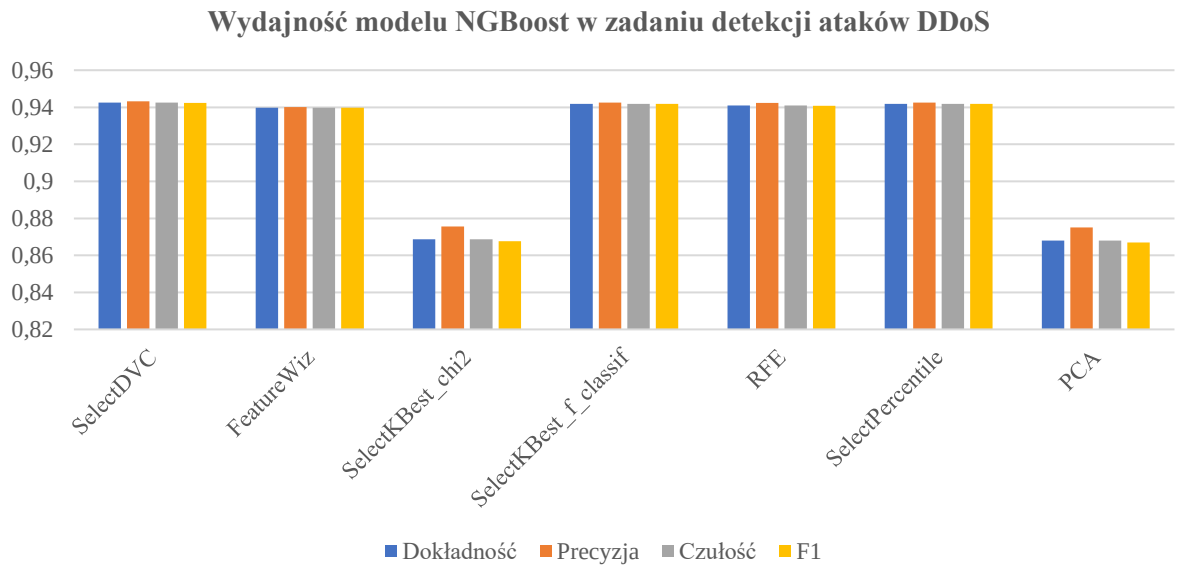
W pierwszym etapie badań przeprowadzono eksperyment walidacyjny, w którym wykorzystano pełny zestaw 77 cech z autorskiego zbioru danych DVCDDoS2022. Dane podzielono za pomocą 5-krotnej crosswalidacji StratifiedKFold, co pozwoliło uzyskać stabilniejszą i bardziej wiarygodną ocenę wydajności modelu. Wiedzę tą, można wykorzystać w celu ustanowienia punktu odniesienia dla późniejszych analiz opartych na zredukowanych zbiorach cech. Mniejsze zestawy cech mogą prowadzić do modeli o niższej złożoności obliczeniowej, co przyspieszy proces trenowania i wnioskowania w proponowanym systemie

detekcji. Na rysunku 6.3 przedstawiono wyniki klasyfikacji na podstawie, których wynika, że wszystkie modele osiągają wysoką dokładność (powyżej 0,94). Najlepsze wyniki osiągnęły modele XGBoost i LightGBM, z dokładnością odpowiednio 0,971 i 0,9754. Podobne wyniki obserwujemy dla precyzji, czułości i F1-score, co wskazuje na wysoką jakość klasyfikacji przez te modele.



Rys. 6.3. Wyniki klasyfikacji dla modelów w oparciu o zbiór wszystkich 77 cech.

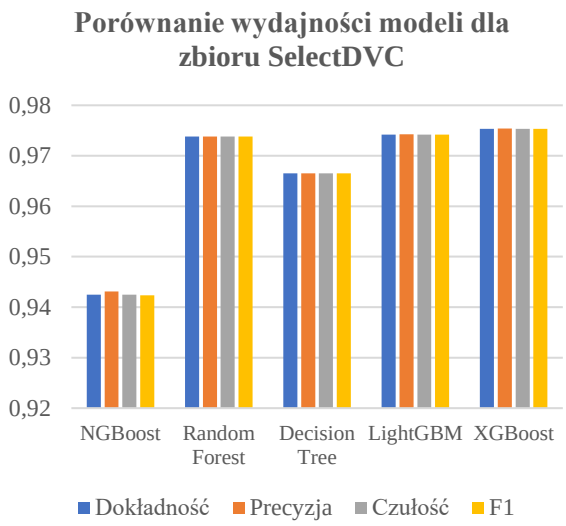
W kolejnym etapie badań oceniono skuteczność różnych metod selekcji cech, takich jak SelectDVC, FeatureWiz, SelectKBest_chi2, SelectKBest_f_classif, RFE oraz PCA. Każdy z wygenerowanych zbiorów liczył 19 cech, a ich efektywność została zweryfikowana za pomocą modelu NGBoost. Mimo, iż wszystkie modele charakteryzują się zrównoważonymi metrykami wydajności, konieczna jest ocena, na ile redukcja cech nie powoduje istotnej utraty jakości klasyfikacji wobec zastosowania pełnego zbioru cech. Celem było również udzielenie odpowiedzi na pytanie, czy model oparty na NGBoost może być z powodzeniem stosowany w zadaniu detekcji ataków DDoS w sieciach SDN?



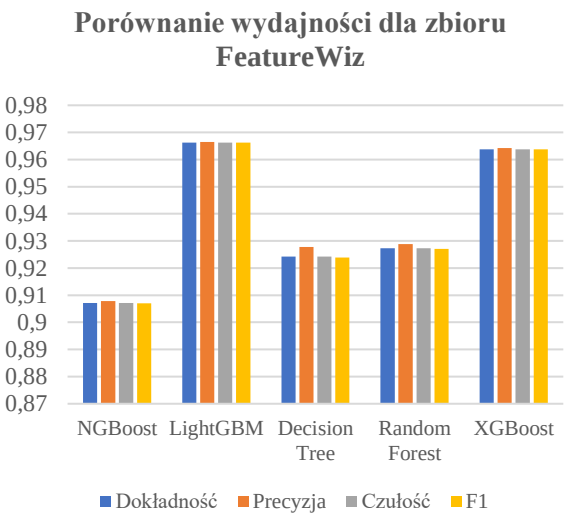
Rys. 6.4 Wydajność modelu NGBoost w zadaniu detekcji ataków DDoS dla każdego ze zbiorów danych

Wyniki klasyfikacji przedstawione na rysunku 6.4 pokazują, iż na skuteczność algorytmu NGBoost w detekcji ataków DDoS mają rzeczywisty wpływ wybrane cechy np. cechy ze zbiorów SelectDVC i SelectKBest_f_classif okazały się najbardziej efektywne (najwyższe dokładności (ponad 0,9418)), a cechy ze zbiorów SelectKBest_chi2 i PCA osiągnęły nieco niższą dokładność (około 0,868). W związku z tym model oparty na algorytmie NGBoost może być z powodzeniem stosowany w zadaniu detekcji ataków DDoS w sieciach SDN utrzymując dobry balans między wykrywaniem ataków (czułość) a unikaniem fałszywych alarmów (precyzja).

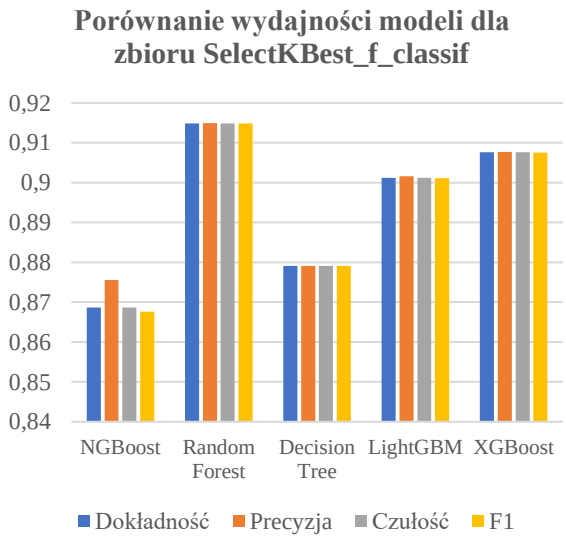
Ostatni etap eksperymentów skupił się na porównaniu skuteczności różnych modeli klasyfikacyjnych działających na zestawie cech SelectDVC. Jak pokazano na rysunku 6.5 wyniki jednoznacznie wskazały, że najlepszymi klasyfikatorami są XGBoost, LightGBM i Random Forest, które osiągnęły dokładności w przedziale 0,973-0.977. Model NGBoost, choć wykazał się solidnymi wynikami, nie dorównał najskuteczniejszym algorytmom boostingowym i Random Forest.



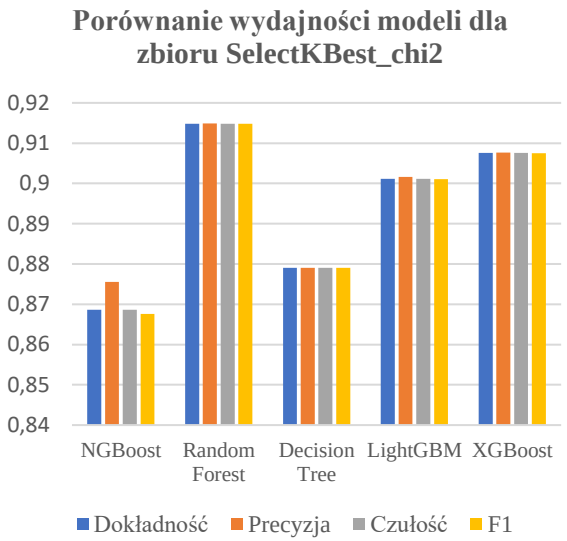
a) Zbiór SelectDVC



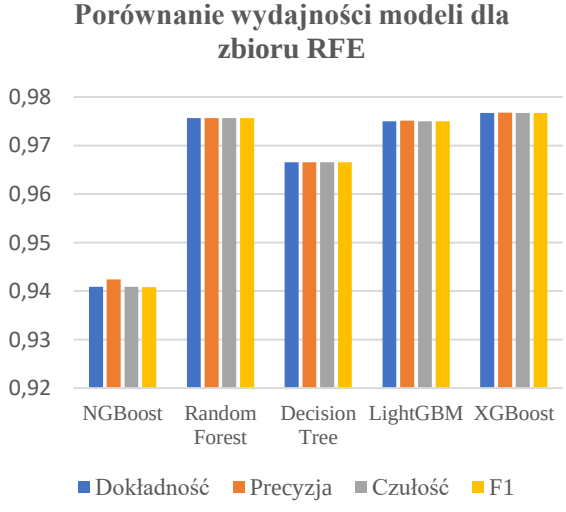
b) Zbiór FeatureWiz



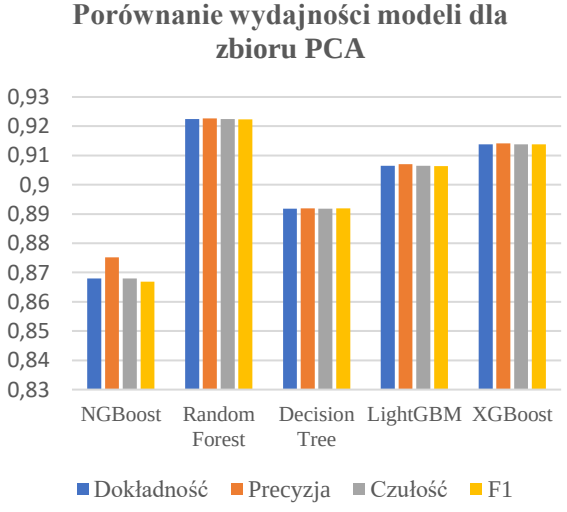
c) Zbiór SelectKBest_f_classfif(Anova)



d) Zbiór SelectKBest_Chi2



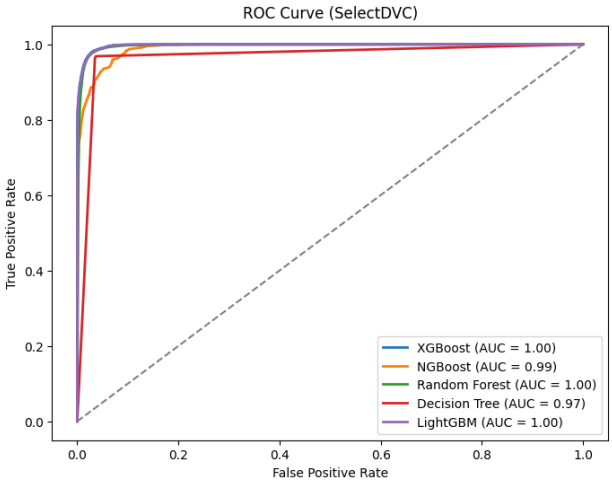
e) Zbiór RFE



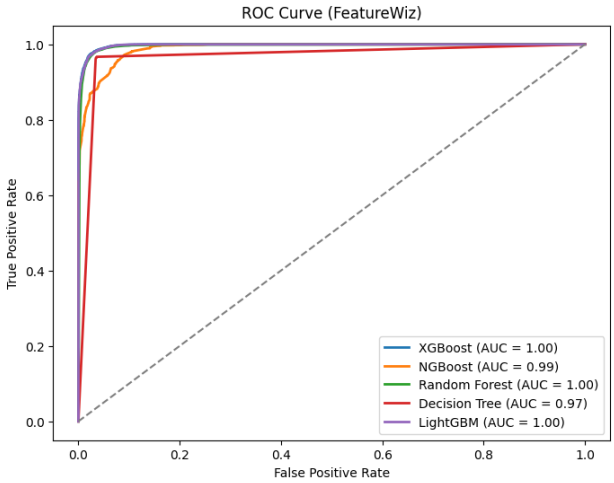
f) Zbiór PCA

Rys. 6.5 Analiza porównawcza wydajności modeli detekcji ataków DDoS w czterech metrykach

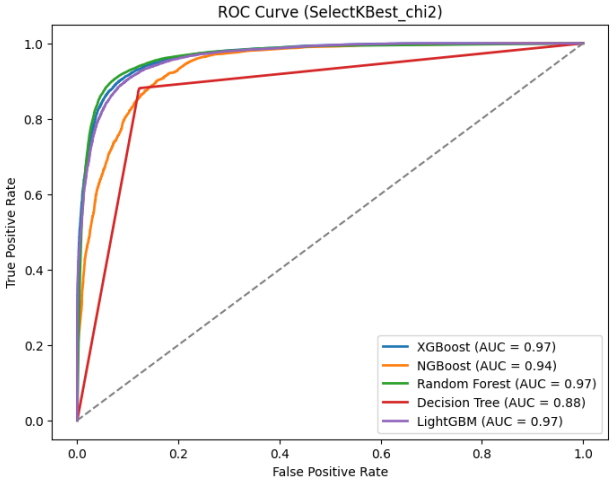
Ostatnim etapem była analiza krzywych ROC i wartości ROC AUC na podstawie rysunku 6.6. Na uwagę zasługują szczególnie modele oparte na XGBoost, Random Forest, NGBoost oraz LightGBM, które osiągnęły wartości AUC bliskie 1.00 dla większości zbiorów cech, potwierdzając przy tym wyjątkową zdolność do rozróżniania ataków od normalnego ruchu w sieci. Co więcej, jak widać po krzywych ROC wszystkie oceniane modele radziły słabiej w oparciu o zestawy cech PCA oraz SelectKBest_chi2.



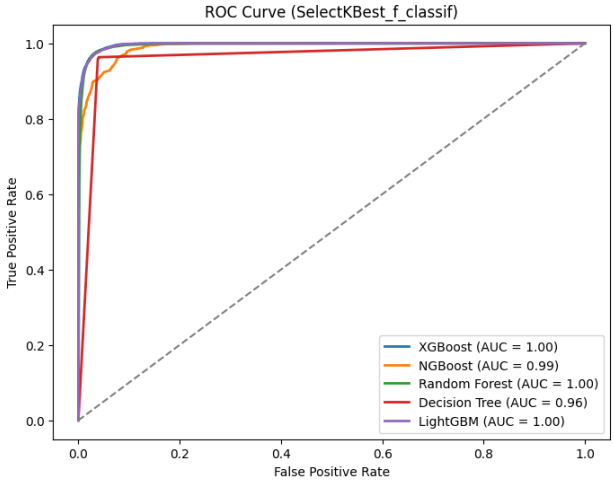
a) SelectDVC



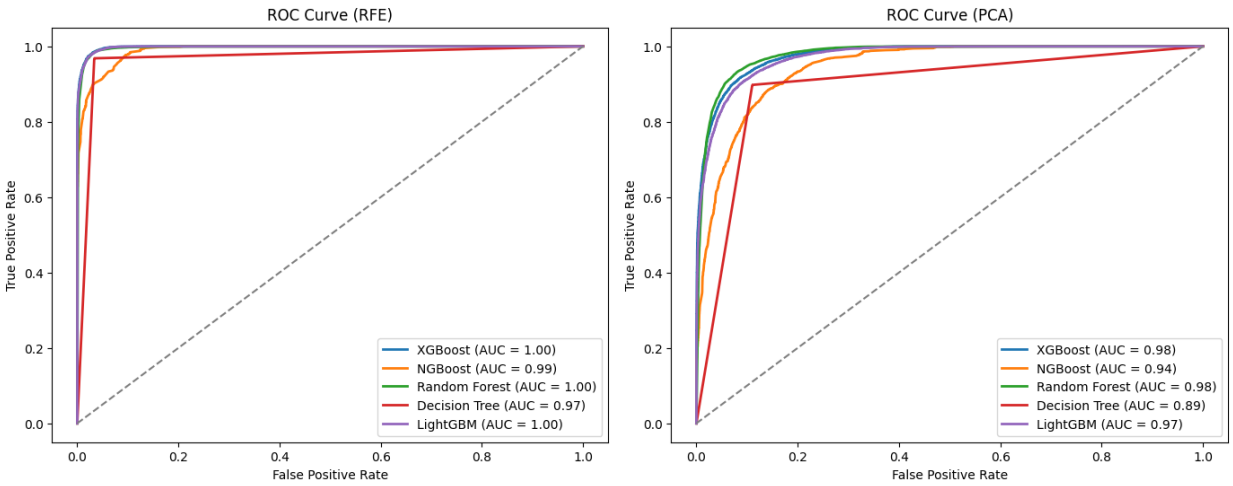
b) FeatureWiz



c) SelectKBest_chi2



d) SelectKBest_f_classif



e) RFE

f) PCA

Rys. 6.6. Analiza krzywych ROC i wartości ROC AUC modeli w zadaniu detekcji ataków DDoS dla wszystkich zbiorów wygenerowanych przez metody selekcji cech.

Zaprezentowane wyniki potwierdziły, że eksperymentowanie z metodami selekcji cech jest niezbędne dla osiągnięcia najlepszych wyników w detekcji ataków DDoS w sieciach SDN. Redukcja cech do 19 nie prowadzi do istotnej utraty jakości klasyfikacji, a jednocześnie znacząco zmniejsza złożoność obliczeniową modeli, co jest ważne dla praktycznych implementacji systemu detekcji. Przykładowo zestaw 19 cech (zbiór SelectDVC) zapewnia modele o dokładności od 0,9425 do 0,9753, a w przypadku pełnego zbioru 77 cech dokładność wzrasta do 0,9771 w przypadku XGBoost. Można, więc uznać, iż zaproponowany zbiór cech SelectDVC zawiera najistotniejsze cechy dla klasyfikatorów prowadząc do lepszych wyników, podobnie jak zbiór pochodzący z metody RFE. Co ciekawe, zbiór FeatureWiz o minimalnej liczbie skorelowanych cech również pozwolił na osiągnięcie wysokiej wydajności klasyfikatorów, podważając konieczność stosowania wyłącznie zbiorów wysokoskorelowanych co jest odpowiedzią na pytanie postawione w poprzednim rozdziale. Niewłaściwy wybór zbioru cech może znacząco pogorszyć skuteczność wykrywania zagrożeń co obrazują wyniki dla zbiorów PCA. Warto zauważyć, że wszystkie badane algorytmy osiągnęły wynik powyżej 0,90 w metryce ROC AUC, co wskazuje, że klasyfikatory zaproponowane do wykrywania ataków DDoS w sieciach definiowanych programowo mogą być skutecznie wykorzystywane w tym zadaniu.

6.5 Analiza wyjaśnialności w oparciu o metrykę D

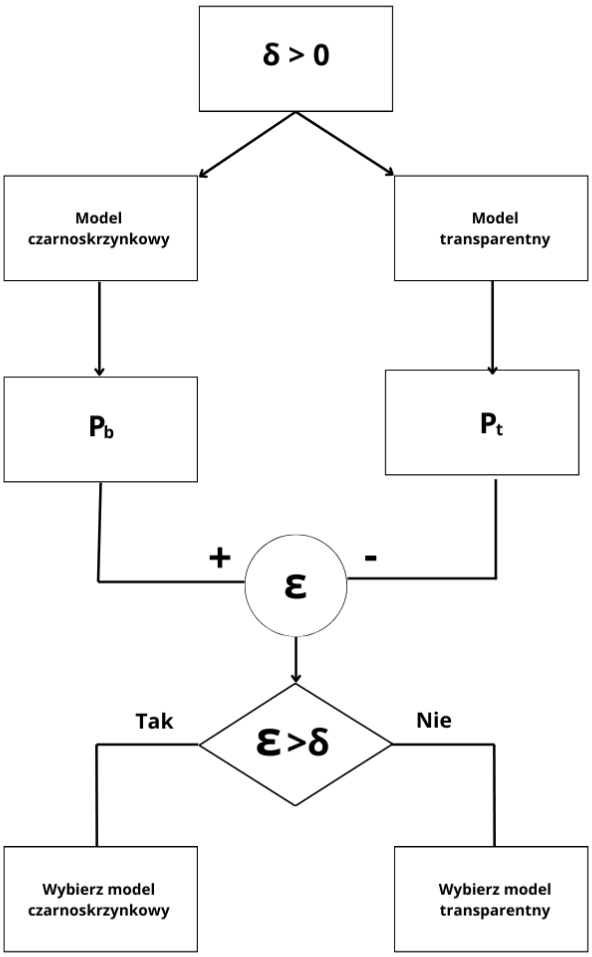
W niniejszym podrozdziale zaproponowano analizę wyjaśnialności modeli uczenia maszynowego w procesie detekcji ataków DDoS w oparciu o autorską implementację metryki D zaproponowaną przez A. Rosenfelda. Metryka ta stanowi ilościową miarę zmiany dokładności klasyfikacji pomiędzy modelem "czarnej skrzynki" a modelem transparentnym. Analizując te różnice, zyskujemy wgląd w to, w jakim stopniu zwiększenie transparentności modeli może wpłynąć na ich skuteczność, nie tracąc przy tym na dokładności ich działania [41]. Modele transparentne, takie jak drzewa decyzyjne czy modele liniowe, pozwalają na łatwiejsze zrozumienie, jak podejmowane są decyzje w modelu. Z kolei modele czarnoskrzynkowe, takie jak XGBoost czy Random Forest, choć oferują lepszą dokładność, ich wewnętrzna struktura jest trudniejsza do zrozumienia.

Wzór metryki D uwzględnia różnicę wydajności modelu "czarnej skrzynki" (Pb) oraz wydajności modelu transparentnego (Pt), to znaczy

$$D = Pb - Pt \quad (6.1)$$

Zastosowanie tej metody wymaga konsultacji z ekspertem dziedzinowym (np. lekarz lub ekonomista), który określi akceptowalną różnicę δ między wydajnościami obu modeli pozwalającą na zastosowanie modelu transparentnego. Określenie wartości tego parametru jest istotne do oceny i odpowiedzi na pytanie, czy korzyści z modelu "czarnej skrzynki" są wystarczające, by uzasadnić jego zastosowanie, szczególnie w przypadku, gdy transparentny model może dostarczyć zbliżoną wydajność.

Ujemne wartości metryki D wskazują, że model "czarnej skrzynki" nie przynosi korzyści w porównaniu z modelem transparentnym i może nawet być mniej efektywny ze względu na swoją złożoność. W niektórych przypadkach prostszy, bardziej transparentny model może być bardziej preferowany ze względu na jego równą lub nawet lepszą wydajność czy złożoność obliczeniową, ale przede wszystkim tam gdzie wyjaśnialność jest równie ważna jak wydajność [41]. Na rysunku 6.7 przedstawiono schemat blokowy implementacji metryki D, gdzie epsilon jest różnicą wydajności między modelem czarnoskrzynkowym i modelem transparentnym.



Rys. 6.7. Schemat blokowy implementacji metryki D.

W poniższej tabeli 6.1 przedstawiono wyniki kombinacji algorytmów Decision Tree czyli modelu transparentnego z algorytmami „czarnoskrzynkowymi" dla zbiorów danych, przyjmując jako miarę wydajności dokładność modelu. W ostatniej kolumnie znajduje się obliczony wynik metryki $D_{(ACC)}$. Dokładność to tylko jedna z metryk oceny, którą można podstawić do wzoru metryki D, przy czym warto wziąć pod uwagę również inne metryki, takie jak precyzja, czułość, F1-score itp.

Tabela 6.1.: Kombinacja modeli w metryce D - model transparentny(Decision Tree) vs modele czarnoskrzynkowe.

Zbiór	Model (Pb)	Dokładność (Pb)	Dokładność (Pt) Decision Tree	D (Pb - Pt)
SelectDVC	NGBoost	0,942455	0,966549	-0,024094
SelectDVC	Random Forest	0,973792	0,966549	0,007243
SelectDVC	LightGBM	0,974196	0,966549	0,007647
SelectDVC	XGBoost	0,975316	0,966549	0,008767
FeatureWiz	NGBoost	0,93975	0,96515	-0,0254
FeatureWiz	Random Forest	0,973668	0,96515	0,008518
FeatureWiz	LightGBM	0,973481	0,96515	0,008331
FeatureWiz	XGBoost	0,974756	0,96515	0,009606
SelectKBest_chi2	NGBoost	0,868619	0,879065	-0,010446
SelectKBest_chi2	Random Forest	0,914848	0,879065	0,035783
SelectKBest_chi2	LightGBM	0,901169	0,879065	0,022104
SelectKBest_chi2	XGBoost	0,907573	0,879065	0,028508
SelectKBest_f_classif	NGBoost	0,941895	0,961263	-0,019368
SelectKBest_f_classif	Random Forest	0,970279	0,961263	0,009016
SelectKBest_f_classif	LightGBM	0,96888	0,961263	0,007617
SelectKBest_f_classif	XGBoost	0,970963	0,961263	0,0097
RFE	NGBoost	0,9409	0,966517	-0,025617
RFE	Random Forest	0,975658	0,966517	0,009141
RFE	LightGBM	0,974974	0,966517	0,008457
RFE	XGBoost	0,976683	0,966517	0,010166
PCA	NGBoost	0,867935	0,891873	-0,023938
PCA	Random Forest	0,922434	0,891873	0,030561
PCA	LightGBM	0,906454	0,891873	0,014581
PCA	XGBoost	0,913853	0,891873	0,02198

Analizując wyniki metryki $D_{(ACC)}$ przedstawione w tabeli 6.1, możemy zauważyć, że w większości przypadków algorytmy czarnoskrzynkowe, takie jak LightGBM, XGBoost i Random Forest, osiągają lepsze wyniki niż model transparentny, jakim jest Decision Tree na co wskazuje dodatnia wartość metryki D. Z drugiej strony, proponowany model NGBoost wykazuje ujemną wartość metryki D w porównaniu do drzewa decyzyjnego, co stanowi wyjątek.

W przypadku sytuacji wymagających wysokiej efektywności klasyfikacyjnej (np. lepsza identyfikacji zagrożeń i szybsza reakcja na nie), modele czarnoskrzynkowe mogą być preferowane, podczas gdy ważna jest wyjaśnialność wyników, model Decision Tree może okazać się bardziej odpowiedni. Warto jednak pamiętać, że drzewa decyzyjne mają tendencję do overfittingu, co może prowadzić do słabej generalizacji na rzeczywistych danych pochodzących z dynamicznych strumieni sieciowych, w przeciwieństwie chociażby do algorytmu XGBoost, który buduje wiele drzew decyzyjnych sekwencyjnie, gdzie każde kolejne drzewo próbuje korygować błędy poprzednich drzew. Ewentualnie należy rozważyć zastosowanie innych algorytmów transparentnych, takich jak stosunkowo nowy Explainable Boosting Machine (dostępny w pakiecie InterpretML) [140] oraz przeprowadzenie dodatkowych badań porównujących ich efektywność w analizowanych warunkach. Pomocne mogłoby okazać się określenie progu, przy którym stosowanie modelu transparentnego jest uzasadnione. Przykładowo w zadaniu detekcji ataków DDoS w sieciach SDN, próg ten mógłby wynieść 2%, a nie proponowane przez A. Rosenfelda 4%. Oznaczałoby to, że jeśli różnica w wydajności między modelem "czarnej skrzynki" a transparentnym jest mniejsza niż 2%, preferowany powinien być model transparentny. Analizując tabelę 6.2, która przedstawia porównanie modeli z uwzględnieniem progu 2% dla różnicy w dokładności na zbiorach danych SelectDVC i RFE, można zauważyć, że różnice w dokładności klasyfikacji pozostają poniżej tego progu.

Tabela 6.2.:Porównanie modeli z uwzględnieniem progu 2% dla różnicy w dokładności.

Model	**$D_{(ACC)}$ (SelectDVC)**	**$D_{(ACC)}$ (RFE)**
Decision Tree vs LightGBM	0.007647 (≈ 0.79%)	0.008457 (≈ 0.88%)
Decision Tree vs XGBoost	0.008767 (≈ 0.91%)	0.010166 (≈ 1.05%)
Decision Tree vs Random Forest	0.007243 (≈ 0.75%)	0.009141 (≈ 0.95%)
Decision Tree vs NGBoost	-0.025617 (≈ -2.65%)	0.025617 (≈ -2.65%)

Przyjęcie takiego podejścia pozwoliłoby na zachowanie wysokiej dokładności klasyfikacji, jednocześnie zwiększając wyjaśnialność i zrozumiałość modelu. Jednak w większości przypadków eliminowałoby to zastosowanie zespołowych algorytmów XGBoost czy LightGBM, które zazwyczaj osiągają znacznie lepszą skuteczność niż algorytm Decision Tree.

6.6 Analiza wyjaśnialności w oparciu o metrykę D w połączeniu z analizą złożoności obliczeniowej.

Analiza złożoności obliczeniowej przedstawiona w tym podrozdziale ma nieco inny charakter niż tradycyjne rozumienie złożoności obliczeniowej, które skupia się na czasie wykonania algorytmu lub wymaganej pamięci. Złożoność obliczeniowa modelu, zaadaptowana według kryteriów AIC (ang. Akaike Information Criterion) i BIC (ang. Bayesian Information Criterion) odnosi się do sposobu, w jaki te kryteria penalizują modele za ich złożoność. Ma to na celu zapobieganiu nadmiernemu dopasowaniu modeli, co może prowadzić do złych predykcji na nowych danych, ale też i promowaniu najlepszego modelu uczenia maszynowego na podstawie porównania wartości ich kryteriów informacyjnych. Pierwotnie te kryteria stosowane były do estymacji rzędu procesów autoregresji oraz zostały zaadaptowane do oceny efektywności systemów neuronowo-rozmytych [117].

Kryterium informacyjne Akaike uwzględnia ilość danych i złożoność modelu, przy czym preferuje modele, które są dobrze dopasowane do danych i jednocześnie mają niską liczbę parametrów [118]. Bayesowskie kryterium informacje faworyzuje modele o niższej liczbie parametrów, ale jednocześnie dobrze dopasowane do danych [119]. Z tego względu, analiza tych wartości pozwala na obiektywne porównanie modeli transparentnych i czarnoskrzynkowych pod kątem efektywności.

Efektywność działania systemu rozumiemy przez osiągniętą dokładność działania takiego systemu w kontekście jego rozmiaru, zaś przez rozmiar rozumiemy liczbę wszystkich parametrów, które podlegają uczeniu. W tych eksperymentach kryteria AIC i BIC są obliczane na podstawie liczby parametrów (k), liczby obserwacji (n), oraz wartości logarytmu funkcji wiarygodności. Chociaż kryteria AIC i BIC klasycznie stosowane są w problemach regresyjnych, to ich wykorzystanie w zadaniach klasyfikacji jest w znikomym stopniu. Mimo to na podstawie dostępnych publikacji, badacze proponują użycie funkcji takich jak Log Loss, Brier Score, Spherical Score oraz Misclassification loss jako wartości logarytmu funkcji wiarygodności [120]. W przypadku regresji najczęściej stosowanymi funkcjami wiarygodności są średni błąd kwadratowy (MSE) oraz średni błąd bezwzględny (MAE) [121], które w tej dziedzinie wykorzystywane są do oceny dopasowania modelu.

Kryteria AIC i BIC są określone przez następujące wzory [118]:

$$AIC = -2 * ln(L) + 2 * k$$

$$BIC = -2 * ln(L) + k * ln\,(n) \qquad (6.2)$$

gdzie:

k to liczba parametrów w modelu,

n to liczba obserwacji,

ln(L) to wartość funkcji logarytmu naturalnego z funkcji wiarygodności.

Przy czym stosujemy logarytmiczną funkcję straty Log Loss jako funkcję wiarygodności(ang. likelihood).

$$ln(L) = -n * LogLoss$$

Funkcja wiarygodności *L* jest iloczynem prawdopodobieństw dla wszystkich obserwacji w zbiorze danych. Znak minus wynika z konwencji dążenia do minimalizacji funkcji straty w uczeniu maszynowym. W przypadku drzew decyzyjnych stosowanie Log Loss jako funkcji oceny modelu może być pewnym uproszczeniem. Wynika to z faktu, że decyzje w drzewie są deterministyczne tzn. węzły przypisują obserwacje do konkretnych klas bez wyraźnego modelowania niepewności. Można jednak założyć, że prawdopodobieństwa są estymowane na podstawie proporcji klas w liściach drzewa. Niemniej jednak, głębokie drzewa decyzyjne mogą nadmiernie dopasowywać się do danych treningowych. Co więcej, modele probabilistyczne, takie jak boostingowe algorytmy XGBoost czy LightGBM, są bardziej reprezentatywne dla Log Loss, ponieważ uczą się rozkładu prawdopodobieństwa na podstawie iteracyjnej optymalizacji. Tak więc, funkcję logarytmicznej straty Log Loss możemy obliczyć ze wzoru dla naszego problemu klasyfikacji binarnej (0 i 1), który wygląda następująco [122]:

$$LogLoss_i = -[y_i \cdot ln(p_i) + (1 - y_i) \cdot ln(1 - p_i)] \qquad (6.3)$$

gdzie:

y_i to rzeczywista etykieta klasy dla obserwacji i,

p_i to przewidywane prawdopodobieństwo, że obserwacja i należy do klasy 1.

Dla całego zestawu danych, funkcja Log Loss jest zazwyczaj uśredniana, co daje ogólny wzór:

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{n} LogLoss_i \quad (6.4)$$

co można zapisać jako

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{n} -[y_i \cdot ln(p_i) + (1 - y_i) \cdot ln(1 - p_i)] \ (6.5)$$

Użycie Log Loss w tym problemie, pozwala określić odpowiedni balans między jakością dopasowania przewidywanych prawdopodobieństw do rzeczywistych klas, a złożonością modelu. Otrzymane wyniki AIC i BIC w praktyce mogą być przydatne tylko wtedy, gdy je używany do porównywania dwóch modeli. Załóżmy, że chcemy porównać złożoność obliczeniową modelu czarnoskrzynkowego z modelem transparentnym. Im mniejsze wartości AIC i BIC, tym model jest lepszy, ponieważ oznacza to, że model jest dobrze dopasowany, ale nie jest zbyt skomplikowany. W związku z powyższym, można spróbować zaadoptować wzór określający metrykę D *(6.1)*, zastępując wydajność wynikami kryteriów złożoności obliczeniowej AIC oraz BIC z uwzględnieniem porównawczym modeli transparentnych(Pt) i czarnoskrzynkowych(Pb).

$$D_{(AIC)} = Pb_{(AIC)} - Pt_{(AIC)}$$
$$D_{(BIC)} = Pb_{(BIC)} - Pt_{(BIC)} \quad (6.7)$$

Przy czym, jeśli wartość AIC oraz BIC modelu czarnoskrzynkowego jest mniejsza od wartości modelu transparentnego wyniki końcowe metryk $D_{(AIC)}$ oraz $D_{(BIC)}$ będą ujemne, co wskazuje na wybór modelu czarnoskrzynkowego, kosztem transparentnego. Wyniki przedstawione w tabeli 6.2 oraz 6.3 pozwolą odpowiedzieć na pytania: czy modyfikacja metryki D z wykorzystaniem kryteriów informacyjnych Akaiki - $D_{(AIC)}$ oraz Bayesa - $D_{(BIC)}$ w celu zastąpienia wydajności modelu przez złożoność obliczeniową premiuje modele czarnoskrzynkowe, w tym NGBoost?

Tabela 6.3.:Wyniki wydajności oraz złożoności obliczeniowej dla klasyfikatorów i zbiorów cech.

Zbiór	Klasyfikator	Dokładność	Precyzja	Czułość	F1	Log Loss	AIC	BIC
SelectDVC	NGBoost	0,942455	0,943124	0,942454766	0,942386656	0,125363	8102,869	8262,063
SelectDVC	Random Forest	0,973792	0,97382	0,973792203	0,973787484	0,094291	6103,96	6263,155
SelectDVC	Decision Tree	0,966549	0,966549	0,96654853	0,966546541	1,140466	73406,46	73565,66
SelectDVC	LightGBM	0,974196	0,974272	0,974196356	0,974188672	0,064058	4158,973	4318,168
SelectDVC	XGBoost	0,975316	0,975373	0,975315551	0,975309264	0,061386	3987,075	4146,27
FeatureWiz	NGBoost	0,93975	0,940057	0,939750047	0,939705318	0,141688	9153,065	9312,26
FeatureWiz	Random Forest	0,973668	0,973696	0,973667848	0,973663107	0,095788	6200,238	6359,432
FeatureWiz	Decision Tree	0,96515	0,965149	0,965149537	0,965149176	1,191518	76690,73	76849,92
FeatureWiz	LightGBM	0,973481	0,973578	0,973481316	0,973472277	0,063695	4135,651	4294,845
FeatureWiz	XGBoost	0,974756	0,974819	0,974755953	0,97474914	0,061263	3979,19	4138,385
SelectKBest_chi2	NGBoost	0,868619	0,87554	0,868619039	0,867609159	0,310719	20027,16	20186,36
SelectKBest_chi2	Random Forest	0,914848	0,914916	0,914847976	0,914815024	0,273584	17638,21	17797,41
SelectKBest_chi2	Decision Tree	0,879065	0,879077	0,879064851	0,879069663	4,261278	274174,6	274333,8
SelectKBest_chi2	LightGBM	0,901169	0,901583	0,901168936	0,901068315	0,241625	15582,25	15741,45
SelectKBest_chi2	XGBoost	0,907573	0,90767	0,907573214	0,90752981	0,226603	14615,81	14775
SelectKBest_f_classif	NGBoost	0,941895	0,942479	0,941895169	0,941831959	0,127697	8253,002	8412,197
SelectKBest_f_classif	Random Forest	0,970279	0,970301	0,970279177	0,970274284	0,097496	6310,109	6469,303
SelectKBest_f_classif	Decision Tree	0,961263	0,961263	0,961263446	0,96126143	1,333215	85806,41	85965,6
SelectKBest_f_classif	LightGBM	0,96888	0,968974	0,968880184	0,968869577	0,072602	4708,646	4867,841
SelectKBest_f_classif	XGBoost	0,970963	0,971018	0,970963129	0,970955733	0,068449	4441,444	4600,639
RFE	NGBoost	0,9409	0,942365	0,94090033	0,940782036	0,126837	8197,699	8356,894
RFE	Random Forest	0,975658	0,97567	0,975657527	0,975654548	0,077892	5048,974	5208,168
RFE	Decision Tree	0,966517	0,966531	0,966517441	0,966519853	1,172348	75457,46	75616,66
RFE	LightGBM	0,974974	0,97509	0,974973575	0,974964224	0,060822	3950,785	4109,979
RFE	XGBoost	0,976683	0,976766	0,976683455	0,97667629	0,056925	3700,13	3859,325
PCA	NGBoost	0,867935	0,875142	0,867935087	0,866885312	0,310655	20023,04	20182,24
PCA	Random Forest	0,922434	0,922626	0,922433626	0,922386042	0,207694	13399,34	13558,53
PCA	Decision Tree	0,891873	0,891893	0,891873407	0,891880204	3,859825	248348,3	248507,5
PCA	LightGBM	0,906454	0,906992	0,90645402	0,906345115	0,217922	14057,36	14216,55
PCA	XGBoost	0,913853	0,914122	0,913853137	0,913787589	0,196876	12703,45	12862,65

Tabela 6.4.: Kombinacja modeli w metryce D z wykorzystaniem kryteriów informacyjnych AIC oraz BIC - model transparentny(Decision Tree) vs modele czarnoskrzynkowe.

Zbiór	Kombinacja modeli w metryce D Model transparentny vs Model czarnoskrzynkowy	Metryka $D_{(ACC)}$= Pb - Pt	Metryka $D_{(BIC)}$ Pb - Pt
SelectDVC	Decision Tree vs NGBoost	-65303,59	-65303,60
SelectDVC	Decision Tree vs LightGBM	-69247,49	-69247,49
SelectDVC	Decision Tree vs XGBoost	-69419,39	-69419,39
SelectDVC	Decision Tree vs Random Forest	-67302,50	-67302,51
Featurewiz	Decision Tree vs NGBoost	-67537,67	-67537,66
Featurewiz	Decision Tree vs LightGBM	-72555,08	-72555,08
Featurewiz	Decision Tree vs XGBoost	-72711,54	-72711,54
Featurewiz	Decision Tree vs Random Forest	-70490,49	-70490,49
Selectkbest_f_classif	Decision Tree vs NGBoost	-65779,25	-65779,24
Selectkbest_f_classif	Decision Tree vs LightGBM	-70224,16	-70224,15
Selectkbest_f_classif	Decision Tree vs XGBoost	-81364,97	-81364,96
Selectkbest_f_classif	Decision Tree vs Random Forest	-79496,30	-79496,30
Selectkbest_chi2	Decision Tree vs NGBoost	-254147,44	-254147,44
Selectkbest_chi2	Decision Tree vs LightGBM	-258592,35	-258592,35
Selectkbest_chi2	Decision Tree vs XGBoost	-259558,79	-259558,80
Selectkbest_chi2	Decision Tree vs Random Forest	-256536,39	-256536,39
RFE	Decision Tree vs NGBoost	-67259,76	-67259,77
RFE	Decision Tree vs LightGBM	-71506,68	-71506,68
RFE	Decision Tree vs XGBoost	-71757,33	-71757,34
RFE	Decision Tree vs Random Forest	-70408,49	-70408,49
PCA	Decision Tree vs NGBoost	-228325,26	-228325,26
PCA	Decision Tree vs LightGBM	-234290,94	-234290,95
PCA	Decision Tree vs XGBoost	-235644,85	-235644,85
PCA	Decision Tree vs Random Forest	-234948,96	-234948,97

Zgodnie z tabelą 6.3 można zauważyć, iż na przykład dla algorytmu NGBoost, wartości AIC są wyraźnie wyższe w porównaniu do innych klasyfikatorów, podczas gdy wartości BIC wykazują bardziej zróżnicowane wyniki, co sugeruje możliwość preferencji modelu w zależności od zastosowanego kryterium. W tabeli również widać, że modele boostingowe (XGBoost, LightGBM, NGBoost) mają niższe wartości Log Loss niż drzewa decyzyjne i w większości przypadków również niż Random Forest. Warto też zwrócić uwagę na przypadek NGBoost, który jako model probabilistyczny powinien dobrze radzić sobie z Log Loss. Jednak w niektórych przypadkach (np. dla zbioru SelectKBest_chi2 Log Loss = 0,310719) ma on wyraźnie wyższe wartości Log Loss niż XGBoost czy LightGBM. Tabela 6.4 obrazuje wyniki metryki D w przypadku porównania modeli czarnoskrzynkowych (NGBoost, Random Forest,

XGBoost, LightGBM) z modelem transparentnym (Decision Tree). Wartości $D_{(AIC)}$ oraz $D_{(BIC)}$ są wyraźnie ujemne co wskazuje na preferencję modeli czarnoskrzynkowych.

W przypadku tej modyfikacji metryki D należałoby również określić próg różnicy, aby ostatecznie podjąć decyzję o wyborze odpowiedniego modelu. Przykładowo, także w tym przypadku próg 2% może być używany do określenia, czy model czarnoskrzynkowy jest wystarczająco lepszy, aby uzasadnić jego zastosowanie zamiast modelu transparentnego.

Mimo że implementacje kryteriów AIC oraz BIC są rzadziej spotykane w literaturze dla problemów klasyfikacyjnych, ich zastosowanie może przynieść wartościowe informacje na temat optymalnego balansu między dokładnością a złożonością modelu. Eksperymenty ukazały istotne różnice w preferencjach modelowych, przy czym kryterium informacyjne Akaike jest bardziej liberalne w preferowaniu modeli czarnoskrzynkowych, co może być korzystne w przypadku osiągania wyższej dokładności przy umiarkowanej złożoności, zaś kryterium informacyjne Bayesa preferuje modele prostsze ze względu na surowszą karę za złożoność, co jest zasadne w systemach wymagających wysokiej wyjaśnialności, nawet kosztem pewnej utraty dokładności. Warto również zauważyć, że różnice między modelami mogą zależeć od zastosowanej metody selekcji cech, co widać w różnicach wyników dla wybranych zbiorów danych. Przeprowadzenie tego badania uwidoczniło również inny aspekt – potrzebę eksperymentowania z liczbą użytych w modelu cech. Zastosowanie Log Loss jako metryki w analizie modeli może prowadzić do preferowania boostingowych metod, ponieważ lepiej reprezentują rozkład prawdopodobieństwa klas. Ze względu na to, że liczba cech w modelu była stała i wynosiła 19, wynik końcowy opierał się na obliczonej wartości logarytmu naturalnego funkcji wiarygodności. Zatem wybór optymalnego modelu powinien uwzględniać zmiany w liczbie użytych cech oraz związane z nimi wartości Log Loss, aby znaleźć równowagę pomiędzy dokładnością, przejrzystością oraz efektywnością obliczeniową, co może pozwolić na bardziej wiarygodną ocenę zarówno skuteczności, jak i wyjaśnialności systemów bezpieczeństwa w sieciach SDN.

7. Wyjaśnialność klasyfikacji ataków DDoS w oparciu o analizę wpływu cech z wykorzystaniem metody SHAP.

Brak przejrzystości modeli uczenia maszynowego w procesie detekcji ataków DDoS stanowi wyzwanie dla ich skutecznego wdrożenia. Nawet wysoko wydajne algorytmy, takie jak XGBoost czy LightGBM, mogą być poddawane krytyce ze względu na ich nieprzejrzysty charakter. Zaproponowana metryka D może okazać się pomocna, jednak sama w sobie nie rozwiązuje problemu kompromisu między dokładnością a przejrzystością modelu. Wśród algorytmów porównawczych analizowanych w niniejszej rozprawie doktorskiej, jedynie drzewa decyzyjne oferują przejrzystość, lecz nie dorównują wydajnościom wymienionym modelom. W celu rozwiązania problemu braku wyjaśnialności modeli czarnoskrzynkowych, w literaturze naukowej proponuje się użycie metody SHAP (SHapley Additive exPlanations).

W niniejszej rozprawie doktorskiej zaadoptowano właśnie tą metodę, która bazuje na wartościach Shapleya z teorii gier i umożliwia głębokie zrozumienie wpływu poszczególnych cech na prognozy modelu uczenia maszynowego. Niezależnie od rodzaju modelu, SHAP analizuje i wyjaśnia globalny wpływ każdej z cech na przewidywania [68].

Dodatkowo, zaproponowano zastosowanie metody SHAP w połączeniu z metryką F - Feature, opracowaną przez A. Rosenfelda [41]. Zakłada się, że skupienie się na mniejszej liczbie cech, zarówno o pozytywnym, jak i negatywnym wpływie na predykcje modelu, przyczyni się do zwiększenia jego przejrzystości i ułatwi interpretację wyników klasyfikacji. Takie podejście pozwala użytkownikowi na lepsze zrozumienie, jak poszczególne cechy wpływają na wynik klasyfikacji, ułatwiając tym samym interpretację i potencjalne dostosowanie modelu. Jednakże, podczas badań skupiono się nie tylko na wartościach ważności cech (ang. Feature Importance), ale na konkretnej identyfikacji cech, które mają zarówno pozytywny, jak i negatywny wpływ na prognozy modelu. Proces ten odbywa się poprzez przypisywanie wartości wpływu poszczególnym cechom, przez dokładne prześledzenie, jak każda z nich przyczynia się do końcowej decyzji klasyfikacyjnej. W związku z tym postawiono następujące pytania badawcze:

- Jak stopniowa redukcja liczby cech, określonych za pomocą analizy SHAP wpływa na kluczowe metryki oceny modelu, takie jak dokładność, precyzja, czułość oraz miara F1?
- Czy istnieje optymalna liczba cech, która zapewnia równowagę między wysoką dokładnością a kompaktowym wyjaśnieniem modelu, zgodnie z metryką F - Feature?

- Czy różnice w liczebności i rodzaju cech pozytywnych i negatywnych między różnymi klasyfikatorami mają zauważalny wpływ na efektywność i wiarygodność predykcji w kontekście ataków DDoS?
- Jakie cechy są najbardziej wpływowe w kontekście wykrywania ataków DDoS według analizy SHAP i w jaki sposób te cechy różnią się między modelami stosującymi różne techniki selekcji cech?

7.1 Analiza wyjaśnialności modeli w oparciu o metodę SHAP.

Metoda SHAP w uczeniu maszynowym polega na obliczeniu wartości Shapleya dla każdego obiektu w przestrzeni wejściowej. Wartość Shapleya dla funkcji i, oznaczona przez φ_i definiuje się jako średni udział cechy i we wszystkich możliwych koalicjach cech. Matematycznie wartość Shapleya można wyrazić w następujący sposób:

$$\phi_i(v) = \sum_{S \subseteq N\{i\}} \frac{|S|!(n-|S|-1)!}{n!} [v(S \cup \{i\}) - v(S)] \quad (7.1)$$

gdzie:

- *$\phi i(v)$ to wartość SHAP dla cechy i*
- *N to zbiór wszystkich cech*
- *S to podzbiór cech bez cechy i*
- *v to funkcja wartości, która mapuje podzbiory cech na przewidywane wartości modelu.*

Wzór ten oblicza średni marginalny wkład cechy i do wszystkich możliwych kombinacji pozostałych cech. Wartość SHAP dla danej cechy reprezentuje jej wpływ na predykcję modelu. Dodatnia wartość SHAP oznacza, że cecha zwiększa przewidywaną wartość, podczas gdy ujemna wartość SHAP wskazuje na zmniejszenie przewidywanej wartości [36]. Właściwości wartości SHAP, które sprawiają, że są one skuteczne w interpretowaniu modeli to addytywność, lokalna dokładność, konsystencja oraz odporność na brakujące dane. Wartości SHAP są addytywne, co oznacza, że udział każdej funkcji w ostatecznym przewidywaniu można obliczyć niezależnie, a następnie zsumować. Wartości SHAP także sumują się do różnicy między oczekiwanymi danymi wyjściowymi modelu a rzeczywistymi danymi wyjściowymi dla danych wejściowych [36].

W celu wyjaśnienia mechanizmów stojących za predykcjami modeli detekcji ataków DDoS zastosowano technikę TreeSHAP z biblioteki SHAP, zaprojektowaną specjalnie do

analizy modeli drzewiastych w tym wybranych do analizy w niniejszej rozprawie doktorskiej algorytmów Decision Tree, Random Forest, NGBoost, XGBoost, LightGBM [124]. TreeSHAP wyróżnia się na tle ogólnego algorytmu SHAP znacznie wyższą wydajnością obliczeniową poprzez inteligentne wykorzystanie struktury drzewa decyzyjnego. Algorytm działa rekurencyjnie, przechodząc od korzenia drzewa do jego liści, i w trakcie tego procesu przypisuje wkłady poszczególnym cechom.

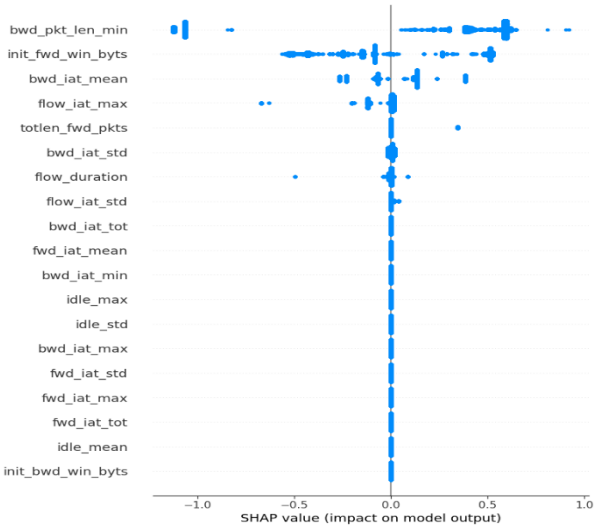
Poniżej zaprezentowano wyniki analizy SHAP dla trzech algorytmów NGBoost, XGBoost oraz LightGBM, zastosowanych do zbioru cech SelectDVC. Wyniki te przedstawiono w postaci diagramów(rys. 7.1) SHAP Summary Plot (po lewej stronie) oraz SHAP Feature Importance (po prawej stronie).

Diagramy SHAP Summary Plot ilustrują wpływ poszczególnych cech na predykcje modelu. Wartości SHAP na osi poziomej określają, jak każda cecha przyczynia się do przewidywanego wyniku, przy czym wartości dodatnie wskazują na zwiększenie przewidywanej wartości, a wartości ujemne na jej zmniejszenie. Cechy są uszeregowane na osi pionowej według malejącego wpływu na wynik modelu.

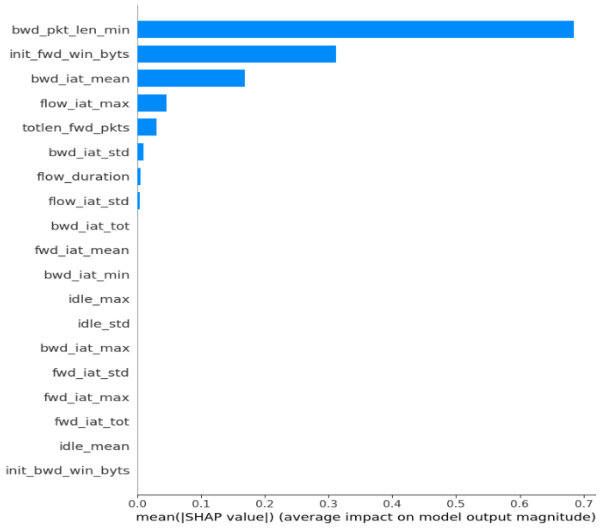
Diagramy SHAP Feature Importance przedstawiają średnie wartości SHAP dla każdej cechy, co pozwala na ocenę ogólnego znaczenia danej cechy dla modelu. Cechy o dużych wartościach bezwzględnych Shapleya są uznawane za ważne. Aby uzyskać globalną ważność, uśredniamy wartości bezwzględne Shapleya dla każdej cechy w całym zbiorze danych:

$$I_j = \frac{1}{n}\sum_{i=n}^{n} |\emptyset_j^{(i)}| \quad (7.2)$$

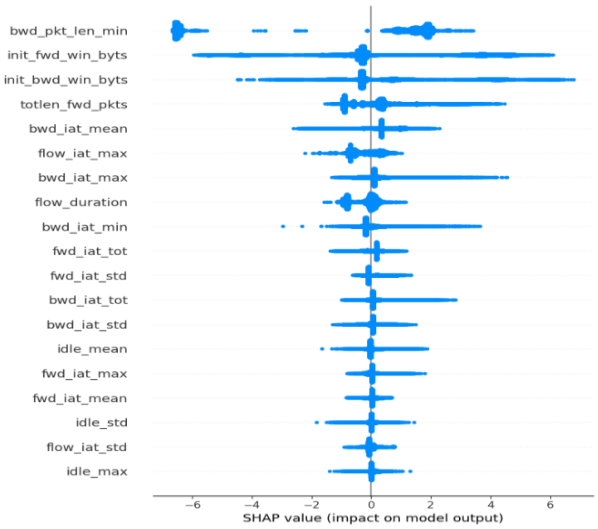
Następnie sortujemy cechy według malejącej ważności i przedstawiamy je na wykresie.



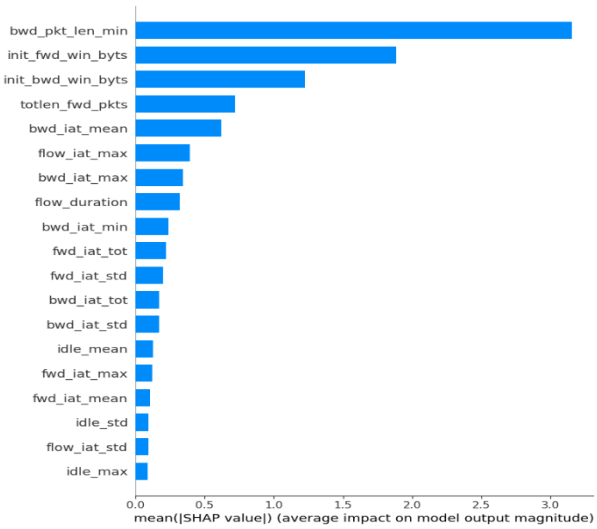
a) SHAP Summary Plot dla modelu NGBoost



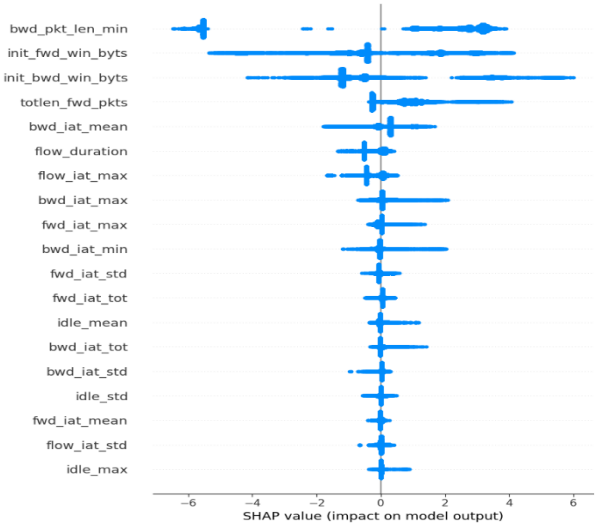
b) SHAP Feature Importance dla modelu NGBoost



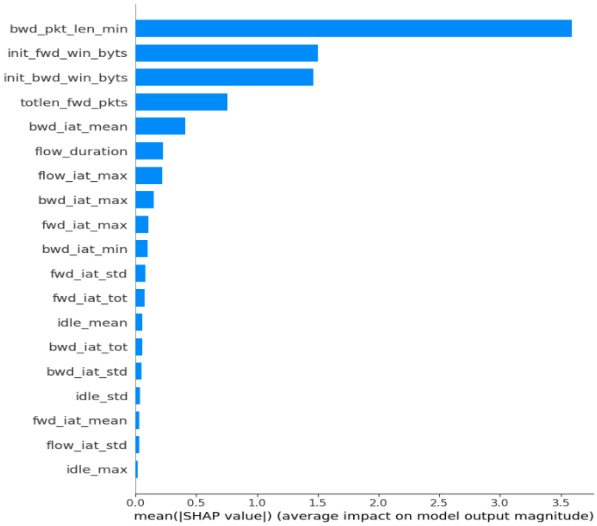
c) SHAP Summary Plot dla modelu XGBoost



d) SHAP Feature Importance dla modelu XGBoost



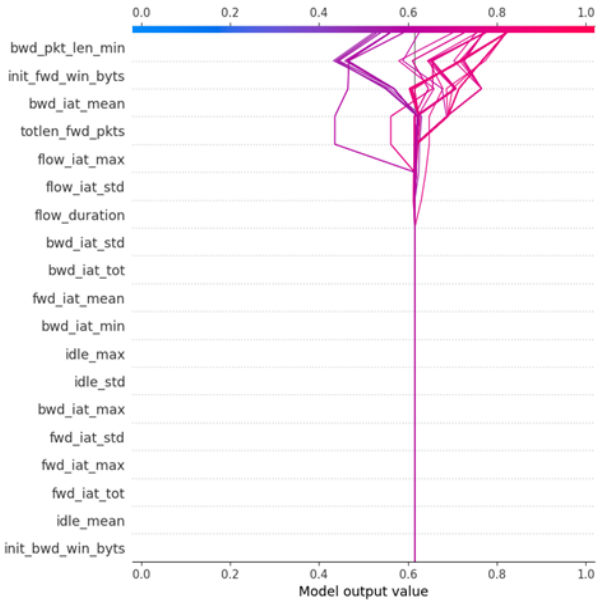
e) SHAP Sumary Plot dla modelu LightGBM



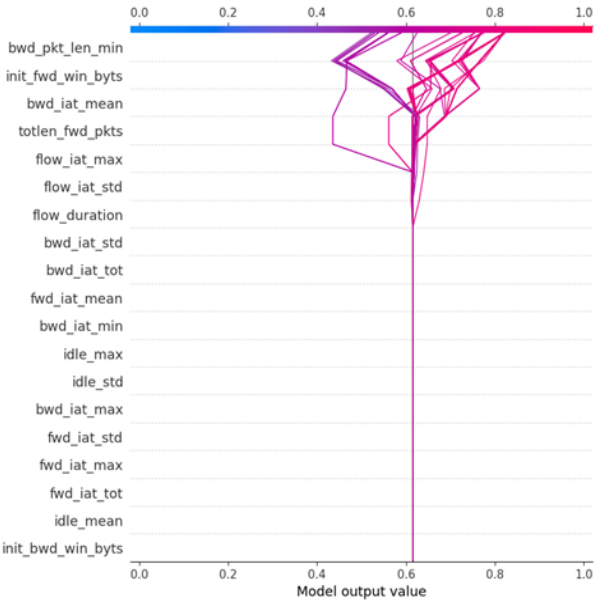
f) SHAP Feature Importance dla modelu LightGBM

Rys. 7.1. Wyniki Summary Plot oraz SHAP Feature Importance trzech modeli NGBoost, XGBoost oraz LightGBM dla zbioru SelectDVC.

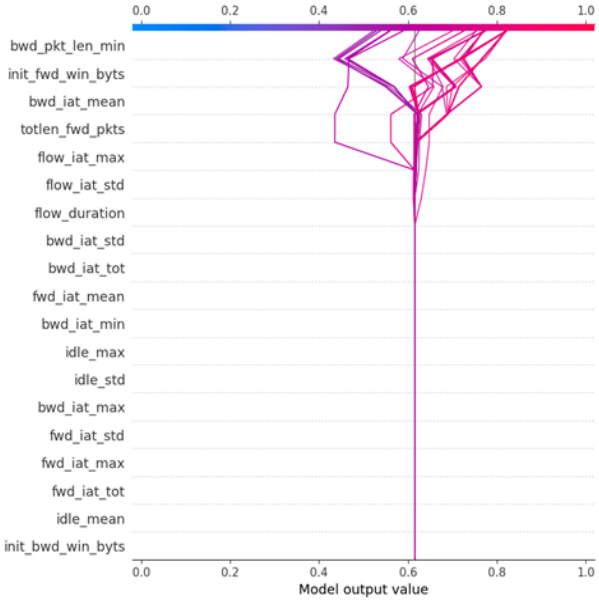
Analiza diagramów SHAP przedstawiona na rysunku 7.1 wskazuje, że kluczową rolę w identyfikacji ataków DDoS odgrywają cechy związane z manipulacją początkowymi rozmiarami okien TCP, takie jak init_fwd_win_byts oraz init_bwd_win_byts, przez co mogą stanowić charakterystyczny wskaźnik ruchu generowanego przez atakujących. Wzrost liczby małych pakietów w stosunku do większych, także może sugerować atak na co wskazuje cecha bwd_pkt_len_min. Z kolei cechy związane z czasem aktywności i bezczynności, w tym fwd_iat_mean (średni czas między kolejnymi pakietami wysyłanymi od klienta do serwera) oraz idle_std (standardowe odchylenie czasu bezczynności), wykazują znacznie mniejszy wpływ na predykcję modeli. Może to wynikać z faktu, że ataki DDoS często imitują normalny ruch sieciowy, co ogranicza zdolność tych parametrów do skutecznego różnicowania między ruchem atakującym a legalnym.



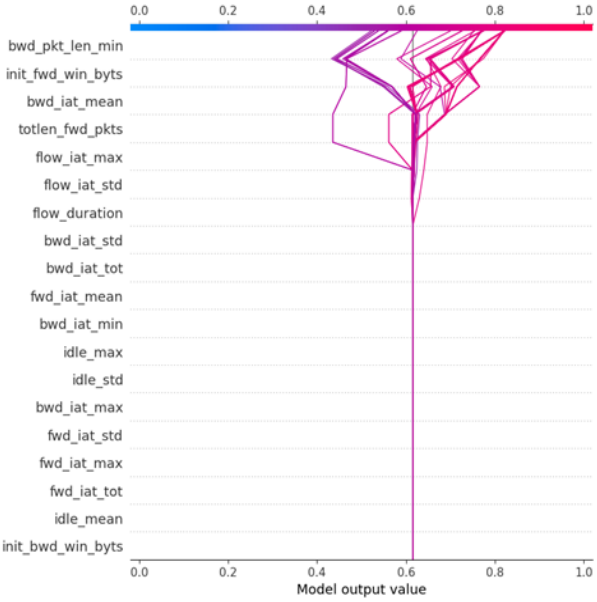
a) SHAP Decision Plot dla modelu NGBoost – predykcja etykiet 1



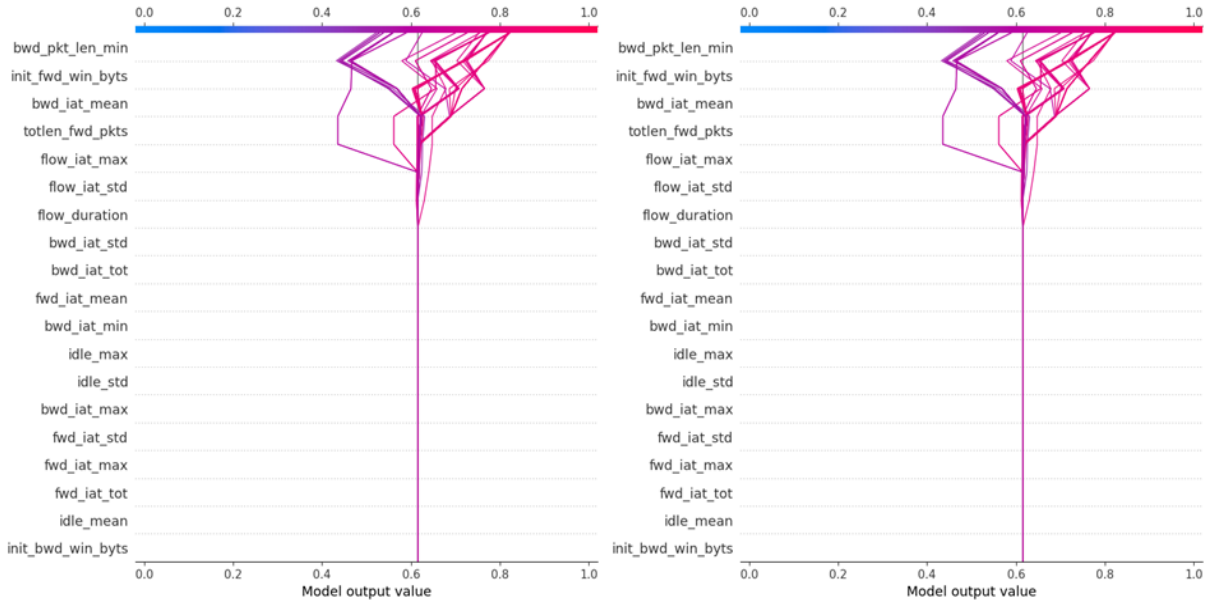
b) SHAP Decision Plot dla modelu NGBoost – predykcja etykiet 1



c) SHAP Decision Plot dla modelu XGBoost – predykcja etykiet 1



d) SHAP Decision Plot dla modelu XGBoost – predykcja etykiet 0



e) SHAP Decision Plot dla modelu LightGBM – predykcja etykiet 1

f) SHAP Decision Plot dla modelu LightGBM – predykcja etykiet 0

Rys. 7.2. Wyniki Decision Plot trzech algorytmów XGBoost, LightGBM oraz NGBoost dla zbioru SelectDVC i predykcji etykiet 1 (po lewej) i etykiet 0 (po prawej).

Interesującą obserwacją jest również wpływ cech takich jak flow_duration (całkowity czas trwania przepływu danych), fwd_iat_std (standardowe odchylenie czasu między pakietami wysyłanymi od klienta do serwera), bwd_iat_max (maksymalny czas między pakietami przesyłanymi od serwera do klienta) oraz fwd_iat_tot (całkowity czas między pakietami wysyłanymi od klienta do serwera). Ich wpływ na modele jest zróżnicowany i kontekstowy – w niektórych scenariuszach ataków DDoS mogą mieć one różny wpływ na te parametry.

Inną formą wizualizacji, która jest pomocna w wyjaśnianiu podejmowania decyzji przez modele jest SHAP Decision Plot (rysunek 7.2), który przedstawia zależności między wartościami cech a wartością wyjściową modelu predykcyjnego. Każda linia łączy wartość konkretnej cechy (na lewym końcu osi poziomej) z wynikiem modelu (na prawym końcu osi poziomej). Kolor linii oznacza siłę lub kierunek wpływu. Z lewej strony przedstawiono wyniki dla predykcji etykiet 1, a z prawej strony predykcji etykiet 0 [125]. Co ciekawe, pomimo ogólnych podobieństw w trendach, algorytmy różnią się w priorytecie cech, co wynika z odmiennych strategii uczenia i optymalizacji.

7.2 Analiza wyjaśnialności w oparciu o metrykę F.

Rosenfeld w swojej pracy [41] proponuje zastosować metrykę opartą na liczbie cech wejściowych używanych przez agenta do generowania wyjaśnienia. Metryka ta została zaprojektowana specjalnie dla wyjaśnień opartych na analizie cech. Zakłada się, że im mniejsza liczba cech, na których użytkownik musi się skupić, tym wyższa jest wyjaśnialność modelu, co sprawia, że model oparty o XAI jest bardziej przejrzysty. Metrykę tą można przedstawić za pomocą wzoru 7.3.

$$F = \lambda \times (n - c)^{+} \qquad (7.3)$$

gdzie:

F - metryka F - Feature,

λ - parametr określający wpływ liczby cech na wynik, w tym przypadku λ=0,05,

n - liczba cech wejściowych użytych do stworzenia wyjaśnienia

c - próg, powyżej którego zastosowanie kary jest możliwe(c=7),

(n−c)+ - funkcja zdefiniowana jako max(0,n−c)

Jako wartość n określa się sumaryczną wartość liczby cech pozytywnych oraz negatywnych obliczonych dla modelu przy pomocy metody SHAP. Jako parametr λ przyjmuje się wartość 0,05. Jest to przykładowa wartość, która może być stosowana, choć Rosenfeld nie określił konkretnej wartości w swojej publikacji. Warto zauważyć, że kara jest nakładana tylko wtedy, gdy różnica między liczbą cech rozważanych, a liczbą cech intepretowanych jako próg jest większe od zera. Jest to zasadne, gdyż kontrolowanie liczby cech wejściowych użytych w procesie wyjaśniania może wpłynąć na wyjaśnialność modelu. Natomiast wartość 7 przyjęto na potrzeby dalszych badań, jako potencjalną wartość progu c, powyżej którego metryka zaczyna rosnąć, co jest zgodne z propozycją Rosenfelda.

Przy zastosowaniu powyższego wzoru (7.3), wartości metryki F- Feature powinny być obliczane dla każdej liczby cech n zgodnie z następującymi krokami:

1. Obliczenie n−c.
2. Zastosowanie funkcji maksymalnej, aby uzyskać (n−c)+.
3. Pomnożenie wyniku przez λ.

W niniejszej rozprawie doktorskiej zaproponowano autorski algorytm usuwania cech (algorytm 2) w oparciu o analizę SHAP w celu znalezienia subiektywnego kompromisu pomiędzy liczbą cech, a wydajnością modeli, która jest zależna od tego, co jest dla nas priorytetowe.

Algorytm 2: Eliminacja cech na podstawie analizy SHAP

Wejście: Wczytaj model z wybranym zbiorem cech(lista *selected_features*)

Procedura:

1. Wykonaj analizę SHAP na modelu z użyciem danych testowych oraz uzyskaj ważność cech.
2. Zainicjalizuj listę *results* do przechowywania wyników iteracji.
3. Dopóki zbiór cech zawiera więcej niż jedną cechę:
    1. Zaktualizuj ranking cech na podstawie ich ważności uzyskanej z analizy SHAP.
    2. Usuń cechę z najniższą ważnością z listy selected_features.
    3. Utwórz nowy zbiór danych *X_selected* zawierający tylko ostatnie cechy z listy *selected_features.*
    4. Podziel zbiór danych *X_selected* na zbiór treningowy i testowy.
    5. Ponownie wytrenuj model na zaktualizowanym zbiorze treningowym.
    6. Wykonaj predykcje na zbiorze testowym i oblicz metryki wydajności i złożoności obliczeniowej
    7. Wykonaj ponownie analizę SHAP i zaktualizuj ranking ważności cech.
    8. Zapisz aktualne wyniki do listy wyników *results*.
4. Wypisz ostateczne wyniki umieszczone w liście *results*.

Koniec Procedury

Wynik: Lista wyników *results* zawierająca metryki wydajności i złożoności obliczeniowej oraz informacje o usuniętych cechach i aktualnej liczbie cech.

W celu uniknięcia losowego wyboru cech w badaniu, w algorytmie zaproponowano iteracyjne zmniejszanie liczby cech o 1, przy czym eliminuje się cechę o najmniejszej ważności, określanej na podstawie średniej wartości bezwzględnej wpływu SHAP dla danej cechy na losowo wybranych próbkach danych. Po usunięciu cechy, zbiór danych jest aktualizowany, a model jest ponownie trenowany i oceniany na zbiorze testowym. Proces ten jest powtarzany iteracyjnie, aż do momentu, gdy pozostanie tylko jedna cecha.

W tabeli 7.1 przedstawiono wyniki metryki F - Feature dla modelu XGBoost i zbioru SelectDVC. W ostatniej kolumnie tabeli przedstawiona została dokładność modelu osiągnięta przez oceniany model dla liczby cech wejściowych od 1 do 19.

Tabela 7.1.: Wyniki metryki F-feature oraz dokładności dla modelu XGBoost na zbiorze SelectDVC.

Klasyfikator	**Liczba cech wejściowych n**	**λ**	**Próg -c**	**F- Feature**	**Dokładność**
XGBoost	19	0,05	7	0,6	0,975316
XGBoost	18	0,05	7	0,55	0,976
XGBoost	17	0,05	7	0,5	0,974974
XGBoost	16	0,05	7	0,45	0,975191
XGBoost	15	0,05	7	0,4	0,975316
XGBoost	14	0,05	7	0,35	0,974507
XGBoost	13	0,05	7	0,3	0,974476
XGBoost	12	0,05	7	0,25	0,970528
XGBoost	11	0,05	7	0,2	0,970404
XGBoost	10	0,05	7	0,15	0,969906
XGBoost	9	0,05	7	0,1	0,969813
XGBoost	8	0,05	7	0,05	0,968849
XGBoost	7	0,05	7	0	0,967077
XGBoost	6	0,05	7	0	0,964186
XGBoost	5	0,05	7	0	0,959118
XGBoost	4	0,05	7	0	0,94746
XGBoost	3	0,05	7	0	0,933439
XGBoost	2	0,05	7	0	0,927625
XGBoost	1	0,05	7	0	0,84465

Z analizy wyników przedstawionych w tabeli 7.1 wynika, że możliwe jest znalezienie optymalnego kompromisu między liczbą cech a dokładnością modelu XGBoost wykorzystując analizę SHAP. Dokładność modelu pozostaje na wysokim poziomie, nawet przy redukcji liczby cech wejściowych. Na przykład, dla 19 cech wejściowych, dokładność modelu wynosi 0,975316, a już przy 18 cechach dokładność wzrasta do 0,976, co pokazuje minimalną różnicę. Zauważalna zmiana następuje dopiero przy 13 cechach, gdzie dokładność spada do 0,974476. Warto jednak zauważyć, że różnice w dokładności między 13 a 19 cechami są niewielkie, co

sugeruje, że redukcja liczby cech o około 30% (z 19 do 13) może być korzystnym rozwiązaniem. Przy takim zmniejszeniu liczby cech uzyskujemy tylko nieznaczny spadek dokładności, osiągając wartość 0,974476 w porównaniu do najwyższej dokładności 0,976 dla 18 cech. W przypadku dalszej redukcji liczby cech, dokładność modelu zaczyna znacząco spadać. Przy 7 cechach dokładność wynosi tylko 0,967077, a przy 6 cechach jest jeszcze niższa, osiągając wartość 0,964186.

Kontrolowanie liczby cech wejściowych przez ich ważność może wpłynąć na wyjaśnialność modelu. Przy czym punkt, gdzie osiągamy zadowalający kompromis pomiędzy liczbą cech a wydajnością modelu jak widzimy, może być dość subiektywny, pomimo sugestii, iż możliwe jest uproszczenie modelu bez istotnej utraty jego efektywności. W związku z tym analizę rozszerzono o dodatkowe kryteria, takie jak kryterium informacyjne Akaike (AIC) czy Bayesowskie kryterium informacyjne (BIC). Wyniki przedstawiono w tabelach 7.2 oraz 7.3.

Tabela 7.2.: Rozszerzenie metryki F - Features oraz wyniki wydajności i kryteria złożoności obliczeniowej AIC oraz BIC dla modelu XGBoost na zbiorze SelectDVC.

				Zbiór SelectDVC				
Klasyfikator	**Dokładność**	**Precyzja**	**Czułość**	**F1**	**AIC**	**BIC**	**Liczba cech wejściowych n**	**Usunięta cecha po iteracji**
XGBoost	0,975316	0,975373	0,975316	0,975309	3987,075	4146,27	19	idle_max
XGBoost	0,976	0,976057	0,976	0,975993	3940,841	4091,657	18	fwd_iat_mean
XGBoost	0,974974	0,975042	0,974974	0,974967	3970,405	4112,842	17	idle_std
XGBoost	0,975191	0,975256	0,975191	0,975184	3947,19	4081,249	16	idle_mean
XGBoost	0,975316	0,975381	0,975316	0,975309	3973,29	4098,969	15	flow_iat_std
XGBoost	0,974507	0,974573	0,974507	0,9745	3953,84	4071,142	14	fwd_iat_max
XGBoost	0,974476	0,974555	0,974476	0,974468	4001,133	4110,056	13	bwd_iat_min
XGBoost	0,970528	0,970586	0,970528	0,97052	4522,2	4622,744	12	fwd_iat_std
XGBoost	0,970404	0,970439	0,970404	0,970397	4662,926	4755,091	11	fwd_iat_tot
XGBoost	0,969906	0,969953	0,969906	0,969899	4633,268	4717,055	10	bwd_iat_std
XGBoost	0,969813	0,96987	0,969813	0,969805	4698,325	4773,733	9	bwd_iat_tot
XGBoost	0,968849	0,968919	0,968849	0,96884	4759,028	4826,057	8	bwd_iat_max
XGBoost	0,967077	0,967113	0,967077	0,96707	4932,818	4991,468	7	flow_duration
XGBoost	0,964186	0,964304	0,964186	0,964172	5151,61	5201,882	6	bwd_iat_mean
XGBoost	0,959118	0,959174	0,959118	0,959107	5520,315	5562,208	5	flow_iat_max
XGBoost	0,94746	0,94934	0,94746	0,947337	6482,846	6516,361	4	init_bwd_win_byts
XGBoost	0,933439	0,93349	0,933439	0,933419	7882,131	7907,267	3	totlen_fwd_pkts
XGBoost	0,927625	0,931564	0,927625	0,927318	9556,748	9573,505	2	init_fwd_win_byts
XGBoost	0,84465	0,880244	0,84465	0,839865	22422,49	22430,87	1	bwd_pkt_len_min

Tabela 7.3.: Rozszerzenie metryki F - Feature oraz wyniki wydajności i kryteria złożoności obliczeniowej AIC oraz BIC dla modelu NGBoost na zbiorze SelectDVC.

Zbiór SelectDVC								
Klasyfikator	**Dokładność**	**Precyzja**	**Czułość**	**F1**	**AIC**	**BIC**	**Liczba cech wejściowych n**	**Usunięta cecha po iteracji**
NGBoost	0,942703	0,9434	0,942703	0,942634	8104,771	8263,965	19	idle_std
NGBoost	0,942703	0,9434	0,942703	0,942634	8102,771	8253,587	18	idle_max
NGBoost	0,942703	0,9434	0,942703	0,942634	8100,771	8243,208	17	fwd_iat_std
NGBoost	0,94404	0,94482	0,94404	0,943968	8080,316	8214,374	16	fwd_iat_max
NGBoost	0,94404	0,94482	0,94404	0,943968	8078,316	8203,996	15	fwd_iat_mean
NGBoost	0,942361	0,943552	0,942361	0,942262	8095,003	8212,304	14	bwd_iat_tot
NGBoost	0,942361	0,943552	0,942361	0,942262	8093,002	8201,925	13	bwd_iat_std
NGBoost	0,942672	0,943353	0,942672	0,942604	8093,679	8194,223	12	bwd_iat_min
NGBoost	0,942703	0,943378	0,942703	0,942635	8151,855	8244,02	11	fwd_iat_tot
NGBoost	0,940558	0,941905	0,940558	0,940446	8180,102	8263,889	10	idle_mean
NGBoost	0,941056	0,942424	0,941056	0,940943	8209,579	8284,987	9	bwd_iat_max
NGBoost	0,940279	0,941702	0,940279	0,940161	8214,331	8281,36	8	flow_duration
NGBoost	0,940279	0,941702	0,940279	0,940161	8227,478	8286,129	7	flow_iat_std
NGBoost	0,940247	0,941776	0,940247	0,940124	8313,243	8363,515	6	bwd_iat_mean
NGBoost	0,928372	0,928447	0,928372	0,928345	9637,044	9678,938	5	flow_iat_max
NGBoost	0,929584	0,93345	0,929584	0,929291	9795,983	9829,498	4	init_bwd_win_byts
NGBoost	0,910838	0,911858	0,910838	0,910865	10650,18	10675,32	3	totlen_fwd_pkts
NGBoost	0,928278	0,930854	0,928278	0,928057	10065,55	10082,3	2	bwd_pkt_len_min
NGBoost	0,84524	0,853995	0,84524	0,844792	22139,87	22148,25	1	init_fwd_win_byts

Warto zauważyć, że wartości AIC i BIC są wrażliwe na liczbę cech, ale również na stopień ich istotności. Początkowa liczba cech wynosi 19, a wartości AIC (3987,075) oraz BIC (4146,27) wskazują na stosunkowo złożony model, jednak już po usunięciu pierwszej cechy (idle_max), obserwujemy wyraźny spadek tych wartości – AIC zmienia się na 3940,841, a BIC na 4091,657. Co więcej, w zakresie 14–18 cech wartości AIC i BIC są najniższe (przy jednoczesnym zachowaniu wysokiej dokładności klasyfikacji (~97,5%)). Oznacza to, iż modele w tym przedziale są najbardziej efektywne pod względem kompromisu między dokładnością a złożonością obliczeniową. Usunięcie kolejnych cech, takich jak fwd_iat_max, bwd_iat_min czy fwd_iat_std, prowadzi do stopniowego pogorszenia wydajności klasyfikatora, co znajduje odzwierciedlenie w rosnących wartościach AIC i BIC. Podobną tendencję można zaobserwować w przypadku modelu NGBoost, gdzie redukcja cech przyczynia się do uproszczenia modelu, jednak zmiany w dokładności klasyfikacji są mniej stabilne niż w przypadku XGBoost. Wskazuje to, że w zakresie 15–16 cech model osiąga najlepszy kompromis między dokładnością a złożonością obliczeniową. Dalsza eliminacja cech

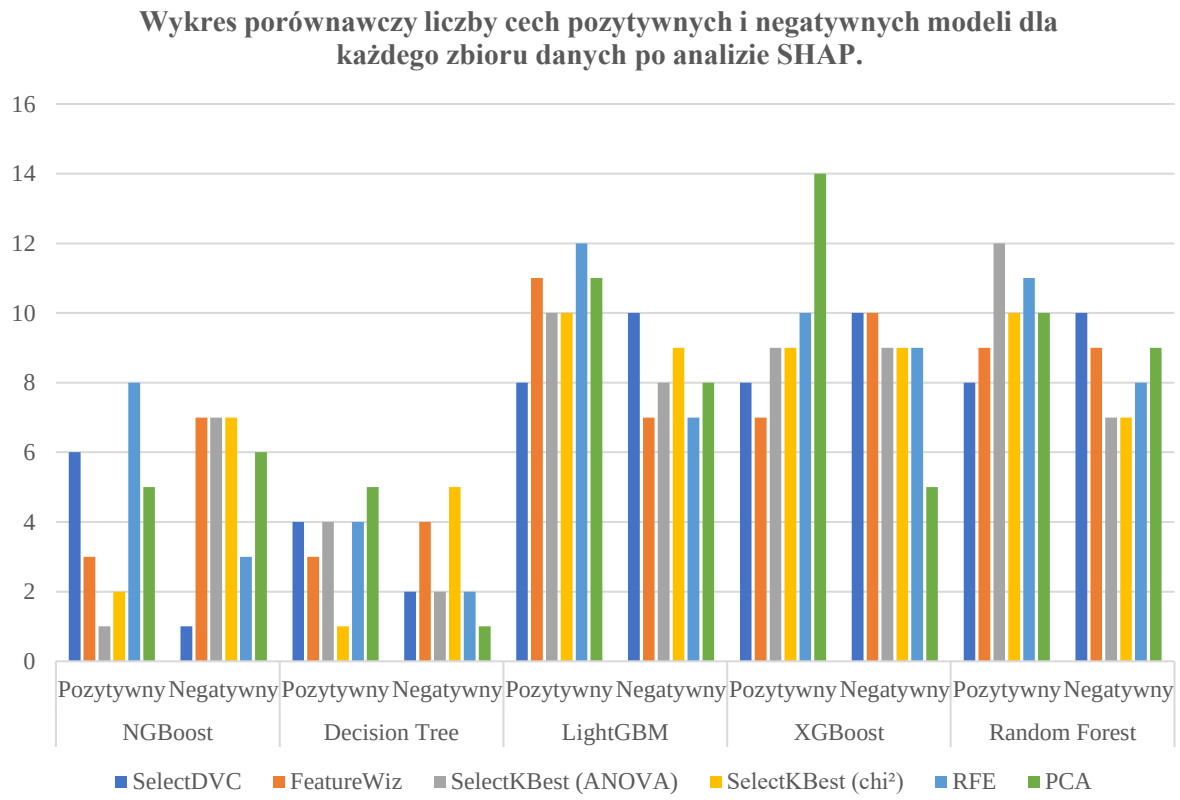
prowadzi do stopniowego wzrostu wartości AIC i BIC, a dokładność zaczyna spadać, szczególnie od 11 cech w dół. W obu klasyfikatorach jako cechy o największym wpływie na model uwidaczniają się cechy totlen_fwd_pkts, init_fwd_win_byts oraz bwd_pkt_len_min.

Analiza wyników jasno pokazuje, że uniwersalne zasady optymalizacji modeli klasyfikacyjnych nie istnieją, jeśli wymagamy znalezienia odpowiedniej równowagi między liczbą cech, a złożonością obliczeniową, przy jednoczesnym zachowaniu wysokiej skuteczności predykcyjnej. Mimo to, możliwe jest eksperymentalnie ustalenie momentu, w którym kompromis między dokładnością, a ilością cech jest najbardziej optymalny, dostosowując go do specyficznych wymagań systemu, choćby w oparciu o usunięcie mniej istotnych cech. W tym świetle, propozycja Rosenfelda dotycząca optymalnego progu 7 cech dla zrozumiałości i przejrzystości modeli może okazać się zbyt restrykcyjna. Eksperymenty wskazują, że dla zachowania wysokiej dokładności i efektywności, modele XGBoost i NGBoost wymagają np. 15-16 cech.

Aby jeszcze lepiej ocenić modele predykcyjne, zaproponowano ulepszenie metryki F. Nowa wersja tej metryki bierze pod uwagę, czy dana cecha pozytywnie, czy negatywnie wpływa na wynik predykcji, co ustala się za pomocą analizy SHAP. Dzięki temu można dokładniej określić, które cechy są najważniejsze i wybrać model, który najlepiej wyjaśnia swoje wyniki. Jest to szczególnie ważne, gdy model wykorzystuje około 7 cech – wtedy preferuje się te, które opierają się na większej liczbie pozytywnych i mniejszej liczbie negatywnych czynników.

Badania określające wpływ cech przeprowadzono w 10 iteracjach, przyjmując, że w każdej z nich cecha mogła mieć różny wpływ – pozytywny w jednej iteracji i negatywny w kolejnej, co odzwierciedla zmienność i zależność danych w próbkach. Założono również, że eksperymenty przeprowadzono dla 19 cech, przy czym możliwa jest redukcja o jedną cechę w każdej iteracji. Pozwoliło to sprawdzić liczbę cech pozytywnych i negatywnych, weryfikując w ten sposób główne założenia. Wyniki końcowe są uśrednionymi rezultatami pochodzącymi z analizy metodą SHAP.

Na załączonym rysunku 7.3 przedstawiono rozkład liczby cech pozytywnych i negatywnych dla 19 cech.



Rys. 7.3. Wykres porównawczy liczby cech pozytywnych i negatywnych modeli dla każdego zbioru danych po analizie SHAP.

Zastosowane metody selekcji cech znacząco wpływają na liczbę i rodzaj cech wykorzystywanych przez różne klasyfikatory. W analizie zauważono, że modele LightGBM, XGBoost oraz Random Forest charakteryzowały się zbliżoną proporcją cech pozytywnie i negatywnie oddziałujących na wyniki predykcji, w przeciwieństwie do klasyfikatorów Decision Tree i NGBoost, które wykazują preferencję dla mniejszego, selektywnego wyboru cech. Modele, w których dominują cechy pozytywnie wpływające na predykcję, zazwyczaj osiągają wyższą dokładność niż te, w których przeważają cechy negatywne. Różnice w strategiach przetwarzania danych przekładają się na zróżnicowaną złożoność modeli oraz ich względem pod względem liczby efektywnie wykorzystywanych cech.

W kolejnym etapie analizy obliczono wartości metryki F, zaproponowanej przez Rosenfelda (41), dla różnych kombinacji klasyfikatorów i zbiorów cech co przedstawiono w tabeli 7.4. Metryka ta w połączeniu z wartościami liczbowymi cech wejściowych pozytywnych i negatywnych wykorzystywanych przez model, pozwala ocenić wyjaśnialność modelu, przy założeniu, że mniejsza liczba cech przekłada się na większą przejrzystość.

Tabela 7.4.: Wyniki zmodyfikowanej metryki F uwzględniającej liczbę cech pozytywnych i negatywnych dla modeli i wszystkich zbiorów cech.

Zbiór danych	NGBoost	XGBoost	LightGBM	Random Forest	Decison Tree
SelectDVC	0	0,6	0,05	0,6	0
FeatureWiz	0,05	0,55	0,6	0,55	0
SelectKBest(ANOVA)	0,1	0,6	0,5	0,5	0
SelectKBest(chi2)	0,3	0,6	0,65	0,55	0
RFE	0,1	0,6	0,6	0,6	0
PCA	0,15	0,6	0,6	0,6	0

Analiza wyników zaprezentowanych w tabeli 7.4 wskazuje, że algorytmy NGBoost oraz Decision Tree, charakteryzują się niższymi wartościami metryki F, przez co są preferowane z punktu widzenia wyjaśnialności, zgodnie z założeniem Rosenfelda, że mniejsza liczba cech wpływających na decyzję modelu przekłada się na jego większą przejrzystość. W tym przypadku algorytm Decision Tree osiągnął w każdym zbiorze danych wynik równy 0.

Sukces algorytmów w zadaniu detekcji ataków DDoS zależy od właściwego doboru cech i ich liczby, aby system mógł skutecznie wykrywać ataki, jednocześnie zachowując wysoką efektywność. W proponowanym systemie, który wymaga natychmiastowej reakcji, równowaga między liczbą cech a szybkością analizy staje się najważniejsza. Zbyt wiele cech może spowodować opóźnienia, a zbyt mało może nie dostarczyć wystarczających informacji do identyfikacji ataku.

W ostatnim etapie eksperymentów wykorzystując metodę SHAP zweryfikowano cechy dające pozytywny i negatywny wpływ dla wszystkich algorytmów, przez co możliwe było zrozumienie, jak poszczególne modele (NGBoost, Decision Tree, LightGBM, XGBoost, Random Forest) różnicują znaczenie poszczególnych cech. Na podstawie wyników przedstawionych w tabelach 7.5-7.9 algorytmy NGBoost i Decision Tree koncentrują się głównie na cechach związanych z czasem między pakietami i długością pakietów takie jak flow_iat_max i bwd_iat_mean, a także z długością pakietów, na przykład bwd_pkt_len_min. Jednak istotną różnicą między tymi algorytmami jest większa zmienność w wyborze cech o ujemnym wpływie w przypadku Decision Tree. Przykładowo, cechy flow_duration oraz flow_iat_max były klasyfikowane zarówno jako pozytywne, jak i negatywne, w zależności od zastosowanej metody selekcji cech. Modele LightGBM i XGBoost zwracają uwagę na rozmiary początkowych okien TCP (cechy takie jak init_fwd_win_bytes i init_bwd_win_bytes), a także z charakterystyką czasową aktywności i bezczynności połączeń

(cechy active_mean, idle_mean oraz idle_std) co może wskazywać na zdolność do wykrywania ataków opartych na krótkich pakietach i długich przerwach. Algorytm Random Forest natomiast wykorzystuje bardziej zróżnicowany zestaw cech, obejmujący zarówno parametry dotyczące podprzepływów (subflow_fwd_pkts), jak i całkowitej liczby pakietów (tot_bwd_pkts), choć niektóre z cech (takie jak fwd_iat_max i fwd_iat_std) mogą być różnie klasyfikowane w zależności od zestawu cech.

Tabela 7.5.:Tabela cech pozytywnych i negatywnych po analizie SHAP dla każdego zbioru cech po analizie SHAP dla modeli NGBoost.

	NGBoost	
Zbiór cech	**Cechy pozytywne**	**Cechy negatywne**
SelectDVC	'flow_iat_max', 'flow_duration', 'bwd_pkt_len_min', 'init_fwd_win_byts', 'flow_iat_std', 'totlen_fwd_pkts'	'bwd_iat_mean'
FeatureWiz	'bwd_pkt_len_min', 'flow_iat_max', 'bwd_pkts_s'	'pkt_size_avg', 'init_fwd_win_byts', 'bwd_pkt_len_mean', 'flow_duration', 'fwd_pkt_len_mean', 'bwd_pkt_len_max', 'pkt_len_max'
SelectKBest(ANOVA)	‘bwd_iat_std'	'flow_duration', 'fwd_pkt_len_min', 'bwd_pkt_len_min', 'pkt_len_min', 'flow_iat_max', 'flow_iat_std', 'init_fwd_win_byts'
SelectKBest(chi2)	'fwd_iat_tot', 'bwd_iat_mean'	'flow_duration', 'flow_iat_mean', 'flow_iat_max', 'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_max', 'idle_mean'
RFE	'flow_duration', 'flow_byts_s', 'fwd_pkt_len_max', 'bwd_pkt_len_min', 'bwd_pkt_len_std', 'pkt_len_max', 'flow_iat_max', 'init_fwd_win_byts'	'fwd_pkts_s', 'pkt_len_mean', 'bwd_iat_mean’
PCA	'fwd_iat_tot', 'flow_iat_max', 'bwd_iat_mean', 'flow_byts_s', 'bwd_blk_rate_avg'	'flow_duration', 'bwd_iat_max', 'fwd_iat_max', 'idle_mean', 'fwd_iat_std', 'fwd_iat_mean'

Tabela 7.6.: Tabela cech pozytywnych i negatywnych po analizie SHAP dla każdego zbioru cech po analizie SHAP dla modeli Decision Tree.

Decision Tree

Zbiór cech	Cechy pozytywne	Cechy negatywne
SelectDVC	'flow_iat_max', 'bwd_pkt_len_min', 'init_fwd_win_byts', 'flow_iat_std'	'bwd_iat_mean', 'totlen_fwd_pkts'
FeatureWiz	'init_fwd_win_byts', 'flow_iat_max', 'bwd_pkts_s'	'bwd_pkt_len_min', 'pkt_size_avg', 'bwd_pkt_len_mean', 'bwd_iat_min'
SelectKBest(ANOVA)	'flow_duration', 'fwd_pkt_len_min', 'flow_iat_max', 'init_fwd_win_byts'	'bwd_pkt_len_min', 'bwd_iat_std'
SelectKBest(chi2)	‘flow_duration'	'flow_iat_mean', 'flow_iat_max', 'fwd_iat_tot', 'fwd_iat_std', 'bwd_iat_mean'
RFE	'flow_byts_s', 'flow_iat_max', 'bwd_iat_mean', 'init_fwd_win_byts'	'bwd_pkt_len_min', 'pkt_len_mean'
PCA	'flow_duration', 'fwd_iat_tot', 'flow_iat_max', 'bwd_iat_mean', 'flow_byts_s'	'fwd_iat_max'

Tabela 7.7.: Tabela cech pozytywnych i negatywnych po analizie SHAP dla każdego zbioru cech po analizie SHAP dla modeli LightGBM.

LightGBM

Zbiór cech	Cechy pozytywne	Cechy negatywne
SelectDVC	'flow_iat_max', 'flow_duration', 'idle_mean', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_iat_mean', 'bwd_iat_std', 'idle_std	'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_max', 'bwd_pkt_len_min', 'flow_iat_std', 'bwd_iat_tot', 'bwd_iat_mean', 'bwd_iat_min', 'totlen_fwd_pkts'
FeatureWiz	'bwd_pkt_len_min', 'subflow_fwd_pkts', 'init_bwd_win_byts', 'flow_iat_max', 'bwd_pkts_s', 'tot_bwd_pkts',	'pkt_size_avg', 'init_fwd_win_byts', 'bwd_pkt_len_mean',

	'fwd_pkt_len_mean', 'idle_mean', 'totlen_fwd_pkts', 'totlen_bwd_pkts', 'fwd_iat_min'	'flow_duration', 'bwd_iat_min', 'bwd_pkt_len_max', 'pkt_len_max'
SelectKBest(ANOVA)	'pkt_len_min', 'flow_iat_std', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_tot', 'bwd_iat_max', 'init_fwd_win_byts', 'init_bwd_win_byts', 'idle_std'	'flow_duration', 'fwd_pkt_len_min', 'bwd_pkt_len_min', 'flow_iat_max', 'fwd_iat_mean', 'bwd_iat_std', 'idle_max', 'idle_mean'
SelectKBest(chi2)	'flow_duration', 'flow_iat_max', 'flow_iat_std', 'fwd_iat_mean', 'bwd_iat_tot', 'active_max', 'active_mean', 'idle_max', 'idle_min', 'idle_std'	'flow_iat_mean', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_max', 'bwd_iat_mean', 'bwd_iat_std', 'active_std', 'idle_mean'
RFE	'flow_duration', 'flow_byts_s', 'fwd_pkts_s', 'bwd_pkt_len_min', 'bwd_pkt_len_std', 'pkt_len_mean', 'flow_iat_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_max', 'init_fwd_win_byts', 'init_bwd_win_byts'	'totlen_fwd_pkts', 'fwd_pkt_len_max', 'pkt_len_max', 'fwd_iat_std', 'bwd_iat_max', 'bwd_iat_min', 'bwd_iat_mean'
PCA	'flow_duration', 'bwd_iat_tot', 'flow_iat_max', 'idle_max', 'bwd_iat_std', 'bwd_iat_mean', 'idle_std', 'active_max', 'flow_iat_std', 'fwd_blk_rate_avg', 'active_mean'	'fwd_iat_tot', 'bwd_iat_max', 'fwd_iat_max', 'idle_mean', 'flow_byts_s', 'fwd_iat_std', 'fwd_iat_mean', 'bwd_blk_rate_avg'

Tabela 7.8.: Tabela cech pozytywnych i negatywnych po analizie SHAP dla każdego zbioru cech po analizie SHAP dla modeli XGBoost.

XGBoost

Zbiór cech	**Cechy pozytywne**	**Cechy negatywne**
SelectDVC	'flow_iat_max', 'flow_duration', 'fwd_iat_max', 'bwd_pkt_len_min', 'flow_iat_std', 'fwd_iat_mean', 'bwd_iat_std', 'bwd_iat_min	'idle_mean', 'fwd_iat_tot', 'fwd_iat_std', 'bwd_iat_max', 'init_fwd_win_byts', 'init_bwd_win_byts', 'bwd_iat_tot', 'idle_max', 'idle_std', 'bwd_iat_mean', 'totlen_fwd_pkts'

FeatureWiz	'bwd_pkt_len_min', 'bwd_pkt_len_mean', 'init_bwd_win_byts', 'bwd_pkts_s', 'idle_mean', 'fwd_iat_min', 'pkt_len_max'	'pkt_size_avg', 'init_fwd_win_byts', 'subflow_fwd_pkts', 'flow_iat_max', 'flow_duration', 'tot_bwd_pkts', 'bwd_iat_min', 'fwd_pkt_len_mean', 'totlen_fwd_pkts', 'bwd_pkt_len_max', 'totlen_bwd_pkts'
SelectKBest(ANOVA)	'fwd_pkt_len_min', 'bwd_pkt_len_min', 'pkt_len_min', 'flow_iat_std', 'fwd_iat_max', 'fwd_iat_mean', 'idle_max', 'idle_mean', 'idle_std'	'flow_duration', 'flow_iat_max', 'fwd_iat_tot', 'fwd_iat_std', 'bwd_iat_tot', 'bwd_iat_max', 'bwd_iat_std', 'init_fwd_win_byts', 'init_bwd_win_byts'
SelectKBest(chi2)	'flow_duration', 'flow_iat_mean', 'flow_iat_std', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'active_mean', 'idle_max', 'idle_min'	'flow_iat_max', 'fwd_iat_mean', 'bwd_iat_tot', 'bwd_iat_max', 'bwd_iat_mean', 'bwd_iat_std', 'active_max', 'active_std', 'idle_mean', 'idle_std'
RFE	'flow_duration', 'fwd_pkts_s', 'totlen_fwd_pkts', 'bwd_pkt_len_min', 'pkt_len_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_std', 'bwd_iat_mean', 'init_fwd_win_byts'	'flow_byts_s', 'fwd_pkt_len_max', 'bwd_pkt_len_std', 'pkt_len_mean', 'flow_iat_max', 'fwd_iat_max', 'bwd_iat_max', 'bwd_iat_min', 'init_bwd_win_byts'
PCA	'flow_duration', 'bwd_iat_tot', 'bwd_iat_max', 'fwd_iat_max', 'flow_iat_max', 'bwd_iat_mean', 'idle_mean', 'flow_byts_s', 'fwd_iat_std', 'active_max', 'flow_iat_std', 'fwd_blk_rate_avg', 'active_mean', 'bwd_blk_rate_avg'	'fwd_iat_tot', 'idle_max', 'bwd_iat_std', 'idle_std', 'fwd_iat_mean'

Tabela 7.9.: Tabela cech pozytywnych i negatywnych po analizie SHAP dla każdego zbioru cech po analizie SHAP dla modeli Random Forest.

Random Forest

Zbiór cech	**Cechy pozytywne**	**Cechy negatywne**
SelectDVC	‘flow_iat_max', 'flow_duration', 'idle_mean', 'fwd_iat_tot', 'init_bwd_win_byts', 'flow_iat_std', 'idle_max', 'idle_std	'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_max', 'bwd_pkt_len_min', 'init_fwd_win_byts', 'fwd_iat_mean', 'bwd_iat_tot', 'bwd_iat_std', 'bwd_iat_mean', 'bwd_iat_min', 'totlen_fwd_pkts'
FeatureWiz	'init_fwd_win_byts', 'subflow_fwd_pkts', 'init_bwd_win_byts', 'flow_iat_max', 'flow_duration', 'bwd_pkts_s', 'idle_mean', 'totlen_fwd_pkts', 'totlen_bwd_pkts	'bwd_pkt_len_min', 'pkt_size_avg', 'bwd_pkt_len_mean', 'tot_bwd_pkts', 'bwd_iat_min', 'fwd_pkt_len_mean', 'bwd_pkt_len_max', 'pkt_len_max', 'tot_fwd_pkts'
SelectKBest(ANOVA)	'flow_duration', 'fwd_seg_size_min', 'flow_iat_max', 'flow_iat_std', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'init_fwd_win_byts', 'init_bwd_win_byts', 'idle_max', 'idle_mean', 'idle_std'	‘fwd_pkt_len_min', 'bwd_pkt_len_min', 'pkt_len_min', 'fwd_iat_mean', 'bwd_iat_tot', 'bwd_iat_max', 'bwd_iat_std'
SelectKBest(chi2)	'flow_duration', 'flow_iat_max', 'fwd_iat_max', 'fwd_iat_std', 'bwd_iat_tot', 'bwd_iat_max', 'active_mean', 'idle_max', 'idle_mean', 'idle_std	'flow_iat_mean', 'flow_iat_std', 'fwd_iat_tot', 'fwd_iat_mean', 'bwd_iat_mean', 'bwd_iat_std', 'idle_min'
RFE	'flow_duration', 'flow_byts_s', 'fwd_pkts_s', 'totlen_fwd_pkts', 'bwd_pkt_len_min', 'flow_iat_max', 'fwd_iat_tot', 'fwd_iat_max', 'fwd_iat_std', 'init_fwd_win_byts', 'init_bwd_win_byts'	fwd_pkt_len_max', 'bwd_pkt_len_std', 'pkt_len_max', 'pkt_len_mean', 'flow_iat_min', 'bwd_iat_max', 'bwd_iat_min', 'bwd_iat_mean'
PCA	'flow_duration', 'fwd_iat_tot', 'fwd_iat_max', 'flow_iat_max', 'fwd_iat_std', 'idle_std', 'active_max', 'flow_iat_std', 'active_mean', 'bwd_blk_rate_avg'	bwd_iat_tot', 'bwd_iat_max', 'idle_max', 'bwd_iat_std', 'bwd_iat_mean', 'idle_mean', 'flow_byts_s', 'fwd_blk_rate_avg', 'fwd_iat_mean'

W konkluzji, wnioski z przeprowadzonych badań wskazują, że największą rolę w detekcji ataków DDoS odgrywają cechy związane z manipulacjami oknami TCP, a także parametry opisujące długość pakietów oraz charakterystykę czasową ruchu sieciowego. Szczególnie należy zwrócić uwagę na parametry init_fwd_win_byts oraz init_bwd_win_byts, które odzwierciedlają początkowe rozmiary okien TCP. Nieprawidłowe wartości tych parametrów mogą wskazywać na anomalie typowe dla ataków SYN Flood, które bazują na masowym nawiązywaniu, lecz niekończeniu połączeń TCP. Dodatkowo nietypowe wzorce w cechach opisujących długość pakietów, w tym bwd_pkt_len_min oraz totlen_fwd_pkts mogą wskazywać na próby przeciążenia sieci poprzez wysyłanie dużej liczby małych pakietów. Znaczącą rolę odgrywają również cechy czasowe, takie jak fwd_iat_min, bwd_iat_min, flow_iat_std, fwd_iat_max, bwd_pkt_len_min oraz flow_iat_max, które opisują interwały czasowe pomiędzy pakietami oraz ich wariancję. Nieregularny rozkład czasowy pakietów, zwłaszcza w ruchu pochodzącym od jednego źródła, może świadczyć o ataku DDoS, w którym atakujący starają się ominąć mechanizmy wykrywania poprzez nieregularne generowanie ruchu.

8. Wyjaśnialność klasyfikacji ataków DDoS w oparciu o celowe perturbacje danych wejściowych.

Modele uczenia maszynowego powinny wykazywać odporność na niewielkie zaburzenia w danych wejściowych, gwarantując, że drobne zmiany nie prowadzą do istotnych modyfikacji predykcji. W celu oceny tej odporności, w niniejszym rozdziale zaprezentowano autorskie implementacje metryk monotoniczności, wierności, niekompletności oraz niewierności, oparte na mechanizmie wprowadzania celowych, kontrolowanych perturbacji do danych wejściowych (generowany jest szum do każdej cechy z rozkładu normalnego $N(0,\sigma)$), a następnie analizowany jest wpływ na predykcje modelu.

Monotoniczność modelu odnosi się do jego zdolności do przewidywalnej zmiany predykcji w odpowiedzi na wzrost lub spadek ważności cech modelu (wagi). Jest to zgodne z intuicją, że zwiększenie wartości cechy powinno prowadzić do monotonicznej zmiany (rosnącej lub malejącej) w predykcjach modelu.

Metryka niewierności pozwala obliczyć różnicę między przewidywaniem oryginalnej próbki, a przewidywaniem próbki zaburzonej. Różnica ta jest podnoszona do kwadratu i mnożona przez wagę zaburzonej cechy.

Metryka wierności weryfikuje, jak dobrze wyjaśnienia odzwierciedlają faktyczne zachowanie modelu. Realizowana jest poprzez dodanie szumu do poszczególnych cech i sprawdzenie, czy wyjaśnienia poprawnie przewidują wynikające z tego zmiany w wynikach modelu.

Metryka niekompletności określa ilościowo, jak wrażliwe są predykcje modelu na małe perturbacje cech wejściowych. Metryka opiera się na średniej zmianie w predykcjach, podczas gdy szum jest dodawany do każdej cechy ważonej przez wyniki ważności cech.

W sieciach SDN dane mogą być podatne na niewielkie zmiany, takie jak błędy pomiarowe lub szum w danych. Pytania badawcze w związku z tym brzmią:

- Jak wrażliwe są modele na małe zmiany w danych wejściowych?
- Czy niewielkie perturbacje cech prowadzą do znaczących zmian w predykcjach?
- Czy modele zachowują się nieprzewidywalnie w odpowiedzi na losowe zmiany w danych?

Zaproponowanie użycia tych metryk przedstawione zostało w badaniach w artykule [126] oraz w dołączonym do artykułu benchmarku. Jednakże dotyczył on kontrolowanych eksperymentów na danych syntetycznych. Autorzy tej publikacji nie przedstawili w artykule

dokładnych wzorów matematycznych, ani też pseudokodu metryk monotoniczności, wierności, niekompletności oraz niewierności. Stąd też zdecydowano się na własną implementację tych metryk w niniejszej rozprawie doktorskiej, aby umożliwić ich praktyczne zastosowanie w ocenie stabilności i wiarygodności modeli uczenia maszynowego w warunkach niepewności.

8.1 Analiza wyjaśnialności w oparciu o metrykę monotoniczności.

Monotoniczność w modelach predykcyjnych jest istotnym aspektem oceny ich stabilności i przewidywalności. Odnosi się do zdolności modelu do przewidywalnej zmiany predykcji w odpowiedzi na wzrost lub spadek ważności cech. W idealnym przypadku, wzrost wartości cechy powinien prowadzić do monotonicznej zmiany (rosnącej lub malejącej) w predykcjach modelu. W niniejszej pracy zbadano monotoniczność klasyfikatorów z wykorzystaniem metryki opartej na wartościach SHAP, gdzie poprzez zastosowanie metryki monotoniczności oceniamy jak dobrze wyniki ważności cech modelu (wagi) są zgodne z kierunkami sugerowanymi przez wartości SHAP.

Autorska propozycja algorytmu(Algorytm 3) pozwala obliczyć średnie wartości bezwzględne SHAP dla każdej cechy, następnie sortuje je malejąco i określa kierunkowość wag cech na podstawie tych posortowanych wartości SHAP. Naruszenia monotoniczności są liczone jako przypadki, w których różnice między kolejnymi wagami są mniejsze od zera. Wynik monotoniczności jest obliczany jako 1 minus stosunek liczby naruszeń do całkowitej liczby możliwych par cech.

Podczas iteracji dla każdej cechy w zestawie danych, algorytm wprowadza zakłócenia poprzez dodanie losowego szumu (generowanego z rozkładu normalnego o średniej 0 i odchyleniu standardowym 1) do wartości danej cechy we wszystkich próbkach. Te wartości zostały dobrane empirycznie w celu zachowania równowagi między czułością testu a jego odpornością na przypadkowe fluktuacje. Perturbacje są losowe, więc każdorazowe wywołanie funkcji może dać nieco inne wyniki ze względu na przypadkowość szumu generowanego z rozkładu normalnego. Na podstawie wartości SHAP przed i po perturbacjach obliczane są wskaźniki monotoniczności, które pokazują, jak stabilne są wagi cech modelu względem wprowadzonych zakłóceń.

Wyniki metryki monotoniczności należy interpretować w następujący sposób:

- Wartości bliskie 1.0 wskazują na silną monotoniczność modelu
- Wartości poniżej 0.8 sugerują potencjalne problemy z przewidywalnością modelu

Poniżej znajduje się pseudokod algorytmu(Algorytm 3) metryki monotoniczności.

Algorytm 3: Algorytm Monotoniczność bez perturbacji

Wejście: Wczytaj model z wybranym zbiorem cech(lista *selected_features*)

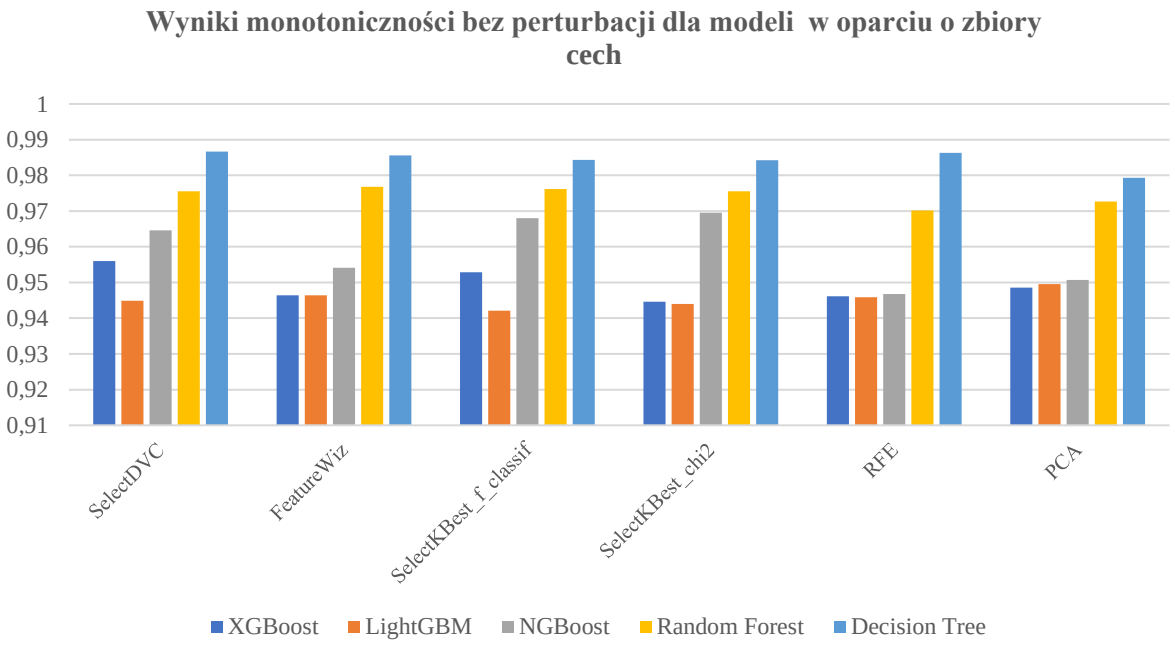
Procedura:

1. Określ liczbę cech *n_features i s*prawdź, czy liczba cech jest zgodna z liczbą wartości SHAP dla każdej próbki. Jeśli nie, zwróć błąd.
2. Oblicz ważność każdej cechy za pomocą *shap_importance* jako średnią wartość bezwzględną jej wartości SHAP.
3. Posortuj cechy malejąco według ważności i zapisz indeksy w liście *sorted_indices*.
4. Utwórz wektor *directional_weights*, mnożąc wagi cech przez znak ich wartości SHAP (dla posortowanych cech).
5. Policz liczbę naruszeń monotoniczności (*violations*), czyli przypadki, gdy sąsiednie wartości w *directional_weights* mają przeciwne znaki.
6. Oblicz liczbę wszystkich możliwych par cech i zapisz w *num_pairs*.
7. Jeśli nie ma żadnych par cech, zwróć wynik 1.0 i (opcjonalnie) posortowane indeksy.
8. Oblicz wynik metryki monotoniczności *(monotonicity_score)*, odejmując od 1 stosunek liczby naruszeń do liczby par.
9. Zwróć wynik metryki monotoniczności*(monotonicity_score),* i opcjonalnie posortowane indeksy cech**(***sorted_indice*s**).**

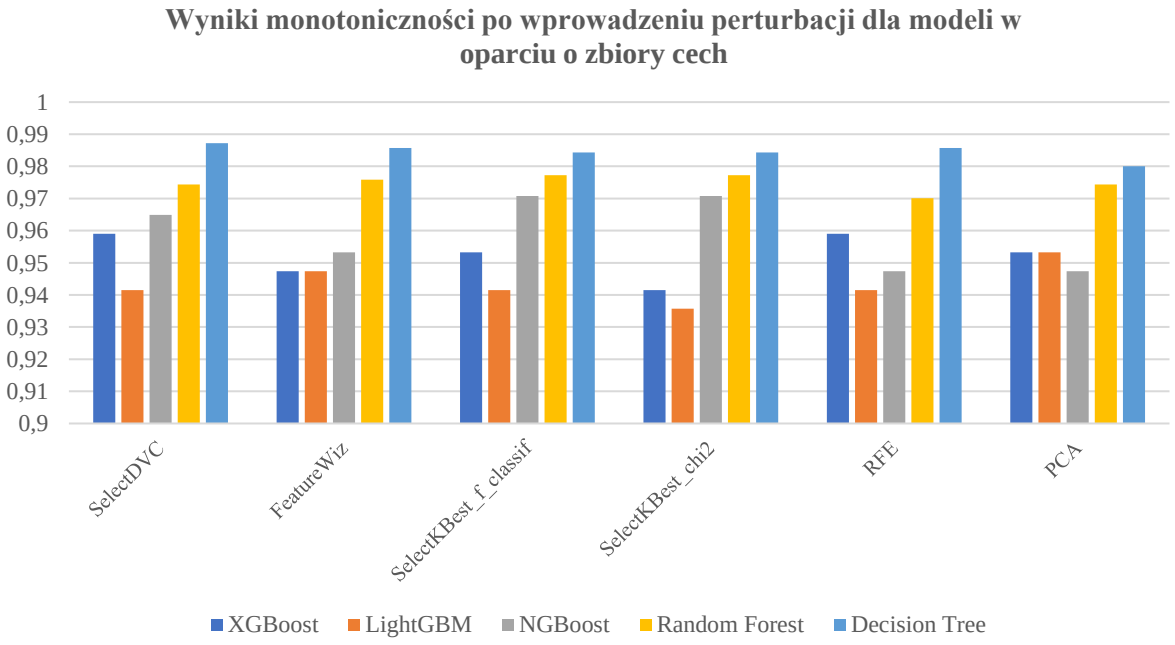
Koniec procedury

Wyjście: Wynik metryki monotoniczności *(monotonicity_score)* i (opcjonalnie) posortowane indeksy cech**(***sorted_indice*s**).**

Perturbacje wprowadzane do danych pozwoliły ocenić stabilność modeli wobec zakłóceń. Wyniki dla ogólnej metryki monotoniczności zostały poniżej przedstawione na wykresie słupkowym(rysunek 8.1) oraz po wprowadzeniu perturbacji dla modeli w każdym zbiorze cech(rysunek 8.2).



Rys. 8.1. Wyniki monotoniczności bez perturbacji dla modeli w oparciu o zbiory cech.



Rys. 8.2. Wyniki monotoniczności po wprowadzeniu perturbacji dla modeli w oparciu o zbiory cech.

Wyniki eksperymentów pokazują, że mimo wprowadzania celowych perturbacji do cech wejściowych, modele te charakteryzują się wysoką monotonicznością(> 0.90) zapewniając silne powiązanie między ważnością cech a wartościami SHAP. Po wprowadzeniu zakłóceń (rysunek 8.2) wartości monotoniczności nieznacznie spadły(średnio o 0.02-0.05), ale pozostały

na wysokim poziomie. Wysokie wartości monotoniczności, utrzymujące się nawet po wprowadzeniu perturbacji, potwierdzają, iż modele zachowują się przewidywalnie i spójnie w odpowiedzi na zmiany ważności cech.

8.2 Analiza wyjaśnialności w oparciu o metrykę wierności.

Metryka wierności jest kolejnym ważnym narzędziem oceny modeli predykcyjnych, umożliwiającym sprawdzenie, czy ich zachowanie jest spójne i przewidywalne w obliczu zmian w danych wejściowych. Zastosowanie tej metryki pozwala ocenić, czy modele utrzymują spójność w przypisywaniu wag poszczególnym cechom oraz czy ich zachowanie jest stabilne i przewidywalne w warunkach zmieniających się danych. Wysoka wierność wskazuje, że model jest przewidywalny i jego wyjaśnienia są zgodne z rzeczywistymi zmianami w danych.

Propozycja autorskiej implementacji metryki wierności(Algorytm 4) uwzględnia systematyczne wprowadzanie losowych szumów do poszczególnych cech w zestawie danych testowych, a następnie ocenie wpływu tych perturbacji na predykcje modelu. Algorytm rozpoczyna działanie od określenia liczby próbek i cech w zestawie danych testowych. Następnie inicjalizowana jest lista która będzie przechowywać wyniki dla poszczególnych prób. Dla każdej iteracji tworzona jest nowa pusta lista, która przechowuje wyniki dla poszczególnych cech. Do każdej cechy w zestawie danych, algorytm wprowadza zakłócenia poprzez dodanie losowego szumu (generowanego z rozkładu normalnego o średniej 0 i odchyleniu standardowym 1) do wartości danej cechy we wszystkich próbkach co zapewnia powtarzalne i interpretowalne wyniki.

Wykorzystując wytrenowany model predykcyjny, generowano dwa zestawy przewidywań. Jeden dla oryginalnych, niezakłóconych danych testowych oraz drugi dla danych z wykorzystaniem perturbacji, gdzie następnie dla obu tych zestawów obliczana jest korelacja Pearsona, będąca miarą zgodności między oryginalnymi a zakłóconymi wynikami. Wartości korelacji dla poszczególnych cech są agregowane, dając średnią miarę wiarę dla danego modelu. Należy zaznaczyć, że w przypadkach, gdy obliczenie korelacji nie jest możliwe z powodu problemów z wymiarami macierzy, daną cechę należy pominąć.

Algorytm 4: Algorytm obliczania metryki wierności

Wejście: Wczytaj model z wybranym zbiorem cech(lista *selected_features*)

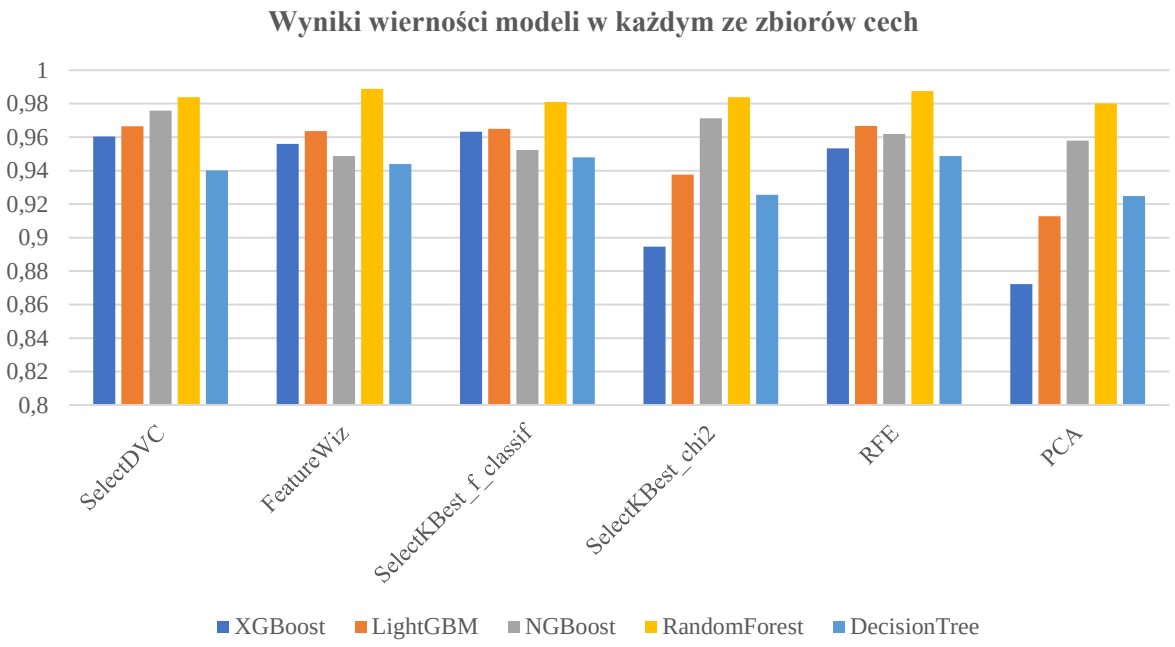
Procedura:

1. Określ liczbę próbek (*num_samples*), liczbę cech (*num_features*) oraz liczbę iteracji n.
2. Stwórz pustą listę do przechowywania wyników wierności (*faithfulness_scores*).
3. Powtórz następujący proces n razy:
    1. Stwórz pustą listę dla wierności cech (*feature_faithfulness*).
    2. Dla każdej cechy (i) od 0 do *num_features* - 1:
    3. Utwórz kopię macierzy X (*X_noisy*).
    4. Dodaj do i-tej kolumny macierzy *X_noisy* szum losowy o rozkładzie normalnym, ze średnią 0 i odchyleniem standardowym σ.
    5. Oblicz nowe przewidywania modelu (*y_noisy*) na podstawie danych z szumem (*X_noisy*).
    6. Oblicz współczynnik korelacji Pearsona między oryginalnymi przewidywaniami (*y_pred*) a przewidywaniami z szumem (*y_noisy*).
    7. Obliczony współczynnik korelacji dodaj do listy *feature_faithfulness*.
    8. Oblicz średnią wartość z listy *feature_faithfulness* i dodaj ją do listy *faithfulness_scores.*
4. Zwróć uśrednianą wartość z listy *faithfulness_scores* jako wynik metryki wierności

Koniec procedury

Wyjście: Wynik metryki wierności.

Wyniki metryki wierności zostały przedstawione na rysunku 8.3. Wskazują one, że modele stosunkowo są odporne na zakłócenia(>0.94 dla większości przypadków), choć niektóre z nich mogą być bardziej wrażliwe na zmiany w danych wejściowych w zależności od zastosowanego zbioru cech.



Rys. 8.3. Wyniki wierności modeli w każdym ze zbiorów cech.

Przykładowo model wykorzystujący algorytm XGBoost osiągnął najgorszy wynik ze wszystkich modeli w przypadku zbioru PCA choć pozostałe modele wykorzystujące boosting osiągają dość dobre wyniki. Jednak należy zauważyć, iż zgodnie z wynikami przedstawionymi na rysunku, modele oparte na algorytmie Random Forest osiągnęły najwyższe wartości w metryce wierności, niezależnie od zastosowanego zestawu cech(wartości wynoszą od 0,980 do 0,988). Uwzględniając korelację między oryginalnymi a zaburzonymi predykcjami, możemy uznać, iż modele wykazują wysoka wierność co prowadzi do ich stabilności i przewidywalności.

8.3 Analiza wyjaśnialności w oparciu o metrykę niewierności.

Metryka niewierności i wierności są dwoma powiązanymi, lecz odmiennymi miarami oceny modeli predykcyjnych. Metryka niewierności pozwala na ocenę rozbieżności między oczekiwanym, a faktycznym zachowaniem modelu, wskazując na stopień jego nieprzewidywalności. Niska wartość tej metryki wskazuje na wysoką przewidywalność modelu, co oznacza, że im niższa wartość niewierności, tym bardziej model jest stabilny i przewidywalny. Innymi słowy, metryka niewierności mierzy wielkość różnic między przewidywaniami modelu dla danych oryginalnych i zakłóconych, umożliwiając identyfikację obszarów, w których model może być mniej stabilny.

W autorskiej propozycji implementacji tej metryki(Algorytm 5), podobnie jak w metryce wierności wprowadza się systematycznie losowe szumy do cech w zestawie danych testowych, a następnie ocenia wpływ tych perturbacji na predykcje modelu. Algorytm rozpoczyna działanie od określenia liczby próbek i cech w zestawie danych testowych, następnie wybiera cechy na podstawie wartości SHAP i dodaje szum do tych cech, aby obliczyć różnice kwadratowe między prognozami modelu dla oryginalnych i zaszumionych danych. Wynik jest ważony przez wagi cech i uśredniony, co daje miarę niewierności modelu.

Algorytm 5: Algorytm obliczania metryki niewierności

Wejście: Wczytaj model z wybranym zbiorem cech(lista *selected_features)*

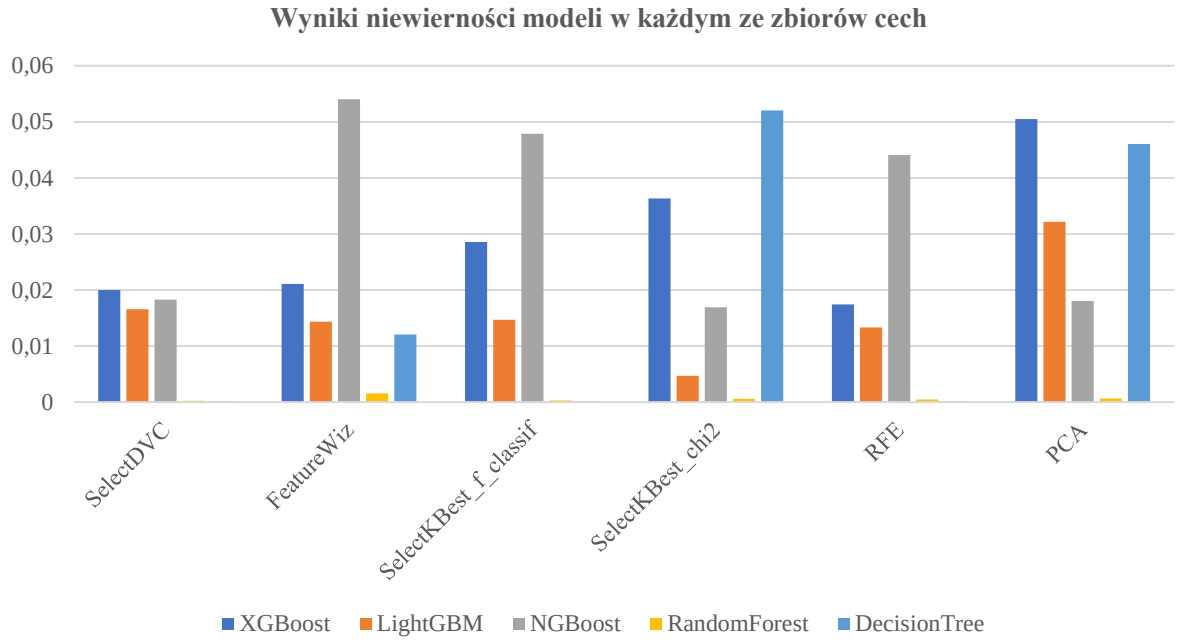
Procedura:

1. Określ liczbę próbek (*num_samples*), liczbę cech (*num_features*) oraz liczbę iteracji n.
2. Stwórz pustą listę do przechowywania wyników niewierności (*infidelities_scores*).
3. Oblicz ważność cech jako średnią wartość bezwzględną wartości SHAP *(feature_importance).*
4. Wylosuj indeksy próbek do zaburzenia (*sample_indices*).
5. Dla każdej wylosowanej próbki (i) powtórz *n* razy:
    1. Utwórz kopię oryginalnej próbki (*x_perturbed*).
    2. Wygeneruj szum losowy o rozkładzie normalnym, ze średnią 0 i odchyleniem standardowym *noise_level*.
    3. Dodaj szum losowy do wartości cechy (*feature_idx*) w próbce *x_perturbed*, ograniczając wynik do przedziału [0, 1].
    4. Oblicz oryginalną predykcję (*y_orig_pred*).
    5. Oblicz predykcję dla zaburzonej próbki (*y_perturbed_pred*).
    6. Oblicz niewierność jako kwadrat różnicy predykcji pomnożony przez wagę cechy *(infidelity).*
    7. Dodaj niewierność do listy *feature_infidelities*.
    8. Dodaj średnią niewierność cech do listy *infidelities_scores*.
6. Zwróć średnią wartość z listy *infidelities_scores* jako wynik metryki niewierności.

Koniec procedury

Wyjście: Wynik metryki niewierności.

Metryka niewierności została zastosowana do oceny wszystkich klasyfikatorów uwzględnionych w niniejszej rozprawie doktorskiej, wraz z różnymi zbiorami cech. Wyniki zostały przedstawione na rysunku 8.4. Podobnie jak w przypadku metryki wierności, wyniki wskazują, że wybór metody selekcji cech ma istotny wpływ na końcowy wynik metryki niewierności modelu.



Rys. 8.4. Wyniki niewierności modeli w każdym ze zbiorów cech.

Analiza wyników ujawniła, że różnice w wartościach niewierności między różnymi metodami selekcji cech były stosunkowo niewielkie, co sugeruje, że modele wykazują dużą odporność na zmiany w danych wejściowych. Modele oparte na algorytmie Random Forest charakteryzowały się również największą stabilnością i przewidywalnością, osiągając konsekwentnie najniższe wartości niewierności, często bliskie 0. Klasyfikatory wykorzystujące metody boostingowe (XGBoost, LightGBM, NGBoost) wykazywały ogólnie wyższą niewierność w porównaniu z modelami Random Forest, choć wartości te były tylko minimalnie wyższe dla niektórych zestawów cech. Żaden z algorytmów nie przekroczył wartości 0,06, a różnice w wartościach niewierności między różnymi metodami selekcji cech były stosunkowo niewielkie.

8.4 Analiza wyjaśnialności w oparciu o metrykę niekompletności.

Metryka niekompletności jest kolejnym narzędziem oceny modeli predykcyjnych, pozwalającym na sprawdzenie, jak wrażliwe są one na subtelne zmiany w danych wejściowych. Zastosowanie jej wraz z innymi metrykami, takimi jak wierność i niewierność, pozwala uzyskać pełniejszy obraz zachowania modelu wobec perturbacji. Głównym założeniem jest

oparcie się na średniej zmianie w predykcjach, podczas gdy szum jest dodawany do każdej cechy ważonej przez wyniki ważności cech. Wagi cech odzwierciedlają ich znaczenie dla modelu. Im niższa wartość metryki niekompletności, tym większa stabilność modelu wobec niewielkich zmian w danych. Autorska implementacja tej metryki(Algorytm 6) została przedstawiona poniżej.

Algorytm 6: Algorytm obliczania metryki niekompletności

Wejście: Wczytaj model z wybranym zbiorem cech(lista *selected_features)*

Procedura:

1. Określ liczbę próbek (*num_samples*), liczbę cech (*num_features*) oraz liczbę iteracji n.
2. Znormalizuj wagi cech *(weights),* aby ich suma wynosiła 1.
3. Stwórz pustą listę do przechowywania wyników niekompletności (*incompleteness_scores*).
4. Stwórz pustą listę dla niekompletności cech (*feature_incompleteness) oraz* utwórz kopię macierzy X (*X_noisy*).
5. Dla każdej wylosowanej próbki (i) powtórz *n* razy

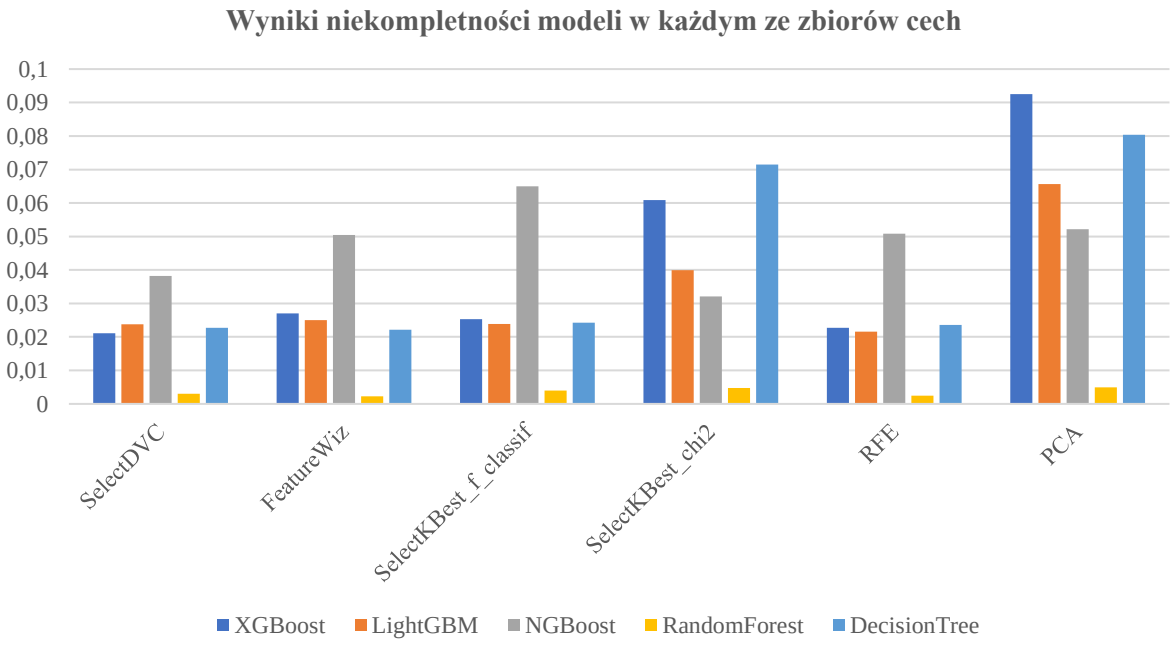
    1. Wygeneruj szum losowy o rozkładzie normalnym, ze średnią 0 i odchyleniem standardowym σ, o rozmiarze takim jak liczba próbek.
    2. Dodaj szum losowy do i-tej kolumny macierzy *X_noisy*.
    3. Oblicz nowe przewidywania modelu (*y_pred_noisy*) na podstawie danych z szumem (*X_noisy*).
    4. Oblicz średnią różnicę bezwzględną między oryginalnymi przewidywaniami (*y_orig_pred*) a przewidywaniami z szumem losowym (*y_pred_noisy*), jako (*prediction_difference*).
    5. Dodaj różnicę predykcji do listy *feature_incompleteness*.
    6. Oblicz średnią wartość z listy feature_incompleteness, pomnożoną przez wagę cechy (*weights[i]*), i dodaj ją do listy *incompleteness_scores*.

6. Zwróć średnią wartość z listy *incompleteness_scores* jako wynik metryki niekompletności

Koniec procedury

Wyjście: Wynik metryki niekompletności.

Wyniki dla metryki niekompletności dla wszystkich klasyfikatorów uwzględnionych w niniejszej rozprawie doktorskiej oraz zbiorach cech zostały poniżej przedstawione na rysunku 8.5.



Rys. 8.5. Wyniki niekompletności modeli w każdym ze zbiorów cech.

Analiza porównawcza modeli uczenia maszynowego, wykazała iż również modele oparte na Random Forest konsekwentnie osiągają najniższe wartości(poniżej 0,01) w metryce niekompletności. Metody selekcji cech również w tym przypadku mają znaczenie. Szczególnie jest to widoczne choćby przy zastosowaniu zbioru PCA, gdzie pozostałe algorytmy zanotowały widoczne pogorszenie wyników oprócz modelu wykorzystującego Random Forest. Należy zwrócić szczególną uwagę, gdyż żaden z algorytmów nie przekroczył wartości 0,10, a w większości przypadków nie przekracza granicy 0.03, przez co można uznać, iż wszystkie testowane metody są w stanie zapewnić dobrą stabilność modelu, co przekłada się na wysoką przewidywalność modeli w wyniku celowych perturbacji.

Badania przeprowadzone z wykorzystaniem czterech komplementarnych metryk (monotoniczność, wierność, niewierność i niekompletność) ujawniają, że wszystkie analizowane algorytmy wykazują zdolność do zapewnienia solidnej stabilności modelu, chociaż niektóre z nich mogą być bardziej podatne na zmiany. Co istotne, eksperymenty dowodzą, że modele na ogół nie reagują w sposób nieprzewidywalny na losowe fluktuacje danych.

9. Wyjaśnialność klasyfikacji ataków DDoS w oparciu stabilność cech wobec ataków zatrucia danych(data poisoning).

Niezawodność modeli uczenia maszynowego oznacza zdolność modelu do zachowania swojej efektywności w obliczu zakłóceń, zmienności danych oraz celowych, potencjalnych ataków manipulacyjnych [127]. Systemy detekcji ataków DDoS oparte na algorytmach klasyfikacji binarnej mogą być niestety podatne na wstrzykiwanie złośliwych danych (data poisoning) do zestawu treningowego w celu zakłócenia procesu uczenia.

Celowe zatrucie danych jest obecnie jednym z najpoważniejszych zagrożeń dla modeli uczenia maszynowego, zwłaszcza w przypadku danych pochodzących ze źródeł niezaufanych. Problem ten nie wynika tylko z potencjalnych błędów w algorytmach klasyfikacji, ale także z procesu pozyskiwania i etykietowania danych, gdzie błędy mogą pojawić się już na etapie zbierania danych. Ataki typu data poisoning możemy podzielić na dwie kategorie uwzględniając ich skutek dla proponowanego systemu detekcji ataków DDoS:

- Atak celowy, gdzie celem jest manipulacja systemem detekcji w określony sposób, np. poprzez wprowadzenie fałszywych danych lub etykiet, aby system klasyfikował niektóre ataki DDoS jako niegroźne. W efekcie, prawdziwe ataki mogą być pomijane, co osłabia bezpieczeństwo sieci.
- Atak niecelowy, którego zadaniem jest ogólne osłabienie wydajności systemu detekcji poprzez dodanie szumu lub nieistotnych danych. To może prowadzić do błędnych alarmów (false positives) lub opóźnienia w reagowaniu na rzeczywiste ataki, DDoS co również negatywnie wpływa na skuteczność ochrony.

Co więcej, tego typu ataki mogą dodatkowo zwiększyć podatność systemów detekcji na inne rodzaje zagrożeń, takie jak ataki backdoor (ukryte mechanizmy umożliwiające obejście zabezpieczeń) czy ataki adversarial (manipulowanie danymi wejściowymi w celu oszukania modelu). Skutkiem tego zjawiska jest także osłabienie wydajności modelu i błędne predykcje detekcji ataków DDoS, co może doprowadzić nie tylko do osłabienia zaufania do modelu, ale także do poważnych konsekwencji. Na przykład takie działania mogą skutecznie pozwolić ominąć istniejące zasady i mechanizmy bezpieczeństwa, otwierając drogę do bardziej zaawansowanych i destrukcyjnych ataków, a także powodując znaczne straty w infrastrukturze sieciowej [127][128]. Konieczne zatem jest wymaganie by tego typu systemy

charakteryzowały się wysokim poziomem gwarancji, odporności, niezawodności w celu zapewnienia nieprzerwalnego działania sieci definiowanych programowo.

W niniejszym rozdziale przedstawiono wyniki eksperymentów opartych na autorskiej propozycji algorytmu przeprowadzania ataków Flip-Label czyli zmianie etykiet w określonej procentowo liczbie rekordów oraz autorskiej propozycji mechanizmu detekcji i ponownego etykietowania podejrzanych etykiet - label sanitization (sanityzacja etykiet, oczyszczanie etykiet), opracowanego w celu łagodzenia skutków tych szkodliwych ataków.

Dzięki eksperymentom dokonano analizy dokładności modelu i podobieństwa predykcji przed i po zastosowaniu ataku oraz po oczyszczeniu zbioru z potencjalnie zmienionych etykiet, wykorzystując do tego metodę SHAP. Zaadaptowano metrykę S, zaproponowaną przez A. Rosenfelda, która mierzy stabilność modelu wobec ataków zatrucia danych [41].

Analiza przedstawiona w tym rozdziale jest ważna z punktu widzenia zapewnienia niezawodności i odporności modeli uczenia maszynowego. Kluczowe stają się zatem pytania:

- Jakie są reakcje modeli uczenia maszynowego na celowo wprowadzone zmiany etykiet?
- Czy istnieją istotne różnice w predykcjach modelu w przypadku pełnego ataku Flip-Label oraz w scenariuszach, gdzie atak dotyczy wyłącznie etykiet oznaczonych jako 1 lub 0?
- Jak zmieniają się wartości ważności cech w modelach poddanych atakom Flip-Label?
- Czy możliwe jest powiązanie tych zmian z wynikami analizy metodą SHAP, aby uzyskać głębsze zrozumienie wpływu takich ataków na działanie modelu?
- Jakie modele wykazują większą stabilność w przypadku celowych ataków Flip-Label?

9.1 Analiza wydajności modeli wobec ataków Flip-Label oraz po oczyszczeniu etykiet.

Atak odwrócenia etykiet Flip-Label występuje w sytuacjach, gdy zmodyfikowane etykiety mogą wprowadzić błąd w działaniu naszego systemu uczącego [128]. W tym eksperymencie przyjmuje się najbardziej niekorzystny scenariusz, w którym atakujący posiada pełną wiedzę na temat sposobu działania systemu detekcji ataków DDoS, użytych w nim danych treningowych, klasyfikatorów oraz cech. Atakujący może dysponować oddzielnym zestawem danych testowych, pochodzących z tego samego rozkładu co dane treningowe i testowe naszego systemu. Możliwe jest też uzyskanie nielegalnego dostępu do kodu źródłowego systemu detekcji przez wstrzyknięcie złośliwego kodu, który powoduje odwrócenie etykiet dla określonego zbioru testującego. Chociaż te założenia mogą wydawać

się nierealne, pozwalają one na sprawdzenie wydajności i odporności użytych klasyfikatorów binarnych w obliczu ataków. W pierwszym scenariuszu, algorytm ataku typu Flip-Label wykorzystuje przeciwstawne przypisanie nowych etykiet do wylosowanych próbek danych. W drugim scenariuszu następuje celowany atak, w którym algorytm wybiera próbki oznaczone etykietą 1 i zmienia ich etykiety na 0. W trzecim scenariuszu, próbki oznaczone etykietą 0 są zmieniane na etykietę równą 1. W każdym z tych przypadków, próbki są dobierane losowo z dostępnych indeksów, przy czym każda próbka jest wybierana tylko raz i nie podlega ponownej modyfikacji. W eksperymentach wybrano reprezentatywny, procentowy poziom odwrócenia etykiet (5%, 10%, 20% i 30%) co umożliwia zbadanie wpływu ataku zmiany etykiet na wydajność i niezawodność systemu w różnych warunkach, rozpoczynając od niewielkich zakłóceń, aż po bardziej skrajne przypadki zachowując porównywalność wyników. Po skutecznie przeprowadzanym ataku, model jest ponownie trenowany w celu oceny zmian w dokładności i podobieństwie predykcji między oryginalnymi i po ataku. Poniżej przedstawiono autorski algorytm pozwalający na wykonanie ataku Flip-Label(Algorytm 7).

Algorytm 7: Algorytm przeprowadzenia ataków Flip-Label

Wejście: Wczytaj model z wybranym zbiorem cech(lista *selected_features)*

Procedura:

1. Wybierz rodzaj ataku: *atak losowej modyfikacji etykiet, atak 1 na 0, atak 0 na 1*
2. Iteruj przez każdą etykietę (0 i 1), która może być zamieniona.
3. Dla wybranego procenta ataku (np.5%,10%,20%,30%):

    1. Znajdź indeksy próbek (*indices_label_to_flip*), które mają etykietę równą etykiecie do zamiany (*flip_label*) oraz wylosuj określoną liczbę indeksów z, zgodnie z wybranym procentem permutacji.
    2. Skopiuj oryginalne etykiety (*y_test*) do listy *permuted_labels* i zamień etykiety wybranych próbek.
    3. Oblicz predykcje modelu z zainfekowanymi etykietami.
    4. Oblicz i zapisz metryki wydajności dla zainfekowanego modelu.
    5. Oblicz wartości SHAP dla zainfekowanego modelu i wyświetl wykres istotności cech (*SHAP summary plot*).

Koniec procedury

Wyjście: Wyniki metryk wydajności oraz wartości SHAP z wykresem istotności cech dla zainfekowanego modelu.

Zaadaptowano także autorski mechanizm detekcji etykiet (Algorytm 8), które potencjalnie mogły zostać zmodyfikowane w wyniku ataków Flip-Label, prowadząc do negatywnego wpływu na wydajność klasyfikatorów uczenia maszynowego. Mechanizm ten w literaturze nazywany jest label-sanitization (sanityzacja etykiet, oczyszczenie etykiet) i działa w oparciu o algorytm k najbliższych sąsiadów (k-NN), przy czym jego zadanie polega na lokalizacji prawdopodobnie odwróconych etykiet oraz zastępowania ich przez etykietę dominującą spośród k najbliższych próbek [129, 130]. Innymi słowy, algorytm zakłada, że etykieta próbki zatrutej będzie zbliżona do etykiety większości próbek w jej otoczeniu.

Algorytm 8: Algorytm sanityzacji etykiet

Wejście: Wczytaj model z zainfekowanym zbiorem cech oraz wygenerowanym zbiorem zaufanym *D_trusted* określonym procentowo.

Procedura:

1. Utwórz model k najbliższych sąsiadów z liczbą sąsiadów równą *k* oraz wybranym algorytmem przeszukania.
2. Dopasuj model na zbiorze treningowym *X_train.*
3. Zidentyfikuj indeksy próbek w zbiorze testowym, dla których etykiety w *permuted_labels* są równe 1 (co oznacza potencjalne zanieczyszczenie).
4. Dla każdej potencjalnie zanieczyszczonej próbki:
    1. Znajdź k najbliższych sąsiadów tej próbki w zbiorze zaufanym *D_trusted* na podstawie ich odległości od zanieczyszczonej próbki.
    2. Wyznacz dominującą etykietę (najczęściej występującą) wśród tych sąsiadów.
    3. Zastąp potencjalnie zanieczyszczoną etykietę dominującą etykietą znalezioną wśród sąsiadów.
    4. Zwróć sanityzowane etykiety (*sanitized_labels*) i zamień etykiety wybranych próbek w zainfekowanym zbiorze.
    5. Oblicz predykcje modelu z oczyszczonymi etykietami.
    6. Oblicz i zapisz metryki wydajności dla oczyszczonego modelu.
    7. Oblicz wartości SHAP dla oczyszczonego modelu i wyświetl wykres istotności cech (*SHAP summary plot*).

Koniec procedury

Wyjście: Wyniki metryk wydajności oraz wartości SHAP z wykresem istotności cech dla oczyszczonego modelu.

Dla każdej z potencjalnie zatrutych próbek, algorytm oblicza odległość między tą próbką, a wszystkimi próbkami w zbiorze zaufanym, które zostały wcześniej wybrane jako próbki referencyjne. W omawianym scenariuszu przyjęto, że 70% danych treningowych będzie stanowiło zbiór zaufany, co jest kompromisem pomiędzy ilością danych uznanych za wystarczająco niezawodne, a skutecznością identyfikacji najbliższych sąsiadów. W wyniku eksperymentów wstępnych co przedstawiono w tabeli 9.1, dla 30% i 50% zaufanych próbek danych treningowych nie osiągnięto znaczącej skuteczności w łagodzeniu skutków ataków odwrócenia etykiet. Dopiero przy użyciu 70% danych zaufanych uzyskano lepsze wyniki oczyszczania danych. Użycie zbyt małego zbioru zaufanego może nie być wystarczające do wiarygodnej identyfikacji sąsiadów, podczas gdy zbyt duży zbiór może spowodować, że proces będzie zbyt zależny od istniejących danych. Zatem użycie 90% lub 100% danych zaufanych mogłoby prowadzić do nadmiernego zaufania do danych, co nie jest pożądane.

Tabela 9.1.: Wpływ rozmiaru zbioru zaufanego na skuteczność sanityzacji etykiet dla zbioru SelectDVC oraz modelu NGBoost.

Klasyfikator NGBoost, Zbiór : SelectDVC, Atak Flip-Label									
	30% zbiór zaufany			**50% zbiór zaufany**			**70% zbiór zaufany**		
% infekcji zbioru	**Oryginalny zbiór**	**Zainfekowany zbiór**	**Oczyszczony zbiór**	**Oryginalny zbiór**	**Zainfekowany zbiór**	**Oczyszczony zbiór**	**Oryginalny zbiór**	**Zainfekowany zbiór**	**Oczyszczony zbiór**
5%	0,94202	0,897469	0,900236	0,940527	0,896879	0,902848	0,939688	0,894578	0,9021637
10%	0,94202	0,853012	0,87608	0,940527	0,852546	0,878692	0,939688	0,852981	0,8814275
20%	0,94202	0,764441	0,830224	0,940527	0,764223	0,835447	0,939688	0,761176	0,8315612
30%	0,94202	0,677081	0,785768	0,940527	0,676366	0,789623	0,939688	0,676988	0,7906485

Również zaproponowano i przeprowadzono dodatkowy eksperyment w celu wybrania wartości k najbliższych sąsiadów. Zdecydowano, iż użycie trzech najbliższych sąsiadów jest odpowiednim wyborem, gdyż większa wartość(k=5, k=7) pomimo uwzględnienia odległych próbek(które mogą zawierać szum), nie poprawiała dokładności detekcji, a także wpływała negatywnie na czas działania algorytmu. Wyniki przedstawiono w tabeli 9.2.

Tabela 9.2.: Wpływ liczby najbliższych sąsiadów k na skuteczność sanityzacji etykiet dla zbioru SelectDVC oraz modelu NGBoost.

Klasyfikator NGBoost, Zbiór : SelectDVC, Atak Flip-Label									
	k = 3			k = 6			k = 9		
% infekcji zbioru	Oryginalny zbiór	Zainfekowany zbiór	Oczyszczony zbiór	Oryginalny zbiór	Zainfekowany zbiór	Oczyszczony zbiór	Oryginalny zbiór	Zainfekowany zbiór	Oczyszczony zbiór
5%	0,939688	0,895394	0,902164	0,941926	0,897469	0,902879	0,940558	0,897003	0,903459
10%	0,939688	0,853633	0,881428	0,941926	0,853883	0,879718	0,940558	0,854038	0,866707
20%	0,939688	0,762024	0,831561	0,941926	0,764068	0,832712	0,940558	0,765778	0,831758
30%	0,939688	0,677122	0,790649	0,941926	0,676833	0,785395	0,940558	0,676864	0,775359

Zaproponowany w niniejszej rozprawie doktorskiej algorytm label-sanitization został oparty na implementacji klasyfikatora k-NN z pakietu Scikit-Learn. W pakiecie tym dla k-NN dostępne są różne algorytmy przeszukiwania, takie jak ‘auto’, ‘ball_tree’, ‘kd_tree’ i ‘brute’ [131].

Aby dokonać świadomego wyboru algorytmu przeszukiwania w procesie oczyszczenia 30% etykiet dla modelu NGBoost, przeprowadzono eksperyment, którego wyniki przedstawiono w tabeli 9.3. Zakres ten został wybrany, gdyż dla 30% etykiet zaobserwowano znaczne różnice w wynikach. Domyślnie klasyfikator k-NN wykorzystuje tryb 'auto', który automatycznie dobiera najbardziej odpowiedni algorytm w zależności od charakterystyki danych. W trybie auto oraz przy wyborze ‘ball_tree’ biblioteka Scikit-Learn przełącza się na metodę brute force. W związku z tym konieczne było manualne wymuszenie użycia algorytmu ball_tree.

Tabela 9.3.: Wpływ wyboru algorytmu przeszukiwania k-NN na skuteczność oczyszczenia etykiet dla zainfekowanego modelu NGBoost w 30%.

Klasyfikator NGBoost, Zbiór : SelectDVC, 30% Label Sanitization				
Algorytm k-NN	Dokładność	Precyzja	Czułość	F1
ball_tree	0.815581670088	0.87766627340	0.815581670086	0.819438619743
kd_tree	0.825747683889	0.88292977481	0.825747683889	0.829370264392
brute force	0.827561151526	0.88297286631	0.825861151526	0.829401340348

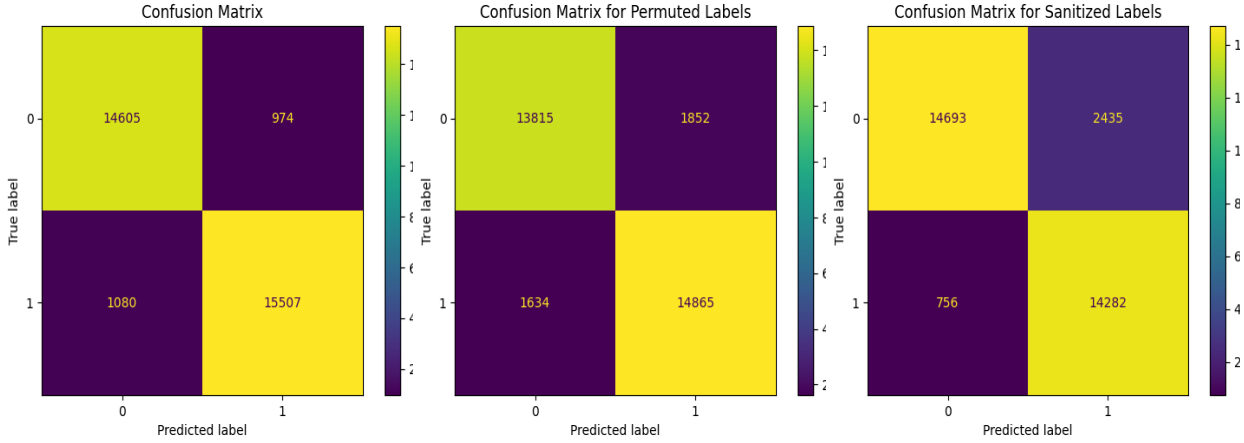
Jak wynika z tabeli 9.3, różnice w algorytmach są minimalne i porównywalne, aczkolwiek metoda 'brute force' jest najprostsza w implementacji, przez co użycie jej może być

korzystne w omawianym algorytmie sanityzacji etykiet. Jednak można uznać, iż wybór metody przeszukiwania nie jest istotny, gdyż nie wpływa znacząco na proces oczyszczenia etykiet.

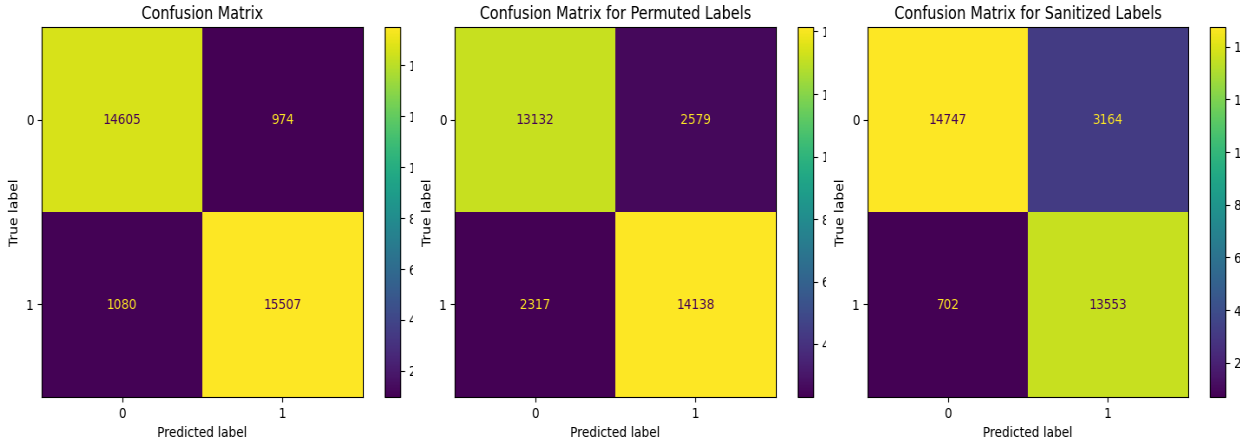
Poniżej na rysunku 9.1 przedstawiono, wygenerowane macierze pomyłek dla modelu NGBoost oraz zbioru SelectDVC w następujących scenariuszach:

- Macierz bazowa (Oryginalny zbiór SelectDVC) - przed wprowadzeniem ataku Flip-Label.
- Macierz po permutacji etykiet (Zbiór SelectDVC po ataku przeciwstawnym Flip-Label) – po przeprowadzaniu ataku Flip-Label w proporcjach infekcji 5%, 10%, 20% i 30% zbioru.
- Macierz po sanityzacji etykiet(Zbiór SelectDVC po użyciu mechanizmu Sanitization Label) – po przeprowadzaniu oczyszczenia etykiet po przeprowadzaniu ataku Flip-Label w proporcjach infekcji 5%, 10%, 20% i 30% zbioru.

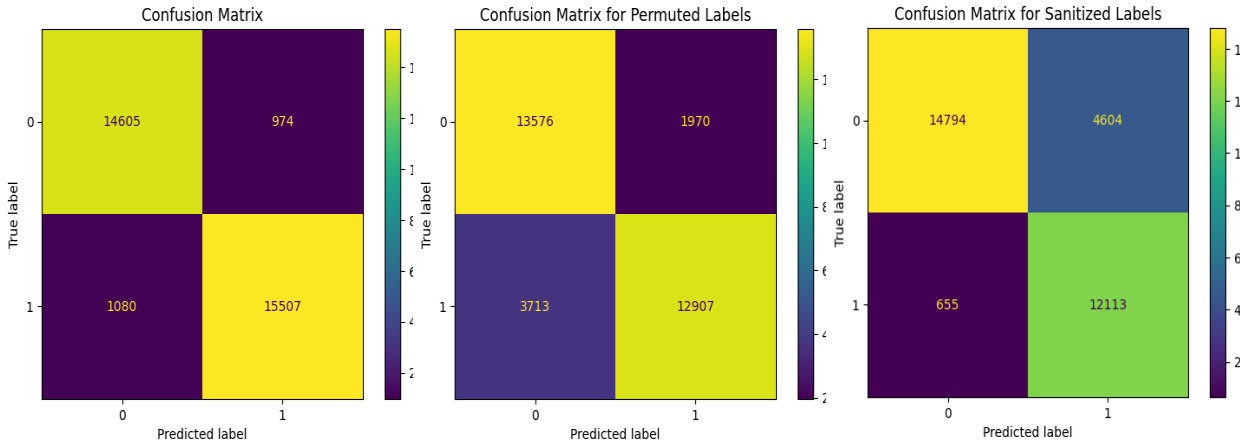
Oryginalny zbiór SelectDVC | Zbiór SelectDVC po ataku przeciwstawnym Flip-Label | Zbiór SelectDVC po użyciu mechanizmu Sanitization Label



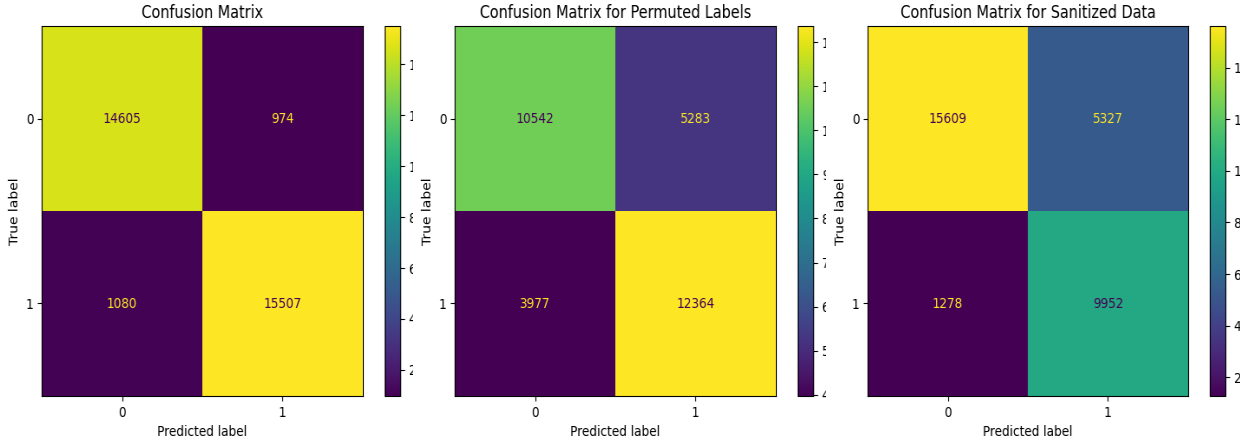
a) NGBoost 5%



b) NGBoost 10%



c) NGBoost 20%

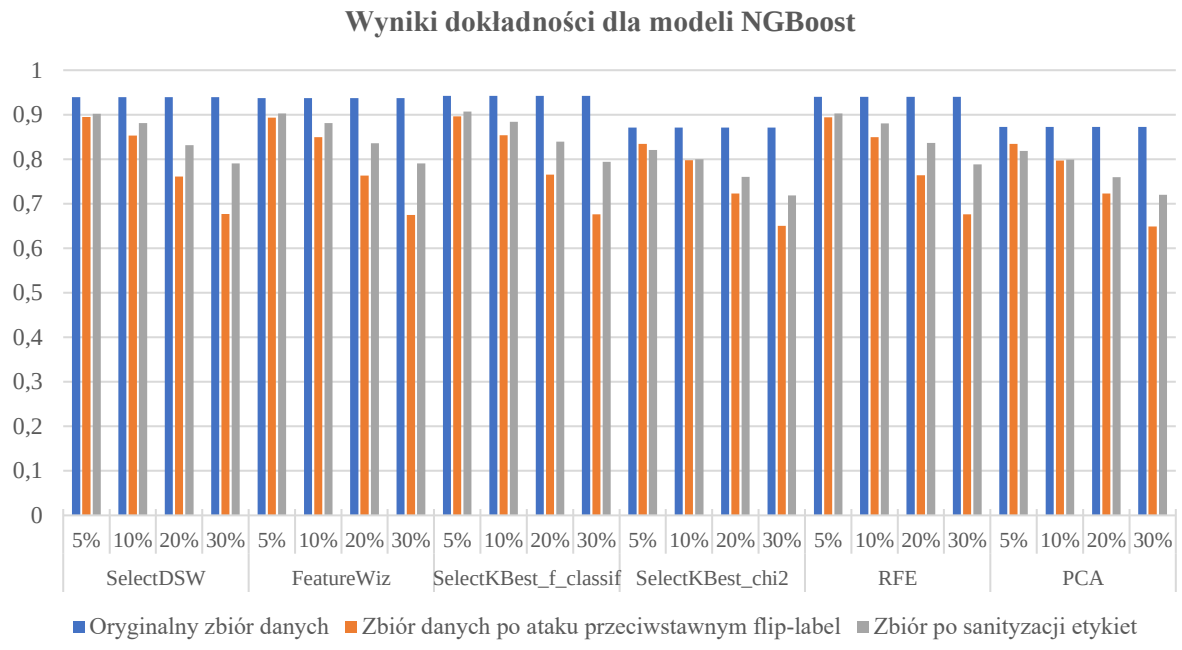


d) NGBoost 30%

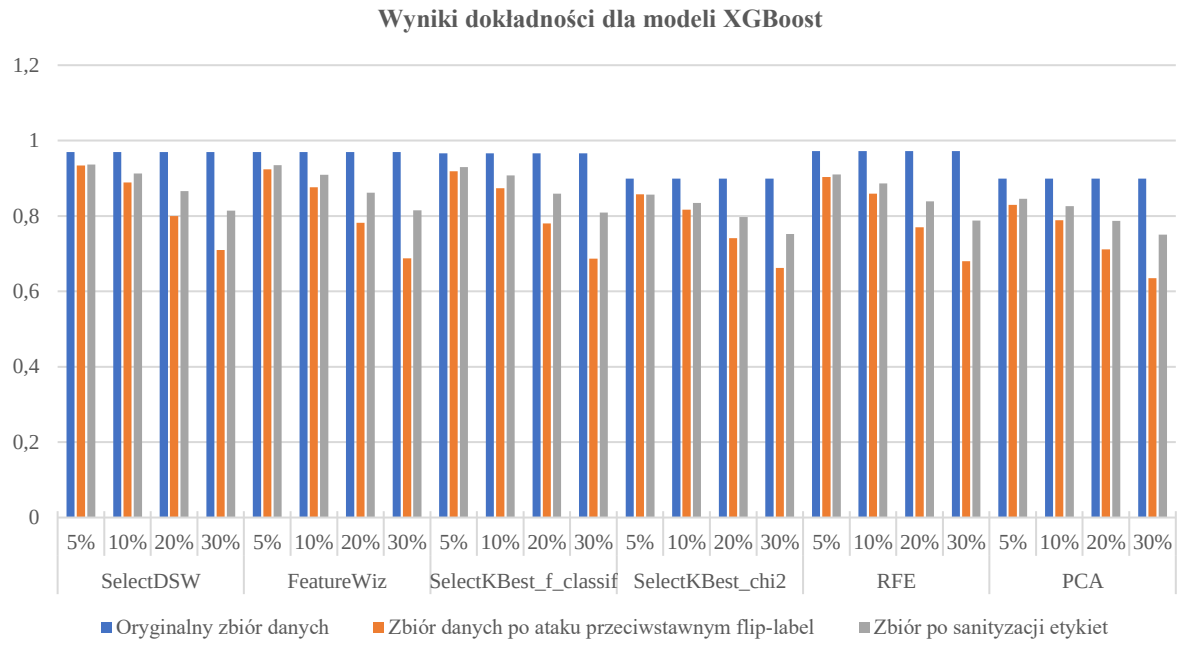
Rys. 9.1. Macierze pomyłek dla modelu NGBoost oraz zbioru SelectDVC przed atakiem Flip-Label, po ataku Flip-Label oraz procesie oczyszczania etykiet.

Dla modelu NGBoost oraz pozostałych klasyfikatorów zaobserwowano, iż przeprowadzenie ataku Flip-Label polegającym na przeciwstawnej zmianie etykiet ma niestety destrukcyjny wpływ na wydajność modelu. Wraz ze wzrostem procentu permutowanych etykiet, obserwuje się spadek liczby prawidłowych klasyfikacji (TP i TN) oraz wzrost liczby błędnych klasyfikacji (FP i FN) Sanityzacja etykiet okazuje się skuteczna w częściowym przywracaniu wydajności modelu(obserwuje się redukcję liczby FN) jednak jej efektywność malała przy wyższych odsetkach permutacji. Przy 30% infekcji dla modelu NGBoost, nie udało się skutecznie zneutralizować skutków ataków odwrócenia etykiet, a wyniki pozostawały dalekie od ideału.

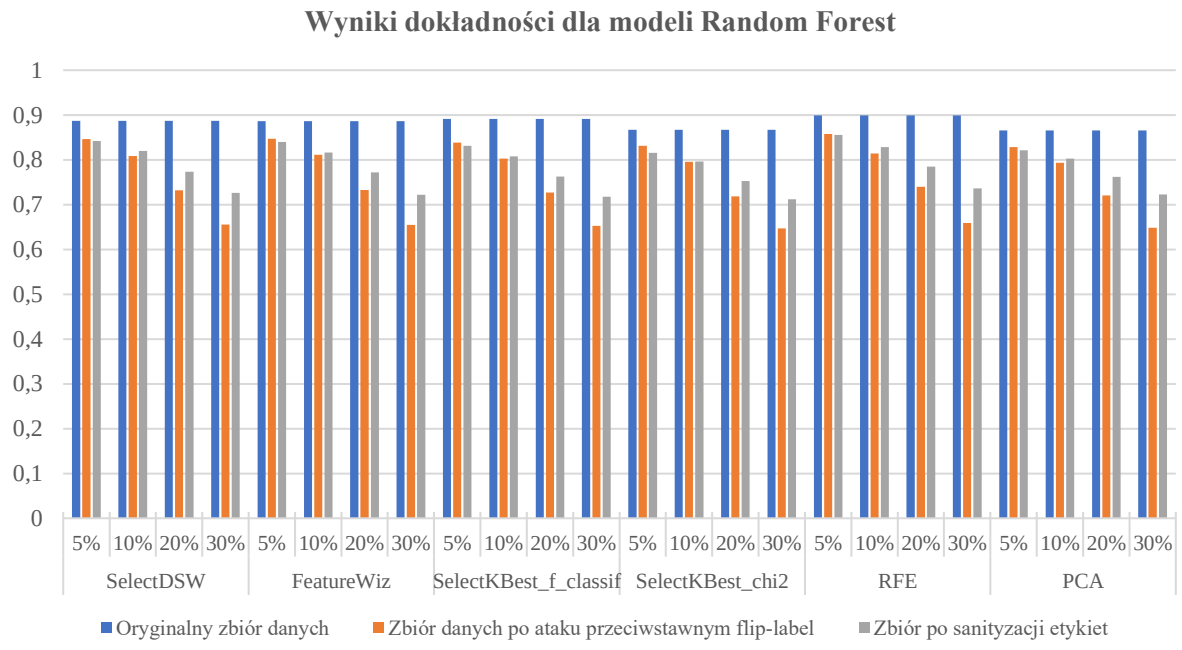
Następnie przeprowadzono analizę wydajności wszystkich modeli wobec ataku typu Flip-Label i po procesie sanityzacji etykiet. Wyniki przedstawione na rysunkach 9.2-9.4 ilustrują różnice w wartości dokładności dla permutacji etykiet wynoszących 5%, 10%, 20% oraz 30%.



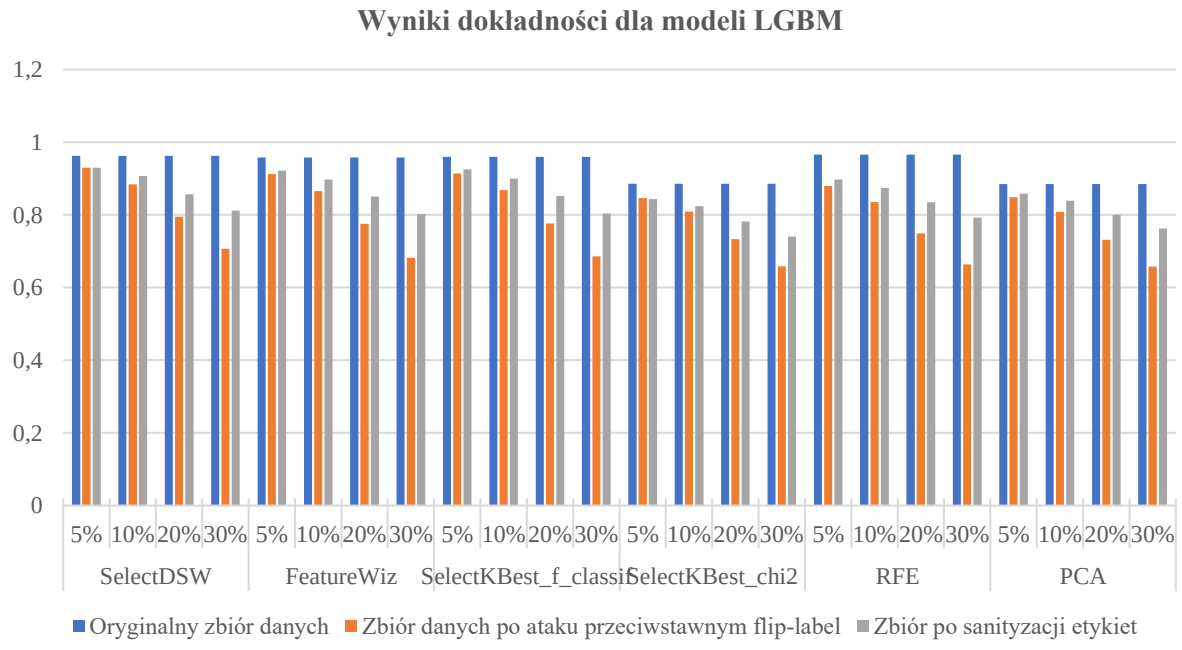
a) Wyniki dokładności dla modeli NGBoost



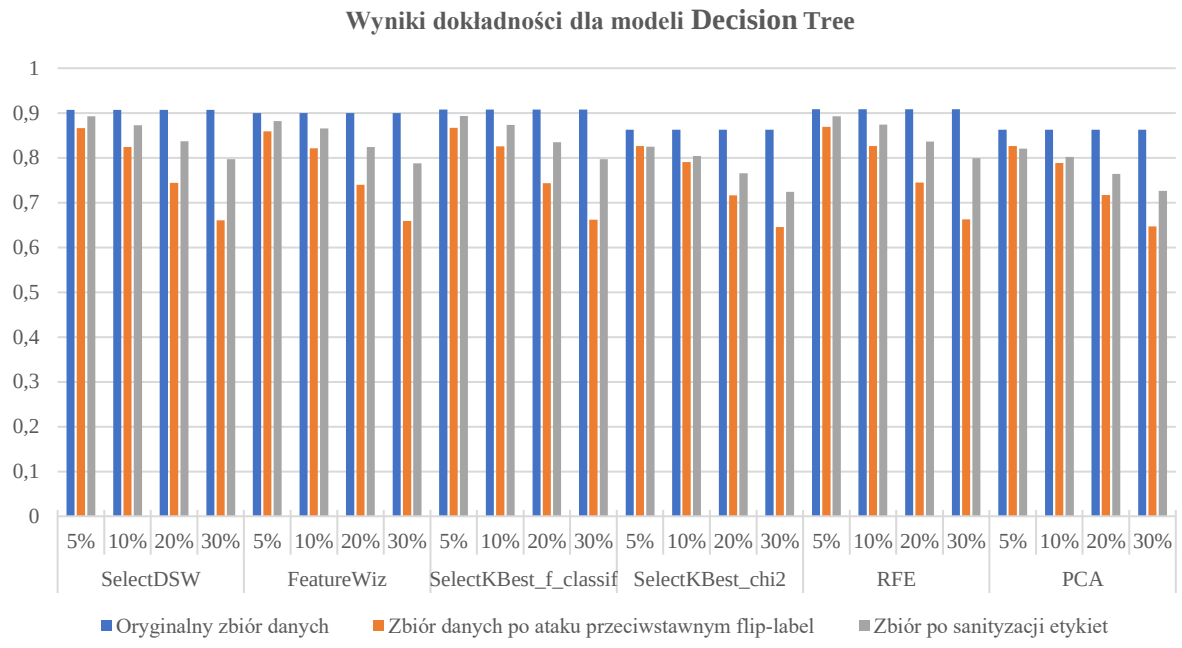
b) Wyniki dokładności dla modeli XGBoost



c) Wyniki dokładności dla modeli Random Forest



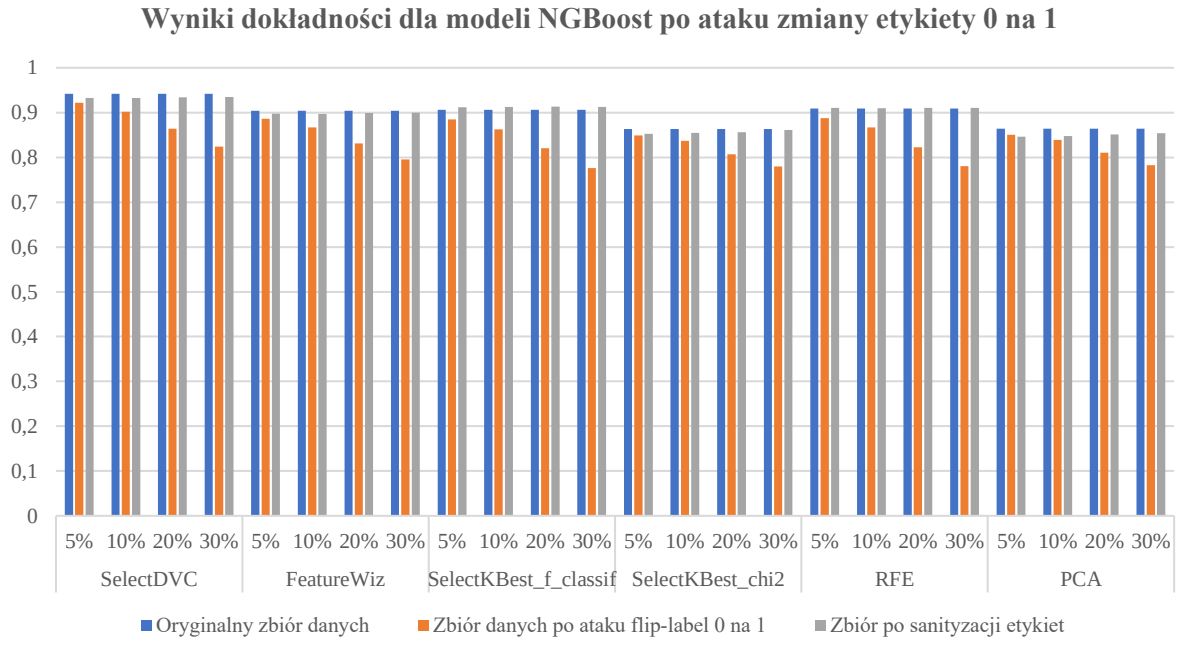
d) Wyniki dokładności dla modeli LightGBM



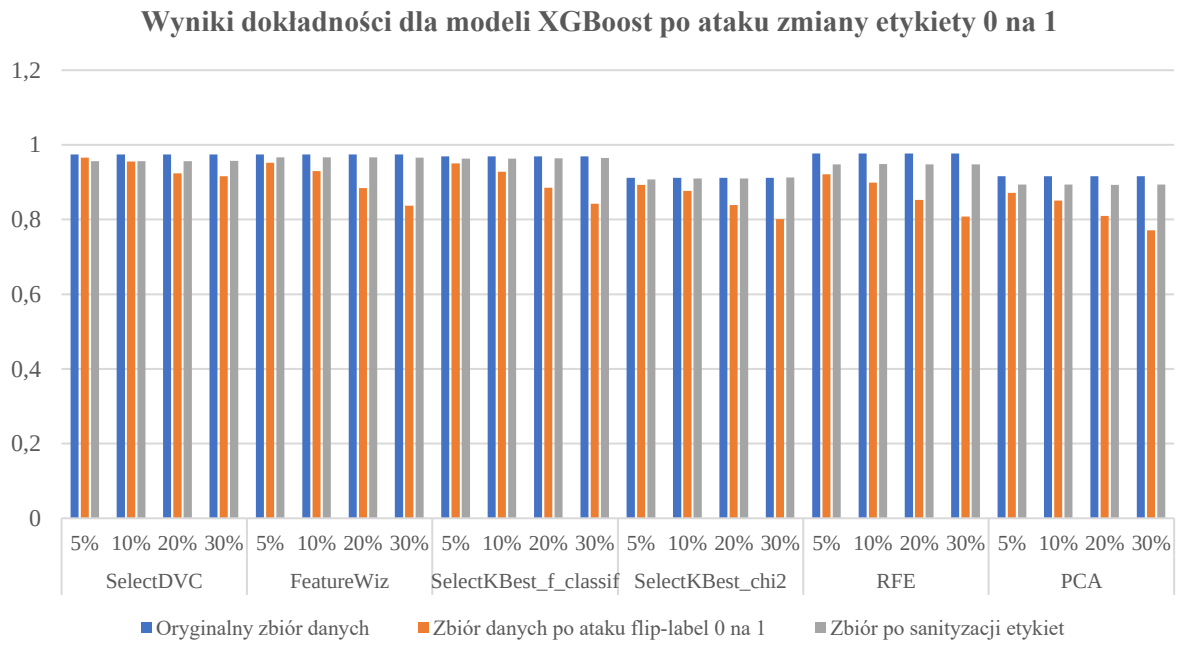
e) Wyniki dokładności dla modeli Decision Tree

Rys. 9.2. Wyniki dokładności dla wszystkich zbiorów danych i modeli na oryginalnym zbiorze, po przeprowadzeniu ataku przeciwstawnego ataku Flip-Label oraz po procesie oczyszczenia etykiet.

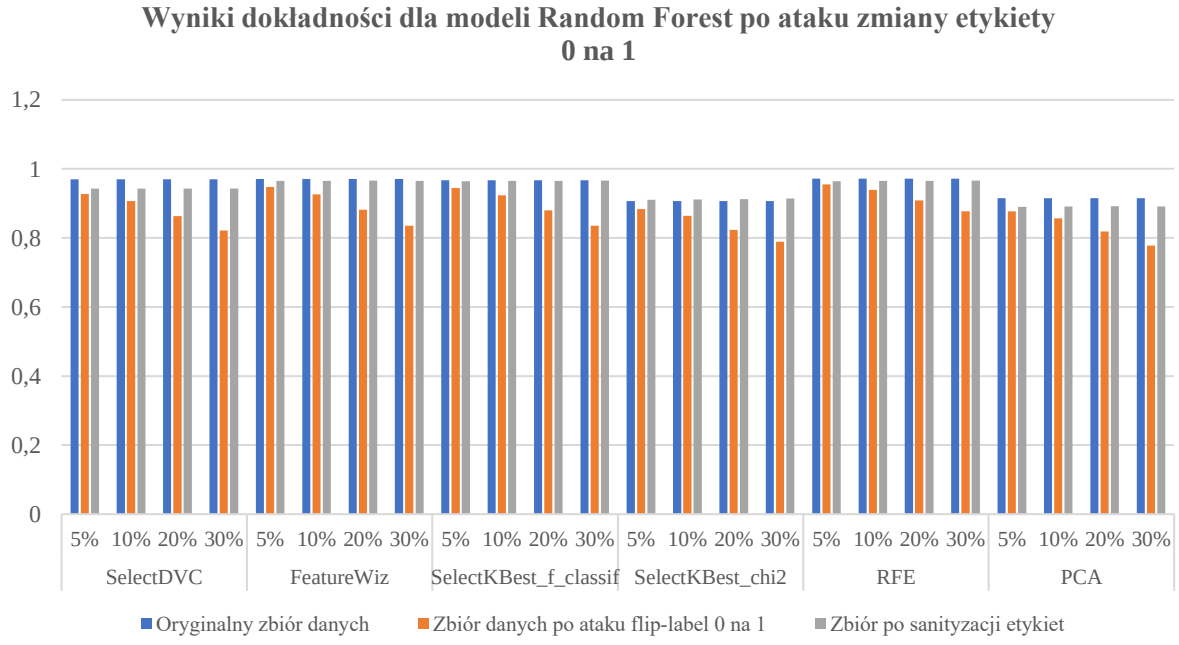
Jak można zauważyć na rysunku 9.2 nawet niewielki odsetek (10%) błędnych etykiet prowadzi do znacznego pogorszenia zdolności modelu do prawidłowego rozróżniania klas. W wielu przypadkach po oczyszczeniu danych przez algorytm label-sanitization udało się znacznie zwiększyć dokładność predykcji ataków DDoS. Następnie zweryfikowano dokładność modeli w przypadku ataków Flip-Label 0 na 1 (rysunek 9.3) oraz 1 na 0(rysunek 9.4).



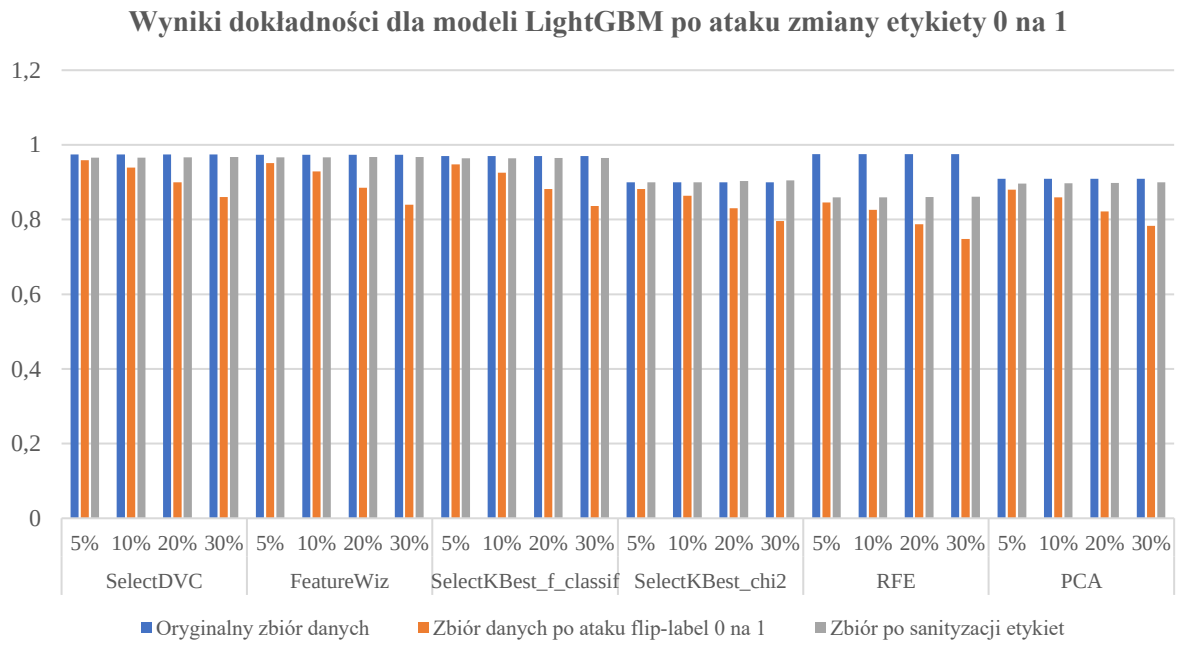
a) Wyniki dokładności dla modeli NGBoost po ataku zmiany etykiety 0 na 1



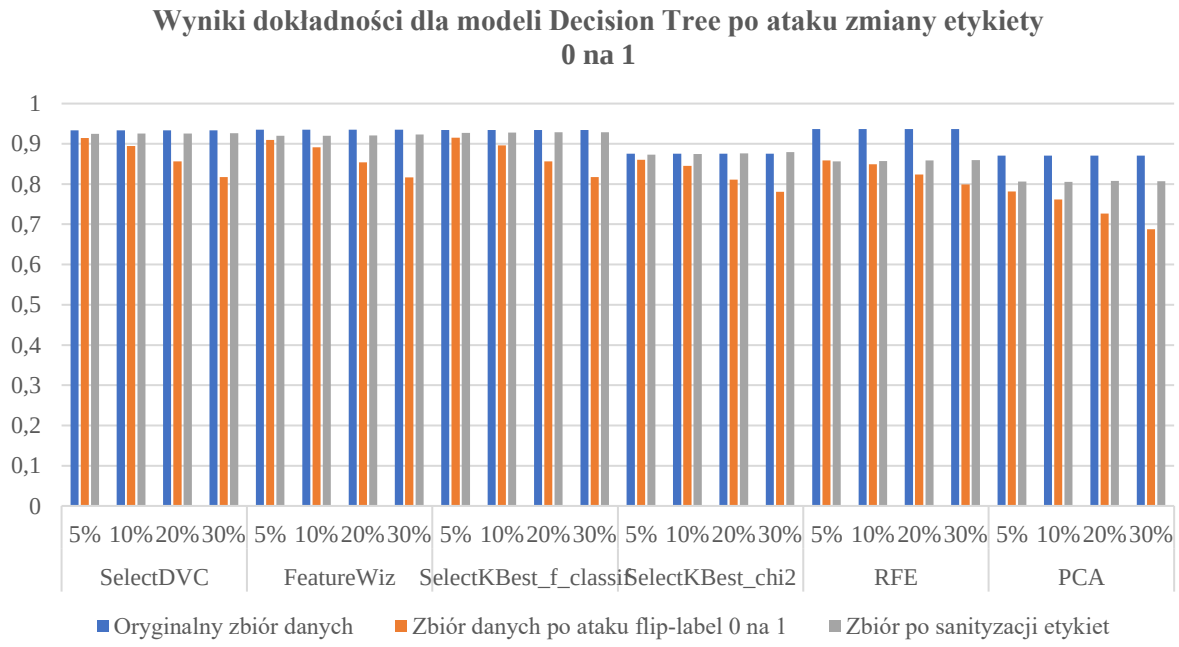
b) Wyniki dokładności dla modeli XGBoost po ataku zmiany etykiety 0 na 1



c) Wyniki dokładności dla modeli Random Forest po ataku zmiany etykiety 0 na 1

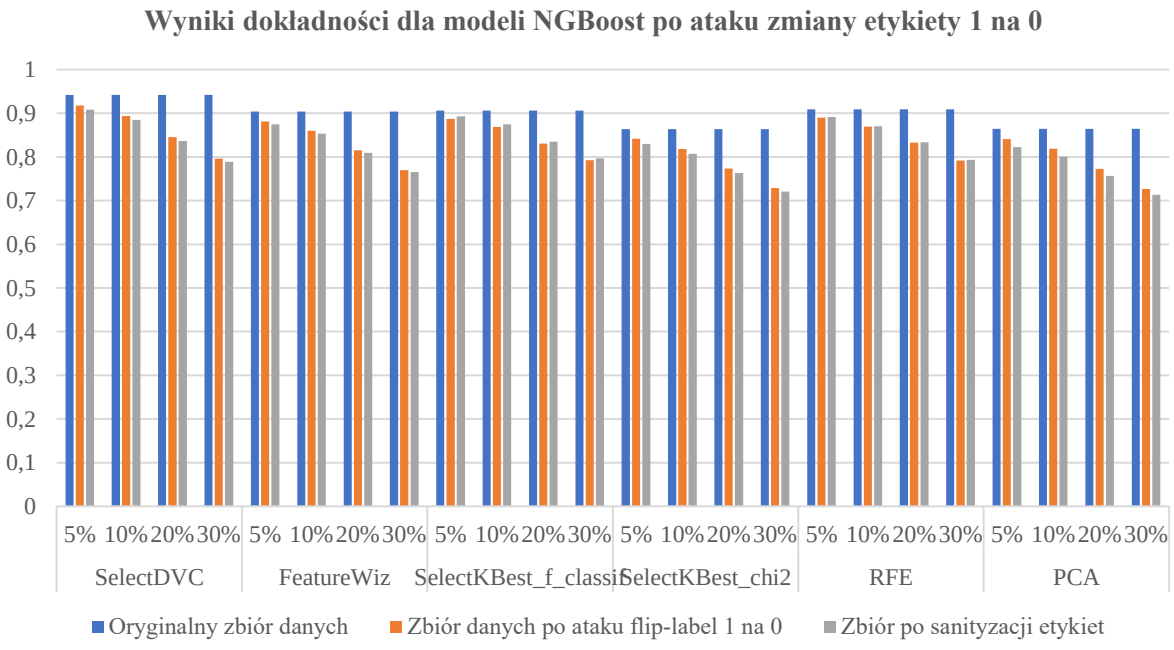


d) Wyniki dokładności dla modeli LightGBM po ataku zmiany etykiety 0 na 1

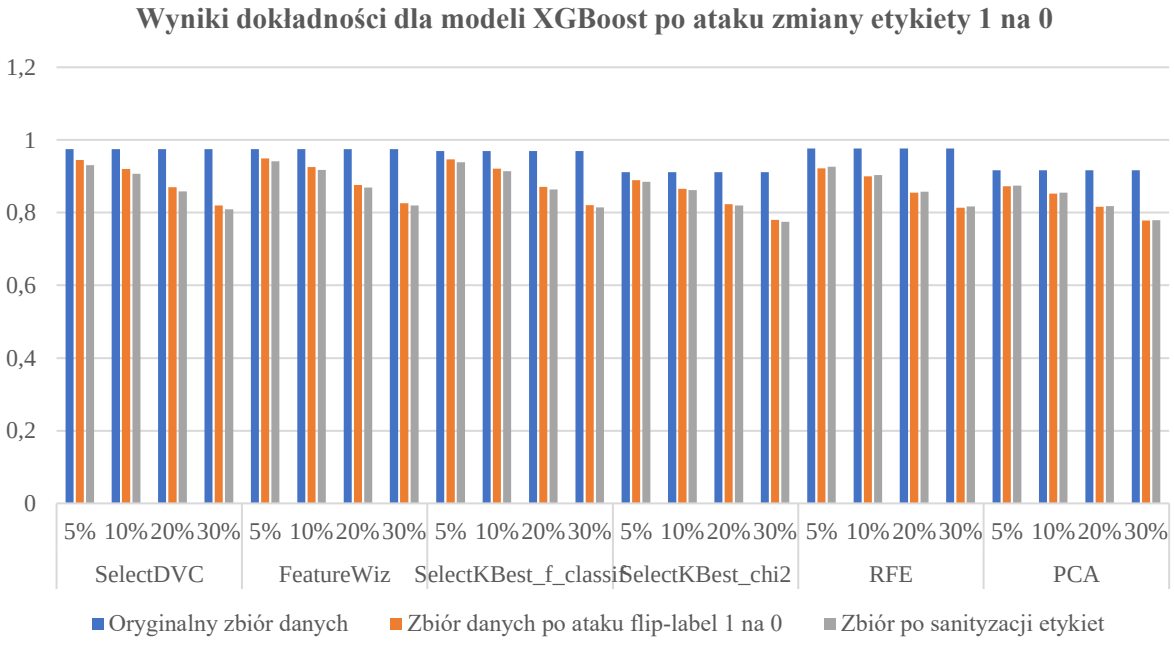


e) Wyniki dokładności dla modeli Decision Tree po ataku zmiany etykiety 0 na 1

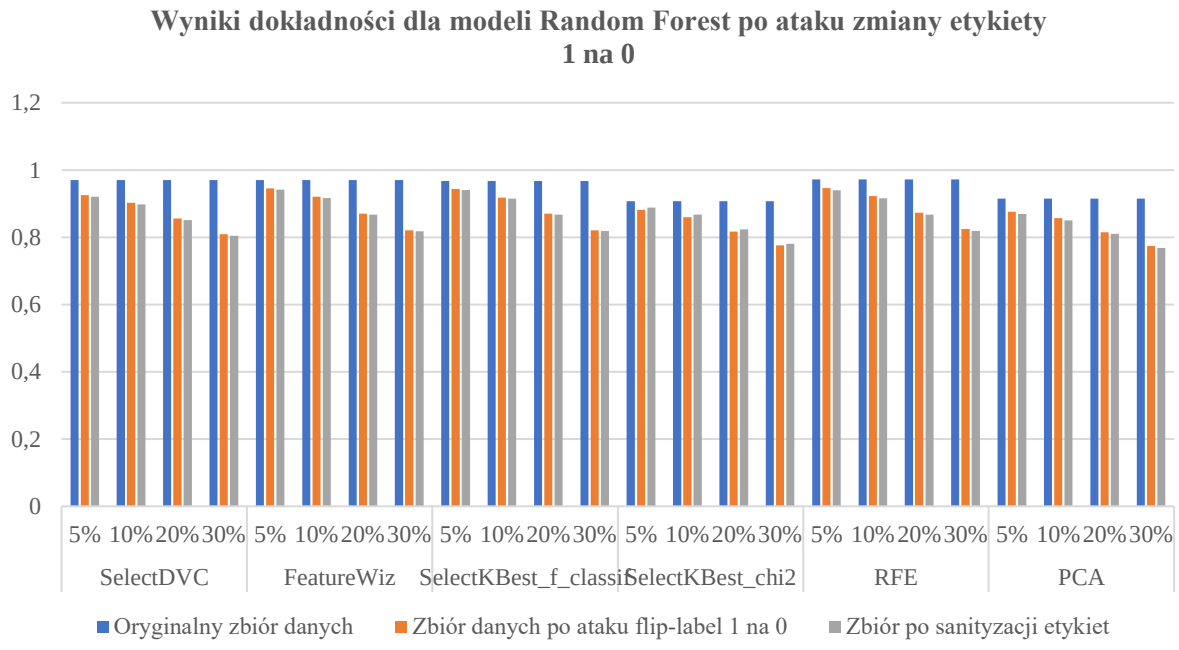
Rys. 9.3. Wyniki dokładności dla wszystkich zbiorów danych i modeli na oryginalnym zbiorze, po przeprowadzeniu ataku Flip-Label zamiany etykiet 0 na 1 oraz po procesie oczyszczenia etykiet.



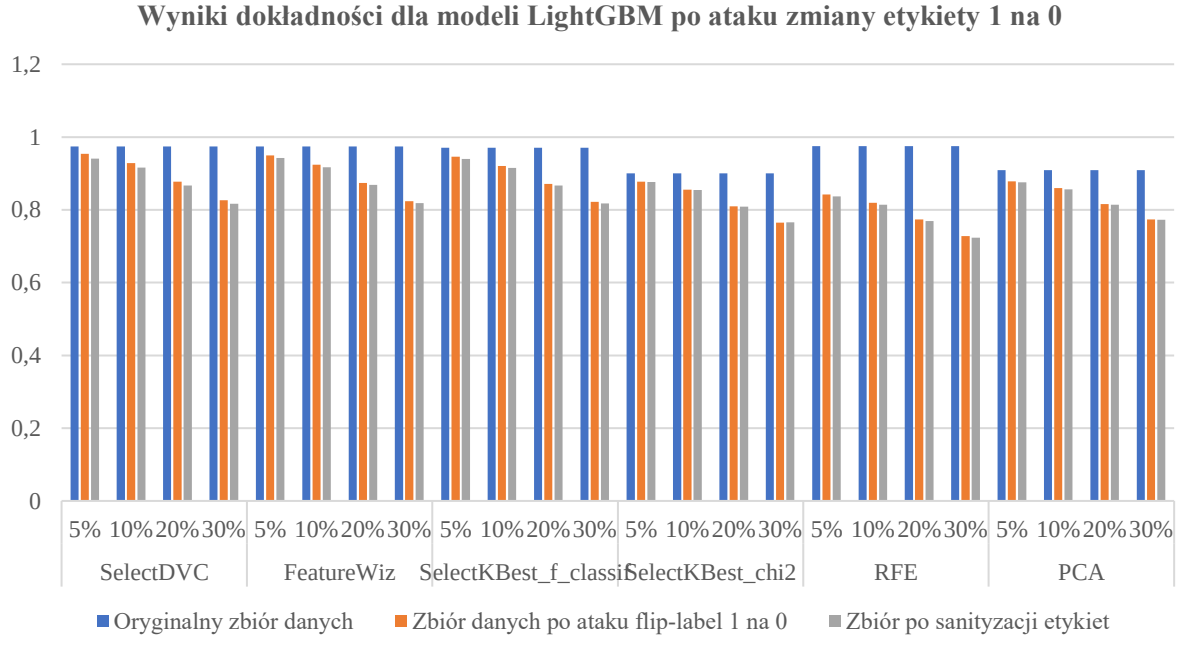
a) Wyniki dokładności dla modeli NGBoost po ataku zmiany etykiety 1 na 0



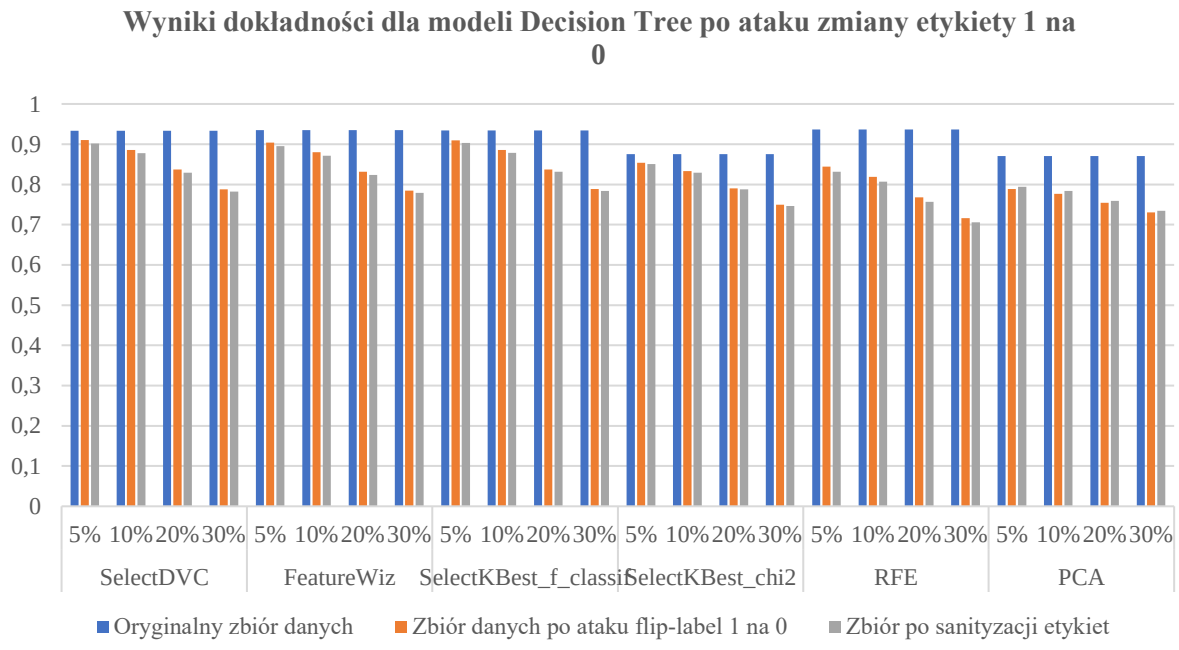
b) Wyniki dokładności dla modeli XGBoost po ataku zmiany etykiety 1 na 0



c) Wyniki dokładności dla modeli Random Forest po ataku zmiany etykiety 1 na 0



d) Wyniki dokładności dla modeli LightGBM po ataku zmiany etykiety 1 na 0



e) Wyniki dokładności dla modeli Decision Tree po ataku zmiany etykiety 1 na 0

Rys. 9.4. Wyniki dokładności dla wszystkich zbiorów danych i modeli na oryginalnym zbiorze, po przeprowadzeniu ataku Flip-Label zamiany etykiet 0 na 1 oraz po procesie oczyszczenia etykiet.

Ataki polegające na odwróceniu etykiet z etykiety 0 na 1 w 5% i 10% miały zauważalny, ale stosunkowo niewielki wpływ na większość modeli. Ataki 20% i 30% spowodowały bardziej znaczące pogorszenie wydajności modelu, czasami zmniejszając dokładność o ponad 10%. Model XGBoost korzystający ze zbioru SelectDVC wykazał się niezwykłą odpornością, zachowując wysoką dokładność przy zainfekowaniu 30% zbioru danych ataku (0,946465 w porównaniu z oryginalnym 0,974538). W przeciwieństwie do tego modelu, model Random Forest korzystający ze zbioru RFE doświadczył dramatycznego spadku pod wpływem ataku (np. 0,87732 przy 30% zainfekowanym zbiorze w porównaniu z oryginalnym 0,97170).

W wielu przypadkach wydajność oczyszczonego zestawu danych była bliska lub nawet nieco lepsza niż wydajność oryginalnego zestawu danych. Jednak dotkliwość tego wpływu była różna, przy czym niektóre kombinacje modeli i technik wyboru cech wykazały większą odporność. Możemy również uznać, iż modele wykorzystujące zbiory cech SelectDVC i FeatureWiz lepiej utrzymywały swoją dokładność.

W przypadku ataków polegających na zmianie etykiet z etykiety 1 na 0 wyniki eksperymentów okazały się całkowicie destrukcyjne nawet dla modelu XGBoost z korzystającego ze zbioru SelectDVC, który choć był odporny na atak zmiany etykiet 0 na 1 przy zainfekowanym w 30% zbiorze, to w tym samym przypadku(również dla infekcji 30%)

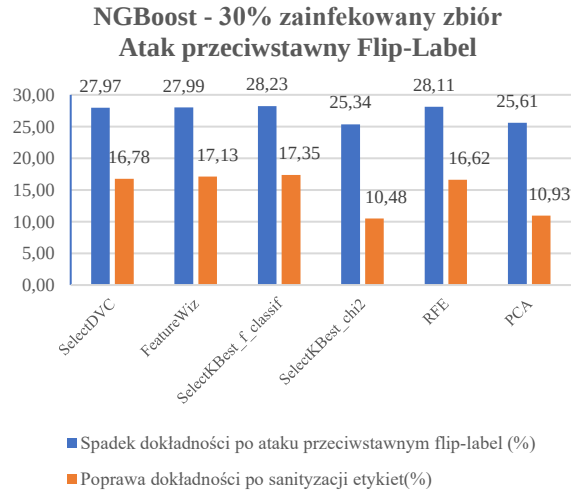
osiągnął dokładność wynoszącą 0.81946 przy pierwotnej dokładności na czystym zbiorze danych wynoszącej 0.97453. Dla większości ocenianych modeli spadki dokładności przy 30% zainfekowanym zbiorze były dość znaczne. Mimo to proces sanityzacji wcale nie przynosił poprawy W tym samym przykładzie sanityzacja etykiet nie była skuteczna, gdyż dokładność wyniosła 0.809053 co jest wynikiem niższym od dokładności wynoszącej 0.81946 po ataku Flip-Label 1 na 0. Niestety niemożliwe było odtworzenie takich samych wyników dokładności po sanityzacji jak na oryginalnym zbiorze danych.

By lepiej zobrazować te wnioski, sprawdzono procentową różnicę dokładności dla modeli przy 30% zainfekowanego zbioru danych według poniższych metryk (wzory 9.1) dla modeli NGBoost oraz XGBoost.

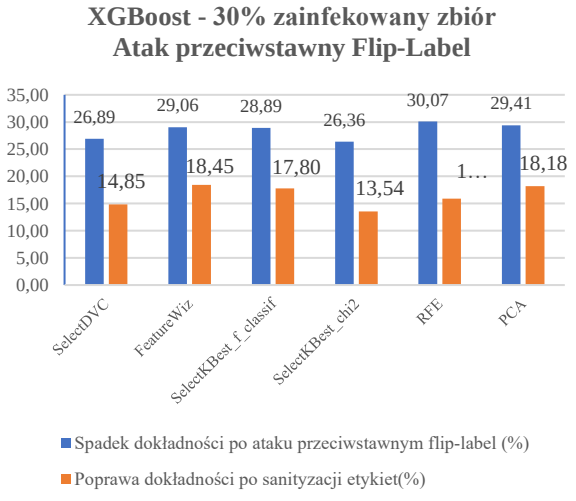
$$Spadek\ dokładności\ po\ ataku = \frac{Oryginalna\ dokkładność - Dokładność\ po\ ataku}{Oryginalna\ dokładność} \times 100$$

$$Poprawa\ dokładności\ po\ sanityzacji = \frac{Dokładność\ po\ sanityzacji - Dokładność\ po\ ataku}{Dokładność\ po\ ataku} \times 100 \quad (9.1)$$

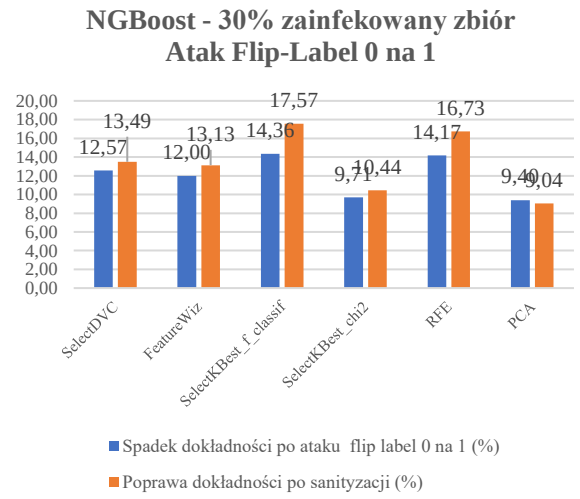
Najpierw obliczono procentowy spadek dokładności po ataku przeciwstawnym w stosunku do oryginalnej dokładności, a następnie procentową poprawę dokładności po sanityzacji w stosunku do dokładności po ataku. Przedstawione wyniki na rysunku 9.5 wskazują na różnice w odporności modeli na ataki oraz efektywności sanityzacji w przywracaniu dokładności.



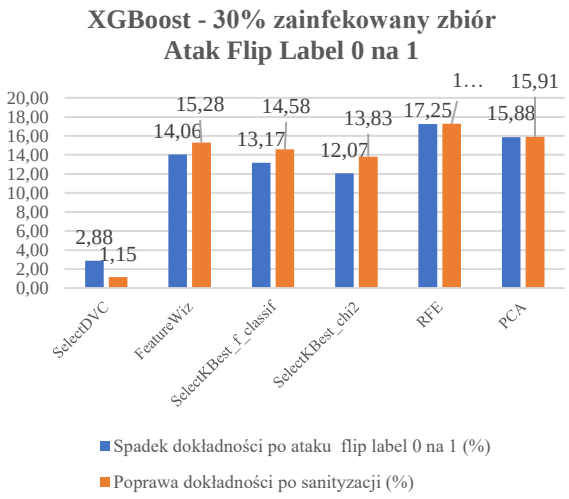
a) NGBoost



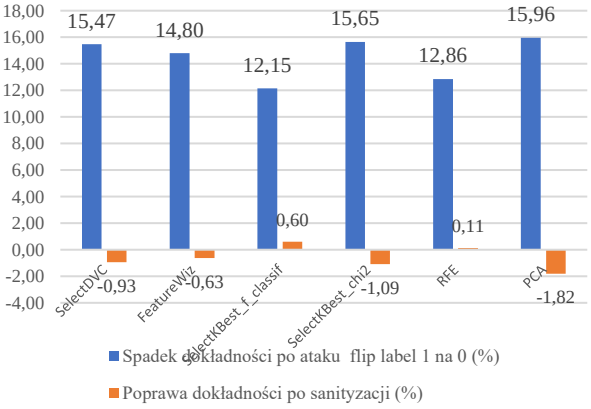
b) XGBoost



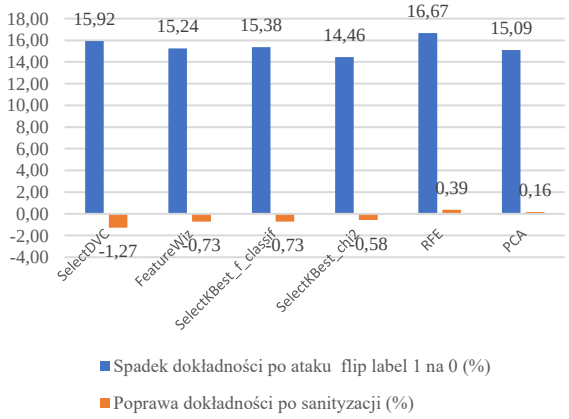
c) NGBoost



d) XGBoost



e) NGBoost



f) XGBoost

Rys. 9.5. Porównanie procentowych różnice dokładności dla modeli NGBoost i XGBoost- 30% zainfekowany zbiór atakami przeciwstawnymi Flip-Label, atakami Flip-Label 0 na 1 oraz atakami Flip-Label 1 na 0.

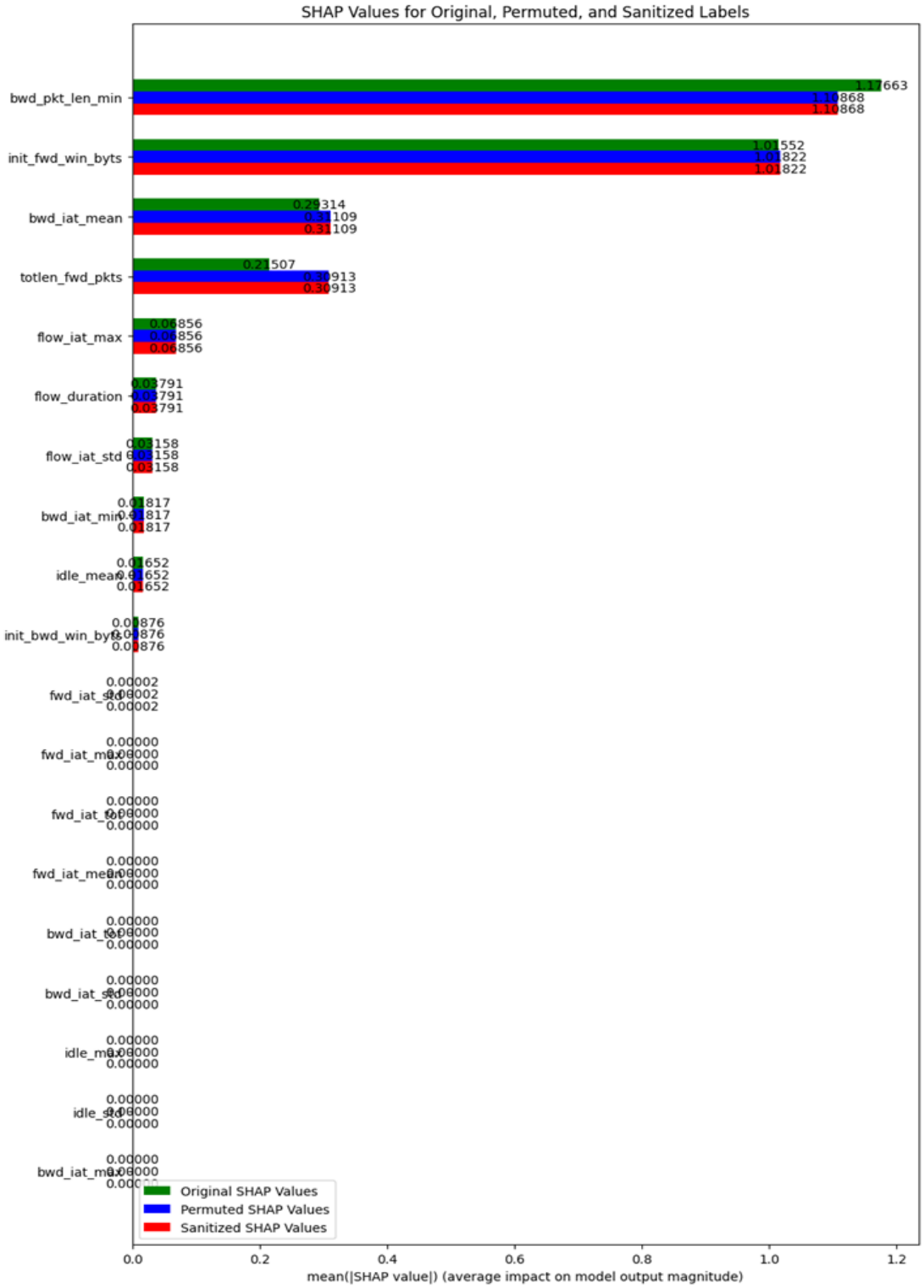
Reakcje modeli uczenia maszynowego na celowo wprowadzone zmiany etykiet oraz różnice w predykcjach w przypadku różnych scenariuszy ataku Flip-Label w znacznym stopniu są negatywne. Badania wykazały, iż stopień tego wpływu może się różnić w zależności od konkretnego scenariusza ataku. Największa różnica w procentowym spadku dokładności między porównywanymi modelami(rysunek 9.5) występowała po ataku przeciwstawnym Flip-Label, gdzie dokładność w zależności od zbioru selekcji cech wahała się między 25% a 30% Poprawa dokładności po sanityzacji wynosiła średnio między 16% a 18%. W przypadku różnic dokładności przy występowaniu ataków zmiany etykiet z 0 na 1 zaobserwowano bardzo wyważone wyniki po ataku i po sanityzacji co pokazuje, iż stosunkowo dobrze radzą sobie modele z tymi atakami.

Atak 1 na 0 okazał się znacznie bardziej szkodliwy niż atak 0 na 1, choć średnie wyniki spadku dokładności po ataku Flip-Label 1 na 0 oscylują wokół 16%, co jest porównywalne z konkurencyjnym atakiem 0 na 1. W wielu przypadkach sanityzacja po ataku zmiany etykiety 1 na 0 nie przyniosła poprawy, a nawet spowodowała dalszy spadek dokładności. Eksperyment jasno pokazuje, iż w tym przypadku mechanizm label-sanitization nie działa zgodnie z oczekiwaniami, gdyż dla większości modeli dokładność modeli po sanityzacji była zazwyczaj gorsza niż po samym ataku. Problem z mechanizmem oczyszczenia etykiet może być uzasadniony faktem, iż modele klasyfikacyjne często są trenowane tak, aby faworyzować większościową klasę, ponieważ minimalizuje to ogólny błąd klasyfikacji. W takim przypadku zmiana etykiety z 1 na 0 ma większy wpływ, ponieważ wprowadza błędną informację do klasy, która już jest mniej reprezentowana w danych. Może to być spowodowane specyfiką algorytmu k-NN zaimplementowanego w mechanizmie label-sanitization, który nie radzi sobie dobrze z przypadkami, gdzie klasa mniejszościowa jest bardziej podatna na zakłócenia.

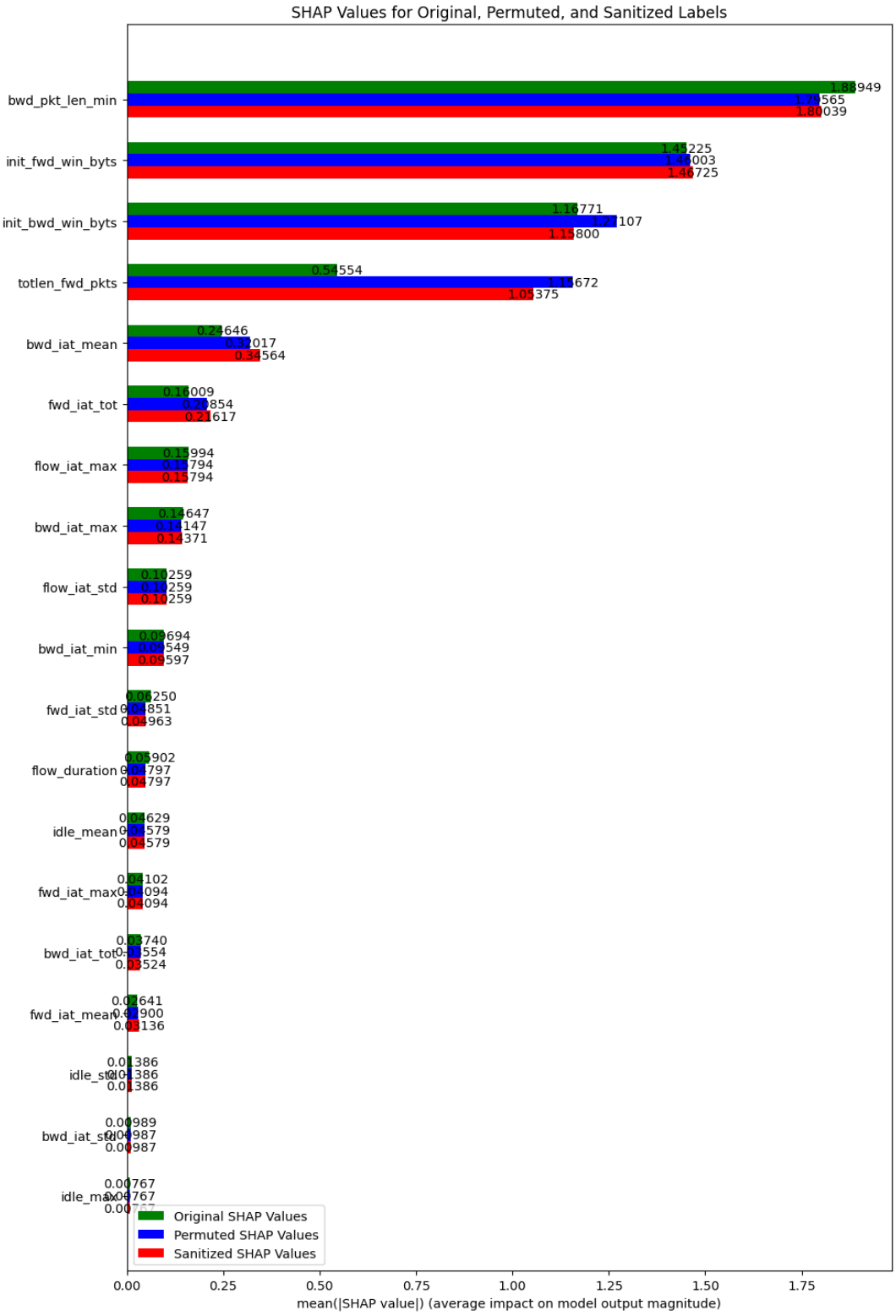
9.2 Analiza wyjaśnialności wpływu ataków Flip-Label oraz procesu sanityzacji etykiet za pomocą metody SHAP.

W końcowej fazie eksperymentów sprawdzano zmiany wartości SHAP w rankingu ważności cech po wykonaniu ataku przeciwstawnego Flip-Label i procesie label-sanitization dla wybranych 3 modeli NGBoost, XGBoost oraz LightGBM opartych na zbiorze SelectDVC. Aby ocenić wpływ tych manipulacji, porównano wartości SHAP przed i po zastosowaniu ataków oraz procesu sanityzacji, analizując zmiany w rankingu ważności cech. Wyniki eksperymentu wykazały, że metoda SHAP pozwala na precyzyjną identyfikację zmian w istotności poszczególnych cech, umożliwiając weryfikację wpływu modyfikacji etykiet na ogólną wyjaśnialność modeli.

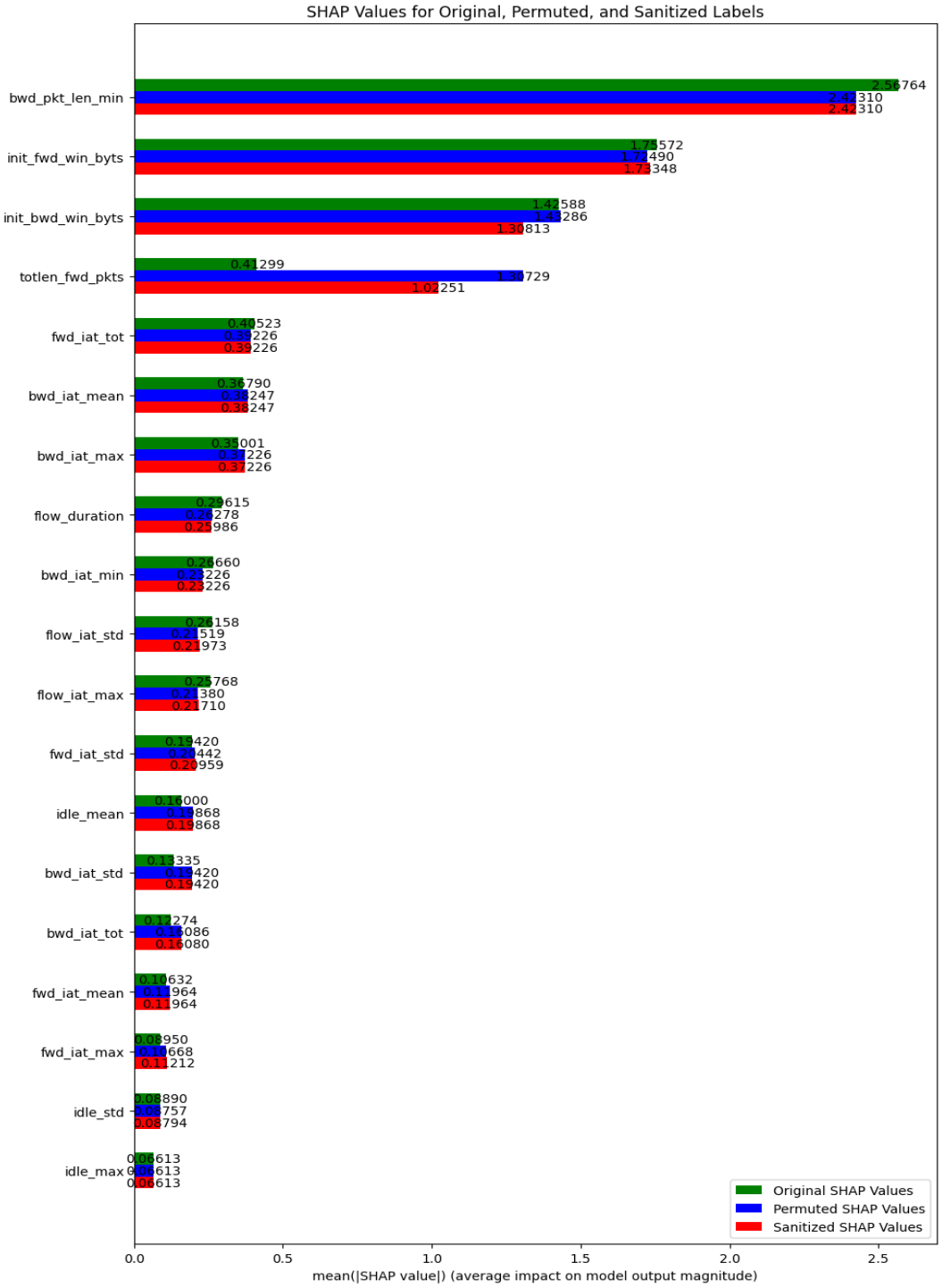
Na poniższych diagramach (rysunki 9.6, 9.7, 9.8) możemy zobaczyć rezultaty tych eksperymentów.



Rys. 9.6. Diagram zmian wartości metody SHAP Feature Importance dla zbioru NGBoost dla oryginalnego zbioru danych(Original SHAP Values), po ataku przeciwstawnym Flip-Label(Permuted SHAP Values) oraz po oczyszczeniu danych(Sanitized SHAP Values).



Rys. 9.7. Diagram zmian wartości metody SHAP Feature Importance dla zbioru XGBoost dla oryginalnego zbioru danych(Original SHAP Values), po ataku przeciwstawnym Flip-Label(Permuted SHAP Values) oraz po oczyszczeniu danych(Sanitized SHAP Values).



Rys. 9.8. Diagram zmian wartości metody SHAP Feature Importance dla zbioru LightGBM dla oryginalnego zbioru danych(Original SHAP Values), po ataku przeciwstawnym Flip-Label(Permuted SHAP Values) oraz po oczyszczeniu danych(Sanitized SHAP Values).

W większości przypadków atak przeciwstawny Flip-Label prowadził do zwiększenia wartości SHAP dla wybranych cech względem wartości uzyskanych na oryginalnym zbiorze danych. Po oczyszczeniu etykiet wartości SHAP utrzymywały się na takim samym bądź bliskim poziomie jak w przypadku wartości po ataku Flip-Label, choć w niektórych

przypadkach obserwowano także spadek tych wartości. Co interesujące, kolejność ważności cech w żadnym z 3 przypadków nie została zmieniona, ani zaburzona, mimo wzrostu wartości SHAP w niektórych przypadkach. Świadczy to o tym, że celowe zmiany etykiet w ataku Flip-Label nie prowadzą do znaczącego zwiększenia wpływu innych cech na predykcje modelu poza cechami dominującymi. Dla cechy najbardziej dominującej bwd_pkt_len_min zaobserwowano widoczny spadek wartości SHAP.

Ostateczne wyniki potwierdzają, że metoda SHAP jest skutecznym narzędziem do wyjaśniania i oceny odporności modeli na ataki zatruwania danych. Umożliwia ona identyfikację zmian w istotności poszczególnych cech oraz analizę wpływu modyfikacji etykiet na wyjaśnialność modeli, co pozwala lepiej zrozumieć konsekwencje manipulacji danymi.

9.3 Analiza wyjaśnialności modeli w oparciu o metrykę S.

Stabilność czyli odporność modelu klasyfikacyjnego na zmiany w etykietach danych treningowych jest metryką, która pokazuje jak bardzo zmieniają się predykcje modelu, gdy część etykiet w danych treningowych zostanie celowo zmieniona w wyniku ataków Flip-Label oraz po dokonaniu procesu sanityzacji. Stabilny model to taki, którego predykcje nie zmieniają się znacząco pomimo wprowadzonych zmian w etykietach. Model niestabilny będzie natomiast wykazywał duże różnice w predykcjach po takiej manipulacji.

Stabilność w metryce S można wyrazić za pomocą miar podobieństwa Jaccarda i/lub Tanimoto. Podobnie jak w przypadku metryki F, która została zaimplementowana i opisana w poprzednim rozdziale, zaleca się wykorzystanie parametru λ do zastosowania kary dla metryki wydajnościowej w zależności od stabilności [41].

$$S = \lambda \times (1 - similarity) \qquad (9.2)$$

gdzie:

S - metryka S,

λ - parametr kary określający wpływ stabilności na wynik,

similarity - miara podobieństwa wyjaśnień między różnymi próbkami (np, Jaccard lub Tanimoto).

Parametr similarity wyznaczyć można za pomocą wzoru na wskaźnik Jaccarda:

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|} \qquad (9,3)$$

gdzie:

J(A,B) to podobieństwo Jaccarda między zbiorami A i B.

|A∩B| oznacza liczbę elementów wspólnych dla zbiorów A i B.

|A∪B| oznacza liczbę elementów w sumie zbiorów A i B.

Parametr λ został użyty w opisanych eksperymentach do zastosowania kary, przy czym założono stałą wartość kary wynoszącą 0,5, a parametr similarity wykorzystuje podobieństwo Jaccarda między oczyszczonym zbiorem danych a zainfekowanym zbiorem danych.

W ramach niniejszej rozprawy doktorskiej zaadaptowano zaproponowaną przez A. Rosenfelda metrykę S w celu określenia stabilności modelu wobec ataków zatrucia danych [41]. Zastosowanie metryki S umożliwi identyfikację modeli o większej stabilności, co jest ważne w przypadku zadania detekcji ataków DDoS, gdzie etykiety danych mogą być podatne na błędy lub manipulacje.

Tabela 9.4.: Wpływ procentowej infekcji zbioru danych atakiem Flip-Label na metrykę S oraz podobieństwo między oczyszczonym a zainfekowanym zbiorem danych SelectDVC.

Klasyfikator	**% infekcji zbioru**	**S**	**Podobieństwo (similarity) między oczyszczonym zbiorem danych a zainfekowanym zbiorem danych**
Decision Tree	0,05	0,119371	0,102399
Decision Tree	0,1	0,151964	0,122034
Decision Tree	0,2	0,207279	0,158206
Decision Tree	0,3	0,258941	0,198933
Random Forest	0,05	0,1202	0,127874
Random Forest	0,1	0,145587	0,145737
Random Forest	0,2	0,193187	0,183731
Random Forest	0,3	0,236293	0,222239
XGBoost	0,05	0,059063	0,059388
XGBoost	0,1	0,095187	0,081812
XGBoost	0,2	0,158791	0,125022
XGBboost	0,3	0,215972	0,173279
LightGBM	0,05	0,062309	0,064761
LightGBM	0,1	0,098425	0,086231

LightGBM	0,2	0,162641	0,133069
LightGBM	0,3	0,216276	0,174577
NGBoost	0,05	0,090475	0,087758
NGBoost	0,1	0,121844	0,106507
NGBoost	0,2	0,185037	0,151332
NGBoost	0,3	0,234368	0,188437

W przeprowadzonych eksperymentach pożądane są wyniki metryki S bliższe zeru, gdyż wskażą one nam modele, które są mniej podatne na zmiany w etykietach danych treningowych. Analizując wyniki przedstawione w tabeli 9.4 przy wykorzystaniu zbioru danych SelectDVC, zaobserwowano wyraźny trend wzrostu wartości metryki S wraz ze wzrostem poziomu permutacji etykiet. Oznacza to, że im większy zakres manipulacji danymi, tym mniej stabilne stają się modele co w konsekwencji prowadzi do obniżenia skuteczności detekcji ataków DDoS. W eksperymencie modele XGBoost oraz LightGBM wyróżniły się najniższymi wartościami metryki S, a z kolei modele Decision Tree oraz Random Forest uzyskały najwyższe wartości w tej metryce. Model oparty na algorytmie NGBoost wykazał umiarkowaną stabilność.

W dalszej części eksperymentów przeprowadzono analizę stabilności modeli w scenariuszach manipulacji etykietami typu "0 na 1" oraz "1 na 0". Co można zaobserwować w poniższych tabelach 9.5 oraz 9.6.

Tabela 9.5.: Wpływ procentowej infekcji zbioru danych atakiem Flip-Label 0 na 1 na metrykę S oraz podobieństwo między oczyszczonym a zainfekowanym zbiorem danych SelectDVC.

Rodzaj ataku Flip-Label	**Klasyfikator**	**% infekcji zbioru**	**S**	**Podobieństwo (similarity) między oczyszczonym zbiorem danych a zainfekowanym zbiorem danych**
0 na 1	Decision Tree	0,05	0,07275	0,066894
0 na 1	Decision Tree	0,1	0,086713	0,066569
0 na 1	Decision Tree	0,2	0,10998	0,066246
0 na 1	Decision Tree	0,3	0,131183	0,065358
0 na 1	LightGBM	0,05	0,036202	0,032152
0 na 1	LightGBM	0,1	0,051359	0,032028
0 na 1	LightGBM	0,2	0,079058	0,030841
0 na 1	LightGBM	0,3	0,102704	0,03024

0 na 1	NGBoost	0,05	0,066458	0,060291
0 na 1	NGBoost	0,1	0,080985	0,06001
0 na 1	NGBoost	0,2	0,104617	0,059041
0 na 1	NGBoost	0,3	0,12697	0,058237
0 na 1	Random Forest	0,05	0,064303	0,053001
0 na 1	Random Forest	0,1	0,078772	0,053018
0 na 1	Random Forest	0,2	0,107531	0,052999
0 na 1	Random Forest	0,3	0,130732	0,052847
0 na 1	XGBoost	0,05	0,029641	0,040445
0 na 1	XGBoost	0,1	0,031457	0,040267
0 na 1	XGBoost	0,2	0,035886	0,03963
0 na 1	XGBoost	0,3	0,039078	0,039072

W przypadku manipulacji "0 na 1", gdzie etykiety prawidłowe (0) są zamieniane na etykiety błędne (1), zaobserwowano, że model XGBoost wykazał się najwyższą stabilnością, utrzymując niskie wartości metryki S nawet przy wysokim poziomie permutacji. Modele LightGBM i NGBoost również wykazały się stosunkowo wysoką stabilnością, podczas gdy Decision Tree i Random Forest były bardziej podatne na tego typu manipulacje.

Tabela 9.6.: Wpływ procentowej infekcji zbioru danych atakiem Flip-Label 1 na 0 na metrykę S oraz podobieństwo między oczyszczonym a zainfekowanym zbiorem danych SelectDVC.

Rodzaj ataku Flip-Label	**Klasyfikator**	**% infekcji zbioru**	**S**	**Podobieństwo (similarity) między oczyszczonym zbiorem danych a zainfekowanym zbiorem danych**
1 na 0	Decision Tree	0,05	0,079335	0,087929
1 na 0	Decision Tree	0,1	0,101283	0,109608
1 na 0	Decision Tree	0,2	0,144514	0,153173
1 na 0	Decision Tree	0,3	0,188728	0,195732
1 na 0	LightGBM	0,05	0,042921	0,055307
1 na 0	LightGBM	0,1	0,066598	0,078629
1 na 0	LightGBM	0,2	0,11389	0,125022
1 na 0	LightGBM	0,3	0,161459	0,171825
1 na 0	NGBoost	0,05	0,072916	0,082246
1 na 0	NGBoost	0,1	0,094499	0,103345
1 na 0	NGBoost	0,2	0,138435	0,146958
1 na 0	NGBoost	0,3	0,18256	0,19049

1 na 0	Random Forest	0,05	0,068423	0,074333
1 na 0	Random Forest	0,1	0,089889	0,095844
1 na 0	Random Forest	0,2	0,13378	0,139605
1 na 0	Random Forest	0,3	0,177487	0,183793
1 na 0	XGBoost	0,05	0,049907	0,063467
1 na 0	XGBoost	0,1	0,072699	0,085523
1 na 0	XGBoost	0,2	0,118895	0,131083
1 na 0	XGBoost	0,3	0,165602	0,17685

W scenariuszu manipulacji "1 na 0", gdzie etykiety błędne (1) są zamieniane na etykiety prawidłowe (0), ponownie model XGBoost okazał się najbardziej stabilny, osiągając najniższe wartości w metryce S. Modele LightGBM i NGBoost również wykazały się dobrą stabilnością, natomiast modele Decision Tree i Random Forest były najmniej odporne na tego typu manipulacje.

Koniecznie należy zwrócić także uwagę na wartości podobieństwa między oczyszczonym zbiorem danych, a zainfekowanym zbiorem danych mierzone miarą Jaccarda. Wysokie wartości podobieństwa wskazują na dużą zgodność między oczyszczonym, a zainfekowanym zbiorem danych, co oznacza, że model jest mniej podatny na zmiany w etykietach. W eksperymentach zaobserwowano, że modele XGBoost i LightGBM utrzymują relatywnie wysokie wartości podobieństwa co przekłada się na ich niskie wartości metryki S i wysoką stabilność. Z kolei modele Decision Tree i Random Forest w kontraście wykazują niższe wartości podobieństwa, co skutkuje wyższymi wartościami metryki S i mniejszą stabilnością.

Należy podkreślić, że wyniki przeprowadzonych eksperymentów związanych ze stabilnością modeli pokazują, iż modele mogą zachowywać się różnie w zależności od typu ataku Flip-Label, zaś zaadaptowana metryka S z powodzeniem może wspomagać ocenę stabilności modeli.

10. Podsumowanie

W niniejszej rozprawie doktorskiej teza brzmi: **"Autorskie propozycje i implementacje kryteriów wyjaśnialnej sztucznej inteligencji umożliwiają skuteczne wyjaśnienie podejmowanych decyzji przez modele uczenia maszynowego w procesie detekcji rozproszonych ataków odmowy usługi w sieciach definiowanych programowo."**

W związku z tą tezą celem pracy było wykazanie, iż odpowiednia integracja algorytmów klasyfikacji, technik selekcji cech oraz metod wyjaśnialności może nie tylko skutecznie wykrywać ataki DDoS, ale również przyczynić się do głębszego zrozumienia, które elementy zaproponowanego systemu bezpieczeństwa SDN są kluczowe.

W ramach badań przeprowadzono serię eksperymentów dotyczących detekcji ataków DDoS w środowiskach SDN z wykorzystaniem zaawansowanych modeli uczenia maszynowego tj. XGBoost, LightGBM, NGBoost, Decision Tree oraz Random Forest z naciskiem na ich wyjaśnialność. Praca wyróżnia się nie tylko analizą wydajności algorytmów i złożoności obliczeniowej, ale przede wszystkim skupieniem na wyjaśnialności procesu decyzyjnego modeli uczenia maszynowego, przez co nawiązuje do aktualnych problemów i osiągnięć sztucznej inteligencji. Istotnym wkładem jest :

- Opracowanie autorskiej metody selekcji cech SelectDVC oraz opracowanie autorskiego zbioru danych DVCDDoS2022.
- Analiza porównawcza algorytmów klasyfikacji zaadaptowanych do zadania detekcji ataków DDoS w sieciach definiowanych programowo.
- Analiza porównawcza wpływu cech sieciowych oraz ich redukcji na predykcje modeli.
- Opracowanie autorskich implementacji metryk D, F, S zdefiniowanych przez A.Rosenfelda oraz autorskich propozycji ich modyfikacji.
- Opracowanie autorskich implementacji metryk monotoniczności, wierności, niewierności oraz niekompletności.
- Opracowanie autorskich algorytmów ataków Flip-Label oraz oczyszczania zainfekowanych etykiet label-sanitization.

Łącznie w pracy przedstawiono 8 autorskich algorytmów.

Przeprowadzone eksperymenty wskazują na to, iż każdy z badanych modeli uczenia maszynowego ma swoje mocne strony i zastosowania, ale również problemy. Algorytm NGBoost pomimo osiągnięcia nie co gorszych wyników w metrykach wydajności niż popularne algorytmy boostingowe (XGBoost, LightGBM) może z powodzeniem być

stosowany w systemach detekcji ataków DDoS ze względu na zapewnienie dobrej wyjaśnialności podejmowanych przez siebie decyzji np. dzięki zastosowaniu metody SHAP.

Zaproponowana autorska metoda selekcji cech SelectDVC, umożliwiła utworzenie zbioru cech, który przyczynił się do osiągnięcia wysokiego poziomu detekcji ataków DDoS przez modele we wszystkich eksperymentach. Zastosowanie zautomatyzowanego narzędzia FeatureWiz do generowania zbioru cech o niskiej korelacji oraz przeprowadzone eksperymenty wykazały, że wysoka korelacja cech nie jest warunkiem koniecznym do osiągnięcia bardzo dobrych wyników przez modele uczenia maszynowego.

Decydującymi cechami w detekcji ataków DDoS są głównie te związane z początkowymi rozmiarami okien TCP, a dodatkowo pozytywny wpływ mogą mieć cechy czasowe oraz parametry związane z ruchem pakietów. Przy czym wybór konkretnych cech decydujących o tym czy atak DDoS ma miejsce, może różnić się w zależności od zastosowanego algorytmu. Co więcej, niektóre cechy mogą mieć zróżnicowany wpływ (negatywny bądź pozytywny) w zależności od użytego algorytmu.

Opracowanie autorskich implementacji metryk D, F, S, zdefiniowanych przez A. Rosenfelda pomimo braku precyzyjnych instrukcji w literaturze, stanowiły istotny element badań w ramach tej rozprawy. Propozycje te wskazują, na to, iż badacze mogą stosować różne metody do ich obliczania, a ocena wyjaśnialności miejscami może być dość subiektywna. Dzięki zastosowaniu metryki D zyskaliśmy wgląd w to, w jakim stopniu zwiększenie transparentności modeli może wpłynąć na ich skuteczność, nie tracąc przy tym na dokładności ich działania. Użycie w metryce D kryteriów informacyjnych AIC oraz BIC, które rzadko pojawiają się w literaturze dla problemów klasyfikacyjnych, dostarczyło wartościowych informacji na temat optymalnego balansu między jakością dopasowania przewidywanych prawdopodobieństw do rzeczywistych klas, a złożonością modelu. Dzięki zastosowaniu metryki F możliwe było ograniczenie liczby cech używanych do wyjaśnienia predykcji modelu razem z analizą SHAP, która oceniała wpływ poszczególnych cech na wynik modelu. Istnieje zatem możliwość identyfikacji optymalnej liczby cech, która zapewni równowagę między wysoką dokładnością a wyjaśnialnością modelu.

Autorskie modele systemu detekcji ataków DDoS nie są całkowicie odporne na perturbacje danych co podkreślono wykorzystując propozycje implementacji metryk monotoniczności, wierności, niewierności i niekompletności. W zależności od poziomu zakłóceń, stabilność modeli maleje. Mimo to, żaden z badanych algorytmów nie wykazuje całkowicie nieprzewidywalnego zachowania w odpowiedzi na losowe zmiany w danych. W przypadku ataków Flip-Label modele stają się bardziej wrażliwe na modyfikacje etykiet w

danych treningowych co prowadzi do pogorszenia wydajności. Spadki te są zgodne z początkowymi założeniami, lecz konieczne jest zwrócenie uwagi na celowane ataki na etykiety zmieniające wartości 1 na 0, które mają szkodliwszy wpływ na model niż ataki zmiany wartości etykiet 0 na 1. Celowe wprowadzenie nawet niewielkich perturbacji cech może prowadzić do zmian w predykcjach i spadków wydajności, ale stopień tych zmian zależy od użytych klasyfikatorów i zastosowanej metody selekcji cech. Warto także dodać, iż stabilność modelu nie zawsze jest proporcjonalna do jego dokładności, a optymalnym rozwiązaniem w systemach detekcji ataków DDoS byłoby użycie algorytmów, które utrzymują dobrą bądź umiarkowaną stabilność na różnych poziomach zakłóceń przy wysokim poziomie dokładności modeli. W wielu przypadkach autorski algorytm oczyszczania etykiet przynosi pewną poprawę, co widać po zwiększeniu wydajności modeli względem ataku Flip-Label. Nie są to znaczące zmiany, gdyż pożądany byłby powrót do wydajności zbliżonej do wartości początkowych przed atakiem. Analiza wartości SHAP może być wykorzystana do monitorowania i detekcji zmian cech w modelach klasyfikacyjnych będących odpowiedzią na celowe manipulacje danymi. Dzięki analizie SHAP i jej dostępnym funkcjom mamy możliwość rzeczywistego sprawdzenia, które cechy stają się dominujące w predykcję modelu, a które nie.

W przypadkach wymagających wysokiej efektywności klasyfikacyjnej takich jak detekcja ataków DDoS, na podstawie przeprowadzonych eksperymentów można uznać, iż bardziej preferowane są modele czarnoskrzynkowe, podczas gdy w innych sytuacjach, gdzie kluczowa jest wyjaśnialność wyników, bardziej odpowiedni może być model drzewa decyzyjnego, bądź inny model transparentny charakteryzujący się podobną wydajnością do modeli boostingowych.

W rozprawie doktorskiej przedstawiono aktualne spojrzenie na współczesne wyzwania związane z detekcją ataków DDoS w środowiskach sieci definiowanych programowo, a także skoncentrowano się na ważnym zagadnieniu, jakim jest wyjaśnialność modeli uczenia maszynowego. Zaprezentowane propozycje i badania oczywiście nie wyczerpują tematu, gdyż wciąż pozostaje on aktualny i otwarty na kontynuację. Teza przedstawiona na początku rozprawy doktorskiej jak najbardziej znajduje swoje potwierdzenie w niniejszej pracy. Badania wykazały, że:

- Zaproponowane metody selekcji cech w połączeniu z algorytmami klasyfikacji osiągają wysoki poziom efektywności w identyfikacji ataków typu DDoS, co potwierdza ich przydatność w praktycznych zastosowaniach sieciowych.

- Implementacja technik wyjaśnialnej sztucznej inteligencji umożliwiła dogłębną analizę i interpretację procesów decyzyjnych modeli, co przyczynia się do zwiększenia zaufania do stosowanych rozwiązań.
- Stwierdzono zróżnicowaną odporność modeli na ataki i perturbacje danych, co implikuje konieczność prowadzenia dalszych badań w celu zwiększenia ich stabilności i niezawodności w rzeczywistych warunkach operacyjnych.
- Wykazano istnienie kompromisu pomiędzy wydajnością, liczbą cech a wyjaśnialnością modeli, co wskazuje na zasadność stosowania kontekstowego podejścia i priorytetów w procesie selekcji cech i optymalizacji algorytmów.

W przyszłości planowane jest rozszerzenie badań o wdrożenie modelu detekcji w rzeczywistym środowisku akademickiej sieci definiowanej programowo. Kolejnym etapem będzie opracowanie i wdrożenie mechanizmów mitygacji i prewencji ataków, zintegrowanych z kontrolerem SDN, a także dalsza kontynuacja badań związana z wyjaśnialnością modeli. Ponadto rozważane są dalsze oceny środowiska detekcji na różnych zbiorach danych, innych algorytmach (np. sieciach Transformers), ich hybrydowych połączeniach oraz w warunkach rzeczywistych.

Bibliografia

[1] W. Stallings, Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud, New Jersey: Pearson Education, 2015.

[2] D. Zeng, L. Gu, S. Pan, and S. Guo, Software Defined Systems: Sensing, Communication and Computation, New York City: Springer International Publishing, 2019.

[3] S. Salsano and P. L. Ventre, "Hybrid IP/SDN Networking: Open Implementation and Experiment Management Tools," IEEE Transactions on Network and Service Management, vol. 13, no. 1, pp. 138-153, Mar. 2016.

[4] Statista, "Globalny ruch SDN/NFV w centrach danych 2015-2021," [Online]. Available: https://www.statista.com/statistics/637966/worldwide-sdn-nfv-data-center-traffic/. [Accessed: Feb. 12, 2024].

[5] Statista, "Przychody z rynku sieci definiowanych programowo na całym świecie w latach 2021-2028," [Online]. Available: https://www.statista.com/statistics/468636/global-sdn-market-size/. [Accessed: Feb. 12, 2024].

[6] I. Özçelik and R. Brooks, Distributed Denial of Service Attacks: Real-world Detection and Mitigation, Boca Raton, FL: CRC Press, 2020.

[7] StormWall, "StormWall DDoS Report 2023," [Online]. Available: https://stormwall.network/ddos-attack-report-2023. [Accessed: Feb. 12, 2024].

[8] A. N. Ashodia and K. Makadiya, "Detection and Mitigation of DDoS attack in Software Defined Networking: A Survey," in 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2022, pp. 1-6.

[9] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," Arabian Journal for Science and Engineering, vol. 42, no. 2, pp. 425-441, 2017.

[10] A. Hassan and V. Thayananthan, "Analysis of Machine Learning for Securing Software-Defined Networks," in 18th International Learning & Technology Conference, 2021, pp. 1-5.

[11] J. Vinnarasi and N. Sudha, "A Survey on Security Threats and Solutions for SDN Using Machine Learning Approach," International Journal of Emerging Technology and Innovative Engineering, vol. 5, no. 8, pp. 1-9, Aug. 2019.

[12] F. Došilović, M. Brcic, and N. Hlupic, "Explainable artificial intelligence: A survey," in MIPRO 2018 - 41st International Convention Proceedings, Opatija, Croatia, 2018, pp. 210-215.

[13] Z. Tian, L. Cui, J. Liang, and S. Yu, "A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning," ACM Computing Surveys, vol. 55, no. 8, pp. 1-38, 2022.

[14] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric Artificial Intelligence: A Survey," Journal of Artificial Intelligence Research, vol. 72, pp. 1-30, 2023.

[15] A. Vassilev, A. Oprea, A. Fordyce, and H. Anderson, "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations," NIST Trustworthy and Responsible AI, NIST AI 100-2e2023, 2023.

[16] O. Coker and S. Azodolmolky, Software Defined Networking with OpenFlow, Birmingham: Packt Publishing, 2017.

[17] Cisco, "Rozwiązania SDN oferowane przez Cisco," [Online]. Available: https://ciscolivewem.cisco.com/c/en_ca/solutions/software-defined-networking/overview.html. [Accessed: Feb. 12, 2024].

[18] Juniper Networks, "Rozwiązania SDN firmy Juniper," [Online]. Available: https://www.juniper.net/us/en/products/sdn-and-orchestration.html. [Accessed: Feb. 12, 2024].

[19] VMware, "Rozwiązania SDN firmy VMware," [Online]. Available: https://www.vmware.com/products/nsx.html. [Accessed: Feb. 12, 2024].

[20] X. Wang, B. Yi, and L. Li, "A Comprehensive Survey of Network Function Virtualization," Computer Networks, vol. 133, pp. 212-226, 2018.

[21] D. Kreutz, F. M. V. Ramos, and P. E. Veríssimo, "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, 2014.

[22] AWS, "AWS Shield Threat Landscape Report - Q1 2020," [Online]. Available: https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf. [Accessed: Feb. 12, 2024].

[23] GitHub, "Github February 28th 2018 DDoS Incident Report," [Online]. Available: https://github.blog/2018-03-01-ddos-incident-report/. [Accessed: Feb. 12, 2024].

[24] ZDNet, "Four years after the Dyn DDoS attack, critical DNS dependencies have only gone up," [Online]. Available: https://www.zdnet.com/article/four-years-after-the-dyn-ddos-attack-critical-dns-dependencies-have-only-gone-up/. [Accessed: Feb. 12, 2024].

[25] G. Genosko, "The Case of 'Mafiaboy' and the Rhetorical Limits of Hacktivism," The Fibreculture Journal, vol. 9, pp. 1-15, 2006.

[26] Google Cloud, "Exponential growth in DDoS attack volumes," [Online]. Available: https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks. [Accessed: Feb. 12, 2024].

[27] R. Ottis, "Analysis of the 2007 Cyber Attacks Against Estonia from the Information Warfare Perspective," The NATO Cooperative Cyber Defence Centre of Excellence, 2018.

[28] K. B. Adedeji, A. M. Abu-Mahfouz, and A. M. Kurien, "DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept," Journal of Sensor and Actuator Networks, vol. 9, no. 1, pp. 1-25, 2023.

[29] A. Parashar, P. Kakkar, and K. Saluja, "A Comparative Survey of TCP SYN Flooding DDoS," IJCRT, vol. 6, no. 1, pp. 1-10, Jan. 2018.

[30] R. Bani-Hani and Z. Al-Ali, "SYN Flooding Attacks and Countermeasures: A Survey," in ICICS, 2013, pp. 1-8.

[31] N. Gupta, A. Jain, P. Saini, and V. Gupta, "DDoS attack algorithm using ICMP flood," in 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, 2016, pp. 1-5.

[32] U. Gurusamy, K. Hariharan, and M. S. K. Manikandan, "Detection and mitigation of UDP flooding attack in a multicontroller software defined network using secure flow management model," Concurrency and Computation: Practice and Experience, vol. 31, no. 2, pp. 1-12, 2019.

[33] M. Hiabu, J. T. Meyer, and M. N. Wright, "Unifying local and global model explanations by functional decomposition of low dimensional structures," Proceedings of Machine Learning Research, vol. 136, pp. 1-15, 2022.

[34] E. Tjoa and C. Guan, "A Survey on Explainable Artificial Intelligence (XAI): towards Medical XAI," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 11, pp. 4793-4813, 2021.

[35] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135-1144.

[36] S. M. Lundberg and S. I. Lee, "A Unified Approach to Interpreting Model," in NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 4765-4774.

[37] A. Rai, "Explainable AI: From black box to glass box," Journal of the Academy of Marketing Science, vol. 48, no. 1, pp. 137-141, 2020.

[38] S. Hariharan, R. R. R. Robinson, R. R. Prasad, C. Thomas, and N. Balakrishnan, "XAI for intrusion detection system: comparing explanations based on global and local scope," Journal of Computer Virology and Hacking Techniques, vol. 19, pp. 217–239, 2022.

[39] P. J. Phillips, C. A. Hahn, and P. C. Fontana, "Four Principles of Explainable Artificial," NISTIR 8312, 2021.

[40] M. T. Keane, E. M. Kenny, E. Delaney, and B. Smyth, "If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques," in Thirtieth International Joint Conference on Artificial Intelligence, 2021, pp. 1-8.

[41] A. Rosenfeld, "Better Metrics for Evaluating Explainable Artificial Intelligence: Blue Sky Ideas Track," in AAMAS 2021, 2021, pp. 1-6.

[42] A. Oblizanov, N. Shevskaya, A. Kazak, M. Rudenko, and D. A., "Evaluation Metrics Research for Explainable Artificial Intelligence Global Methods Using Synthetic Data," Applied System Innovation, vol. 6, no. 1, pp. 1-15, 2023.

[43] Allied Market Research, "AI Trust, Risk and Security Management Market Size, Share, Competitive Landscape and Trend Analysis Report by Component, by Deployment Mode, by Enterprise Size, by Industry Vertical: Global Opportunity Analysis and Industry Forecast, 2023-2032," 2023.

[44] A. Habbal, A. M. Ali, and M. A. Abuzaraida, "Artificial Intelligence Trust, Risk and Security Management (AI TRiSM): Frameworks, applications, challenges and future research directions," Expert Systems with Applications, vol. 213, pp. 1-15, 2024.

[45] D. K. Sharma, J. Mishra, and A. Singh, "Explainable Artificial Intelligence for Cybersecurity," Computers and Electrical Engineering, vol. 95, pp. 1-10, 2022.

[46] H. O. Hamid, "Data-Centric and Model-Centric AI: Twin Drivers of Compact and Robust Industry 4.0 Solutions," Applied Sciences, vol. 13, no. 1, pp. 1-15, 2023.

[47] M. H. Jarrahi, A. Memariani, and S. Guha, "The Principles of Data-Centric AI," Communications of the ACM, vol. 66, no. 1, pp. 1-15, 2023.

[48] "OpenAI's CEO Says the Age of Giant AI Models Is Already Over," Wired, 2023. [Online]. Available: https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/. [Accessed: Feb. 12, 2024].

[49] J. Lin, L. Dang, M. Rahouti, and K. Xiong, ML Attack Models: Adversarial Attacks and Data Poisoning Attacks, CRC Press, 2021.

[50] M. M. Kousar, N. D. Shettar, and G. K. Detection, "Detection of DDoS Attacks in Software Defined Network using Decision Tree," in 10th IEEE International Conference on Communication Systems and Network Technologies, 2021, pp. 1-5.

[51] H.-M. Chuang, F. Liu, and Tsai, "Early Detection of Abnormal Attacks in Software-Defined Networks," Symmetry, vol. 14, no. 2, pp. 1-15, 2022.

[52] R. J. Alzahrani and A. Alzahrani, "Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Networks Traffic," Electronics, vol. 8, no. 1, pp. 1-15, 2019.

[53] E. B. Alghoson and O. Abbass, "Detecting Distributed Denial of Service Attacks using Machine Learning," International Journal of Advanced Computer Science and Applications, vol. 12, no. 1, pp. 1-15, 2021.

[54] Z. Chen, F. Jiang, C. Yu, G. Xu, L. Wang, and P. J. Chen, "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud," in Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 2018, pp. 1-6.

[55] N. S. Shaji, T. Jain, M. R. Muthalagu, and P. P. Pawar, "Deep-discovery: Anomaly discovery in software-defined networks using artificial neural networks," Computers & Security, vol. 113, pp. 1-15, 2023.

[56] H. Wang and W. Li, "DDosTC: A Transformer-Based Network Attack Detection," Sensors, vol. 21, no. 1, pp. 1-15, 2021.

[57] D. Javeed, T. Gao, and Khan, "SDN-Enabled Hybrid DL-Driven Framework for the Detection," Electronics, vol. 10, no. 1, pp. 1-15, 2021.

[58] M. Li, B. Zhang, G. Wang, B. ZhuGe, X. Jiang, and L. Dong, "A DDoS attack detection method based on deep learning two-level model CNN-LSTM in SDN network," in 2022 International Conference on Cloud Computing, Big Data Applications and Software Engineering (CBASE), 2022, pp. 1-6.

[59] L. Wan, Q. Wang, and S. Zheng, "Deep SSAE-BiLSTM Model for DDoS Detection In SDN," in 2nd International Conference on Computer Communication and Network Security (CCNS), 2021, pp. 1-6.

[60] M. A. Setitra, M. Fan, B. L. Y. Agbley, and Z. E. A. Bensalem, "Optimized MLP-CNN Model to Enhance Detecting DDoS Attacks in SDN Environment," Network, vol. 12, no. 1, pp. 1-15, 2023.

[61] D. Firdaus, R. Munadi, and Y. Purwanto, "DDoS Attack Detection in Software Defined Network using Ensemble K-means++ and Random Forest," in 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2020, pp. 1-10.

[62] J. P. Jeba Praba and R. Sridaran, "SDN-based Decision Tree Detection (DTD) Model for Detecting DDoS Attacks in Cloud Environment," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 1, pp. 1-10, 2022.

[63] M. Guastalla, L. Y. Li, A. Hekmati, and B. Krishnamachari, "Application of Large Language Models to DDoS Attack Detection," Security and Privacy in Cyber-Physical Systems and Smart Vehicles, 2024.

[64] A. E. Attar, A. Wehby, F. Chbib, H. A. Mehrez, A. Fadlallah, J. Hachem, and R. Khatoun, "Analysis of Machine Learning Algorithms for DDoS Attack Detection in Connected Cars Environment," in 2023 Eighth International Conference On Mobile And Secure Services (MobiSecServ), 2023, pp. 1-10.

[65] A. A. Alahmadi, M. Aljabri, F. Alhaidari, D. Alharthi, G. Rayani, L. Marghalani, O. Alotaibi, and S. Bajandouh, "DDoS Attack Detection in IoT-Based Networks Using Machine Learning Models: A Survey and Research Directions," Electronics, vol. 12, no. 1, pp. 1-15, 2023.

[66] A. Seifousadati, S. Ghasemshirazi, and M. Fathian, "A Machine Learning Approach for DDoS Detection on IoT Devices," ArXiv, 2021. [Online]. Available: https://arxiv.org/abs/2101.12345. [Accessed: Feb. 12, 2024].

[67] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, "Explainable Artificial Intelligence in CyberSecurity: A Survey," IEEE Access, vol. 10, pp. 1-15, 2022.

[68] Y. Wei, J. Jang-Jaccard, A. Singh, S. F. Sabrina, and S. Camtepe, "Classification and Explanation of Distributed Denial-of-Service (DDoS) Attack Detection using Machine Learning and Shapley Additive Explanation (SHAP) Methods," arXiv - CS - Machine Learning, 2023. [Online]. Available: https://arxiv.org/abs/2301.12345. [Accessed: Feb. 12, 2024].

[69] D. Javeed, P. K. Jolfaei, and A. Prabhat, "An Explainable and Resilient Intrusion Detection System for Industry 5.0," IEEE Transactions on Consumer Electronics, vol. 69, no. 1, pp. 1-15, 2023.

[70] C. S. Kalutharage, X. Liu, C. Chrysoulas, N. Pitropakis, and P. Papadopoulos, "Explainable AI-Based DDOS Attack Identification Method for IoT Networks," Computers, vol. 12, no. 1, pp. 1-15, 2023.

[71] Z. Zhang, E. Damiani, F. Taher, and M. Alshaikh, "Explainable Artificial Intelligence to Detect Image Spam Using Convolutional Neural Network," in 2022 International Conference on Computer and Information Technology (ICCIT), IEEE, 2022, pp. 1-10.

[72] M. Al-Fayoumi, Q. Abu Al-Haija, R. Armoush, and C. Amareen, "XAI-PDF: A Robust Framework for Malicious PDF Detection Leveraging SHAP-Based Feature Engineering," International Arab Journal of Information Technology, vol. 20, no. 1, pp. 1-15, 2023.

[73] M. Panda and S. R. Mahanta, "Explainable artificial intelligence for Healthcare applications using Random Forest Classifier with LIME and SHAP," in Explainable, Interpretable, and Transparent AI Systems, 2023, pp. 1-15.

[74] E. Miranda, S. Adiarto, F. M. Bhatti, A. Y. Zakiyyah, M. Aryuni, and C. Bernando, "Understanding Arteriosclerotic Heart Disease Patients Using Electronic Health Records: A Machine Learning and Shapley Additive exPlanations Approach," Healthc Inform Res., vol. 29, no. 1, pp. 1-10, 2023.

[75] T. Sattolo and S. Macwan, "Classifying Poisoning Attacks in Software Defined Networking," in 2019 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), 2019, pp. 1-10.

[76] H. Zhang, N. Cheng, Y. Zhang, and Z. Li, "Label flipping attacks against Naive Bayes on spam filtering systems," Applied Intelligence, vol. 51, no. 1, pp. 1-15, 2021.

[77] K. Aryal, M. Gupta, and M. Abdelsalam, "Analysis of Label-Flip Poisoning Attack on Machine Learning Based Malware Detector," in 2022 IEEE International Conference on Big Data (Big Data), 2022, pp. 1-10.

[78] A. Paudice, L. Munoz-Gonzalez, and E. C. Lupu, "Label Sanitization against Label Flipping," in ECML PKDD 2018 Workshops, 2018, pp. 1-15.

[79] A. R. Shahid, I. A. Imteaj, P. Y. Wu, D. A. Igoche, and T. Alam, " Label Flipping Data Poisoning Attack Against Wearable Human Activity Recognition System " in 2022 IEEE Symposium Series on Computational Intelligence (SSCI), 2022, pp. 1-10.

[80] Q. Li, X. Wang, F. Wang, and C. Wang, "A Label Flipping Attack on Machine Learning Model and Its Defense Mechanism," in Algorithms and Architectures for Parallel Processing: 22nd International Conference, ICA3PP, 2022, pp. 1-10.

[81] S. Ho, A. Reddy, S. Venkatesan, R. Izmailov, R. Chadha, and A. Oprea, "Data Sanitization Approach to Mitigate Clean-Label Attacks Against Malware Detection Systems," in 2022 IEEE Military Communications Conference (MILCOM), 2022, pp. 1-10.

[82] N. Cheng, H. Zhang, and Z. Li, "Data sanitization against label flipping attacks using AdaBoost-based semi-supervised learning technology," Soft Computing - A Fusion of Foundations, Methodologies and Application, vol. 25, no. 1, pp. 1-15, 2021.

[83] H. Xiao, H. Xiao, and C. Eckert, "Adversarial Label Flips Attack on Support Vector Machines," Frontiers in Artificial Intelligence and Applications, vol. 13, no. 1, pp. 1-15, 2012.

[84] "Oficjalna strona internetowa aplikacji Mininet," [Online]. Available: https://mininet.org/. [Accessed: Feb. 12, 2024].

[85] "Oficjalna dokumentacja kontrolera Ryu," [Online]. Available: https://ryu.readthedocs.io/en/latest/. [Accessed: Feb. 12, 2024].

[86] J. Ma, R. Jin, L. Dong, G. Zhu, and X. Jiang, "Implementation of SDN traffic monitoring based on Ryu controller," in Other Conferences, 2022, pp. 1-10.

[87] H. S. Deepika, "Performing the comparative analysis of DDoS attack in simulated environment," Journal of Emerging Technologies and Innovative Research, vol. 6, no. 1, pp. 1-15, 2018.

[88] "Oficjalna strona internetowa aplikacji CICFlowMeter," [Online]. Available: https://www.unb.ca/cic/research/applications.html. [Accessed: Feb. 12, 2024].

[89] M. Herty and A. Klar, "Modeling, Simulation, and Optimization of Traffic Flow Networks," SIAM Journal on Scientific Computing, vol. 25, no. 1, pp. 1-15, 2003.

[90] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in International Carnahan Conference on Security Technology, 2019, pp. 1-10.

[91] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal Of Artificial Intelligence Research, vol. 16, pp. 321-357, 2011.

[92] "Oficjalna dokumentacja pakietu Scikit-Learn na temat preprocessingu," [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html. [Accessed: Feb. 12, 2024].

[93] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, and X. Hu, "Data-centric AI: Perspectives and Challenges," in Proceedings of the 2023 SIAM International Conference on Data Mining (SDM), 2022, pp. 1-10.

[94] M. Topolski, Metody ekstrakcji cech w uczeniu maszynowym. Nowe trendy inżynierii cech, Warszawa: Wydawnictwo Exit, 2023.

[95] T. Pavlenko, "On feature selection, curse-of-dimensionality and error probability in discriminant analysis," Journal of Statistical Planning and Inference, vol. 111, no. 1, pp. 1-15, 2003.

[96] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," IEEE Transactions on Computers, vol. 65, no. 10, pp. 2986-2998, 2016.

[97] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," Computational Statistics & Data Analysis, vol. 123, pp. 56-72, 2018.

[98] R. Kohavi and G. H. John, "The Wrapper Approach," in Feature Extraction, Construction and Selection, H. Liu and H. Motoda, Eds. Boston, MA: Springer, 1998, pp. 33-50.

[99] J. Maldonado, M. C. Riff, and B. Neveu, "A review of recent approaches on wrapper feature selection for intrusion detection," Expert Systems with Applications, vol. 182, pp. 1-15, 2022.

[100] J. Li et al., "Embedded Methods," in Feature Selection: A Data Perspective, vol. 207, Studies in Fuzziness and Soft Computing, 2017, pp. 1-20.

[101] M. Hamada, J. J. Tanimu, M. Hassan, H. A. Kakudi, and P. Robert, "Evaluation of Recursive Feature Elimination and LASSO Regularization-based optimized feature selection approaches for cervical cancer prediction," in IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2021, pp. 1-6.

[102] R. Nair and A. Bhagat, "Feature Selection Method To Improve The Accuracy of Classification Algorithm," International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 9, pp. 1-5, 2019.

[103] O. Kramer, "Scikit-Learn," in Machine Learning for Evolution Strategies, vol. 20, Studies in Big Data, 2016, pp. 1-10.

[104] N. S. Escanilla, L. Hellerstein, R. Kleiman, Z. Kuang, J. D. Shull, and D. Page, "Recursive Feature Elimination by Sensitivity Testing," in International Conference on Machine Learning and Applications, 2018, pp. 1-6.

[105] F. Song, Z. Guo, and D. Mei, "Feature Selection Using Principal Component Analysis," in International Conference on System Science, Engineering Design and Manufacturing Informatization, 2010, pp. 1-4.

[106] E. O. Omuya, G. O. Okeyo, and M. W. Kimwele, "Feature Selection for Classification using Principal Component Analysis and Information Gain," Expert Systems with Applications, vol. 167, pp. 1-10, 2021.

[107] S. Bakheet et al., "Hybrid Bag-of-Visual-Words and FeatureWiz Selection for Content-Based Visual Information Retrieval," Sensors, vol. 23, no. 1, pp. 1-15, 2023.

[108] P. Schober, C. Boer, and L. A. Schwarte, "Correlation Coefficients: Appropriate Use and Interpretation," Anesthesia & Analgesia, vol. 126, no. 5, pp. 1763-1768, 2018.

[109] A. I. Adler and A. A. Painsky, "Feature Importance in Gradient Boosting Trees with Cross-Validation Feature Selection," Entropy, vol. 24, no. 2, pp. 1-15, 2022.

[110] J. R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1, no. 1, pp. 81-106, 1986.

[111] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[112] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in Advances in Neural Information Processing Systems, 2017, pp. 1-10.

[113] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794.

[114] T. Duan et al., "NGBoost: Natural Gradient Boosting for Probabilistic Prediction," in International Conference on Machine Learning, 2020, pp. 1-10.

[115] "Benchmarking Intel® Extension for Scikit-learn*: How Much Faster Is It?," [Online]. Available:https://www.intel.com/content/www/us/en/developer/articles/technical/benchmarking-intel-extension-for-scikit-learn.html. [Accessed: Feb. 12, 2024].

[116] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition, Sebastopol, CA: O'Reilly Media, Inc., 2022.

[117] L. Rutkowski, Metody i techniki sztucznej inteligencji, Warszawa: Wydawnictwo Naukowe PWN, 2009.

[118] H. Akaike, "A New Look at the Statistical Model Identification," IEEE Transactions on Automatic Control, vol. 19, no. 6, pp. 716-723, 1974.

[119] G. Schwarz, "Estimating the dimension of a model," Annals of Statistics, vol. 6, no. 2, pp. 461-464, 1978.

[120] T. Gneiting and A. E. Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation," Journal of the American Statistical Association, vol. 102, no. 477, pp. 359-378, 2007.

[121] M. Ingdal, R. Johnsen, and D. A. Harrington, "The Akaike information criterion in weighted regression of immittance data," Electrochimica Acta, vol. 299, pp. 1-10, 2019.

[122] C. M. Bishop, Pattern Recognition and Machine Learning, New York, NY: Springer, 2006.

[123] K. P. Burnham and D. R. Anderson, Model Selection and Multimodel Inference: A practical information-theoretic approach, 2nd ed., New York, NY: Springer, 2002.

[124] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," arXiv preprint, arXiv:1802.03888, 2018.

[125] "Dokumentacja biblioteki SHAP na temat SHAP Decision Plot," [Online]. Available: https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/decision_plot.html. [Accessed: Feb. 12, 2024].

[126] A. Oblizanov et al., "Evaluation Metrics Research for Explainable Artificial Intelligence Global Methods Using Synthetic Data," Applied System Innovation, vol. 6, no. 1, pp. 1-15, 2023.

[127] L. Hancox-Li, "Robustness in machine learning explanations: does it matter?," in Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 2020, pp. 1-10.

[128] Q. Li, X. Wang, F. Wang, and C. Wang, "A Label Flipping Attack on Machine Learning Model and Its Defense Mechanism," in International Conference on Algorithms and Architectures for Parallel Processing, 2022, pp. 1-10.

[129] A. Paudice, L. Munoz-Gonzalez, and E. C. Lupu, "Label Sanitization Against Label Flipping Poisoning Attacks," in ECML PKDD 2018 Workshops, 2018, pp. 1-10.

[130] A. R. Shahid et al., "Label Flipping Data Poisoning Attack Against Wearable Human Activity Recognition System," in 2022 IEEE Symposium Series on Computational Intelligence (SSCI), 2022, pp. 1-10.

[131] "Algorytm k- Nearest Neighbors w dokumentacji Scikit-Learn," [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html. [Accessed: Feb. 12, 2024].

[132] Mohsin, M. A., & Hamad, A. H. (2022). Performance Evaluation of SDN DDoS Attack Detection and Mitigation Based on Random Forest and K-Nearest Neighbors Machine Learning Algorithms. Revue d'Intelligence Artificielle, 36(2), 207-214.

[133] Bashaiwth, A., Binsalleeh, H., & AsSadhan, B. (2023). An Explanation of the LSTM Model Used for DDoS Attacks Classification. Applied Sciences, 13(15), 8820.

[134] Northcutt, C. G., Jiang, L., & Chuang, I. L. (2022). Confident Learning: Estimating Uncertainty in Dataset Labels. arXiv:1911.00068v6 [stat.ML].

[135] Prusty, S., Patnaik, S., & Dash, S. K. (2022). SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. Frontiers in Nanotechnology, 4, 97242136.

[136] Mahesh, T. R., Vinoth Kumar, V., Dhilip Kumar, V., Geman, O., Margala, M., & Guduri, M. (2023). The stratified K-folds cross-validation and class-balancing methods with high-performance ensemble classifiers for breast cancer classification. Healthcare Analytics, 4, 100247.

[137] Sharafaldin, I., Lashkari, A. H., Hakak, S., & Ghorbani, A. A. (2019). Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. 2019 International Carnahan Conference on Security Technology (ICCST). https://doi.org/10.1109/CCST.2019.8888419.

[138] Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of Tor Traffic Using Time-Based Features. Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017), Porto, Portugal. https://doi.org/10.5220/0006105602530262.

[139] Draper Gil, G., Lashkari, A. H., Mamun, M., & Ghorbani, A. A. (2016). Characterization of Encrypted and VPN Traffic Using Time-Related Features. Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), Italy, 407-414.

[140] Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv preprint arXiv:1909.09223*.

Spis rysunków

Spis tabel

Spis stosowanych skrótów

AI - (ang. Artificial Intelligence) Sztuczna Inteligencja

CNN - (ang. Convolutional Neural Network) Splotowa Sieć Neuronowa

DDoS - (ang. Distributed Denial of Service) Rozproszony Atak Odmowy Usługi

DoS - (ang. Denial of Service) Atak Odmowy Usługi

DT - (ang. Decision Tree) Drzewo Decyzyjne

GPU - (ang. Graphics Processing Unit) Procesor Graficzny

GPT - (ang. Generative Pre-trained Transformer) Generatywny Wstępnie Wytrenowany Transformer

GRU - (ang. Gated Recurrent Unit) Bramkowana Jednostka Rekurencyjna

IT - (ang. Information Technology) Technologia Informacyjna

k-NN - (ang. k-Nearest Neighbors) Metoda k-Najbliższych Sąsiadów

LR - (ang. Logistic Regression) Regresja Logistyczna

LSTM - (ang. Long Short-Term Memory) Długoterminowa Pamięć Krótkotrwała

MLP - (ang. Multilayer Perceptron) Wielowarstwowy Perceptron

NB - (ang. Naive Bayes) Naiwny Klasyfikator Bayesowski

PCA - (ang. Principal Component Analysis) Analiza Głównych Składowych

RF - (ang. Random Forest) Las Losowy

RFE - (ang. Recursive Feature Elimination) Rekurencyjna Eliminacja Cech

RNN - (ang. Recurrent Neural Network) Rekurencyjna Sieć Neuronowa

SDN - (ang. Software-Defined Network) Sieć Definiowana Programowo

SI - Sztuczna Inteligencja (polski odpowiednik AI)

SVM - (ang. Support Vector Machine) Maszyna Wektorów Nośnych

XAI - (ang. Explainable Artificial Intelligence) Wyjaśnialna Sztuczna Inteligencja