Uczenie maszynowe w bezpieczeństwie

Projekt 2

Michał Łaskawski

Oznaczenia matematyczne

- $N(\mu, \sigma)$ - rozkład normalny o średniej μ i odchyleniu standardowym σ
- Poisson(λ) - rozkład Poissona o parametrze λ
- Exp(λ) - rozkład wykładniczy o parametrze λ
- TP, TN, FP, FN - prawdziwie dodatnie, prawdziwie ujemne, fałszywie dodatnie, fałszywie ujemne
- β - współczynnik regresji logistycznej
- β_0 - wyraz wolny
- τ - próg decyzyjny
- H - entropia Shannona
- ρ - współczynnik korelacji Pearsona
- X - macierz cech (dane wejściowe)
- y - wektor etykiet (0 - normalny ruch, 1 - atak)
- n - liczba próbek
- d - liczba cech
- $\hat{\mu}$ - estymator średniej
- $\hat{\sigma}$ - estymator odchylenia standardowego
- ϵ - mała stała numeryczna zapobiegająca dzieleniu przez zero (typowo 10^{-10})
- D - ramka danych (dataframe)
- D_{Label} - kolumna etykiet w ramce danych
- D_{DestPort} - kolumna portów docelowych w ramce danych
- L - wektor etykiet (tekstowych lub numerycznych)
- L_{norm} - etykieta klasy normalnej (wartość oznaczająca normalny ruch)
- T_{type} - kolumna typu ataku w ramce danych
- T_{DDoS} - wartość etykiety oznaczająca atak DDoS
- T_{Port} - wartość etykiety oznaczająca atak Port Scan

Notacja funkcji w algorytmach

- $\mathbb{E}[X]$ - wartość oczekiwana (średnia) wektora X
- $\sigma[X]$ - odchylenie standardowe wektora X
- $\text{Var}[X]$ - wariancja wektora X
- $\text{Med}[X]$ - mediana wektora X
- $\oplus$ - konkatenacja (łączenie) struktur danych
- π - permutacja losowa elementów
- $\pi(X, y)$ - aplikacja permutacji losowej do danych X i etykiet y
- $\text{sort}(X, \text{key})$ - sortowanie zbioru X według klucza
- $\text{argsort}(X)$ - zwraca indeksy sortujące wektor X rosnąco
- $\text{clip}(x, a, b)$ - obcięcie wartości x do przedziału $[a, b]$
- $\mathbb{1}(\cdot)$ - funkcja wskaźnikowa (1 gdy warunek prawdziwy, 0 w przeciwnym przypadku)
- $\triangleright$ - symbol komentarza w algorytmach

- $\mathcal{M}(y, \hat{y})$ - macierz pomyłek zwracająca (TP, TN, FP, FN) dla etykiet y i predykcji $\hat{y}$
- $H(\{p_i\})$ - entropia Shannona: $H = -\sum_i p_i \log_2(p_i)$
- $\arg\min_x f(x)$ i $\arg\max_x f(x)$ - argument minimalizujący/maksymalizujący funkcję f
- x^* - notacja oznaczająca wartość optymalną (po optymalizacji)
- $\mathbf{0}_{n \times m}$ - macierz zer o wymiarach $n \times m$
- $\mathbf{0}_n$ - wektor zer o długości n
- unique(S) - zwraca zbiór unikalnych elementów ze zbioru S
- read_csv(f) - wczytuje dane z pliku CSV o ścieżce f
- is_numeric(X) - predykat zwracający prawdę gdy X jest typu numerycznego
- num_cols(D) - zwraca liczbę kolumn w macierzy/ramce danych D
- sample(S, k) - zwraca losowy podzbiór k elementów ze zbioru S
- stratified_split(X, y, p) - podział ze stratyfikacją w proporcji $p : (1 - p)$
- $|$ - separator w definicjach matematycznych (odpowiednik "gdzie")

Operatory algorytmiczne

- $\leftarrow$ - operator przypisania (strzałka w lewo): przypisanie wartości do zmiennej
- $\sim$ - operator losowania: $x \sim N(\mu, \sigma)$ oznacza wylosowanie wartości x z rozkładu normalnego
- $\wedge$ - koniunkcja logiczna (AND): $a \wedge b$ jest prawdziwe gdy oba warunki są spełnione
- $\vee$ - alternatywa logiczna (OR): $a \vee b$ jest prawdziwe gdy przynajmniej jeden warunek jest spełniony
- $\setminus$ - różnica zbiorów: $A \setminus B$ zawiera elementy z A nieobecne w B
- $\in$ - przynależność do zbioru: $x \in S$ oznacza że element x należy do zbioru S
- $\exists$ - kwantyfikator egzystencjalny: istnieje
- $\forall$ - kwantyfikator uniwersalny: dla każdego

ZADANIE 1: EKSPERYMENT Z DANYMI IDEALNYMI

Treść zadania

Wygeneruj syntetyczny zbiór danych symulujący ruch sieciowy składający się z 1000 próbek, gdzie 800 próbek reprezentuje normalny ruch a 200 próbek reprezentuje ataki. Każda próbka musi zawierać 7 cech numerycznych charakteryzujących ruch sieciowy: liczbę pakietów na sekundę, średni rozmiar pakietu w bajtach, entropię portów docelowych, stosunek pakietów SYN do wszystkich pakietów, liczbę unikalnych adresów IP docelowych, czas trwania połączenia w sekundach oraz liczbę powtórzonych połączeń.

Normalny ruch wygeneruj losując wartości z rozkładów o następujących parametrach:

- liczba pakietów na sekundę: $N(\mu = 50, \sigma = 15)$
- średni rozmiar pakietu: $N(\mu = 800, \sigma = 200)$
- entropia portów: $N(\mu = 2.5, \sigma = 0.5)$
- stosunek SYN: $N(\mu = 0.2, \sigma = 0.05)$, obcięte do przedziału $[0, 1]$
- liczba unikalnych IP: Poisson$(\lambda = 5) + 1$
- czas trwania połączenia: Exp$(\lambda = 1/30)$
- liczba powtórzeń: Poisson$(\lambda = 2)$

Ataki wygeneruj używając tych samych typów rozkładów ale z przesuniętymi parametrami tak aby były wyraźnie oddzielone od normalnego ruchu:

- liczba pakietów na sekundę: $N(\mu = 250, \sigma = 30)$
- średni rozmiar pakietu: $N(\mu = 300, \sigma = 100)$

- entropia portów: $N(\mu = 4.0, \sigma = 0.3)$
- stosunek SYN: $N(\mu = 0.8, \sigma = 0.05)$, obcięte do przedziału $[0, 1]$
- liczba unikalnych IP: Poisson$(\lambda = 50) + 1$
- czas trwania połączenia: Exp$(\lambda = 1/2)$
- liczba powtórzeń: Poisson$(\lambda = 20)$

Podziel wygenerowane dane na zbiór treningowy zawierający 70% próbek oraz zbiór testowy zawierający 30% próbek zachowując proporcje klas w obu zbiorach. Znormalizuj wszystkie cechy obliczając dla zbioru treningowego średnią $\hat{\mu}$ i odchylenie standardowe $\hat{\sigma}$ każdej cechy, a następnie stosując transformację $z_{\text{train}} = (x_{\text{train}} - \hat{\mu})/\hat{\sigma}$ oraz $z_{\text{test}} = (x_{\text{test}} - \hat{\mu})/\hat{\sigma}$.

Wytrenuj model regresji logistycznej z regularyzacją L2 o sile regularyzacji $C = 1.0$. Oblicz prognozy na zbiorze testowym i wyznacz macierz pomyłek, dokładność Accuracy $= (TP+TN)/(TP+TN+FP+FN)$, precyzję Precision $= TP/(TP+FP)$, czułość Recall $= TP/(TP+FN)$ oraz miarę F1 F1 $= 2 \cdot \text{Precision} \cdot \text{Recall}/(\text{Precision} + \text{Recall})$ osobno dla obu klas. Oblicz pole pod krzywą ROC oznaczone jako AUC. Wyodrębnij współczynniki $\beta_1, \beta_2, \ldots, \beta_7$ dla każdej cechy oraz wyraz wolny β_0 z wytrenowanego modelu.

Wygeneruj następujące wizualizacje: 4 wykresy pokazujące rozkłady gęstości dla liczby pakietów na sekundę, średniego rozmiaru pakietu, entropii portów oraz stosunku SYN, gdzie każdy wykres zawiera 2 krzywe przedstawiające rozkład dla normalnego ruchu i dla ataków w różnych kolorach; wykres słupkowy poziomy przedstawiający współczynniki $\beta_1, \ldots, \beta_7$ uporządkowane według wartości $|\beta_i|$ z adnotacjami liczbowymi; mapę ciepła macierzy pomyłek z liczbami w komórkach; krzywą ROC z zaznaczonym punktem optymalnym i wartością AUC w legendzie; histogram prawdopodobieństw predykcji $P(y = 1|x)$ osobno dla próbek normalnych i ataków z linią pionową przy progu $\tau = 0.5$; 3 wykresy słupkowe pokazujące wpływ każdej cechy $\beta_i \cdot x_i$ na decyzję modelu dla 3 wybranych próbek testowych; macierz korelacji Pearsona ρ_{ij} między wszystkimi cechami jako mapę ciepła.

Przeanalizuj wyniki odpowiadając na pytania: które cechy mają największe współczynniki $|\beta_i|$ i co to oznacza dla detekcji ataków, ile błędów typu FN i FP popełnia model, czy wszystkie 7 cech jest potrzebnych do osiągnięcia wysokiej dokładności, oraz które pary cech mają najwyższe wartości $|\rho_{ij}|$.

Procedura postępowania

Szczegółowa procedura postępowania dla tego zadania została opisana w Algorytmie 1.

ZADANIE 2: EKSPERYMENT Z DANYMI REALISTYCZNYMI

Treść zadania

Wygeneruj syntetyczny zbiór danych symulujący realistyczny scenariusz gdzie normalny ruch stanowi 950 próbek a ataki tylko 50 próbek, odzwierciedlając proporcje występujące w rzeczywistych sieciach. Ataki podziel na 3 kategorie o różnym stopniu trudności wykrycia: 20 ataków oczywistych których cechy są oddalone od rozkładu normalnego ruchu o 4σ, 15 ataków średnio subtelnych oddalonych o 2σ oraz 15 ataków bardzo subtelnych oddalonych tylko o 1σ.

Normalny ruch wygeneruj tak samo jak w zadaniu 1 używając rozkładów $N(\mu_{\text{norm}}, \sigma_{\text{norm}})$ dla każdej cechy. Dla ataków oczywistych wygeneruj wartości z rozkładów $N(\mu_{\text{norm}} \pm 4\sigma_{\text{norm}}, \sigma_{\text{norm}})$ gdzie znak zależy od tego czy atak charakteryzuje się wyższymi (+) czy niższymi (-) wartościami danej cechy. Dla ataków średnio subtelnych użyj przesunięcia $\pm 2\sigma_{\text{norm}}$ a dla ataków bardzo subtelnych użyj przesunięcia $\pm 1\sigma_{\text{norm}}$.

Konkretnie dla każdego typu ataku użyj następujących parametrów:

- Ataki oczywiste: $N(\mu_{\text{norm}} + k \cdot \sigma_{\text{norm}}, \sigma_{\text{norm}})$ gdzie $k = 4$ dla cech rosnących przy ataku, $k = -4$ dla malejących
- Ataki średnie: $k = 2$
- Ataki subtelne: $k = 1$

Podziel dane na zbiory treningowy i testowy w proporcjach 70:30 i znormalizuj analogicznie jak w zadaniu 1. Wytrenuj 3 różne modele regresji logistycznej stosując różne strategie: pierwszy model standardowy bez specjalnej obsługi niezbalansowania, drugi model z parametrem `class_weight='balanced'`, trzeci model standardowy ale z ręcznie dostosowanym progiem decyzyjnym τ_{opt}.

Dla trzeciego modelu znajdź optymalny próg testując wartości $\tau \in \{0.01, 0.02, \ldots, 0.99\}$. Dla każdego progu oblicz liczbę $FN(\tau)$ i $FP(\tau)$, następnie oblicz całkowity koszt $C(\tau) = 100 \cdot FN(\tau) + 1 \cdot FP(\tau)$. Wybierz próg $\tau_{\text{opt}} = \arg\min C(\tau)$ i użyj go do generowania prognoz: $\hat{y} = 1$ jeśli $P(y = 1|x) \geq \tau_{\text{opt}}$, inaczej $\hat{y} = 0$.

Algorithm 1 Eksperyment z danymi idealnymi

Wejście: $n_{\text{norm}} = 800$, $n_{\text{atak}} = 200$, $d = 7$
Wyjście: M, β, β_0, metryki
▷ Generowanie danych
$X \leftarrow \mathbf{0}_{1000\times 7}$, $y \leftarrow \mathbf{0}_{1000}$
for $i = 0$ **to** 799 **do**
    $X[i,0] \sim N(50, 15)$
    $X[i,1] \sim N(800, 200)$
    $X[i,2] \sim N(2.5, 0.5)$
    $X[i,3] \sim \text{clip}(N(0.2, 0.05), 0, 1)$
    $X[i,4] \sim \text{Poisson}(5) + 1$
    $X[i,5] \sim \text{Exp}(1/30)$
    $X[i,6] \sim \text{Poisson}(2)$
    $y[i] \leftarrow 0$
end for
for $i = 800$ **to** 999 **do**
    $X[i,0] \sim N(250, 30)$
    $X[i,1] \sim N(300, 100)$
    $X[i,2] \sim N(4.0, 0.3)$
    $X[i,3] \sim \text{clip}(N(0.8, 0.05), 0, 1)$
    $X[i,4] \sim \text{Poisson}(50) + 1$
    $X[i,5] \sim \text{Exp}(1/2)$
    $X[i,6] \sim \text{Poisson}(20)$
    $y[i] \leftarrow 1$
end for
$(X, y) \leftarrow \pi(X, y)$
▷ Podział i normalizacja
$I_{\text{train}}, I_{\text{test}} \leftarrow \text{stratified\_split}(\{0, \ldots, 999\}, y, 0.7)$
$X_{\text{train}} \leftarrow X[I_{\text{train}}, :]$, $y_{\text{train}} \leftarrow y[I_{\text{train}}]$
$X_{\text{test}} \leftarrow X[I_{\text{test}}, :]$, $y_{\text{test}} \leftarrow y[I_{\text{test}}]$
for $j = 0$ **to** 6 **do**
    $\hat{\mu}_j \leftarrow \mathbb{E}[X_{\text{train}}[:, j]]$
    $\hat{\sigma}_j \leftarrow \sigma[X_{\text{train}}[:, j]]$
    $X_{\text{train}}[:, j] \leftarrow (X_{\text{train}}[:, j] - \hat{\mu}_j)/\hat{\sigma}_j$
    $X_{\text{test}}[:, j] \leftarrow (X_{\text{test}}[:, j] - \hat{\mu}_j)/\hat{\sigma}_j$
end for
▷ Model regresji logistycznej z regularyzacją L2
$(\beta^*, \beta_0^*) \leftarrow \arg\min_{\beta\in\mathbb{R}^7, \beta_0\in\mathbb{R}} \mathcal{L}(\beta, \beta_0)$
$\mathcal{L}(\beta, \beta_0) = -\sum_{i\in I_{\text{train}}} [y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i))] + \frac{1}{2C}\|\beta\|^2$
$z_i = \beta_0 + \sum_{j=0}^{6} \beta_j X_{\text{train}}[i, j] \mid \sigma(z) = 1/(1 + e^{-z}), C = 1.0$
$\beta \leftarrow \beta^*$, $\beta_0 \leftarrow \beta_0^*$
▷ Predykcja i ewaluacja
for $i \in I_{\text{test}}$ **do**
    $z_i \leftarrow \beta_0 + \sum_{j=0}^{6} \beta_j X_{\text{test}}[i, j]$
    $P[i] \leftarrow \sigma(z_i) = 1/(1 + e^{-z_i})$
    $\hat{y}_{\text{test}}[i] \leftarrow \mathbb{1}(P[i] \geq 0.5)$
end for
$(TP, TN, FP, FN) \leftarrow \mathcal{M}(y_{\text{test}}, \hat{y}_{\text{test}})$
Accuracy $\leftarrow (TP + TN)/(TP + TN + FP + FN)$
Precision $\leftarrow TP/(TP + FP)$
Recall $\leftarrow TP/(TP + FN)$
$F1 \leftarrow 2 \cdot \text{Precision} \cdot \text{Recall}/(\text{Precision} + \text{Recall})$
AUC $\leftarrow \int_0^1 \text{TPR}(\text{FPR}^{-1}(x))dx$ dla krzywej ROC(P, y_{test})
return $M = (\beta, \beta_0)$, {Accuracy, Precision, Recall, F1, AUC}, (TP, TN, FP, FN)

Dla każdego z 3 modeli oblicz pełny zestaw metryk oraz wygeneruj osobne macierze pomyłek. Porównaj szczególnie liczbę FN między modelami aby ocenić skuteczność każdej strategii w redukcji pominiętych ataków.

Wygeneruj następujące wizualizacje: 3 macierze pomyłek obok siebie dla porównania strategii z wyraźnymi adnotacjami liczb FN i FP; wykres krzywych ROC dla modelu standardowego i modelu balanced nałożone na jedną figurę z wartościami AUC; histogram rozkładu prawdopodobieństw $P(y = 1|x)$ dla modelu balanced pokazujący separację między klasami; wykres słupkowy porównujący współczynniki β między modelem standardowym a balanced; 3 wykresy rozkładów gęstości dla wybranych cech pokazujące nakładkę między normalnym ruchem $(\mu_{\text{norm}}, \sigma_{\text{norm}})$ a atakami wszystkich 3 typów w różnych kolorach; wykres liniowy pokazujący jak Precision(τ), Recall(τ) i $F1(\tau)$ zmieniają się w funkcji progu decyzyjnego $\tau \in [0, 1]$; wykres liniowy pokazujący $C(\tau)$ w funkcji τ z zaznaczonym minimum przy τ_{opt}.

Przeanalizuj wyniki odpowiadając na pytania: jaka jest wartość τ_{opt} i ile wynosi oszczędność kosztów $\Delta C = C(\tau = 0.5) - C(\tau_{\text{opt}})$, ile ataków subtelnych udało się wykryć w każdym modelu i dlaczego model balanced radzi sobie lepiej z tą kategorią, jak parametr $\text{class}_{\text{weight}}$ wpływa na wartości współczynników β_i porównując β_{std} vs β_{balanced}, czy zwiększenie liczby próbek treningowych pomogłoby w wykrywaniu ataków subtelnych i dlaczego.

Procedura postępowania

Szczegółowa procedura postępowania dla tego zadania została opisana w Algorytmach 2, 3 i 4.

ZADANIE 3: EKSPERYMENT Z DANYMI RZECZYWISTYMI

Treść zadania

Użyj rzeczywistego zbioru danych CICIDS2017 lub UNSW-NB15 do wytrenowania i ewaluacji modelu detekcji ataków. Oba zbiory zawierają nagrania rzeczywistego ruchu sieciowego z oznaczonymi atakami różnych typów. Dla CICIDS2017 użyj plików zawierających ataki DDoS (Friday-WorkingHours-Afternoon-DDoS) oraz Port Scan (Friday-WorkingHours-Afternoon-PortScan), a także próbki normalnego ruchu (Monday-WorkingHours). Dla UNSW-NB15 użyj gotowych zbiorów treningowego i testowego.

Przed przystąpieniem do analizy wykonaj czyszczenie danych: usuń kolumny identyfikacyjne (IP addresses, Flow ID, Timestamp, Source Port) które nie powinny być używane jako cechy predykcyjne, zamień wartości nieskończone $\pm\infty$ na NaN i wypełnij wartości brakujące medianą każdej kolumny, usuń kolumny o zerowej lub prawie zerowej wariancji $\sigma^2 < 10^{-10}$ ponieważ nie niosą informacji, przekształć etykiety tekstowe na binarne gdzie 0 oznacza normalny ruch a 1 oznacza atak niezależnie od typu.

Następnie utwórz 7 cech bazowych które mają odpowiadać cechom z zadań syntetycznych. Dla CICIDS2017 oblicz cechy bezpośrednio z dostępnych kolumn lub poprzez agregację w oknach czasowych. Dla $\text{packets}_{\text{persec}}$ użyj kolumny Flow Packets/s. Dla $\text{avg}_{\text{packetsize}}$ użyj kolumny Average Packet Size. Dla $\text{connection}_{\text{duration}}$ przekształć Flow Duration z mikrosekund na sekundy dzieląc przez 10^6.

Dla $\text{syn}_{\text{ratio}}$ oblicz:

$$\text{syn\_ratio} = \frac{\text{SYN Flag Count}}{\text{Total Fwd Packets} + \text{Total Backward Packets} + \epsilon}$$

gdzie $\epsilon = 10^{-10}$ zapobiega dzieleniu przez zero, następnie obetnij do przedziału $[0, 1]$.

Dla $\text{port}_{\text{entropy}}$ pogrupuj rekordy według Source IP i przedziałów czasowych $\Delta t = 300s$, następnie dla każdej grupy oblicz entropię Shannona rozkładu wartości Destination Port:

$$H = -\sum_i p_i \cdot \log_2(p_i)$$

gdzie p_i to częstość wystąpienia i-tego unikalnego portu w grupie, $\sum_i p_i = 1$.

Dla $\text{unique}_{\text{dstips}}$ pogrupuj według Source IP i przedziałów $\Delta t = 3600s$ licząc liczbę unikalnych wartości Destination IP w każdej grupie. Dla $\text{repeated}_{\text{connections}}$ znajdź maksymalną liczbę powtórzeń tej samej pary (Source IP, Destination IP) w przedziałach $\Delta t = 900s$.

Dla UNSW-NB15 bezpośrednio użyj kolumny dur jako $\text{connection}_{\text{duration}}$. Oblicz $\text{packets}_{\text{persec}}$ = spkts / (dur + ϵ). Oblicz $\text{avg}_{\text{packetsize}}$ = (smeansz + dmeansz) / 2. Użyj $\text{ct}_{\text{srvsrc}}$ jako przybliżenia $\text{repeated}_{\text{connections}}$. Dla $\text{port}_{\text{entropy}}$ oblicz analogicznie jak dla CICIDS2017 używając kolumn sport i dsport. Cechy $\text{syn}_{\text{ratio}}$ i $\text{unique}_{\text{dstips}}$ są trudne do obliczenia

Algorithm 2 Generowanie danych realistycznych

Wejście: $n_{\text{norm}} = 950$, $n_{\text{obv}} = 20$, $n_{\text{med}} = 15$, $n_{\text{sub}} = 15$
Wyjście: $X \in \mathbb{R}^{1000\times 7}$, $y \in \{0,1\}^{1000}$
$\mu_{\text{norm}} \leftarrow [50, 800, 2.5, 0.2, 5, 30, 2]^T$
$\sigma_{\text{norm}} \leftarrow [15, 200, 0.5, 0.05, -, -, -]^T$
▷ Normalny ruch
for $i = 0$ **to** 949 **do**
    $X[i,0] \sim N(\mu_{\text{norm}}[0], \sigma_{\text{norm}}[0])$
    $X[i,1] \sim N(\mu_{\text{norm}}[1], \sigma_{\text{norm}}[1])$
    $X[i,2] \sim N(\mu_{\text{norm}}[2], \sigma_{\text{norm}}[2])$
    $X[i,3] \sim \text{clip}(N(\mu_{\text{norm}}[3], \sigma_{\text{norm}}[3]), 0, 1)$
    $X[i,4] \sim \text{Poisson}(\mu_{\text{norm}}[4]) + 1$
    $X[i,5] \sim \text{Exp}(1/\mu_{\text{norm}}[5])$
    $X[i,6] \sim \text{Poisson}(\mu_{\text{norm}}[6])$
    $y[i] \leftarrow 0$
end for
▷ Ataki oczywiste ($k = 4$)
for $i = 950$ **to** 969 **do**
    $X[i,0] \sim N(\mu_{\text{norm}}[0] + 4\sigma_{\text{norm}}[0], \sigma_{\text{norm}}[0])$
    $X[i,1] \sim N(\mu_{\text{norm}}[1] - 4\sigma_{\text{norm}}[1], \sigma_{\text{norm}}[1])$
    $X[i,2] \sim N(\mu_{\text{norm}}[2] + 4\sigma_{\text{norm}}[2], \sigma_{\text{norm}}[2])$
    $X[i,3] \sim \text{clip}(N(\mu_{\text{norm}}[3] + 4\sigma_{\text{norm}}[3], \sigma_{\text{norm}}[3]), 0, 1)$
    $X[i,4] \sim \text{Poisson}(\mu_{\text{norm}}[4] + 4 \cdot 10) + 1$
    $X[i,5] \sim \text{Exp}(1/(\mu_{\text{norm}}[5]/4))$
    $X[i,6] \sim \text{Poisson}(\mu_{\text{norm}}[6] + 4 \cdot 5)$
    $y[i] \leftarrow 1$
end for
▷ Ataki średnie ($k = 2$)
for $i = 970$ **to** 984 **do**
    $X[i,0] \sim N(\mu_{\text{norm}}[0] + 2\sigma_{\text{norm}}[0], \sigma_{\text{norm}}[0])$
    $X[i,1] \sim N(\mu_{\text{norm}}[1] - 2\sigma_{\text{norm}}[1], \sigma_{\text{norm}}[1])$
    $X[i,2] \sim N(\mu_{\text{norm}}[2] + 2\sigma_{\text{norm}}[2], \sigma_{\text{norm}}[2])$
    $X[i,3] \sim \text{clip}(N(\mu_{\text{norm}}[3] + 2\sigma_{\text{norm}}[3], \sigma_{\text{norm}}[3]), 0, 1)$
    $X[i,4] \sim \text{Poisson}(\mu_{\text{norm}}[4] + 2 \cdot 10) + 1$
    $X[i,5] \sim \text{Exp}(1/(\mu_{\text{norm}}[5]/2))$
    $X[i,6] \sim \text{Poisson}(\mu_{\text{norm}}[6] + 2 \cdot 5)$
    $y[i] \leftarrow 1$
end for
▷ Ataki subtelne ($k = 1$)
for $i = 985$ **to** 999 **do**
    $X[i,0] \sim N(\mu_{\text{norm}}[0] + 1\sigma_{\text{norm}}[0], \sigma_{\text{norm}}[0])$
    $X[i,1] \sim N(\mu_{\text{norm}}[1] - 1\sigma_{\text{norm}}[1], \sigma_{\text{norm}}[1])$
    $X[i,2] \sim N(\mu_{\text{norm}}[2] + 1\sigma_{\text{norm}}[2], \sigma_{\text{norm}}[2])$
    $X[i,3] \sim \text{clip}(N(\mu_{\text{norm}}[3] + 1\sigma_{\text{norm}}[3], \sigma_{\text{norm}}[3]), 0, 1)$
    $X[i,4] \sim \text{Poisson}(\mu_{\text{norm}}[4] + 1 \cdot 10) + 1$
    $X[i,5] \sim \text{Exp}(1/(\mu_{\text{norm}}[5]/1))$
    $X[i,6] \sim \text{Poisson}(\mu_{\text{norm}}[6] + 1 \cdot 5)$
    $y[i] \leftarrow 1$
end for
$(X, y) \leftarrow \pi(X, y)$
return X, y

Algorithm 3 Optymalizacja progu decyzyjnego

Wejście: β, β_0 (parametry modelu), X_{test}, y_{test}, $w_{FN} = 100$, $w_{FP} = 1$
Wyjście: τ_{opt}, $\hat{y}_{\text{opt}}$
for $i = 0$ **to** $|I_{\text{test}}| - 1$ **do**
    $z_i \leftarrow \beta_0 + \sum_{j=0}^{6} \beta_j X_{\text{test}}[i, j]$
    $P[i] \leftarrow 1/(1 + e^{-z_i})$
end for
$C_{\min} \leftarrow \infty$, $\tau_{\text{opt}} \leftarrow 0.5$
for $\tau = 0.01$ **to** 0.99 **step** 0.01 **do**
    $\hat{y} \leftarrow \mathbb{1}(P \geq \tau)$
    $(TP, TN, FP, FN) \leftarrow \mathcal{M}(y_{\text{test}}, \hat{y})$
    $C(\tau) \leftarrow w_{FN} \cdot FN + w_{FP} \cdot FP$
    if $C(\tau) < C_{\min}$ **then**
        $C_{\min} \leftarrow C(\tau)$
        $\tau_{\text{opt}} \leftarrow \tau$
    end if
end for
$\hat{y}_{\text{opt}} \leftarrow \mathbb{1}(P \geq \tau_{\text{opt}})$
return τ_{opt}, $\hat{y}_{\text{opt}}$

Algorithm 4 Porównanie trzech strategii

Wejście: X_{train}, y_{train}, X_{test}, y_{test}
Wyjście: Metryki dla 3 modeli
▷ Model 1: Standardowy ($w_0 = w_1 = 1$)
$(\beta^*_{\text{std}}, \beta^*_{0,\text{std}}) \leftarrow \arg\min_{\beta,\beta_0} \mathcal{L}_1(\beta, \beta_0)$
$\mathcal{L}_1(\beta, \beta_0) = -\sum_i [y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i))] + \frac{1}{2C}\|\beta\|^2$
$z_i = \beta_0 + \sum_j \beta_j X_{\text{train}}[i, j] \mid C = 1.0$
$\beta_{\text{std}} \leftarrow \beta^*_{\text{std}}$, $\beta_{0,\text{std}} \leftarrow \beta^*_{0,\text{std}}$
for $i \in I_{\text{test}}$ **do**
    $P_{\text{std}}[i] \leftarrow 1/(1 + \exp(-\beta_{0,\text{std}} - \sum_j \beta_{\text{std},j} X_{\text{test}}[i, j]))$
    $\hat{y}_{\text{std}}[i] \leftarrow \mathbb{1}(P_{\text{std}}[i] \geq 0.5)$
end for
$(TP_1, TN_1, FP_1, FN_1) \leftarrow \mathcal{M}(y_{\text{test}}, \hat{y}_{\text{std}})$
$\text{metryki}_{\text{std}} \leftarrow \{\text{Accuracy}_1, \text{Precision}_1, \text{Recall}_1, \text{F1}_1\}$
▷ Model 2: Balanced (ważone wagi klas)
$w_0 \leftarrow |I_{\text{train}}|/(2 \cdot |\{i : y_{\text{train}}[i] = 0\}|)$
$w_1 \leftarrow |I_{\text{train}}|/(2 \cdot |\{i : y_{\text{train}}[i] = 1\}|)$
$(\beta^*_{\text{bal}}, \beta^*_{0,\text{bal}}) \leftarrow \arg\min_{\beta,\beta_0} \mathcal{L}_2(\beta, \beta_0)$
$\mathcal{L}_2(\beta, \beta_0) = -\sum_i w_{y_i} [y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i))] + \frac{1}{2C}\|\beta\|^2$
$\beta_{\text{bal}} \leftarrow \beta^*_{\text{bal}}$, $\beta_{0,\text{bal}} \leftarrow \beta^*_{0,\text{bal}}$
for $i \in I_{\text{test}}$ **do**
    $P_{\text{bal}}[i] \leftarrow 1/(1 + \exp(-\beta_{0,\text{bal}} - \sum_j \beta_{\text{bal},j} X_{\text{test}}[i, j]))$
    $\hat{y}_{\text{bal}}[i] \leftarrow \mathbb{1}(P_{\text{bal}}[i] \geq 0.5)$
end for
$(TP_2, TN_2, FP_2, FN_2) \leftarrow \mathcal{M}(y_{\text{test}}, \hat{y}_{\text{bal}})$
$\text{metryki}_{\text{bal}} \leftarrow \{\text{Accuracy}_2, \text{Precision}_2, \text{Recall}_2, \text{F1}_2\}$
▷ Model 3: Próg optymalny (używa modelu 1 z dostosowanym progiem)
$\tau_{\text{opt}}, \hat{y}_{\text{thr}} \leftarrow$ Algorytm 3$(\beta_{\text{std}}, \beta_{0,\text{std}}, X_{\text{test}}, y_{\text{test}})$
$(TP_3, TN_3, FP_3, FN_3) \leftarrow \mathcal{M}(y_{\text{test}}, \hat{y}_{\text{thr}})$
$\text{metryki}_{\text{thr}} \leftarrow \{\text{Accuracy}_3, \text{Precision}_3, \text{Recall}_3, \text{F1}_3\}$
return $\{\text{metryki}_{\text{std}}, \text{metryki}_{\text{bal}}, \text{metryki}_{\text{thr}}\}$

z tego zbioru danych więc możesz użyć innych dostępnych cech jako substytutów lub pominąć je dokumentując to ograniczenie.

Podziel dane na zbiory treningowy, walidacyjny i testowy w proporcjach 60:20:20 zachowując stratyfikację. Znormalizuj cechy obliczając $\hat{\mu}_j$ i $\hat{\sigma}_j$ tylko na zbiorze treningowym, następnie stosując $z = (x - \hat{\mu})/\hat{\sigma}$ do wszystkich zbiorów. Wytrenuj model regresji logistycznej z parametrem `class_weight='balanced'` oraz maksymalną liczbą iteracji równą 1000. Oblicz prognozy $P(y = 1|x)$ na zbiorze testowym i wyznacz pełny zestaw metryk.

Zidentyfikuj wszystkie próbki sklasyfikowane jako FN oraz FP. Wybierz 10 przykładów z każdej kategorii i dla każdego przykładu wypisz wartości wszystkich 7 cech x_j oraz prawdopodobieństwo $P(y = 1|x)$ zwrócone przez model. Przeanalizuj te przypadki próbując zidentyfikować wspólne wzorce. Dla FN odpowiedz dlaczego model nie rozpoznał ataku mimo że próbka była rzeczywiście atakiem. Dla FP odpowiedz jakie cechy normalnego ruchu spowodowały że model błędnie oznaczył go jako atak.

Jeśli zbiór danych zawiera osobne etykiety dla ataków DDoS i Port Scan oblicz $\text{Recall}_{\text{DDoS}}$ i $\text{Recall}_{\text{PortScan}}$ osobno. Zidentyfikuj który typ jest trudniejszy do wykrycia i wyjaśnij dlaczego na podstawie analizy wartości $\hat{\mu}_j$ dla obu typów.

Porównaj wyniki uzyskane na danych rzeczywistych z wynikami z zadań 1 i 2 tworząc tabelę zawierającą Accuracy, Recall, FN oraz AUC dla wszystkich 3 eksperymentów. Przeanalizuj różnice odpowiadając na pytania: ile wynosi $\Delta_{\text{Accuracy}} = \text{Accuracy}_{Z3} - \text{Accuracy}_{Z1}$ oraz $\Delta_{\text{Recall}} = \text{Recall}_{Z3} - \text{Recall}_{Z1}$, co jest główną przyczyną tego spadku analizując rozkłady cech i nakładkę między klasami, które cechy zachowały swoją moc dyskryminacyjną ($|\beta_{Z3}| \approx |\beta_{Z1}|$) a które straciły na znaczeniu ($|\beta_{Z3}| \ll |\beta_{Z1}|$), czy regresja logistyczna jest wystarczająca dla tego problemu czy potrzeba bardziej złożonych modeli.

Wygeneruj następujące wizualizacje: 7 histogramów pokazujących rozkłady każdej z 7 cech bazowych z podziałem na normalny ruch i ataki z zaznaczonymi μ i σ dla obu klas; mapę ciepła macierzy korelacji ρ_{ij} między wszystkimi 7 cechami ze skalą od -1 do +1; macierz pomyłek z liczbami i procentami; krzywą ROC z wartością AUC; wykres słupkowy współczynników β_j uporządkowanych według $|\beta_j|$; wykres słupkowy pokazujący Recall osobno dla normalnego ruchu, ataków DDoS i ataków Port Scan jeśli dostępne; wykres krzywej uczenia pokazujący Accuracy(n) na zbiorze treningowym i walidacyjnym w funkcji rozmiaru zbioru treningowego $n \in \{0.1N, 0.25N, 0.5N, 0.75N, N\}$.

Procedura postępowania

Szczegółowa procedura postępowania dla tego zadania została opisana w Algorytmach 5, 6 i 7. Algorytm 5 opisuje czyszczenie danych, Algorytm 6 inżynierię cech, a Algorytm 7 analizę błędów. Podział danych (60:20:20), normalizację i trenowanie modelu wykonaj analogicznie do Algorytmu 1 (linie 27-40), dostosowując proporcje podziału oraz dodając parametr `class_weight='balanced'` w regresji logistycznej.

DODATEK A: SZCZEGÓŁOWE DEFINICJE METRYK

Macierz pomyłek

Dla problemu klasyfikacji binarnej gdzie $y \in \{0, 1\}$ oraz $\hat{y} \in \{0, 1\}$:

$$TP = \sum_{i=1}^{N} \mathbb{1}(y_i = 1 \wedge \hat{y}_i = 1) \quad \text{(prawdziwie dodatnie)}$$

$$TN = \sum_{i=1}^{N} \mathbb{1}(y_i = 0 \wedge \hat{y}_i = 0) \quad \text{(prawdziwie ujemne)}$$

$$FP = \sum_{i=1}^{N} \mathbb{1}(y_i = 0 \wedge \hat{y}_i = 1) \quad \text{(fałszywie dodatnie)}$$

$$FN = \sum_{i=1}^{N} \mathbb{1}(y_i = 1 \wedge \hat{y}_i = 0) \quad \text{(fałszywie ujemne)}$$

gdzie $\mathbb{1}(\cdot)$ jest funkcją wskaźnikową zwracającą 1 gdy warunek jest spełniony, 0 w przeciwnym przypadku.

Algorithm 5 Czyszczenie i przygotowanie danych rzeczywistych

Wejście: Ścieżki f_1, f_2, f_3 do plików CSV (CICIDS2017 lub UNSW-NB15)
Wyjście: $X \in \mathbb{R}^{n \times d'}$, $y \in \{0,1\}^n$
▷ Wczytanie i konkatenacja
$D_1 \leftarrow \text{read\_csv}(f_1)$
$D_2 \leftarrow \text{read\_csv}(f_2)$
$D_3 \leftarrow \text{read\_csv}(f_3)$
$D \leftarrow D_1 \oplus D_2 \oplus D_3$
▷ Czyszczenie
$\text{cols}_{\text{remove}} \leftarrow \{\texttt{'Flow ID'}, \texttt{'Source IP'}, \texttt{'Destination IP'}, \texttt{'Timestamp'}, \texttt{'Source Port'}\}$
$D \leftarrow D \setminus \text{cols}_{\text{remove}}$
for $c = 0$ **to** $\text{num\_cols}(D) - 1$ **do**
  if $\text{is\_numeric}(D[:, c])$ **then**
    for $i = 0$ **to** $n - 1$ **do**
      if $D[i, c] = \pm\infty$ **then**
        $D[i, c] \leftarrow \text{NaN}$
      end if
    end for
    $m \leftarrow \text{Med}[\{D[i, c] : D[i, c] \neq \text{NaN}\}]$
    for $i = 0$ **to** $n - 1$ **do**
      if $D[i, c] = \text{NaN}$ **then**
        $D[i, c] \leftarrow m$
      end if
    end for
  end if
end for
▷ Usuwanie kolumn o zerowej wariancji
for $c = 0$ **to** $\text{num\_cols}(D) - 1$ **do**
  if $\text{Var}[D[:, c]] < 10^{-10}$ **then**
    $D \leftarrow D \setminus \{\text{col}_c\}$
  end if
end for
▷ Ekstrakcja etykiet
$L \leftarrow D_{\text{Label}}$
$y \leftarrow \mathbb{1}(L \neq L_{\text{norm}})$
$\text{cols}_{\text{label}} \leftarrow \{D_{\text{Label}}, D_{\text{DestPort}}\}$
$X \leftarrow D \setminus \text{cols}_{\text{label}}$
return X, y

Algorithm 6 Inżynieria cech dla danych rzeczywistych

Wejście: X_{raw} (oryginalne cechy), metadane (timestamp, IPs)
Wyjście: $X_{\text{base}} \in \mathbb{R}^{n \times 7}$ (7 cech bazowych)
▷ Cechy proste
$X_{\text{base}}[:, 0] \leftarrow X_{\text{raw}}[\texttt{'Flow Packets/s'}]$
$X_{\text{base}}[:, 1] \leftarrow X_{\text{raw}}[\texttt{'Average Packet Size'}]$
$X_{\text{base}}[:, 2] \leftarrow X_{\text{raw}}[\texttt{'Flow Duration'}]/10^6$
▷ SYN ratio
$\text{total} \leftarrow X_{\text{raw}}[\texttt{'Total Fwd'}] + X_{\text{raw}}[\texttt{'Total Bwd'}] + 10^{-10}$
$X_{\text{base}}[:, 3] \leftarrow \text{clip}(X_{\text{raw}}[\texttt{'SYN Flag Count'}]/\text{total}, 0, 1)$
▷ Port entropy (agregacja w oknach czasowych $\Delta t = 300s$)
$\pi \leftarrow \text{argsort}(t)$
$t \leftarrow t[\pi]$, source_ip $\leftarrow$ source_ip$[\pi]$, dest_port $\leftarrow$ dest_port$[\pi]$
for $i = 0$ **to** $n - 1$ **do**
  $W \leftarrow \{j : t[i] - 300 \leq t[j] \leq t[i] \wedge \text{source\_ip}[j] = \text{source\_ip}[i]\}$
  $\{u_1, \ldots, u_k\} \leftarrow \text{unique}(\{\text{dest\_port}[j] : j \in W\})$
  for $\ell = 1$ **to** k **do**
    $p_\ell \leftarrow |\{j \in W : \text{dest\_port}[j] = u_\ell\}|/|W|$
  end for
  $X_{\text{base}}[i, 4] \leftarrow -\sum_{\ell=1}^{k} p_\ell \log_2(p_\ell)$
end for
▷ Unique destination IPs (okna $\Delta t = 3600s$)
for $i = 0$ **to** $n - 1$ **do**
  $W \leftarrow \{j : t[i] - 3600 \leq t[j] \leq t[i] \wedge \text{source\_ip}[j] = \text{source\_ip}[i]\}$
  $X_{\text{base}}[i, 5] \leftarrow |\{\text{dest\_ip}[j] : j \in W\}|$
end for
▷ Repeated connections (okna $\Delta t = 900s$)
for $i = 0$ **to** $n - 1$ **do**
  $W \leftarrow \{j : t[i] - 900 \leq t[j] \leq t[i] \wedge \text{source\_ip}[j] = \text{source\_ip}[i]\}$
  $\{(s_1, d_1), \ldots, (s_m, d_m)\} \leftarrow \text{unique}(\{(\text{source\_ip}[j], \text{dest\_ip}[j]) : j \in W\})$
  for $\ell = 1$ **to** m **do**
    $\text{cnt}_\ell \leftarrow |\{j \in W : (\text{source\_ip}[j], \text{dest\_ip}[j]) = (s_\ell, d_\ell)\}|$
  end for
  $X_{\text{base}}[i, 6] \leftarrow \max_\ell \text{cnt}_\ell$
end for
return X_{base}

Algorithm 7 Analiza błędów klasyfikacji

Wejście: X_{test}, y_{test}, $\hat{y}_{\text{test}}$, P_{test}, μ_{norm}, σ_{norm}
Wyjście: Statystyki błędów FN i FP
▷ Identyfikacja błędów
$I_{FN} \leftarrow \{i : y_{\text{test}}[i] = 1 \wedge \hat{y}_{\text{test}}[i] = 0\}$
$I_{FP} \leftarrow \{i : y_{\text{test}}[i] = 0 \wedge \hat{y}_{\text{test}}[i] = 1\}$
▷ Analiza False Negatives
$k_{FN} \leftarrow \min(10, |I_{FN}|)$
$S_{FN} \leftarrow \text{sample}(I_{FN}, k_{FN})$
for $j = 0$ **to** 6 **do**
    $\mu_{FN}[j] \leftarrow \frac{1}{|I_{FN}|} \sum_{i \in I_{FN}} X_{\text{test}}[i, j]$
end for
$\text{Cechy}_{FN} \leftarrow \{j \in \{0, \ldots, 6\} : |\mu_{FN}[j] - \mu_{\text{norm}}[j]| < \epsilon\}$
▷ Analiza False Positives
$k_{FP} \leftarrow \min(10, |I_{FP}|)$
$S_{FP} \leftarrow \text{sample}(I_{FP}, k_{FP})$
for $j = 0$ **to** 6 **do**
    $\mu_{FP}[j] \leftarrow \frac{1}{|I_{FP}|} \sum_{i \in I_{FP}} X_{\text{test}}[i, j]$
end for
$\text{Cechy}_{FP} \leftarrow \{j \in \{0, \ldots, 6\} : |\mu_{FP}[j] - \mu_{\text{norm}}[j]| > 2\sigma_{\text{norm}}[j]\}$
▷ Porównanie typów ataków (jeśli dostępne etykiety typu)
if $\exists T_{\text{type}} \in D$ **then**
    $I_{\text{DDoS}} \leftarrow \{i : y_{\text{test}}[i] = 1 \wedge T_{\text{type}}[i] = T_{\text{DDoS}}\}$
    $I_{\text{Port}} \leftarrow \{i : y_{\text{test}}[i] = 1 \wedge T_{\text{type}}[i] = T_{\text{Port}}\}$
    $\text{Recall}_{\text{DDoS}} \leftarrow \frac{|\{i \in I_{\text{DDoS}} : \hat{y}_{\text{test}}[i] = 1\}|}{|I_{\text{DDoS}}|}$
    $\text{Recall}_{\text{Port}} \leftarrow \frac{|\{i \in I_{\text{Port}} : \hat{y}_{\text{test}}[i] = 1\}|}{|I_{\text{Port}}|}$
    for $j = 0$ **to** 6 **do**
        $\mu_{\text{DDoS}}[j] \leftarrow \frac{1}{|I_{\text{DDoS}}|} \sum_{i \in I_{\text{DDoS}}} X_{\text{test}}[i, j]$
        $\mu_{\text{Port}}[j] \leftarrow \frac{1}{|I_{\text{Port}}|} \sum_{i \in I_{\text{Port}}} X_{\text{test}}[i, j]$
    end for
    return $\{S_{FN}, S_{FP}, \mu_{FN}, \mu_{FP}, \text{Cechy}_{FN}, \text{Cechy}_{FP}, \mu_{\text{DDoS}}, \mu_{\text{Port}}\}$
else
    return $\{S_{FN}, S_{FP}, \mu_{FN}, \mu_{FP}, \text{Cechy}_{FN}, \text{Cechy}_{FP}\}$
end if

Podstawowe metryki

$$\begin{aligned}
\text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\
\text{Precision} &= \frac{TP}{TP + FP} \\
\text{Recall (Czułość)} &= \frac{TP}{TP + FN} \\
\text{Specificity (Swoistość)} &= \frac{TN}{TN + FP} \\
\text{F1-score} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}
\end{aligned}$$

Regresja logistyczna

Model regresji logistycznej oblicza prawdopodobieństwo przynależności do klasy pozytywnej:

$$P(y = 1|x) = \sigma(\beta_0 + \sum_{j=1}^{d} \beta_j x_j) = \frac{1}{1 + \exp(-(\beta_0 + \sum_{j=1}^{d} \beta_j x_j))}$$

gdzie $\sigma(z) = 1/(1 + e^{-z})$ jest funkcją sigmoidalną.

Decyzja klasyfikacyjna:

$$\hat{y} = \begin{cases} 1 & \text{jeśli } P(y = 1|x) \geq \tau \\ 0 & \text{jeśli } P(y = 1|x) < \tau \end{cases}$$

gdzie τ jest progiem decyzyjnym (domyślnie $\tau = 0.5$).

Krzywa ROC i AUC

Krzywa ROC (Receiver Operating Characteristic) przedstawia zależność między czułością $\text{TPR}(\tau) = TP(\tau)/(TP(\tau) + FN(\tau))$ a współczynnikiem fałszywych alarmów $\text{FPR}(\tau) = FP(\tau)/(FP(\tau) + TN(\tau))$ dla różnych wartości progu decyzyjnego $\tau \in [0, 1]$.

Pole pod krzywą ROC (Area Under Curve):

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(x))\, dx$$

Interpretacja: AUC = 0.5 odpowiada losowemu klasyfikatorowi, AUC = 1.0 odpowiada idealnemu klasyfikatorowi.

Entropia Shannona

Dla dyskretnej zmiennej losowej X przyjmującej wartości $\{x_1, x_2, \ldots, x_n\}$ z prawdopodobieństwami $\{p_1, p_2, \ldots, p_n\}$ gdzie $\sum_{i=1}^{n} p_i = 1$:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

z konwencją $0 \log_2(0) = 0$. Entropia jest miarą niepewności lub losowości rozkładu, wyrażoną w bitach.

Współczynnik korelacji Pearsona

Dla dwóch zmiennych X i Y reprezentowanych przez wektory $(x_1, \ldots, x_n)$ i $(y_1, \ldots, y_n)$:

$$\rho_{XY} = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

gdzie $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ oraz $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ są średnimi.

Zakres: $\rho_{XY} \in [-1, 1]$ gdzie $|\rho_{XY}| = 1$ oznacza doskonałą korelację liniową, $\rho_{XY} = 0$ oznacza brak korelacji liniowej.

DODATEK B: WYMAGANIA TECHNICZNE

Struktura projektu

Organizacja katalogów:

```
projekt/
  data/
    CICIDS2017/
      Friday-WorkingHours-Afternoon-DDoS.pcap_ISCX.csv
      Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv
      Monday-WorkingHours.pcap_ISCX.csv
    UNSW-NB15/
      UNSW_NB15_training-set.csv
      UNSW_NB15_testing-set.csv
  notebooks/
    zadanie1_dane_idealne.ipynb
    zadanie2_dane_realistyczne.ipynb
    zadanie3_dane_rzeczywiste.ipynb
  src/
    data_generation.py
    feature_engineering.py
    model_training.py
    visualization.py
  results/
    figures/
    metrics/
  README.md
```

Format raportu końcowego

Raport powinien zawierać:

1. Streszczenie (maksymalnie 1 strona)
2. Opis metodologii dla każdego zadania z odwołaniami do Algorytmów 1-7
3. Wyniki numeryczne w tabelach
4. Wszystkie wymagane wizualizacje
5. Odpowiedzi na wszystkie pytania analityczne
6. Wnioski i dyskusja
7. Referencje do wykorzystanych zbiorów danych

Format: PDF lub Jupyter Notebook wyeksportowany do HTML

DODATEK C: WSKAZÓWKI I NAJCZĘSTSZE BŁĘDY

Typowe problemy

1. **Wycieki danych (data leakage)**: Nigdy nie obliczaj parametrów normalizacji ($\hat{\mu}$, $\hat{\sigma}$) na całym zbiorze. Zawsze obliczaj je tylko na zbiorze treningowym i stosuj do wszystkich zbiorów (patrz linie 33-36 w Algorytmie 1).
2. **Niezbalansowane klasy**: Dla realistycznych danych gdzie ataki stanowią $< 5\%$ próbek, zawsze używaj stratyfikacji przy podziale danych i rozważ użycie `class_weight='balanced'` (Algorytm 3, linia 9).
3. **Interpretacja współczynników**: Współczynniki β_j są interpretowalne tylko dla znormalizowanych cech. Im większe $|\beta_j|$, tym większy wpływ cechy j na decyzję modelu.
4. **Wybór progu**: Domyślny próg $\tau = 0.5$ nie zawsze jest optymalny. Dla problemów gdzie koszt FN jest dużo większy niż koszt FP, użyj niższego progu (Algorytm 2 pokazuje optymalizację progu).

Optymalizacja wydajności

Dla dużych zbiorów danych (> 1 milion próbek):

- Użyj próbkowania losowego do wybrania podzbioru dla początkowych eksperymentów
- Rozważ użycie algorytmu SGD zamiast standardowej regresji logistycznej
- Przetwarzaj dane w partiach przy obliczaniu cech wymagających agregacji czasowej (Algorytm 6, linie 15-30)

Wizualizacje - dobre praktyki

1. Zawsze podpisuj osie (x, y) z jednostkami
2. Dodaj legendy wyjaśniające kolory i symbole
3. Używaj rozmiarów czcionek czytelnych w druku ($\geq 10pt$)
4. Zapisuj wykresy w wysokiej rozdzielczości (DPI ≥ 300)
5. Dla macierzy korelacji używaj dywergentnej skali kolorów (np. czerwony-biały-niebieski)

DODATEK D: DODATKOWE ZADANIA OPCJONALNE

Rozszerzenie 1: Inżynieria cech

Wymień cechy dodatkowe, które mogą poprawić wykrywanie ataków:

- Cechy oparte na statystykach okien czasowych: średnia ruchoma, odchylenie standardowe w oknie
- Cechy oparte na grafach: stopień węzła w grafie połączeń, współczynnik klasteryzacji
- Cechy behawioralne: odstępstwa od profilu normalnego zachowania dla danego IP

Zaimplementuj co najmniej 3 nowe cechy i oceń ich wpływ na Recall i AUC.

Rozszerzenie 2: Porównanie modeli

Wytrenuj i porównaj następujące modele:

- Drzewo decyzyjne (`DecisionTreeClassifier`)
- Las losowy (`RandomForestClassifier`)
- Gradient Boosting (`XGBClassifier` lub `LightGBM`)
- Sieć neuronowa (`MLPClassifier`)

Dla każdego modelu oblicz te same metryki i stwórz wykres porównawczy. Odpowiedz: który model osiąga najlepszy kompromis między Recall a Precision?

Rozszerzenie 3: Wykrywanie w czasie rzeczywistym

Zaprojektuj system wykrywania ataków działający w czasie rzeczywistym:

1. Okno buforowe: agreguj pakiety w oknach $\Delta t = 10s$
2. Obliczanie cech: dla każdego okna oblicz 7 cech bazowych
3. Predykcja: użyj wytrenowanego modelu do klasyfikacji okna
4. Alarm: jeśli $P(y = 1|x) > \tau_{\text{alarm}}$, wygeneruj alert

Zasymuluj działanie systemu na danych testowych obliczając opóźnienie detekcji $\Delta t_{\text{detect}} = t_{\text{alarm}} - t_{\text{atak}}$ oraz liczbę fałszywych alarmów na godzinę.

Rozszerzenie 4: Analiza adversarial

Zbadaj podatność modelu na ataki adversarial:

1. Wybierz próbkę ataku prawidłowo sklasyfikowaną: $\hat{y} = 1$
2. Znajdź minimalną perturbację δ taką że $\hat{y}(x + \delta) = 0$

3. Wymień cechy najbardziej wrażliwe na perturbacje

Metoda: dla każdej cechy j zmień wartość x_j o $\pm 0.1\sigma_j$ i sprawdź czy klasyfikacja się zmienia.

DODATEK E: PRZEWODNIK IMPLEMENTACYJNY

Ten dodatek zawiera mapowanie symboli matematycznych używanych w algorytmach na konkretny kod Python. Jego celem jest ułatwienie przełożenia formalnych algorytmów na działającą implementację.

Podstawowe operacje statystyczne

Symbol	Kod Python	Uwagi
$\mathbb{E}[X]$	`np.mean(X)` lub `X.mean()`	Średnia arytmetyczna
$\sigma[X]$	`np.std(X)` lub `X.std()`	Odchylenie standardowe
$\mathrm{Var}[X]$	`np.var(X)` lub `X.var()`	Wariancja
$\mathrm{Med}[X]$	`np.median(X)` lub `X.median()`	Mediana

Operacje na danych

Symbol	Kod Python
read_csv(f)	`pd.read_csv(f)`
$D_1 \oplus D_2$	`pd.concat([D1, D2], ignore_index=True)`
$X \setminus \{\text{kol}\}$	`X.drop(columns=[kol])`
clip(x, a, b)	`np.clip(x, a, b)`
unique(S)	`np.unique(S)` lub `set(S)`
argsort(X)	`np.argsort(X)`
$\pi(X, y)$	`sklearn.utils.shuffle(X, y, random_state=42)`
$\mathbf{0}_{n \times m}$	`np.zeros((n, m))`
$\mathbf{0}_n$	`np.zeros(n)`
is_numeric(X)	`pd.api.types.is_numeric_dtype(X)`
num_cols(D)	`D.shape[1]` lub `len(D.columns)`
sample(S, k)	`np.random.choice(S, k, replace=False)`

Czyszczenie danych - wypełnianie wartości brakujących

W Algorytmie 5 (linie 419-429) wypełnianie wartości NaN medianą jest realizowane przez pętlę:

```
m <- Med[{D[i,c] : D[i,c] != NaN}]
FOR i = 0 TO n-1
    IF D[i,c] = NaN
        D[i,c] <- m
```

W Pythonie można to uprościć używając metody `fillna()`:

```
m = D.iloc[:, c].median()  # Med[...]
D.iloc[:, c] = D.iloc[:, c].fillna(m)  # wypełnienie NaN
```

lub krócej dla wszystkich kolumn numerycznych:

```
D = D.fillna(D.median(numeric_only=True))
```

Podział i transformacje danych

Symbol	Kod Python

$\text{stratified\_split}(I, y, p)$	`from sklearn.model_selection import train_test_split`
	`I_train, I_test = train_test_split(`
	`  I, train_size=p, random_state=42,`
	`  stratify=y`
	`)`

$\mathbb{1}(P \geq \tau)$	`(P >= tau).astype(int)` lub `np.where(P >= tau, 1, 0)`
$\arg\min_x f(x)$	Użyj optymalizatora, np. `scipy.optimize.minimize` lub solwera ML

Regresja logistyczna w implementacji

W algorytmach regresja logistyczna jest definiowana matematycznie przez optymalizację $\arg\min \mathcal{L}(\beta, \beta_0)$. W implementacji Python używamy gotowego solwera:

Operacja w algorytmie	Kod Python

$(\beta^*, \beta_0^*) \leftarrow \arg\min_{\beta,\beta_0} \mathcal{L}(\beta, \beta_0)$	`from sklearn.linear_model import LogisticRegression`
	`M = LogisticRegression(`
	`  C=1.0,`
	`  class_weight='balanced',  # opcjonalnie`
	`  max_iter=1000,`
	`  random_state=42`
	`)`
	`M.fit(X_train, y_train)`
	`beta = M.coef_[0]`
	`beta_0 = M.intercept_[0]`

$P[i] \leftarrow \sigma(z_i)$
$\hat{y}[i] \leftarrow \mathbb{1}(P[i] \geq \tau)$
	`P = M.predict_proba(X_test)[:, 1]`
	`y_hat = M.predict(X_test)`
	lub
	`y_hat = (P >= tau).astype(int)`

Ekstrakcja parametrów	`beta = M.coef_[0]` `beta_0 = M.intercept_[0]`

Uwaga: W algorytmach optymalizacja jest zapisana jako $\arg\min \mathcal{L}(\beta, \beta_0)$, co odpowiada wywołaniu `M.fit(X_train, y_train)` w implementacji. Parametry β i β_0 są dostępne przez `M.coef_[0]` i `M.intercept_[0]`.

Macierz pomyłek i metryki

Symbol	Kod Python

$\mathcal{M}(y, \hat{y})$	`from sklearn.metrics import confusion_matrix`
	`tn, fp, fn, tp = confusion_matrix(`
	`  y, y_hat`
	`).ravel()`

Metryki klasyfikacji	`from sklearn.metrics import (`
	`  accuracy_score, precision_score,`
	`  recall_score, f1_score, roc_auc_score`
	`)`
	`accuracy = accuracy_score(y, y_hat)`
	`precision = precision_score(y, y_hat)`
	`recall = recall_score(y, y_hat)`
	`f1 = f1_score(y, y_hat)`
	`auc = roc_auc_score(y, y_proba)`

Uwaga: W algorytmach metryki obliczane są bezpośrednio ze wzorów (np. Accuracy $\leftarrow (TP+TN)/(TP+TN+FP+FN)$). W implementacji używamy gotowych funkcji z `sklearn.metrics`.

Entropia i operacje na zbiorach

Symbol	Kod Python

$H(\{p_i\})$	`from scipy.stats import entropy`
	`H = entropy(p, base=2)`
	lub
	`H = -np.sum(p * np.log2(p + 1e-10))`

$\lvert\{x_i : i \in I\}\rvert$	`len(set(x[i] for i in I))` lub `x[I].nunique()`

Pełny przykład implementacji

Poniżej znajduje się kompletny przykład implementacji Algorytmu 1 w języku Python:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, accuracy_score,
                             precision_score, recall_score,
                             f1_score, roc_auc_score)
from sklearn.utils import shuffle

# Generowanie danych syntetycznych (Algorytm 1)
np.random.seed(42)
n_norm, n_atak = 800, 200
X = np.zeros((1000, 7))
y = np.zeros(1000)

# Normalny ruch
for i in range(800):
    X[i, 0] = np.random.normal(50, 15)
    X[i, 1] = np.random.normal(800, 200)
    X[i, 2] = np.random.normal(2.5, 0.5)
    X[i, 3] = np.clip(np.random.normal(0.2, 0.05), 0, 1)
    X[i, 4] = np.random.poisson(5) + 1
    X[i, 5] = np.random.exponential(30)
    X[i, 6] = np.random.poisson(2)

# Ataki
for i in range(800, 1000):
    X[i, 0] = np.random.normal(250, 30)
    X[i, 1] = np.random.normal(300, 100)
    X[i, 2] = np.random.normal(4.0, 0.3)
    X[i, 3] = np.clip(np.random.normal(0.8, 0.05), 0, 1)
    X[i, 4] = np.random.poisson(50) + 1
    X[i, 5] = np.random.exponential(2)
    X[i, 6] = np.random.poisson(20)
    y[i] = 1

# Permutacja losowa
X, y = shuffle(X, y, random_state=42)

# Podział danych
indices = np.arange(1000)
I_train, I_test = train_test_split(
    indices, train_size=0.7, random_state=42, stratify=y
)
X_train, X_test = X[I_train], X[I_test]
```

```
y_train, y_test = y[I_train], y[I_test]

# Normalizacja
mu_hat = X_train.mean(axis=0)
sigma_hat = X_train.std(axis=0)
X_train = (X_train - mu_hat) / sigma_hat
X_test = (X_test - mu_hat) / sigma_hat

# Trenowanie modelu
M = LogisticRegression(C=1.0, max_iter=1000, random_state=42)
M.fit(X_train, y_train)
beta = M.coef_[0]
beta_0 = M.intercept_[0]

# Ewaluacja
y_hat_test = M.predict(X_test)
P = M.predict_proba(X_test)[:, 1]
tn, fp, fn, tp = confusion_matrix(y_test, y_hat_test).ravel()

metrics = {
    'Accuracy': accuracy_score(y_test, y_hat_test),
    'Precision': precision_score(y_test, y_hat_test),
    'Recall': recall_score(y_test, y_hat_test),
    'F1': f1_score(y_test, y_hat_test),
    'AUC': roc_auc_score(y_test, P)
}

print(f"TP={tp}, TN={tn}, FP={fp}, FN={fn}")
for metric, value in metrics.items():
    print(f"{metric}: {value:.4f}")
```

Najważniejsze wskazówki praktyczne

1. **Zawsze używaj `random_state`**: We wszystkich funkcjach losowych ustaw parametr `random_state` aby wyniki były powtarzalne. Dotyczy to między innymi funkcji `train_test_split` oraz `LogisticRegression`.
2. **Stratyfikacja dla niezbalansowanych danych**: Gdy klasy są niezbalansowane (jak w zadaniu 2 i 3), ZAWSZE używaj `stratify=y` w funkcji `train_test_split` aby zachować proporcje klas.
3. **Normalizacja - kluczowe zasady**:
   - Oblicz parametry normalizacji $(\hat{\mu}, \hat{\sigma})$ TYLKO na zbiorze treningowym
   - Zastosuj te same parametry do zbiorów testowego i walidacyjnego
   - NIGDY nie używaj `fit_transform()` na zbiorze testowym - tylko `transform()`
4. **Nazewnictwo zgodne z notacją**: Możesz używać nazw zmiennych zbliżonych do symboli matematycznych:

   ```
   mu_hat = X_train.mean()      # zamiast "mean"
   sigma_hat = X_train.std()    # zamiast "std"
   y_hat = M.predict(X_test)    # zamiast "y_pred"
   ```

5. **Dokumentacja z odwołaniami do algorytmów**: W komentarzach możesz odwoływać się do numerów algorytmów:

   ```
   # Algorytm 1: normalizacja cech
   mu_hat = X_train.mean(axis=0)
   sigma_hat = X_train.std(axis=0)
   X_train_scaled = (X_train - mu_hat) / sigma_hat
   ```

Częste błędy i ich unikanie

Błąd	**Poprawne rozwiązanie**
Normalizacja na całym zbiorze	Normalizuj tylko na zbiorze treningowym
`fit_transform(X_test)`	Użyj `transform(X_test)`
Brak `random_state`	Zawsze ustaw `random_state=42`
Brak stratyfikacji	Użyj `stratify=y` dla niezbalansowanych danych
Dzielenie przez zero	Dodaj małą stałą: `x / (y + 1e-10)`