

POLITECHNIKA LUBELSKA
WYDZIAŁ MATEMATYKI I INFORMATYKI
TECHNICZNEJ

Kierunek: Inżynieria i Analiza Danych



Projekt Zaliczeniowy z Zakresu Metod Walidacji

Praca wykonana przez:

Mateusz Drozd, Nr albumu: s100966

Wprowadzenie

W dzisiejszych czasach platformy społecznościowe, fora internetowe i sekcje komentarzy na stronach internetowych są zalewane setkami tysięcy opinii—w tym także treściami o charakterze nienawiści. Ręczne moderowanie jest czasochłonne i kosztowne, dlatego coraz częściej stosuje się **automatyczne modele klasyfikacji tekstu**.

Celem tego projektu jest **zbudowanie modelu uczenia maszynowego, który będzie klasyfikował komentarze jako hejt(1) lub treści neutralne (0)**. W tym celu wykorzystamy **zaawansowane techniki przetwarzania języka naturalnego (NLP) oraz modele uczenia maszynowego**.

Zbiór danych

Wpisy obejmują tweety zebrane z publicznie dostępnych dyskusji na **Twitterze**. Zbiór ten stanowi część polskiego benchmarku **KLEJ** (Kompleksowa Lista Ewaluacji Językowych) dostępnego pod adresem: <https://klejbenchmark.com/tasks/>. Ponieważ automatyczne wykrywanie cyberprzemocy często zależy od doboru cech (feature selection) i inżynierii cech (feature engineering), tweety są udostępniane w **surowej formie, z minimalnym przetwarzaniem**. Przetwarzanie dotyczy głównie sytuacji, gdy **ujawniane są informacje o osobach prywatnych**. Zbiór z KLEJ służy jako narzędzie do testowania modeli NLP w **zakresie identyfikacji szkodliwych treści w języku polskim**.

Dane treningowe pochodzą z pliku `train.tsv`, który zawiera następujące kolumny:

- **sentence** – zawiera tekst komentarza.
- **target** – etykieta przypisana do komentarza, gdzie:
 - `0` oznacza treść neutralną,
 - `1` oznacza komentarz nacechowany hejtem.

Wstępna analiza danych

Przed rozpoczęciem przetwarzania i budowy modelu wykonujemy wstępную eksplorację danych, która obejmuje:

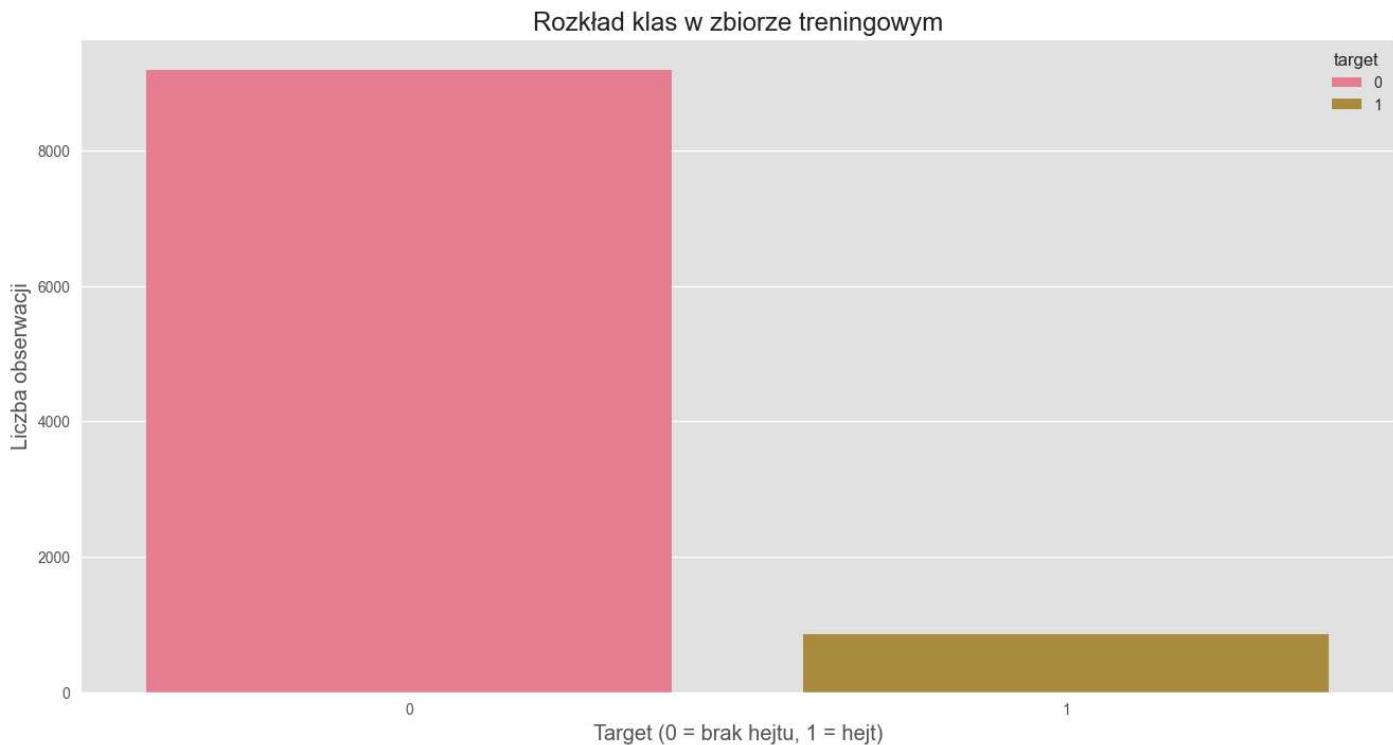
- **Wczytanie danych:** Importujemy dane do obiektu DataFrame przy użyciu biblioteki pandas.
- **Podstawowe statystyki i podgląd rekordów:** Sprawdzamy informacje o typach danych, rozkład poszczególnych klas oraz podstawowe statystyki, co pomaga zidentyfikować potencjalne problemy (np. brakujące dane, niezbalansowany rozkład etykiet).
- **Analiza długości tekstu i liczby słów:** Dodanie kolumn takich jak `word_count` oraz `text_length` umożliwia lepsze zrozumienie struktury komentarzy oraz identyfikację ewentualnych outlierów.

Dzięki temu etapowi uzyskujemy wstępny obraz jakości danych, co pozwala na świadome podejmowanie decyzji dotyczących dalszego przygotowania danych.

1. Wizualizacja danych

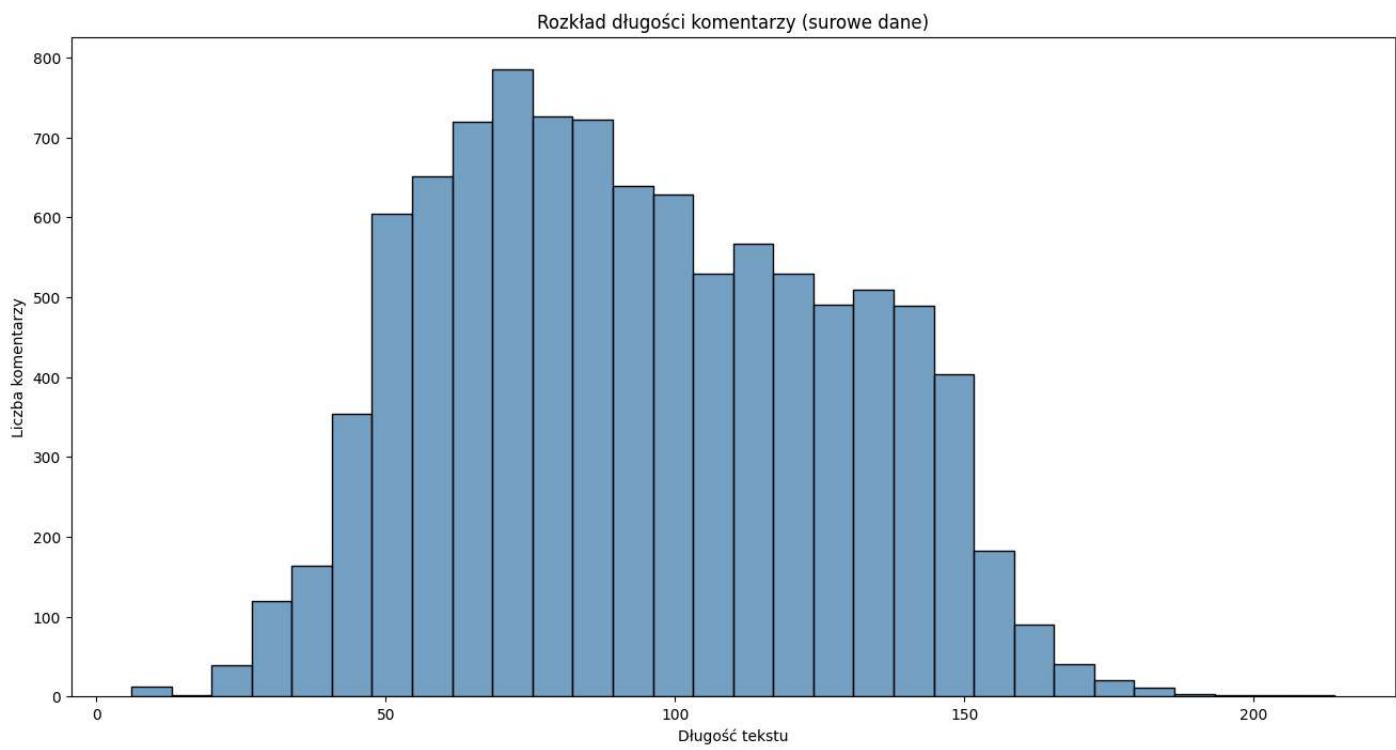
Celem tego etapu jest wizualizacja rozkładów danych oraz prezentacja najczęściej występujących słów. Dzięki tej analizie możemy:

- Szybko zidentyfikować rozkład długości komentarzy i liczbę słów,
- Sprawdzić, czy występują ewentualne anomalie (np. komentarze z bardzo małą lub bardzo dużą liczbą słów),
- Zobaczyć, jak rozkładają się klasy (np. hejt vs. brak hejtu),
- Zyskać intuicyjny podgląd na dominujące słowa i kluczowe tematy poprzez chmurę słów (wordcloud).



Rozkład klas w zbiorze treningowym

Powyższy wykres prezentuje liczbę obserwacji w poszczególnych klasach komentarzy (0 – brak hejtu, 1 – hejt). Na osi X znajdują się etykiety klas, a na osi Y – liczba przykładów. Na podstawie wykresu widać, że klasa 0 (brak hejtu) stanowi zdecydowaną większość danych (9190 obserwacji), a klasa 1 jest znaczco mniej liczna (851 obserwacji).



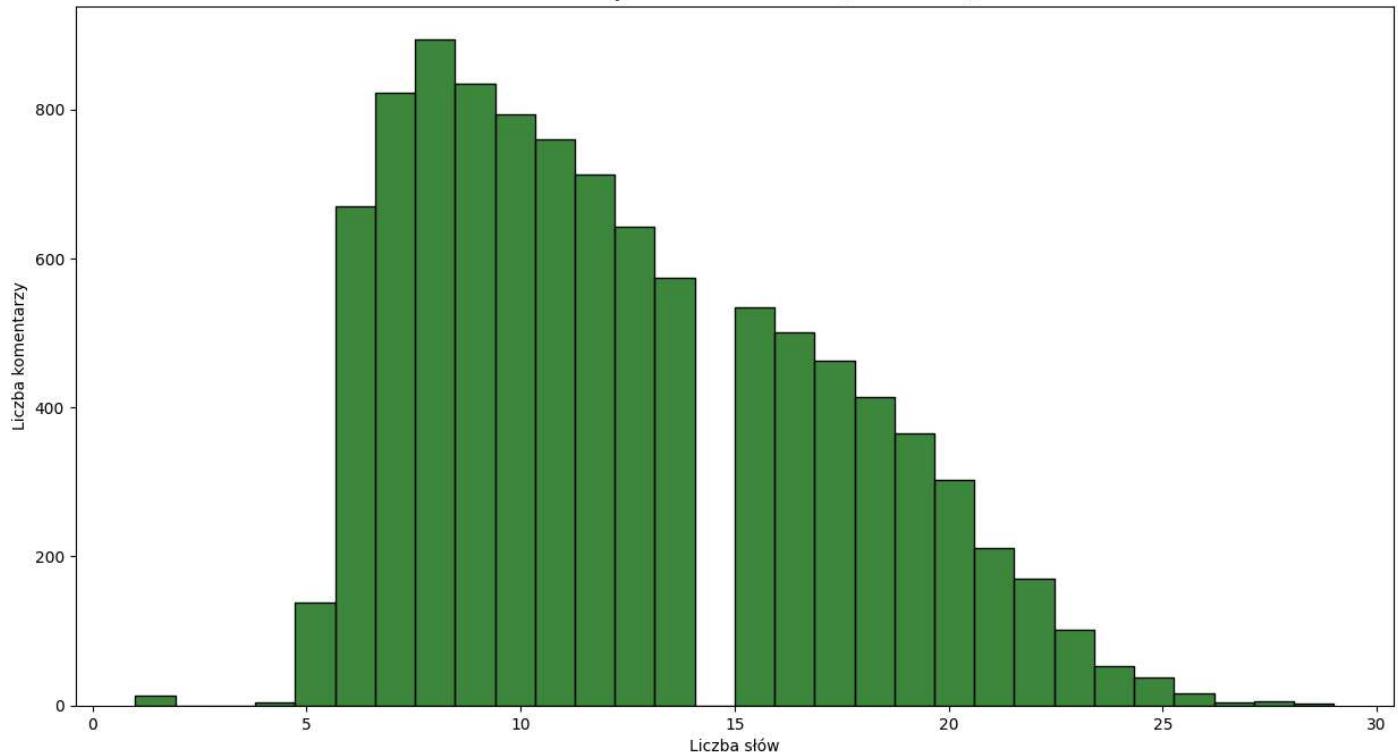
Rozkład długości komentarzy (surowe dane)

Na powyższym wykresie przedstawiono histogram długości komentarzy (w liczbie znaków). Na osi X znajduje się długość tekstu, a na osi Y – liczba komentarzy o danej długości.

Możemy zauważyć, że:

- **Najwięcej komentarzy** ma długość w zakresie około 30–90 znaków.
- Rozkład jest **zblizony do rozkładu normalnego** z długim ogonem po prawej stronie (dla wyjątkowo długich komentarzy).
- **Większość komentarzy** ma długość mieszczącą się w przedziale 10–150 znaków; tylko pojedyncze wpisy są **krótsze lub dłuższe**.

Rozkład liczby słów w komentarzach (surowe dane)



Rozkład liczby słów w komentarzach (surowe dane)

Na poniższym wykresie zaprezentowano histogram liczby słów w komentarzach. Oś X przedstawia liczbę słów w komentarzu, natomiast na osi Y widoczna jest liczba komentarzy przypadających na daną liczbę słów.

Główne obserwacje:

- **Najwięcej komentarzy** zawiera około **5–8 słów**. Jest to szczyt rozkładu.
- Rozkład opada stopniowo wraz ze wzrostem liczby słów – stosunkowo niewiele komentarzy liczy ponad 15 słów.
- Bardzo krótkie komentarze (np. 1–2 słowa) stanowią niewielki odsetek zbioru.
- Komentarze wyjątkowo długie (powyżej 20–25 słów) również występują rzadko i tworzą długi ogon po prawej stronie histogramu.

2. Podział na zbiór treningowy i testowy

Aby sprawdzić zdolność modelu do uogólniania wiedzy, dzielimy dane na:

- **Zbiór treningowy (80%)** – na którym model będzie się uczyć.
- **Zbiór testowy (20%)** – na którym ocenimy, jak model radzi sobie z nowymi, niewidzianymi wcześniej danymi.

Dlaczego dzielimy zbiór?

Dzięki temu możemy porównać wyniki na danych, na których model był trenowany, z wynikami na danych testowych. Jeśli model osiąga **bardzo wysokie** wyniki na treningu, ale **słabe** na testowym, wskazuje to na problem **overfittingu (przetrenowania)**, czyli sytuację, gdy model "zapamiętuje" dane treningowe, zamiast **uogólniać** wzorce. Z kolei, gdy model osiąga **niższe** wyniki na danych treningowych niż na testowych, mamy do czynienia z **underfittingiem (niedotrenowaniem)**. Wskazuje to, że model jest **zbyt prosty**, aby uchwycić **istotne wzorce** w danych. **Underfitting** wskazuje na to, że nie dobraliśmy **optymalnych hiperparametrów** modelu. Aby **poprawić** zdolność modelu do generalizacji, warto przeprowadzić **strojenie hiperparametrów**, na przykład za pomocą Grid Search lub Randomized Search

3. Czyszczenie i przygotowanie danych

Opis:

Cel:

Ujednolicenie i oczyszczenie tekstu w zbiorach danych przed dalszym przetwarzaniem. Proces ten obejmuje:

- Usuwanie specyficznych tokenów (np. "RT", wzmianki typu "@anonymized_account") oraz emotikonów, które mogą zakłócać analizę.
- Konwersję tekstu do małych liter, co pozwala na traktowanie różnych form tego samego słowa jednolicie.
- Usuwanie interpunkcji, cyfr oraz zbędnych białych znaków, aby ograniczyć szum w danych.
- Eliminowanie wybranych stop words charakterystycznych dla języka polskiego (np. **i, oraz, a, czy, to**), dzięki czemu pozostają jedynie kluczowe terminy użyte w komentarzach.

Korzyści:

- **Jednolitość tekstu:**

Dzięki konwersji do małych liter i usunięciu zbędnych znaków, ten sam wyraz w różnych formach jest traktowany identycznie.

- **Redukcja szumu:**

Usuwanie specjalnych tokenów, interpunkcji, cyfr oraz stop words zmniejsza szum, co przekłada się na lepszą wydajność modelu podczas uczenia.

- **Poprawa jakości danych:**

Eliminacja pustych rekordów oraz uaktualnienie statystyk takich jak liczba słów i długość tekstu pozwala na lepsze zrozumienie i kontrolę nad danymi przed ich dalszym przetwarzaniem.

Dzięki powyższym krokom przygotowanie danych zapewnia, że model nauczy się analizować czyste, uporządkowane i bardziej reprezentatywne cechy tekstu.

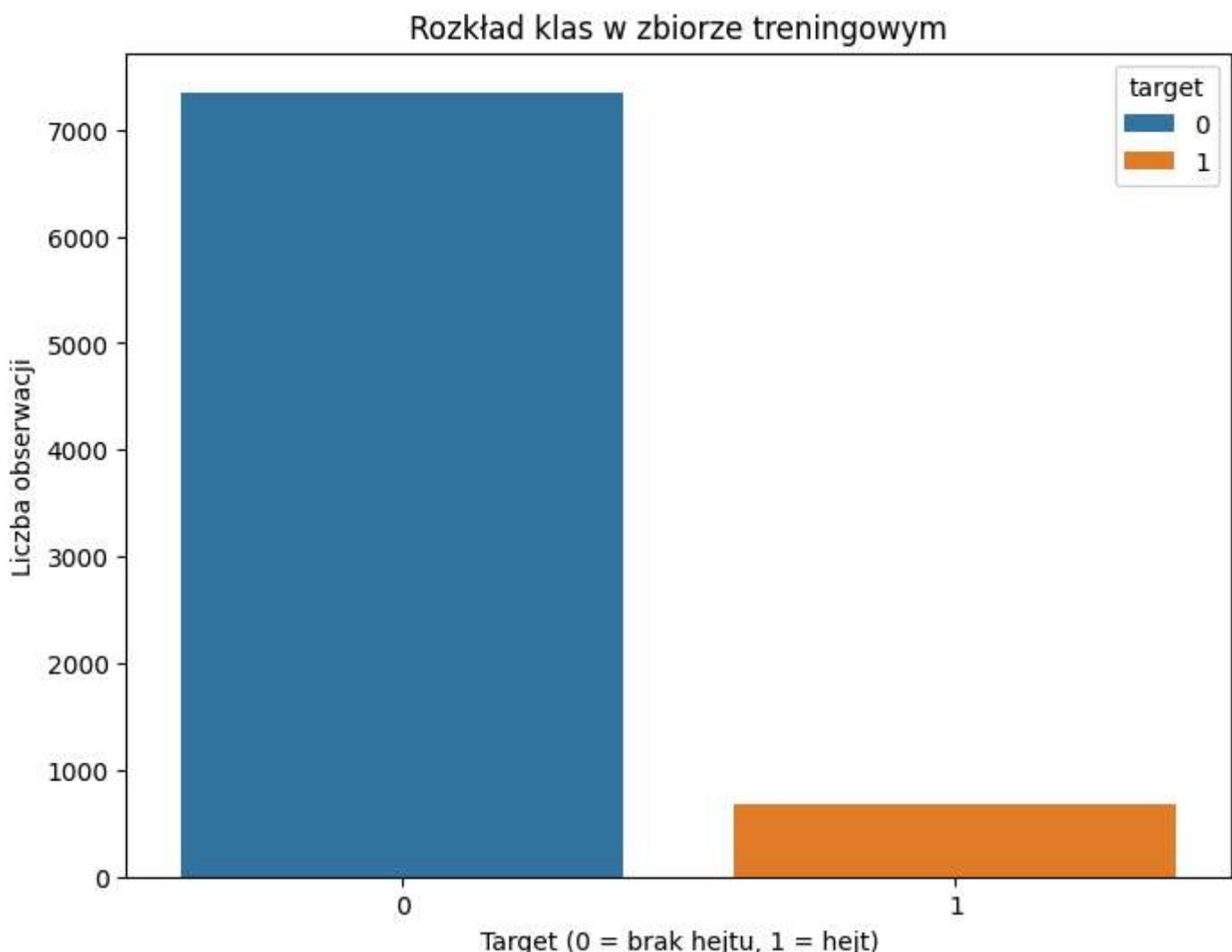
4. Lematyzacja

Co to jest lematyzacja?

Lematyzacja to proces przekształcania różnych form gramatycznych danego wyrazu do jego podstawowej (lematu). Przykładowo, formy takie jak "**biegnie**", "**biegł**", "**biegając**" są sprowadzane do jednego lematu "**biegać**". Dzięki temu model nie traktuje różnych odmian tego samego słowa jako oddzielnych cech, co prowadzi do bardziej spójnej i efektywnej reprezentacji tekstu.

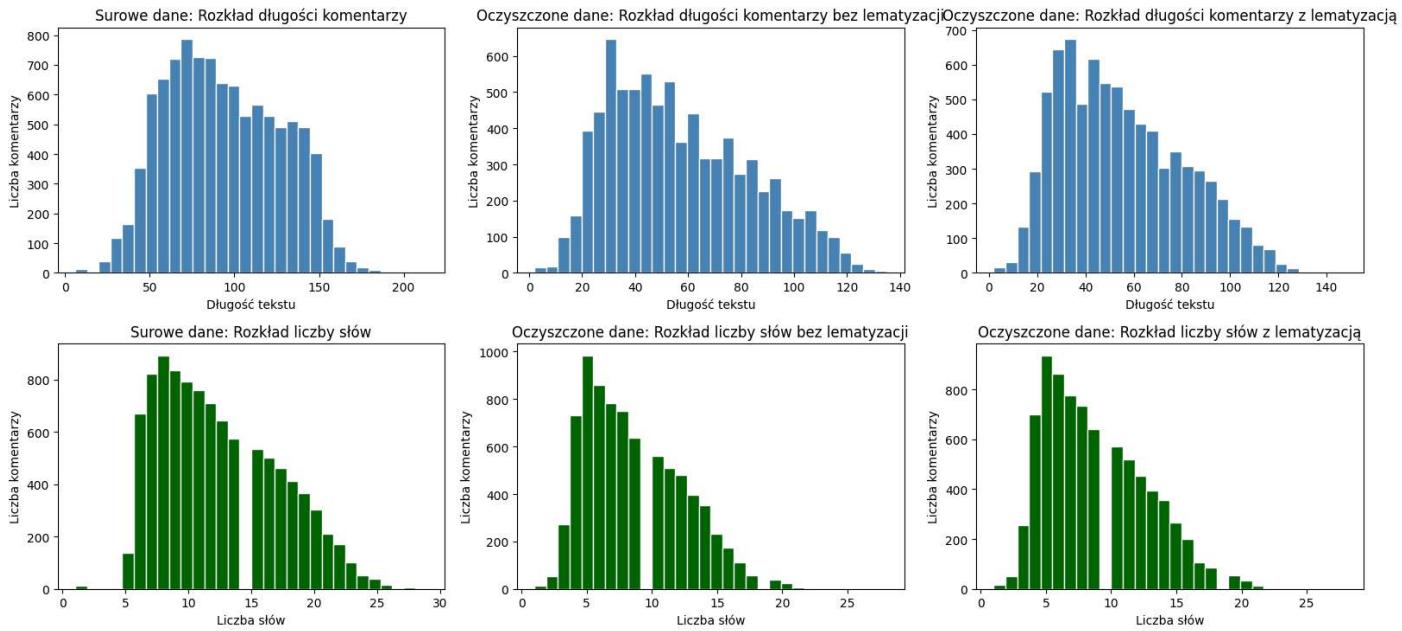
Dlaczego warto stosować lematyzację?

- **Ujednolicenie danych:** Wszystkie odmiany jednego wyrazu są łączone, co zmniejsza liczbę unikalnych słów (cech) w słowniku.
- **Redukcja szumu:** Eliminacja zbędnych wariantów słów pomaga modelowi skupić się na istotnych informacjach, zamiast na przypadkowych różnicach morfologicznych.
- **Poprawa jakości modelu:** Ujednolicona reprezentacja tekstu może przyczynić się do lepszych wyników klasyfikacyjnych, ponieważ model uczy się na bardziej spójnych danych.



Analiza efektu oczyszczania komentarzy

- Oczyszczenie komentarzy spowodowało **zmniejszenie ogólnej liczby komentarzy**.
- Po wstępnej obróbce danych i po redukcji, **dysproporcja między klasami** pozostała na podobnym poziomie.



Porównanie histogramów: Surowe vs Oczyszczone dane

Powyższy zestaw sześciu histogramów przedstawia różnice w długościach komentarzy (mierzonej liczbą znaków) oraz liczbie słów pomiędzy danymi **surowymi** a danymi **oczyszczonymi** (z lematyzacją lub bez niej). Histogramy zostały ułożone w 2 wierszach × 3 kolumnach:

1. Góry rząd:

- **(Lewa kolumna) Surowe dane:** Rozkład długości komentarzy w nieprzetworzonym zbiorze (wartości osi X zwykle większe, występują spacje, emotikony, interpunkcja).
- **(Środkowa kolumna) Oczyszczone dane:** Rozkład długości komentarzy zmniejszył się w zbiorze po usunięciu znaków specjalnych, interpunkcji i słów nieistotnych (stop words), ale **przed lematyzacją**.
- **(Prawa kolumna) Oczyszczone dane z lematyzacją:** Rozkład długości komentarzy w zbiorze, w którym dodatkowo zastosowano lematyzację, co skraca jeszcze bardziej średnią długość tekstu.

2. Dolny rząd:

- **(Lewa kolumna) Surowe dane:** Histogram liczby słów w surowym zbiorze. Ze względu na brak czyszczenia tekstu, niektóre „słowa” mogły być liczone wielokrotnie lub pojawiać się spacje i znaki spoza słownika.
- **(Środkowa kolumna) Oczyszczone dane:** Histogram liczby słów w komentarzach po usunięciu elementów nieistotnych, lecz **bez lematyzacji**. Widać zazwyczaj niższe wartości w porównaniu z surowymi danymi (zniknęły duplikaty spowodowane interpunkcją czy emotikonami).
- **(Prawa kolumna) Oczyszczone dane z lematyzacją:** Rozkład liczby słów w komentarzach po pełnej obróbce, łącznie z lematyzacją. Histogram jest przesunięty jeszcze bardziej w lewo (krótsze komentarze pod względem liczby tokenów).

Najważniejsze obserwacje:

- **Zauważalna różnica w długości:** Po usunięciu elementów takich jak interpunkcja i stop words, a następnie przeprowadzeniu lematyzacji, komentarze stają się krótsze (spada średnia długość tekstu).
- **Spadek liczby słów:** W przypadku oczyszczonych danych (z lematyzacją lub bez niej) histogram liczby słów przesuwa się w lewo. Liczba słów często się zmniejsza, ponieważ usuwane są wielokrotne wersje tego samego słowa w różnych formach lub słowa nieistotne.



Wordcloud – Surowe dane vs Oczyszczone dane bez lematyzacji vs Oczyszczone dane z lematyzacją

Powyższa ilustracja przedstawia trzy chmury słów (Wordcloudy) wygenerowane z naszego zbioru komentarzy. Każda chmura prezentuje najczęściej występujące wyrazy w danych, przy czym ich rozmiar jest proporcjonalny do częstotliwości występowania. Możemy zaobserwować, jak zmienia się dominacja poszczególnych słów w zależności od etapu przetwarzania.

1. Wordcloud – Surowe dane (lewa część):

- Zawiera nieprzetworzone komentarze, w tym wzmianki, słowa stop, interpunkcję, emotikony itp.
- Często widzimy słowa „anonymized_account” (przykład usuwanych wzmiankowanych kont) lub inne artefakty, które mogą być powtarzane wielokrotnie w surowym zbiorze.
- W efekcie chmura może być zdominowana przez mniej istotne terminy, nieodzwierciedlające faktycznej treści komentarzy.

2. Wordcloud – Oczyszczone dane bez lematyzacji (środkowa część):

- Usunięto zbędne elementy (interpunkcję, niepotrzebne tokeny typu „RT” czy wzmianki).
- Usunięto również stop words (słowa najczęściej występujące i nienoszące znaczenia), dzięki czemu widoczne stają się bardziej kluczowe terminy (np. „nie”, „być”, „tak”).
- **Brak lematyzacji** oznacza, że różne formy wyrazów (np. „mówić”, „mówilem”, „mówili”) nadal występują osobno.

3. Wordcloud – Oczyszczone dane z lematyzacją (prawa część):

- Dodatkowo przeprowadzono lematyzację, czyli sprowadzanie słów do ich formy podstawowej (np. „mówić” zamiast „mówilem”).
- Dzięki temu słowa takie jak „być” czy „mieć” pojawiają się w ich formie lematycznej, co pomaga w ujednoliceniu wariantów językowych.
- Ostatecznie w chmurze dominuje mniejsza liczba unikalnych słów, ale lepiej ukazują one najistotniejsze pojęcia i tematy w zbiorze.

Kluczowe wnioski:

- **Usuwanie niepotrzebnych elementów** (interpunkcji, wzmiankowanych kont, emotikonów) znaczco oczyszcza dane i pozwala skupić się na właściwych słowach.
- **Eliminacja stop words oraz lematyzacja** zmniejszają liczbę wyrazów, co pozwala na wyraźniejsze ujawnienie słów kluczowych i tematów.
- Dzięki takim chmurom słów możemy szybko porównać etapy przetwarzania tekstu i ocenić, że **oczyszczone dane z lematyzacją** umożliwiły ujawnienie **faktycznie ważnych treści** w komentarzach.

5. Wektoryzacja tekstu (TF-IDF) i przygotowanie danych do modelowania

Co to w ogóle znaczy "wektoryzacja tekstu"?

Komputer nie rozumie słów, tak jak człowiek – dla niego tekst to tylko ciąg liter.

Żeby nauczyć model maszynowy rozróżniać komentarze, musimy najpierw **zamienić tekst na liczby**, które można policzyć.

To właśnie robi **wektoryzacja**: zmienia każde zdanie (komentarz) w **wektor liczb**, który opisuje, jakie słowa tam występują i jak bardzo są one istotne.

Krok po kroku: Jak działa TF-IDF?

1. Tokenizacja

Czyli rozdzielamy tekst na słowa.

Przykład: Kocham psy, ale nie lubię kotów" → ["kocham", "psy", "ale", "nie", "lubię", "kotów"]

2. Budowa słownika

Tworzymy listę wszystkich unikalnych słów we wszystkich komentarzach.

Przykład słownika: ["kocham", "psy", "kotów", "nienawiść", "super", ...]

3. Macierz dokument-term

Dla każdego komentarza zapisujemy, które słowa z tego słownika w nim występują.

Przykład (1 = słowo występuje, 0 = nie występuje):

komentarz	kocham	psy	kotów	nienawiść
tekst 1	1	1	1	0
tekst 2	0	0	0	1

TF-IDF (Term Frequency - Inverse Document Frequency)

TF-IDF to technika przekształcania tekstu w reprezentację numeryczną, która pozwala określić wagę poszczególnych słów w dokumencie, uwzględniając, jak często występują w jednym dokumencie (Term Frequency, TF) oraz jak rzadkie są w całym korpusie (Inverse Document Frequency, IDF).

Jak to działa?

1. Term Frequency (TF):

- **Definicja:** Liczba wystąpień danego słowa w dokumencie, zwykle dzielona przez łączną liczbę słów w tym dokumencie.
- **Cel:** Aby znormalizować częstotliwość – dokumenty o różnej długości mają porównywalne miary.

$$TF(t, d) = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

$n_{t,d}$ = liczba wystąpień termu t w dokumencie d ,

$\sum_k n_{k,d}$ = łączna liczba wszystkich termów w dokumencie d .

2. Inverse Document Frequency (IDF):

- **Definicja:** Miara, która obliczana jest jako logarytm ilorazu liczby wszystkich dokumentów przez liczbę dokumentów zawierających dane słowo.
- **Cel:** Aby nadać mniejszą wagę słowom, które pojawiają się bardzo często w całym zbiorze (np. "i", "oraz", "to"). Jeśli słowo występuje we wszystkich dokumentach, jego wartość IDF jest bardzo niska (lub bliska 0).

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

$|D|$ = liczba dokumentów w korpusie D ,
 $|\{d \in D : t \in d\}|$ = liczba dokumentów zawierających term t .

3. TF-IDF:

- **Interpretacja:**

- Słowa, które są charakterystyczne dla danego dokumentu (wysokie TF) i jednocześnie rzadko występują w innych dokumentach (wysokie IDF), dostają wysoką wagę.
- Słowa występujące powszechnie mają niską wartość IDF, a więc nawet jeśli mają wysoką TF, ich wynikowy TF-IDF jest niski.

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

$\text{TF}(t, d)$ = względna częstość termu t w dokumencie d ,

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}.$$

Dlaczego TF-IDF jest ważny?

W przeciwieństwie do podejścia "bag-of-words", które jedynie sprawdza, czy słowo występuje w dokumencie (lub liczy jego wystąpienia), TF-IDF uzupełnia informację o kontekście globalnym. Dzięki temu model lepiej rozumie, które słowa są istotne (np. obrazliwe lub unikalne terminy w komentarzach hejtowych), a które są tzw. stop words, czyli słowami powszechnie występującymi, ale mało informacyjnymi.

Prawidłowe użycie TF-IDF

Aby zapobiec wyciekowi informacji („data leakage”) z danych testowych do procesu trenowania, wektoryzator TF-IDF jest **dopasowywany** (`fit`) wyłącznie na danych treningowych, a następnie na tych samych parametrach wykonujemy jedynie **transformację** danych testowych.

Dopasowanie i wektoryzacja na zbiorze treningowym

X_train = vectorizer.fit_transform(train_data)

Metoda `fit_transform` oblicza statystyki TF-IDF (IDF) **wyłącznie** na danych treningowych i od razu zwraca macierz TF-IDF dla tych danych.

Dzięki temu **żadne informacje** z zestawu testowego **nie „przeciekają”** podczas uczenia IDF.

Tylko transformacja na zbiorze testowym

X_test = vectorizer.transform(test_data)

Metoda `transform` wykorzystuje już wyliczone na **etapie treningu** ****wartości IDF i nie pozwala, by testowe przykłady **wpłynęły** na te wagi.

Zapobiega to tzw. „**train-test contamination**” (**zanieczyszczeniu testu**) i sztucznie zawyżonym wynikom modelu.

6. Język i Morfologia

Ablacja i Eksperymenty

Ablation study to metoda eksperimentalna, polegająca na porównaniu różnych modeli z różnymi elementami. W kontekście lematyzacji oznacza to:

- **Porównanie podejścia z lematyzacją vs. bez lematyzacji:**

Dzięki temu możesz sprawdzić, czy dodatkowy krok przygotowania danych (lemmatyzacja) naprawdę wpływa pozytywnie na wyniki modelu.

- Jeżeli model pracujący na lematyzowanych danych osiąga lepsze metryki (np. wyższy F1 score) – oznacza, że lematyzacja pomaga ujednolicić dane i poprawić jakość modelu.
- Jeżeli wyniki są podobne lub model bez lematyzacji radzi sobie lepiej – może się okazać, że lematyzacja nie jest niezbędna lub nawet wprowadza pewien szum.

Czas i Zasoby

- **Czas przetwarzania:**

Lematyzacja, szczególnie dla dużych zbiorów danych, może wydłużyć czas przygotowania danych.

- **Koszty obliczeniowe:**

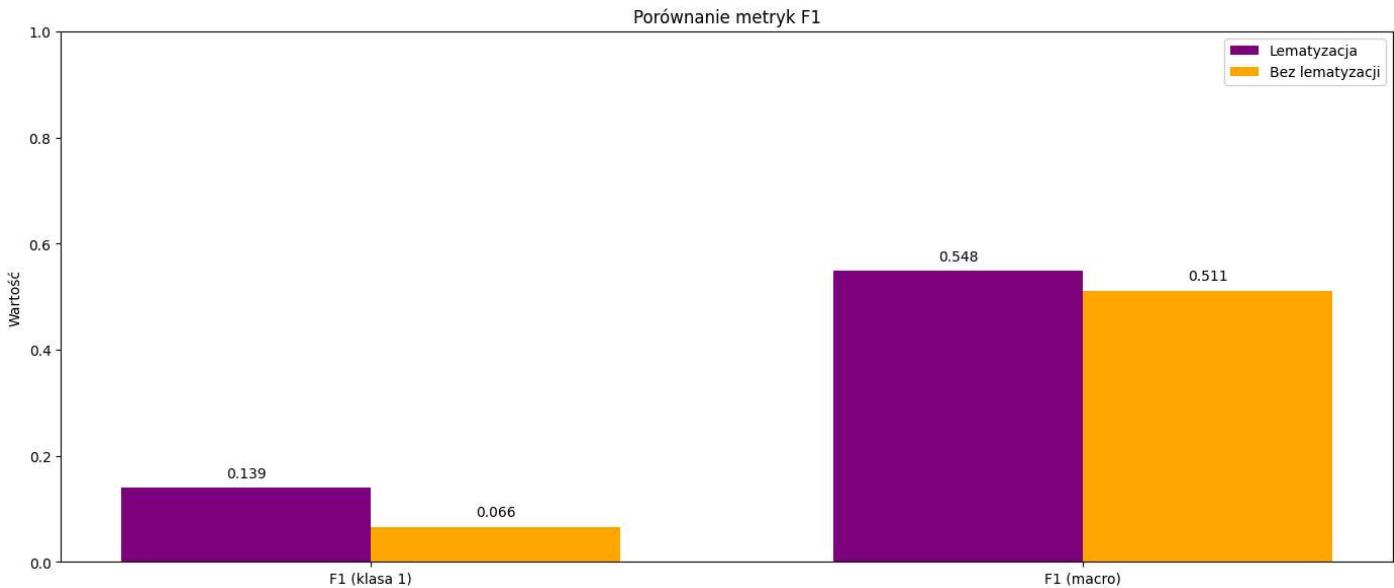
Jeżeli efekty lematyzacji nie są znaczco lepsze, można rozważyć pominięcie tego kroku, co uprości cały pipeline i zaoszczędzi zasoby.

Podsumowanie

Zbudowanie modelu na danych z lematyzacją może:

- **Poprawić jakość reprezentacji** tekstu, poprzez ujednolicenie odmian słów,
- **Zredukować wymiarowość** zbioru cech, co często przekłada się na lepsze wyniki modelu.

Jeżeli eksperymenty pokażą, że model z lematyzacją osiąga wyraźnie lepsze wyniki, warto zachować ten krok. W przeciwnym razie, dla uproszczenia i oszczędności zasobów, można rozważyć pominięcie lematyzacji. Takie podejście zapewnia, że decyzje są oparte na danych oraz faktycznych metrykach, a nie tylko na domniemaniach.



Wykres słupkowy (Bar Chart) – Porównanie F1 dla klasy 1 oraz Macro F1

- **Opis:**

Pierwsze dwa słupki po lewej stronie pokazują słupki reprezentujące F1-score (dla klasy 1), a dwa słupki po prawej stronie reprezentują F1-score (w ujęciu macro) dla dwóch wersji danych:

- **Lematyzacja** (fioletowy słupek)
- **Bez Lematyzacji** (pomarańczowy słupek)

- **Wnioski:**

Możemy łatwo dostrzec, że lematyzacja dla klasy 1 znacznie podniosła poziom F1-score, ale już dla metryki F1-score(macro) nie ma tak dużej różnicy.

Jak interpretować Macierz Pomyłek:

- **True Positives (TP):** Przypadki, gdzie model poprawnie przewidział klasę pozytywną.
- **True Negatives (TN):** Przypadki, gdzie model poprawnie przewidział klasę negatywną.
- **False Positives (FP):** Przypadki, gdzie model błędnie przypisał pozytywną etykietę (błąd typu I).
- **False Negatives (FN):** Przypadki, gdzie model błędnie przypisał negatywną etykietę (błąd typu II).
- **Metryki wyliczane z macierzy:**

Na podstawie tej macierzy można obliczyć między innymi:

- **Accuracy:**

$$\boxed{\frac{TP + TN}{TP + TN + FP + FN}}$$

Procent poprawnych klasyfikacji w całym zbiorze.

- **Precision:**

$$\boxed{\frac{TP}{TP + FP}}$$

Jaki odsetek przykładów zaklasyfikowanych jako „hejt” rzeczywiście nim jest.

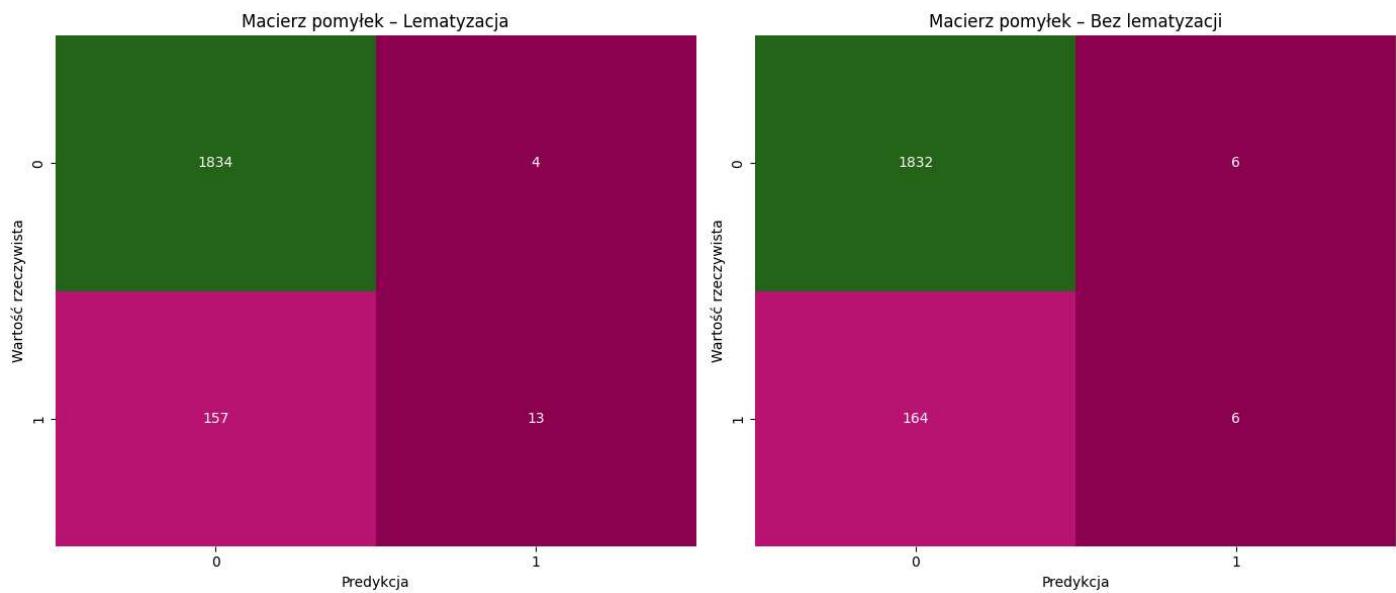
- **Recall:**

$$\boxed{\frac{TP}{TP + FN}}$$

Jaki odsetek rzeczywistego hejtu został poprawnie wykryty przez model.

- **F1-score:** Średnia harmoniczna Precision i Recall.

$$\boxed{F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}}$$



Porównanie macierzy pomyłek:

Na powyższych dwóch wykresach znajdują się macierze pomyłek (Confusion Matrix) dla każdego wariantu danych:

- **Lewa macierz (Lematyzacja):** Pokazuje, jak model radzi sobie na danych poddanych lematyzacji z prawidłowym dopasowaniem (True Positives i True Negatives) oraz błędny (False Positives i False Negatives) klasyfikowaniem próbek.
- **Prawa macierz (Bez Lematyzacji):** Analogicznie dla modelu trenowanego na danych niepodanych lematyzacji.
- **Wnioski:**
Dzięki tym macierzom możemy stwierdzić, że model z lematyzacją trochę lepiej rozpoznaje klasy, więc dalej utwierdzamy się w przekonaniu, że model z lematyzacją jest lepszym wyborem w naszym problemie.

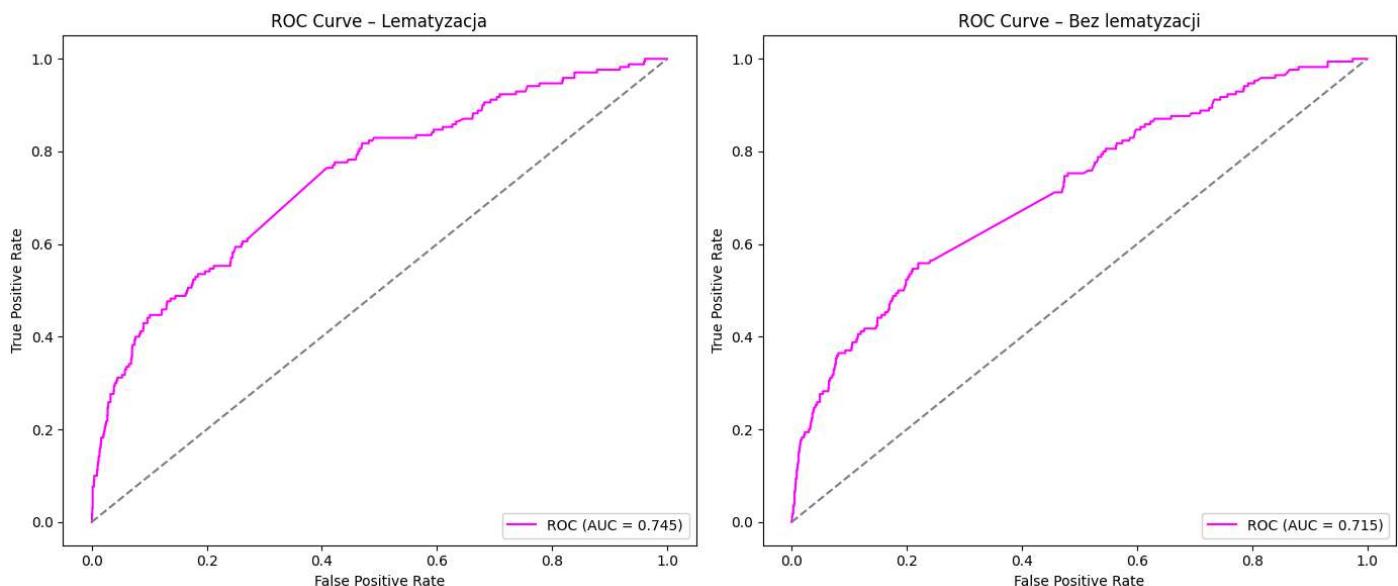
Krzywa ROC (Receiver Operating Characteristic)

- **Co to jest:**

Krzywa ROC pokazuje, jak zmieniają się wskaźniki **True Positive Rate (TPR)** i **False Positive Rate (FPR)** przy różnych progach decyzyjnych.

- **Jak interpretować:**

- **TPR (czułość/recall):** Odsetek rzeczywistych pozytywów poprawnie wykrytych przez model.
- **FPR:** Odsetek negatywnych przypadków błędnie sklasyfikowanych jako pozytywne.
- **Idealny model:** Krzywa przebiega bardzo blisko lewej górnej krawędzi, co oznacza wysoki TPR przy niskim FPR.
- **AUC (Area Under the Curve):** Podsumowuje wydajność modelu – wartość bliska 1 wskazuje na bardzo dobry model, natomiast wartość 0.5 sugeruje, że model nie lepiej radzi sobie niż losowe zgadywanie.



Prówanie krzywych:

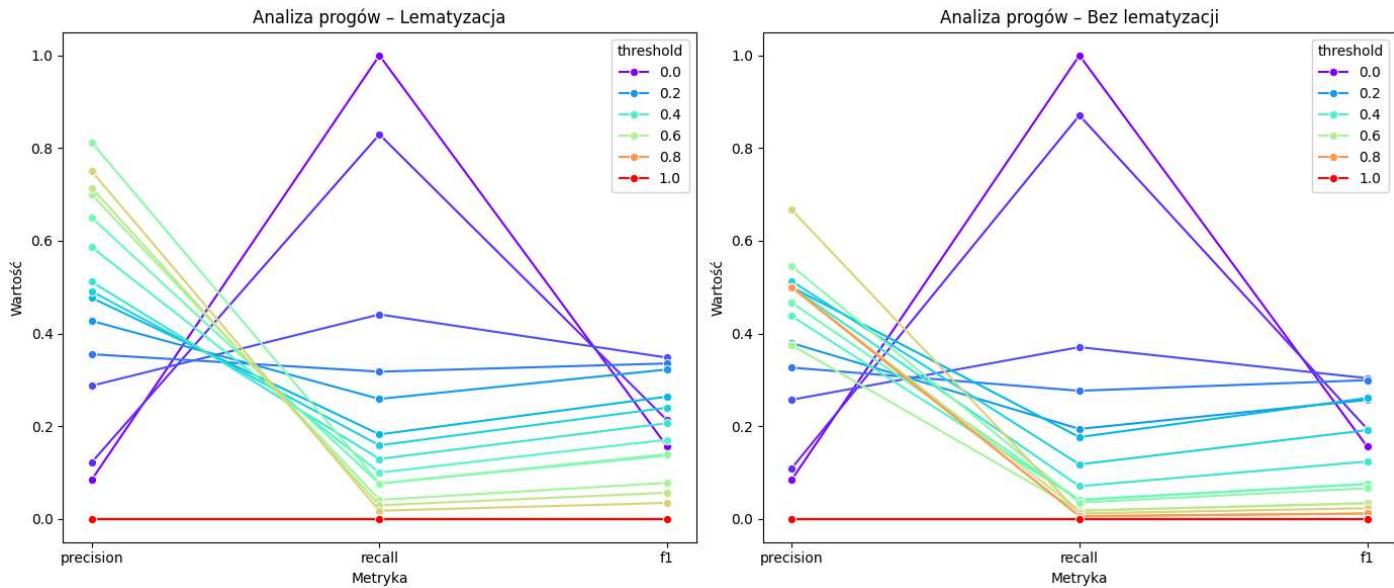
Widzimy dwie krzywe ROC – jedną dla **Lematyzacji**, drugą dla **Bez Lematyzacji**. Krzywa ROC obrazuje zależność między **True Positive Rate (TPR)** a **False Positive Rate (FPR)** przy różnych progach decyzyjnych. Linie diagonalne (szare, przerywane) reprezentują klasyfikację losową ($AUC = 0.5$).

- **Metryka AUC:**

Każdy wykres ma podaną wartość AUC (Area Under the Curve). Im wyższa wartość, tym lepsza zdolność modelu do odróżniania klas pozytywnych i negatywnych na różnych progach.

- **Wnioski:**

Krzywa dla wersji z lematyzacją leży wyżej, oznacza to, że model lepiej rozróżnia klasy.



Powyższy wykres przedstawia wpływ zmiany progu decyzyjnego (threshold) na metryki klasyfikacji: **precision, recall i f1-score**.

- Każdy kolor odpowiada innemu progu (threshold) przy klasyfikacji binarnej.
- Im wyższy próg (np. 0.8, 1.0), tym bardziej model „ostrożny” w przypisywaniu pozytywnej klasy.
- Im niższy próg (np. 0.0, 0.2), tym bardziej „liberalny” – łatwiej przypisuje klasę pozytywną.

Interpretacja wykresu

Lewa strona – z lematyzacją

Wpływ lematyzacji: sprowadzenie słów do formy podstawowej zmniejsza liczbę rzadkich wariantów, co z reguły poprawia precision kosztem recall. W rezultacie przy niższych progach (0.1–0.3) recall jest niższy niż bez lematyzacji, ale precision jest wyższe.

Podsumowanie: przy lematyzacji model staje się mniej „agresywny” w oznaczaniu komentarzy jako hejt (mniej fałszywych alarmów), ale wymaga nieco niższych progów, by utrzymać przyzwoity recall.

Prawa strona – bez lematyzacji

Recall przy niższych progach jest wyższy niż z lematyzacją, ale precision spada.

Maksimum F1 osiągane jest przy niższych progach (ok. 0.3), co świadczy o bardziej „agresywnym” wykrywaniu klasy pozytywnej.

Wnioski

- Lematyzacja pomaga w balansowaniu między **precision** a **recall** – widać wyższy **f1-score** przy środkowych progach.
- Wybór **odpowiedniego progu decyzyjnego** ma ogromny wpływ na jakość modelu – domyślny próg 0.5 niekoniecznie jest optymalny.

Podsumowanie

1. **Porównanie macro-F1 (Bar Chart)** ukazuje ogólną jakość klasyfikacji w obu wariantach.
2. **Macierze pomyłek** pomagają dostrzec różnice w charakterze błędów (FP, FN) po lematyzacji i bez niej.
3. **Krzywe ROC** wskazują, na ile dobrze model rozróżnia klasy przy różnych progach (poprzez wartość AUC).
4. **Wykresy metryk vs. próg decyzyjny** umożliwiają wybór odpowiedniego progu w zależności od wymaganego kompromisu między precision i recall

W związku z powyższym, ze względu na lepszą wydajność, wyższą jakość klasyfikacji oraz korzystniejszy charakter błędów, postanawiam używać danych, które zostały poddane lematyzacji.

7. Opis warunków uczenia modeli

Celem tego opisu jest przedstawienie kluczowych warunków uczenia modeli oraz podsumowanie specyficznych ustawień, które mogą mieć wpływ na wydajność i wyniki. Dzięki temu można szybko porównać różne podejścia (np. LightGBM, XGBoost, Random Forest, SVM, Logistic Regression) pod kątem ich parametrów, specyfiki treningu i uwag praktycznych.

Opis modeli

Poniższa tabela przedstawia przykładowy podział opisów warunków uczenia dla różnych modeli:

Model	Parametry / Specyfika	Wskazówki / Uwagi
LightGBM	- Liczba drzew - Maksymalna głębokość - Learning rate - Regularizacja	Szybki trening, bardzo dobra skalowalność na dużych zbiorach danych; może wymagać dostrojenia hiperparametrów
XGBoost	- Liczba iteracji - Maksymalna głębokość - Learning rate - Subsampling	Wysoka precyza; dobrze radzi sobie z danymi o złożonych zależnościach, ale jest bardziej podatny na overfitting
Random Forest	- Liczba drzew - Maksymalna głębokość - Metoda losowania próbek - Losowość próbkowania	Stabilny model; dobrze radzi sobie z dużą liczbą cech; stosunkowo niewrażliwy na overfitting
Support Vector Machines (SVM)	- Rodzaj jądra (kernel) - Parametr C - Gamma	Skuteczny przy separacji nielinowej; wymaga standaryzacji cech; optymalizacja parametrów może być czasochłonna
Logistic Regression (LR)	- Rodzaj regularizacji (L1/L2) - Parametr C - Liczba iteracji	Prosty model, szybki trening; interpretacja wyników jest przejrzysta dzięki liniowej zależności, idealny przy dużych zbiorach

8. Benchmarking modeli

W ramach tego etapu przeprowadzamy **benchmarking modeli**, czyli systematyczne porównanie wyników różnych algorytmów klasyfikacyjnych trenowanych na tych samych danych wejściowych. Celem benchmarkingu jest ocena, który z modeli osiąga najlepsze wyniki w zadaniu klasyfikacji komentarzy jako "hejt" (1) lub treści neutralne (0).

Główne cele benchmarkingu:

1. Porównanie różnych algorytmów:

Testujemy wszystkie wyżej przedstawione modele. Każdy z nich uczy się na tych samych danych, co pozwala porównać ich skuteczność i dopasowanie.

2. Analiza metryk:

Dla każdego modelu obliczamy kluczowe metryki, takie jak:

- **Accuracy:** Procent poprawnych przewidywań.
- **Precision i Recall:** Ocena, jak dobrze model identyfikuje przypadki pozytywne.
- **F1 Score:** Średnia harmoniczna precyzji i recall, która jest szczególnie przydatna przy niezbalansowanych zbiorach danych.
- **ROC Curve oraz AUC:** Wizualizacja i ocena zdolności modelu do rozróżniania klas przy różnych progach decyzyjnych.

Dlaczego benchmarking jest kluczowy?

• Obiektywna ocena:

Porównanie wyników na tym samym zbiorze danych treningowych i testowych pozwala obiektywnie ocenić, który model lepiej uogólnia wiedzę, a który jest bardziej podatny na overfitting.

• Wybór najlepszego podejścia:

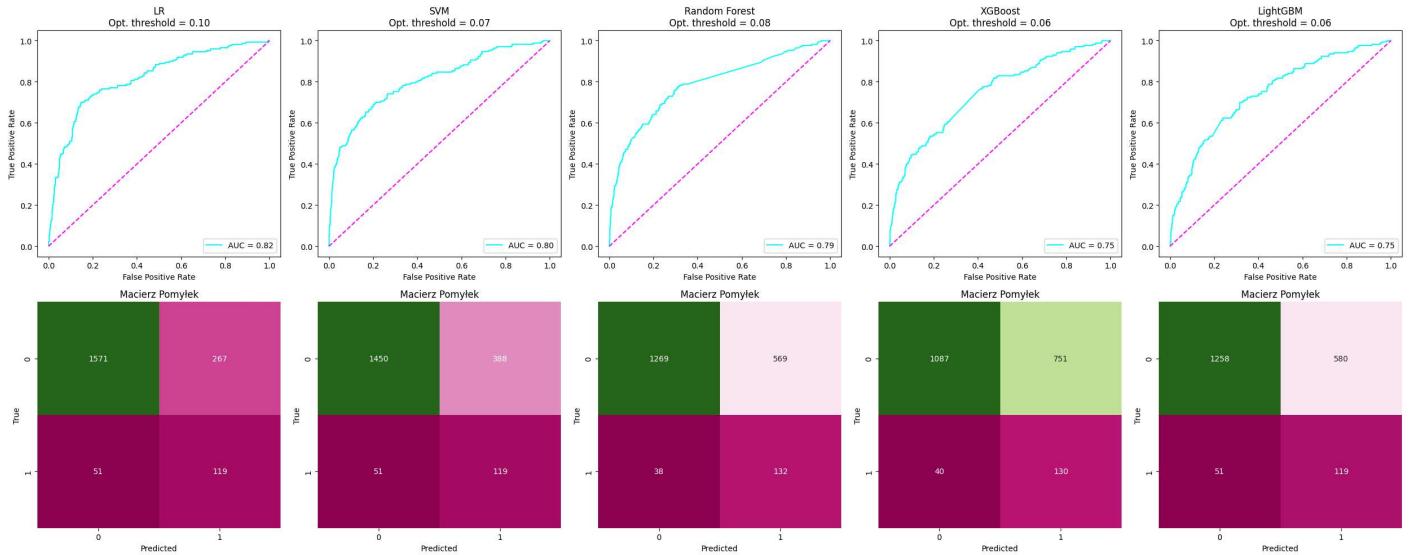
Benchmarking umożliwia wybór optymalnego modelu i dalszą konfigurację hiperparametrów. Porównanie metryk, takich jak F1 Score zarówno dla danych treningowych, jak i testowych, pozwala ocenić, czy model nie jest przetrenowany i czy dobrze uogólnia wiedzę na nowych danych.

Podsumowanie

Benchmarking modeli to kompleksowe podejście polegające na:

- Trenowaniu wielu modeli na tych samych danych.
- Obliczaniu i porównywaniu kluczowych metryk.
- Wybieraniu najlepszego modelu na podstawie wyników uzyskanych na zbiorze testowym.

Dzięki tej metodzie możemy rzetelnie ocenić, które rozwiązanie najlepiej odpowiada potrzebom zadania klasyfikacji komentarzy i zapewnia optymalną skuteczność oraz generalizację.



Porównanie metryk modeli – Benchmarking

Wzór na optymalny próg (indeks Youdena), który został użyty do macierzy pomyłek

$$t^* = \arg \max_t (TPR(t) - FPR(t))$$

- t — dowolny próg decyzyjny stosowany do przekształcenia prawdopodobieństwa ($P(y = 1 | x)$) na etykietę klasy (0 lub 1).
- t^* — wartość progu (t), która maksymalizuje różnicę ($TPR(t) - FPR(t)$); czyli tzw. optymalny próg według indeksu Youdena.

$$TPR(t) = \frac{TP(t)}{TP(t) + FN(t)}$$

- $TPR(t)$ (True Positive Rate) — czułość (sensitivity), czyli odsetek poprawnie wykrytych pozytywów przy progu (t):

$$FPR(t) = \frac{FP(t)}{FP(t) + TN(t)}$$

- $FPR(t)$ (False Positive Rate) — współczynnik fałszywych alarmów, czyli odsetek negatywów błędnie oznaczonych jako pozytywy przy progu (t):

Podsumowanie modeli

Poniższa tabela przedstawia metryki dla różnych modeli klasyfikacyjnych uzyskane na zbiorze treningowym oraz testowym.

Model	Train Accuracy	Train F1 (macro)	Test Accuracy	Test F1 (macro)
Logistic Regression	92.0%	55.6%	91.6%	54.9%
SVM	92.4%	57.8%	91.5%	48.3%
Random Forest	75.2%	61.6%	69.8%	55.5%
XGBoost	59.3%	49.8%	60.6%	49.0%
LightGBM	70.5%	58.5%	68.6%	53.7%

Wnioski i wybór najlepszego modelu:

- **Logistic Regression:**
 - Na zbiorze treningowym: F1 Score (macro-average) ~0.556
 - Na zbiorze testowym: F1 Score (macro-average) ~0.549
- **SVM:**
 - Model osiąga na zbiorze treningowym (F1 ~0.578), ale na zbiorze testowym spada do ~0.483.
 - Duża różnica wskazuje na overfitting.
- **Random Forest:**
 - Wyniki są umiarkowane: F1 (macro-average) ~0.616 na treningu oraz ~0.555 na teście.
- **XGBoost:**
 - Najniższe wyniki zarówno na zbiorze treningowym, jak i testowym (F1 ~0.498 na treningu, ~0.490 na teście).
- **LightGBM:**
 - Wyniki są podobne do Random Forest: F1 (macro-average) ~0.585 na treningu oraz ~0.537 na teście.

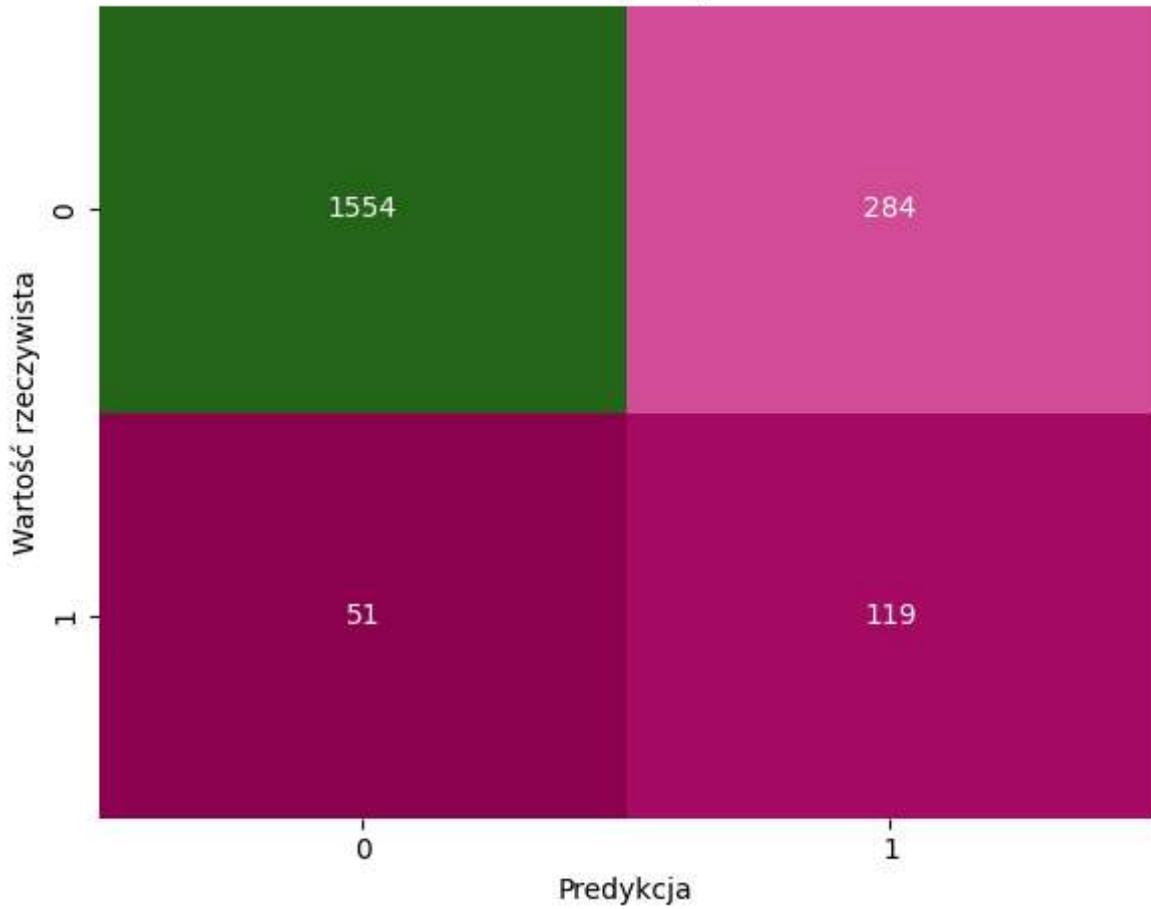
Podsumowanie wyboru: Na podstawie uzyskanych metryk, **model Logistic Regression** wykazuje najlepsze ogólne wyniki (prawie najwyższe F1 Score – 0.549 na zbiorze testowym), co sugeruje, że mimo pewnych niedociągnięć w klasyfikacji, porównując metryki w tym podejściu, uznaję że model ten generalizuje lepiej niż pozostałe podejścia.

Dlatego, spośród ocenianych modeli, **Logistic Regression** wydaje się być najlepszym wyborem, ponieważ:

- Utrzymuje wyższe uśrednione wyniki na danych testowych.
- Jest prosty, a zarazem skuteczny, co może mieć dodatkowe korzyści interpretacyjne i obliczeniowe.

W **dalszych obliczeniach** porównując możliwość dostrajania i dokładność modelu na tym etapie używa modelu **Logistic Regression**

Macierz pomyłek

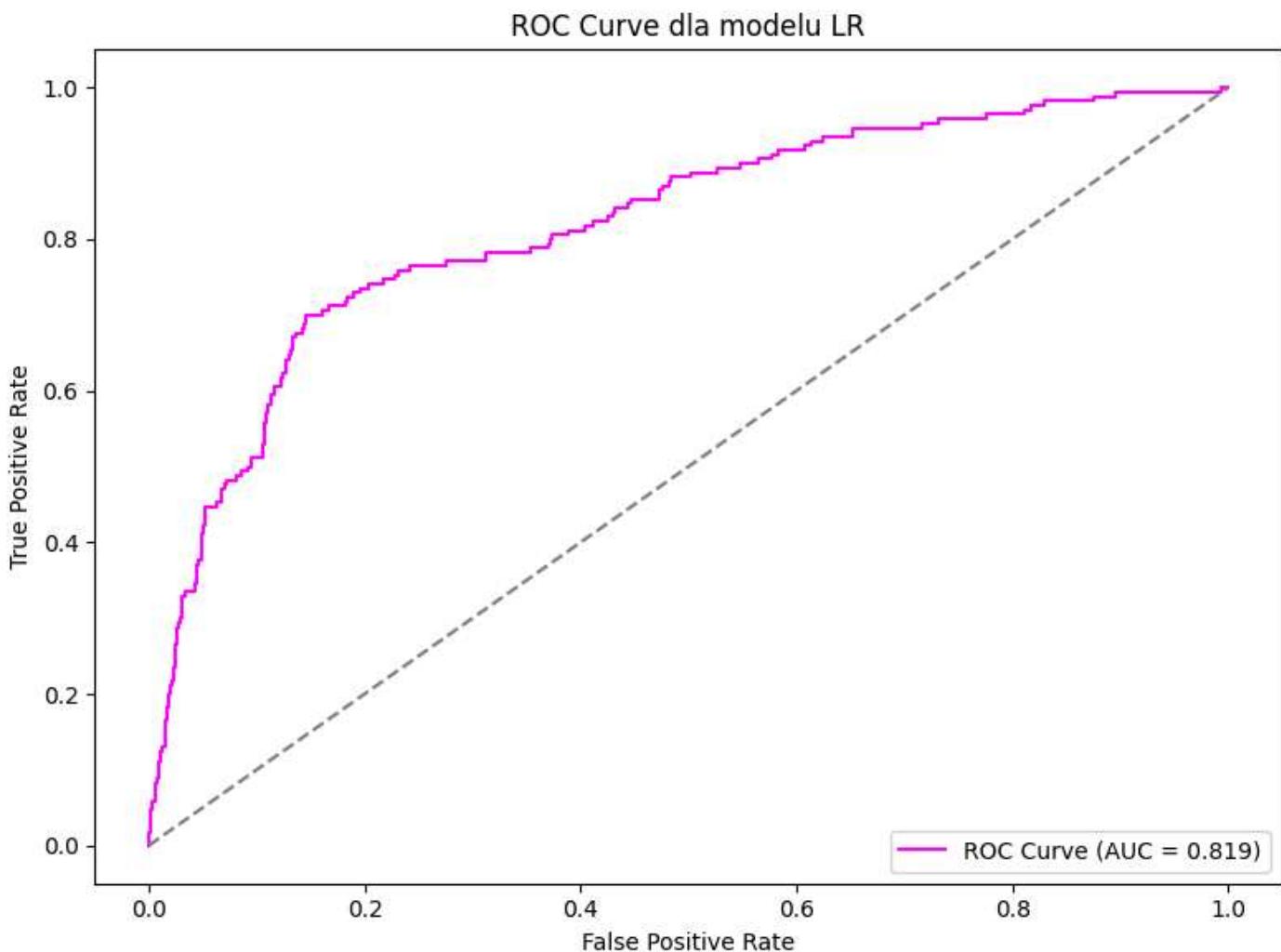


Wnioski z Macierzy Pomyłek

- **True Negatives (TN = 1554):** Model bardzo dobrze klasyfikuje komentarze jako "brak hejtu" (duża liczba poprawnych klasyfikacji negatywnych).
- **False Positives (FP = 284):** Istnieje spora liczba przypadków, w których model błędnie klasyfikuje komentarze "brak hejtu" jako "hejt".
- **False Negatives (FN = 51):** Model rzadko przeocza komentarze hejtowe, czyli niewiele przypadków, które rzeczywiście są hejtowe, zostało zaklasyfikowanych jako "brak hejtu".
- **True Positives (TP = 119):** Model poprawnie wykrywa komentarze hejtowe, choć liczba tych przypadków jest niższa niż FP.

Podsumowanie:

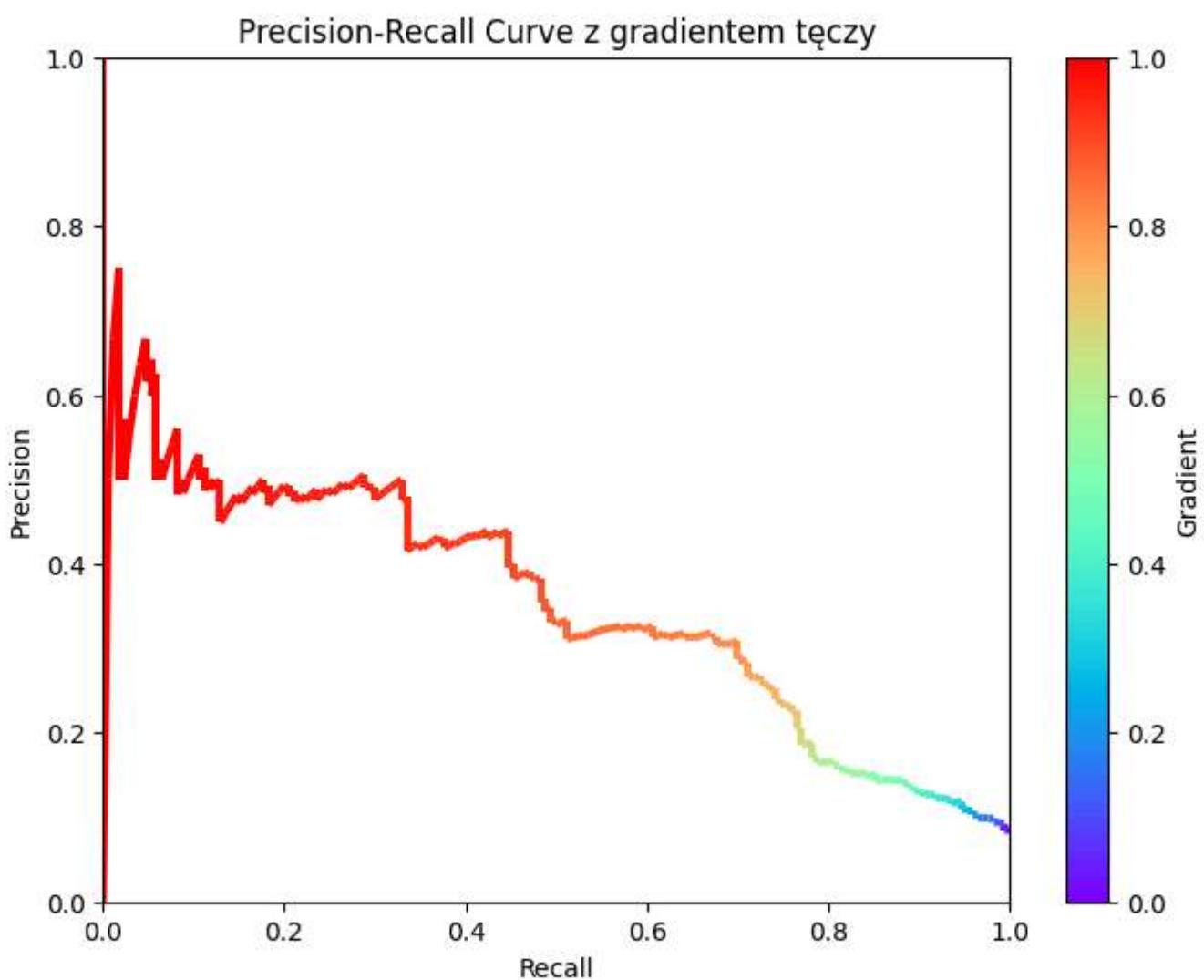
Model bardzo dobrze radzi sobie z identyfikacją komentarzy negatywnych (brak hejtu), ale ma tendencję do nadmiernego przypisywania etykiety hejt (wysoki FP). Niska liczba FN jest pozytywna, ale konieczne jest dalsze dostrojenie, aby zredukować fałszywe alarmy i poprawić precyzję klasyfikacji komentarzy hejtowych.



Analiza ROC Curve dla modelu LR

- **AUC (Area Under Curve) = 0.819** oznacza, że model w około 81,9% przypadków poprawnie rozróżni próbki pozytywne od negatywnych (w ujęciu rankingowym).
- Wartość AUC powyżej 0.8 jest zazwyczaj uznawana za przyzwoity poziom efektywności modelu – wskazuje na dobrą zdolność klasyfikacyjną w kontekście analizowanego zadania.
- Im większa powierzchnia pod krzywą, tym mniejsze prawdopodobieństwo błędnej klasyfikacji.

Podsumowując, model LR wykazuje się **dobrymi** zdolnościami predykcyjnymi, co potwierdza wartość AUC wynoszącą **0.819**. Dalsze dostrajanie hiperparametrów, zmiana progu decyzyjnego, a także ewentualne modyfikacje w zakresie przygotowania danych mogą jeszcze poprawić wyniki klasyfikacji.



Wykres Precision-Recall Curve z gradientem tęczy

Powyższy wykres ilustruje zależność między **Precision** a **Recall** (czułością) przy różnych progach decyzyjnych modelu:

- **Oś X (Recall):** pokazuje, jaki procent przykładów faktycznie pozytywnych (np. hejtowych) model poprawnie wykrywa.
- **Oś Y (Precision):** wskazuje, jaki odsetek przykładów zaklasyfikowanych jako pozytywne rzeczywiście należy do klasy pozytywnej.

W miarę przesuwania progu decyzyjnego model:

- Zwiększa **Recall** (wykrywa więcej przypadków pozytywnych),
- Jednak obniża się **Precision** (więcej przykładów błędnie klasyfikowanych jako pozytywne).

Na wykresie każdemu segmentowi linii przypisano kolor z palety tęczy, co pozwala zorientować się w zakresie wartości na krzywej:

- Czerwona część krzywej wskazuje obszar wysokiej precyzji i niskiego recall,
- W miarę zbliżania się do zielono-niebieskich odcieni, model zwiększa recall, ale precision spada.

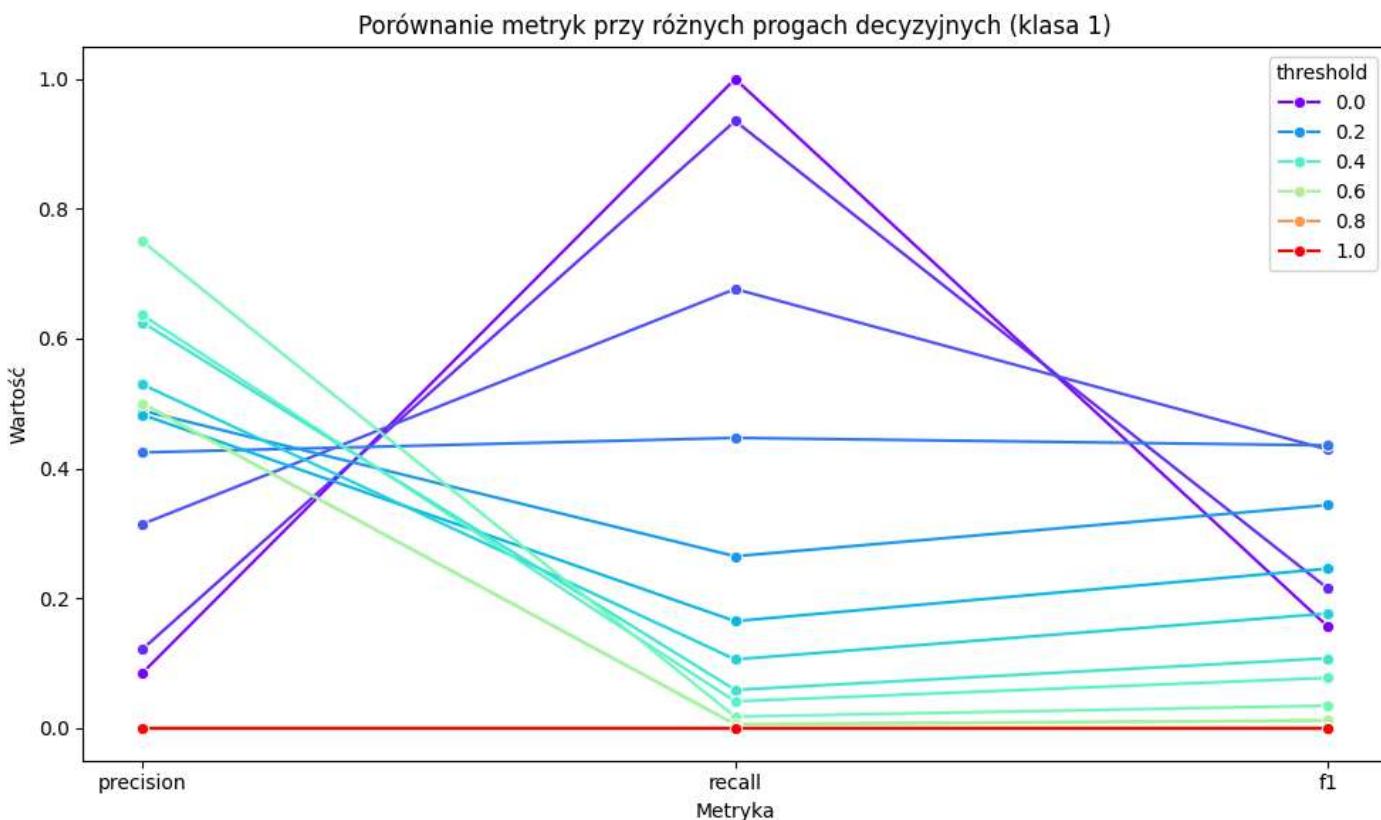
Taki wykres jest szczególnie przydatny w zadaniach z niezrównoważonymi klasami (np. klasyfikacja hejtu), gdzie sam wskaźnik Accuracy nie oddaje w pełni jakości modelu. Analiza zmiany **Precision** i **Recall** przy różnych progach pozwala lepiej dopasować zachowanie klasyfikatora do priorytetów projektu (np. minimalizowanie przeoczenia hejtu vs. ograniczanie fałszywych alarmów).

Analiza wykresu

1. **Wartości wysokiej precyzji (Precision)** świadczą o tym, że spośród klasyfikowanych przez model jako „pozytywne” większość rzeczywiście jest pozytywna.
2. **Wartości wysokiej czułości (Recall)** oznaczają, że spośród wszystkich faktycznie pozytywnych przypadków model wykrywa ich dużą część.
3. **Obserwacje krzywej:**

- **Początkowo** (z lewej strony, przy wysokiej precyzji) możemy zauważyc, że precyzja jest wysoka, ale recall nierzadko niższy. Oznacza to, że model wychwytuje niewiele próbek pozytywnych (bo jest zachowawczy), ale te, które zostaną uznane za pozytywne, rzeczywiście częściej nimi są.
- **W miarę przesuwania się w prawo** (rośnie recall) zwykle spada precyzja, ponieważ model wyłapuje coraz więcej próbek pozytywnych, ale również zaczyna dopuszczać większą liczbę fałszywie pozytywnych.
- Każdy segment linii reprezentuje inny próg, a intensywność koloru wskazuje na sekwencję (czy „krok”) w tych progach.

Ogólnie rzecz biorąc, **im bliżej lewego górnego rogu znajduje się krzywa Precision-Recall**, tym lepszą równowagę pomiędzy precyzją a czułością osiąga model w różnych zakresach progów. W zadaniach z nierównomierną liczbą przykładów w klasach krzywa Precision-Recall bywa bardziej miarodajna niż ROC, ponieważ kładzie nacisk na wydajność wykrywania pozytywnych (rzadkich) przypadków.



Porównanie metryk przy różnych progach decyzyjnych (klasa 1)

Na powyższym wykresie przedstawiono wartości **precision**, **recall** oraz **F1-score** dla klasy pozytywnej (hejtu) przy różnych progach decyzyjnych modelu. Kolor linii odpowiada określonemu progu w przedziale [0.0, 1.0], a każda linia pokazuje, jak zmieniają się trzy metryki (wyświetlone na osi X) dla tego konkretnego progu.

- **Oś X (Metryka):**

Wykres prezentuje trzy wybrane miary: **precision**, **recall** oraz **F1 dla klasy 1**.

- **Oś Y (Wartość):**

Przedstawia numeryczną wartość danej metryki, od 0 do 1.

- **Kolory linii (threshold)**

- Każda linia odpowiada konkretnemu progowi decyzyjnemu, w tym przypadku wartościami: 0.0, 0.2, 0.4, 0.6, 0.8 i 1.0.
- Niższy próg (blisko 0) oznacza, że model łatwiej klasyfikuje przypadki jako pozytywne, co skutkuje zazwyczaj **wyższym recall**, lecz niższą precision.
- Wyższy próg (blisko 1) sprawia, że model dużo rzadziej przypisuje etykietę pozytywną, więc **precision** bywa wyższa, ale recall drastycznie spada.

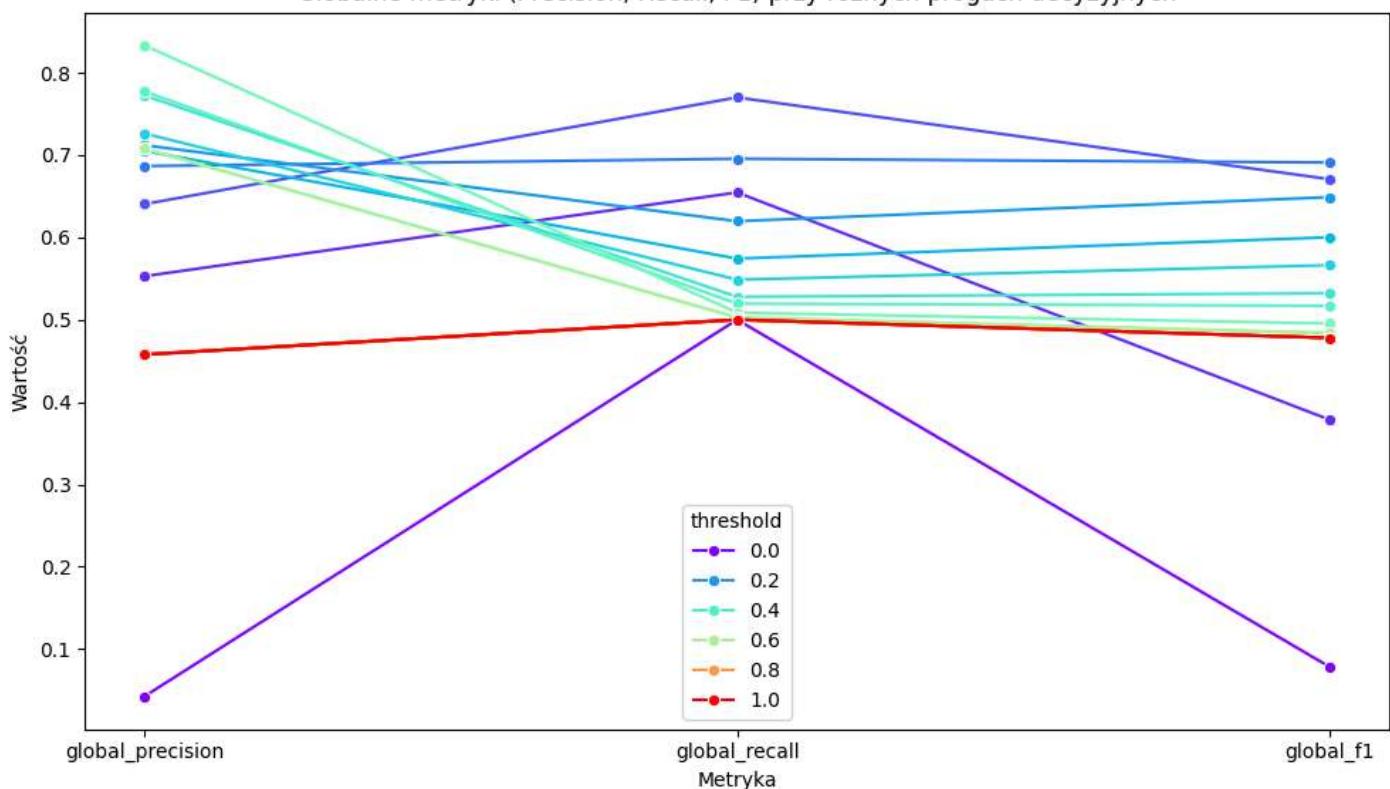
Interpretacja

Wartości pośrednie (0.2, 0.4, 0.6, 0.8)

- Dają różne kompromisy między **precision** a **recall**.
- Przy niższych progach model częściej klasyfikuje próbki jako pozytywne (zwiększa się recall, spada precision).
- Przy wyższych progach model jest bardziej „ostrożny” i wymaga wyższego prawdopodobieństwa przypisania pozytywnej etykiety, co z reguły obniża recall, ale może podnosić precision.

Wnioski

- Wybór **optymalnego progu** decyzyjnego ma kluczowe znaczenie w zadaniach klasyfikacji binarnej, zwłaszcza przy nierównomiernym rozkładzie klas.
- Najlepsza wartość **F1-score** (równowaga między precyją i czułością) nie zawsze przypada w progu 0.5; wykres jasno pokazuje, że mogą być warte rozważenia inne wartości.
- Taka wizualizacja pozwala lepiej zrozumieć działanie modelu i dopasować próg do konkretnych potrzeb.
- **Wykres umożliwia obserwację kompromisu między precyją a czułością dla klasy 1 w zależności od progu** decyzyjnego. Pomaga to w świadomym wyborze progu, tak by zoptymalizować metrykę F1-score, która jest najważniejsza w kontekście naszego zadania.



Globalne metryki (Precision, Recall, F1) przy różnych progach decyzyjnych

Powyższy wykres przedstawia, jak **globalna precyzja (precision)**, **globalna czułość (recall)** oraz **globalny F1-score** zmieniają się w zależności od wybranego progu decyzyjnego. Każda linia na wykresie reprezentuje zbiór wartości metryk osiąganych przy jednym progu (np. 0.0, 0.2, 0.4, 0.6, 0.8, 1.0), natomiast na osi X widzimy trzy kluczowe metryki: **global_precision**, **global_recall** i **global_f1**.

1. Oś X – Metryka

Na osi X znajdują się trzy kategorie:

- **global_precision** – globalna precyzja,
- **global_recall** – globalna czułość,
- **global_f1** – globalny F1-score,

które odpowiadają średnim wartościom metryk wyznaczonym dla różnych klas (np. macro-average lub micro-average).

2. Oś Y – Wartość

Wartość każdej z metryk (0–1), gdzie 1 oznacza wynik idealny.

3. Interpretacja

- **global_precision** rośnie zazwyczaj, gdy próg decyzyjny staje się wyższy (ale spada recall).
- **global_recall** zachowuje się odwrotnie – obniża się wraz ze wzrostem progu (co może mieć wpływ na F1-score).
- **global_f1** to średnia harmoniczna precision i recall, więc z reguły przyjmuje wartości pośrednie i reaguje, gdy jedna z metryk się pogarsza.

Dzięki temu wykresowi możemy porównać efekty zmiany progu decyzyjnego na trzy najważniejsze globalne miary jakości klasyfikatora, co jest przydatne w podejmowaniu decyzji o optymalnym punkcie pracy modelu.

Ogólne wnioski

- Wraz ze wzrostem progu decyzyjnego zwykle rośnie **precyzja**, ale spada **czułość**.
- **Miarę F1** warto traktować jako kompromis pomiędzy tymi dwoma skrajnymi wartościami.
- Różnokolorowe linie przecinające się w różnych punktach pokazują, że **efekt zmiany progu** bywa nieliniowy i zależy w dużej mierze od rozkładu danych (proporcji klas, jakości cech i dopasowania modelu).

W praktyce, aby wybrać optymalny próg:

- Można spojrzeć na ten wykres i wybrać najbardziej odpowiadający nam **punkt „zbalansowania”** między precyją i czułością.
- Alternatywnie przeanalizować wartości **F1** dla poszczególnych progów i zobaczyć, w którym miejscu ta miara jest najwyższa.

Podsumowanie:

Wykres pozwala łatwo porównać, jak kształtują się trzy kluczowe miary jakości klasyfikacji (**Precision, Recall, F1**) dla różnych wartości progu decyzyjnego. **Każda linia** reprezentuje inną wartość progu, a na osi x widzimy, jak ta sama wartość progu przekłada się na **każdy z trzech wskaźników**.

Strojenie hiperparametrów modelu regresji logistycznej

Proces strojenia

- **RandomizedSearchCV:**

Użyłem do szerokiego przeszukania przestrzeni hiperparametrów. Wyniki pozwoliły zidentyfikować obiecujące obszary dla dalszych poszukiwań.

- **Zawężenie zakresu:**

Na podstawie wyników RandomizedSearchCV sprecyzowałem zakresy wartości hiperparametrów, eliminując mniej obiecujące kombinacje. Dzięki temu kolejne strojenie skoncentrowano na najbardziej efektywnych przedziałach wartości.

- **GridSearchCV:**

W zawężonym zakresie zastosowałem GridSearchCV, w poszukiwaniu najlepszych wartości. Pozwoliło to na precyzyjne dobranie hiperparametrów w obrębie wcześniej wybranych zakresów.

- **Zapobieganie przetrenowaniu:**

Aby mieć pewność, że model nie jest przetrenowany, w obu etapach używałem walidacji krzyżowej (CV) oraz monitorowałem szereg metryk, takich jak **F1-score**, **precision** i **recall**.

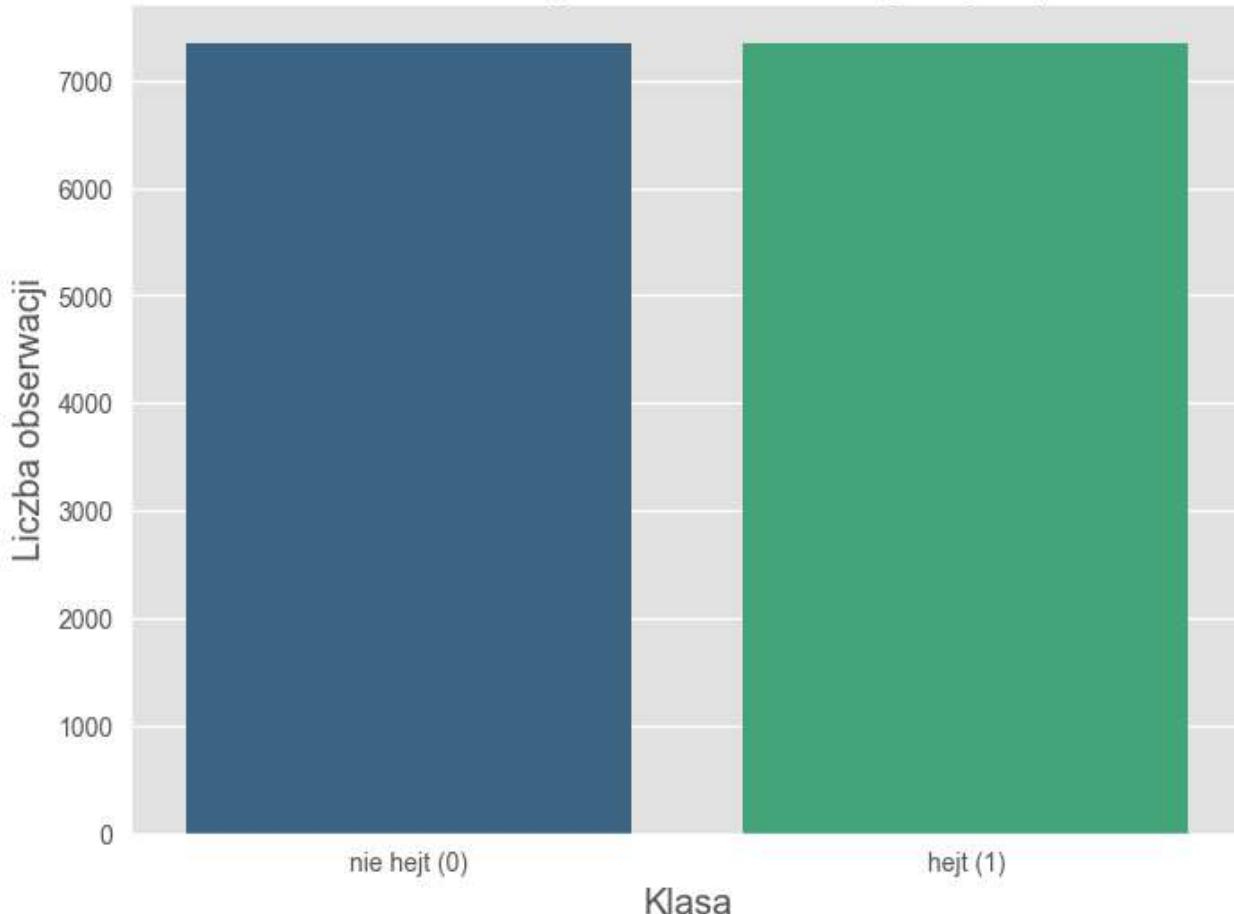
Kryteria wyboru modelu

- **Maksymalizacja F1-score (makro):** Podstawowym celem było maksymalizowanie makro F1-score na zbiorze testowym.
- **Kontrola przeuczenia:** Dodatkowo monitorowano różnicę F1-score między zbiorem treningowym a testowym, aby ograniczyć ryzyko przeuczenia.
- **Ostateczny wybór modelu:** Spośród badanych modeli wybrano ten, który uzyskał najwyższy F1-score na zbiorze testowym, przy jednoczesnym spełnieniu powyższych kryteriów. Dzięki temu finalny model uzyskał dobre wyniki zarówno na zbiorze testowym, jak i zachował odpowiednią generalizację.

Oversampling

Polega na zwiększaniu liczby przykładów w mniejszościowej klasie, aby uzyskać bardziej zrównoważony zbiór danych. Dzięki oversamplingowi model ma więcej przykładów, z których może uczyć się reprezentacji rzadszej klasy, co pomaga w zmniejszeniu błędów wynikających z dominacji klasy większościowej.

Rozkład klas w zbalansowanym zbiorze treningowym po oversamplingu



Podsumowanie

1. Przygotowanie danych:

Najpierw dokonuje się oversamplingu, czyli zwiększenia liczby przykładów z klasy mniejszościowej.

2. Trening modelu:

Model Logistic Regression jest trenowany na wzbogaconym zbiorze danych. Dzięki zbalansowaniu klas, model uczy się bardziej ogólnych reprezentacji, co zmniejsza ryzyko przetrenowania.

3. Korzyści:

- Lepsza generalizacja na danych testowych.
- Poprawa metryk klasyfikacyjnych, dzięki balansowaniu zbioru danych.

Oversampling + Synomiczne Zmiany

Dodatkową techniką tego podejścia oversamplingu to:

Synomiczne zmiany (Synonimiczna Augmentacja):

To technika wzbogacania danych, w której niektóre słowa w tekście są zastępowane ich synonimami. Dzięki tej modyfikacji, każdy oryginalny przykład może zostać rozszerzony o jego warianty, co zwiększa różnorodność danych bez konieczności pozyskiwania dodatkowych danych. Synonimiczna augmentacja pozwala modelowi na lepsze uogólnienie i zwiększa odporność na wariacje językowe.

Zastosowanie

1. Przygotowanie danych:

Najpierw dokonuje się oversamplingu. Następnie oryginalne przykłady są modyfikowane przez zastępowanie wybranych słów ich synonimami. Efektem jest powstanie rozszerzonego zbioru danych, który lepiej reprezentuje pełne spektrum wariacji językowych.

2. Trening modelu:

Model Logistic Regression jest trenowany na wzbogaconym zbiorze danych. Dzięki różnorodności danych, jakie wprowadza synonimiczna augmentacja, model uczy się bardziej ogólnych reprezentacji, co zmniejsza ryzyko przetrenowania (overfitting).

3. Korzyści:

- Lepsza generalizacja na danych testowych.
- Wyższa odporność na różnice językowe w rzeczywistych danych.
- Poprawa metryk klasyfikacyjnych (np. F1-score), dzięki skutecznieszemu balansowaniu zbioru danych.

Oversampling + Parafrazowanie

W modelu "z oversamplingiem + parafrazowaniem" do oversamplingu dołączamy dodatkowo:

Parafrazowanie:

- Parafrazowanie polega na generowaniu nowych wersji istniejących tekstów poprzez przekształcenie ich struktury lub słownictwa przy zachowaniu pierwotnego znaczenia.
- Ta technika wzbogaca dane o syntaktyczne i semantyczne warianty, co sprawia, że model staje się bardziej odporny na wariancje językowe występujące w rzeczywistych danych.
- Parafrazowanie pozwala na zwiększenie różnorodności językowej bez konieczności ręcznego zbierania nowych przykładów, co jest szczególnie cenne przy ograniczonej liczbie danych.

Podsumowanie

Model z oversamplingiem + parafrazowaniem stanowi skuteczne podejście do:

- **Zbalansowania klasy mniejszościowej.**
- **Wzbogacenia danych**, poprzez generowanie parafraszów pierwotnych przykładów, co poprawia różnorodność i reprezentatywność zbioru danych.
- **Poprawy generalizacji modelu**, dzięki czemu staje się on mniej podatny na przetrenowanie i bardziej odporny na zmiany językowe w danych wejściowych.

Oversampling + Back Translation

Aby dodatkowo wzbogacić dane i zwiększyć różnorodność lingwistyczną, można zastosować technikę **back translation**.

Kluczowe dodatkowe elementy podejścia do oversamplingu

Back Translation:

- Back translation to technika augmentacji danych, w której tekst oryginalny jest tłumaczony na inny język, a następnie z powrotem na język źródłowy.
- Dzięki temu procesowi tworzone są nowe warianty tekstu, które zachowują pierwotne znaczenie, ale różnią się stylistycznie oraz słownictwem.
- Back translation jest szczególnie przydatny w NLP, ponieważ pomaga zwiększyć różnorodność danych bez utraty kluczowych informacji semantycznych.

Zalety podejścia

- **Zwiększenie różnorodności zbioru danych:** Back translation wprowadza nowe warianty tekstu, zachowując przy tym jego oryginalne znaczenie.
- **Lepsza generalizacja modelu:** Model uczony na wzbogaconym zbiorze danych może lepiej radzić sobie z różnymi formami wyrażania tej samej treści.
- **Zmniejszenie ryzyka przetrenowania:** Oversampling w połączeniu z augmentacją danych pomaga w osiągnięciu optymalnego balansu, co umożliwia budowanie modeli bardziej odpornych na nadmierne dopasowanie do zbioru treningowego.

Model z Wagami Klas

W sytuacjach, gdy zbiór danych charakteryzuje się nierównomiernym rozkładem klas, model może faworyzować klasę większościową, co skutkuje słabą detekcją klasy mniejszościowej. Jednym ze sposobów przeciwdziałania temu problemowi jest zastosowanie **wag klas** (class weights) podczas treningu modelu.

Kluczowe aspekty podejścia:

1. Cel stosowania wag:

- **Zbalansowanie wpływu poszczególnych klas:** Przypisanie większych wag dla klasy mniejszościowej sprawia, że błędy popełnione przy jej klasyfikacji są „karane” bardziej, co zmusza model do lepszego uczenia się cech tej klasy.
- **Redukcja tendencyjności modelu:** Dzięki zastosowaniu wag model nie będzie nadmiernie optymalizował predykcji dla klasy większościowej, co pozwala na osiągnięcie bardziej wyważonych wyników.

2. Jak to działa:

- Podczas treningu algorytm uwzględnia dodatkowe informacje określające, jak ważna jest dana klasa. Wagi te mogą być określone ręcznie lub automatycznie (np. poprzez ustawienie opcji `class_weight='balanced'` w wielu algorytmach, takich jak Logistic Regression czy drzewa decyzyjne).
- Model, korygując swoje parametry, stara się minimalizować skorygowany koszt błędu, w którym błędy dla rzadziej występujących klas mają wyższy koszt.

3. Zastosowanie w praktyce:

- **Przygotowanie danych:** Dane są dzielone na zestawy treningowe i testowe. Przy inicjalizacji modelu ustawiana jest odpowiednia wartość dla parametrów wag klas.
- **Trening modelu:** Podczas treningu model bierze pod uwagę te wagi przy optymalizacji, co skutkuje lepszym dopasowaniem do rzadko występujących przykładów.
- **Ewaluacja:** Metryki takie jak precision, recall oraz F1-score, szczególnie dla klasy mniejszościowej, są monitorowane, aby ocenić skuteczność zastosowania wag.

SMOTE (Synthetic Minority Oversampling Technique)

SMOTE (Synthetic Minority Oversampling Technique) to metoda generowania sztucznych próbek klasy mniejszościowej poprzez interpolację pomiędzy istniejącymi przykładami tej klasy.

Została opisana przez Nitesha Chawla i współautorów w 2002 roku jako ulepszenie prostego powielania próbek mniejszościowej klasy, które nie wnoszą dodatkowej informacji do modelu

Jak działa SMOTE?

1. Wybór próbek

Dla każdej próbki mniejszościowej (x_i) wybierana jest losowa podgrupa (k) najbliższych sąsiadów (w tej samej klasie) przy użyciu odległości euklidesowej.

2. Interpolacja

Z wybranych sąsiadów losowany jest jeden punkt (x_{nn}), a następnie tworzona jest nowa próbka:

$$x_{\text{new}} = x_i + \delta \times (x_{nn} - x_i),$$

gdzie ($\delta \sim \mathcal{U}(0, 1)$).

3. Powtarzanie

Krok interpolacji powtarza się tyle razy, aby uzyskać żądaną współczynnik nadpróbkowania dla klasy mniejszościowej.

Kluczowe parametry

- **k_neighbors:** liczba sąsiadów używana do interpolacji (domyślnie 5).
- **sampling_strategy:** określa, jaka część próbek ma być sztucznie dodana (np. „auto” oznacza wyrównanie liczebności klas).
- **random_state:** ziarno generatora liczb losowych, zapewniające powtarzalność wyników.

Podsumowanie

- **Zastosowanie metody Smote redukuje nadmierne dopasowanie** (overfitting) w porównaniu do prostego duplikowania próbek klasy mniejszościowej.

ADASYN (Adaptive Synthetic Sampling Technique)

ADASYN (Adaptive Synthetic Sampling) to adaptacyjna metoda oversamplingu klasy mniejszościowej, która automatycznie decyduje, ile syntetycznych próbek wygenerować dla każdego przykładu na podstawie „trudności” jego otoczenia w przestrzeni cechowej. Została zaproponowana przez He i współautorów w 2008 roku jako rozwinięcie SMOTE, mające na celu lepsze wyrównanie rozkładu klas przy jednoczesnym skupieniu się na obszarach trudno uczących się.

Jak działa ADASYN?

1. Obliczenie współczynnika trudności (r_i)

Dla każdej próbki mniejszościowej (x_i) oblicza się liczbę przykładów innych klas wśród jej (k)-najbliższych sąsiadów, tworząc surowy współczynnik ($r_i = \frac{\text{liczba sąsiadów z klasą większości}}{k}$).

2. Normalizacja współczynników

Surowe wartości (r_i) są normalizowane, by suma wszystkich (\hat{r}_i) równała się 1:

$$\hat{r}_i = \frac{r_i}{\sum_j r_j}.$$

Dzięki temu uzyskujemy rozkład wag, który określa, ile próbek wygenerować dla każdego (x_i).

3. Wyznaczenie liczby nowych próbek

Całkowitą liczbę syntetycznych próbek (G) definiuje współczynnik oversamplingu (np. wyrównanie klas). Następnie dla każdej próbki (x_i) oblicza się

$$g_i = \hat{r}_i \times G,$$

czyli ile nowych punktów wygenerować w otoczeniu (x_i).

4. Generowanie syntetycznych próbek

Dla każdej próbki (x_i) i dla każdej z (g_i) iteracji losuje się jednego z jej (k)-najbliższych sąsiadów (x_{nn}) i interpoluje nowy punkt:

$$s = x_i + \lambda \times (x_{nn} - x_i), \quad \lambda \sim \mathcal{U}(0, 1).$$

W rezultacie powstają syntetyczne próbki rozmiieszczone tam, gdzie oryginalne dane były „trudne” do sklasyfikowania.

Kluczowe parametry

- **n_neighbors (k)**: liczba sąsiadów używana do oszacowania (r_i) i generowania nowych próbek (domyślnie 5).
- **sampling_strategy**: określa docelowy rozkład klas po oversamplingu (np. 'auto' dla pełnego wyrównania).
- **random_state**: ziarno generatora liczb losowych, gwarantujące powtarzalność wyników.

Podsumowanie

- **Adasyn** generuje próbki w rejonach trudniejszych do klasyfikacji, zmniejszając ryzyko nadmiernego dopasowania w łatwych obszarach oraz waży generowanie próbek według lokalnej gęstości błędów.

SMOTETomek

SMOTETomek jest to hybrydowa metoda radzenia sobie z nierównowagą klas, która łączy oversampling za pomocą SMOTE z oczyszczaniem granic decyzyjnych przez usuwanie par Tomek Links.

Jak działa metoda po użyciu Smote

Usuwanie Tomek Links

Następnie identyfikowane są pary próbek z różnych klas, które są wzajemnie swoimi najbliższymi sąsiadami (Tomek Links). Usunięcie takich par oczyszcza obszary graniczne i redukuje nakładanie się klas.

Kluczowe parametry

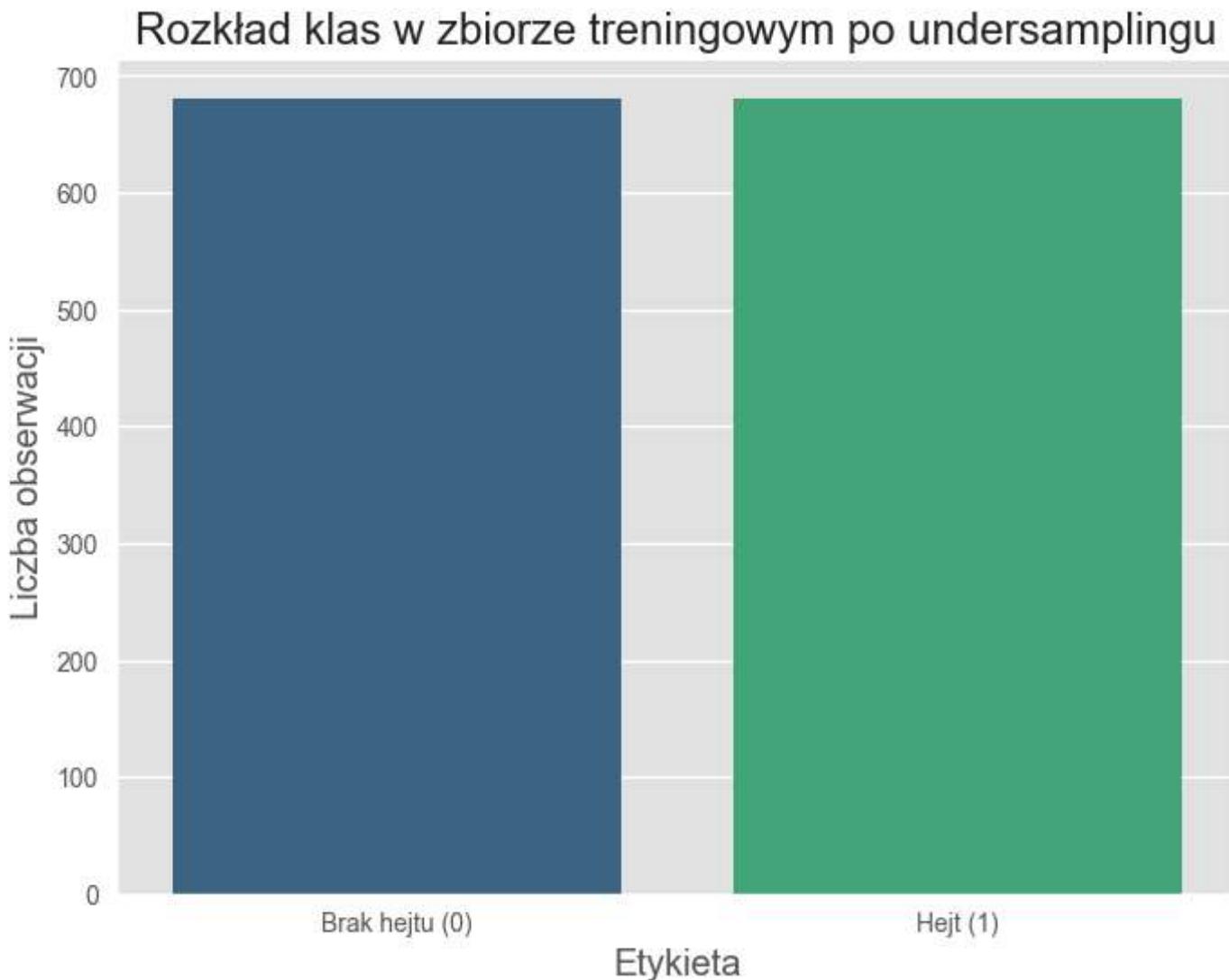
- **sampling_strategy**: docelowy rozkład klas po resamplingu, np. `'auto'` dla wyrównania liczebności.
- **random_state**: ziarno generatora losowego, zapewniające powtarzalność wyników.
- **smote**: instancja klasy SMOTE z takimi parametrami jak `k_neighbors` decydującymi o liczbie sąsiadów do interpolacji.
- **tomek**: instancja klasy TomekLinks, określająca sposób detekcji i usuwania par Tomek Links.
- **n_jobs**: liczba wątków do równoległego przetwarzania, przyspieszająca działanie na dużych zbiorach.

Podsumowanie

- **Usunięcie Tomek Links** eliminuje próbki graniczne oraz potencjalne outlierы, poprawiając jakość danych treningowych.
- **Połączenie** oversamplingu z oczyszczaniem granic prowadzi do **wyraźniej** rozdzielonych klas i **lepszych** wyników klasyfikacji.

Undersampling

Celem tej metody jest zredukowanie liczby przykładów z klasy dominującej, aby uzyskać bardziej zbalansowany zbiór danych.



Główne aspekty undersamplingu:

1. Redukcja nadmiaru danych:

- Metoda polega na losowym (lub strategicznym) usunięciu części przykładów z klasy większościowej.
- Dzięki zmniejszeniu liczby przykładów klasy dominującej model otrzymuje bardziej zrównoważony zbiór, co ułatwia naukę reprezentacji obu klas.

2. Zalety undersamplingu:

- **Poprawa równowagi klas:** Przekształcenie zbioru danych do bardziej zbalansowanego stanu może pozytywnie wpływać na wyniki modeli klasyfikacyjnych, szczególnie na metryki dotyczące klasy mniejszościowej.
- **Skrócenie czasu treningu:** Mniejsza liczba przykładów oznacza szybsze uczenie modelu, co jest korzystne przy dużych zbiorach danych.

- **Redukcja ryzyka przetrenowania:** Mniej danych może czasami pomóc w uniknięciu nadmiernego dopasowania do dużej, dominującej klasy.

3. Wady undersamplingu:

- **Utrata informacji:** W wyniku redukcji danych z klasy większościowej można stracić cenne informacje, co może wpływać na zdolność modelu do generalizacji.
- **Słabsza reprezentacja klasy większościowej:** Jeśli próbki usunięte w procesie undersamplingu zawierały unikalne cechy, model może ich nie nauczyć się poprawnie, co może skutkować gorszym rozpoznawaniem przypadków klasy większościowej.

Cluster Centroids

Cluster Centroids to technika *undersamplingu*, która służy do równoważenia zbiorów danych o nierównomiernym rozkładzie klas. Działa ona na zasadzie klasteryzacji próbek należących do klasy większościowej oraz zastępowania poszczególnych klastrów ich centroidami (czyli środkowymi punktami). Dzięki temu:

- **Redukujemy liczbę próbek** z klasy większościowej, eliminując redundancję.
- **Zachowujemy reprezentatywne cechy** danych, ponieważ centroydy oddają ogólny rozkład danych w klastrze.
- **Ułatwiamy działanie algorytmów klasyfikacyjnych**, które mogą lepiej radzić sobie z bardziej zbalansowanym zbiorem danych.

Zalety

- Redukcja rozmiaru zbioru danych klasy większościowej.
- Zmniejszenie ryzyka przeuczenia (overfitting) przez eliminację nadmiarowych danych.
- Utrzymanie ogólnych właściwości danych.

Wady

- Możliwa utrata szczegółowych informacji, ponieważ centroid może nie oddać wszystkich niuansów oryginalnych próbek.
- Możliwe problemy przy bardzo złożonych lub rozproszonych klastrach – centroid może nie reprezentować właściwie całości rozkładu.

Tomek Links oraz Edited Nearest Neighbors (ENN)

W procesie przetwarzania i balansowania zbioru treningowego, oprócz metod oversamplingu lub undersamplingu (które modyfikują liczbę próbek), bardzo ważne jest także oczyszczenie danych – czyli usunięcie przykładów, które znajdują się na granicy klas. Takie przykłady mogą być trudne do sklasyfikowania i wprowadzać szum, co z kolei może negatywnie wpływać na jakość modelu. Do tego celu stosuje się metody, takie jak **Tomek Links** i **Edited Nearest Neighbors (ENN)**.

Tomek Links

Tomek Links to metoda usuwania przykładów, która identyfikuje pary próbek (Tomek Linki) z różnych klas, które są sobą najbliższe – to znaczy, że są wzajemnymi najbliższymi sąsiadami. Założeniem jest, że próbki takie mogą znajdować się blisko granicy decyzyjnej, często będąc potencjalnym szumem lub źródłem niejednoznaczności. W praktyce usuwa się jedną (najczęściej z klasy większościowej) z każdej takiej pary, co pomaga w:

- **Czyszczeniu danych** – usunięcie trudnych przypadków oraz przykładów mogących wprowadzać zakłócenia.
- **Wyoszczędzeniu granicy decyzyjnej** – dzięki eliminacji punktów blisko granicy, klasyfikator może lepiej zdefiniować separację między klasami.

Edited Nearest Neighbors (ENN)

Edited Nearest Neighbors (ENN) to metoda, która opiera się na analizie otoczenia każdego przykładu. Dla każdej próbki rozpatruje się jej k -najbliższych sąsiadów i jeśli dana próbka nie zgadza się z większością swoich sąsiadów (tzn. jej etykieta odbiega od etykiet dominujących w otoczeniu), zostaje usunięta. Pozwala to na:

- **Usunięcie szumów i nietypowych przykładów** – eliminacja przykładów, które mogą negatywnie wpływać na granicę decyzyjną.
- **Poprawę czystości zbioru danych** – zmniejszenie liczby błędnie sklasyfikowanych punktów w zbiorze treningowym, co w rezultacie może wpływać na lepszą generalizację klasyfikatora.

Podsumowanie wyników modeli

Poniższa tabela przedstawia kluczowe metryki (Accuracy, Precision, Recall i F1-score - wszystkie w ujęciu macro-average) dla zbioru treningowego i testowego dla różnych podejść stosowanych przy klasyfikacji komentarzy (przyjmując, że 1 oznacza „hejt” a 0 – „brak hejtu”), **dla wszystkich podejść** przy budowaniu modelu została użyta **siatka parametrów losowych oraz wąska siatka parametrów**, w celu uzyskania jak **najlepszych metryk dla każdego podejścia**. Wyniki zostały zaokrąglone do czterech miejsc dziesiętnych.

Model	Train Accuracy	Train Precision	Train Recall	Train F1	Test Accuracy	Test Precision	Test Recall	Test F1
LR zwykły	0.9200	0.8343	0.5419	0.5566	0.9163	0.7298	0.5379	0.5494
LR z hiperparametrami	0.9269	0.8691	0.5883	0.6283	0.9188	0.7494	0.5766	0.6063
Oversampling	0.6650	0.6996	0.6650	0.6498	0.8237	0.5794	0.6394	0.5928
Oversampling ze zmianami synonicznymi	0.6185	0.6149	0.5972	0.5913	0.7574	0.5474	0.6086	0.5433
Oversampling z parafrazowaniem	0.7907	0.8241	0.7907	0.7851	0.8944	0.6243	0.5873	0.6010
Model oversampling + back translation	0.6835	0.7352	0.6835	0.6651	0.8641	0.5984	0.6215	0.6079
Oversampling SMOTE	0.6295	0.7138	0.6295	0.5889	0.8800	0.5899	0.5768	0.5825
Oversampling ADASYN	0.6306	0.7125	0.6328	0.5934	0.8780	0.5921	0.5837	0.5876
Oversampling SMOTE-Tomek	0.6302	0.7143	0.6302	0.5900	0.8800	0.5899	0.5768	0.5825
Model z waga klas	0.8592	0.6593	0.8045	0.6949	0.8441	0.6211	0.7173	0.6455
Undersampling	0.8202	0.6343	0.8052	0.6597	0.7923	0.5894	0.6997	0.6007
Undersampling + ClusterCentroids	0.5624	0.5624	0.5624	0.5622	0.5667	0.5297	0.5951	0.4503
Unersampling + Tomeklinks i ENN	0.9295	0.8913	0.6076	0.6554	0.9173	0.7312	0.5865	0.6172

Interpretacja wyników

- **LR zwykły:**

Osiąga bardzo wysoką dokładność (Accuracy ~0.92) na obu zbiorach, jednak recall dla klasy hejt jest niski, co skutkuje niskim F1-score (0.5566 na treningu, 0.5494 na teście). Model wykazuje wysoki poziom precyzji, ale zaniedbuje wiele przypadków klasy hejt.

- **LR z hiperparametrami:**

Dostosowanie hiperparametrów (np. zwiększenie regularizacji) poprawia recall i podnosi F1-score do poziomu 0.6283 na treningu i 0.6063 na teście.

Wyniki wskazują na lepszy kompromis między precision a recall.

- **Oversampling:**

Zastosowanie oversamplingu (bez augmentacji) znaczco obniża wyniki na zbiorze treningowym (Accuracy ~**0.6650**) i testowym (Accuracy ~**0.8237**), co może wskazywać na problemy ze stratą informacji przy powielaniu przykładów.

- **Oversampling z synonicznymi zamianami:**

Użycie augmentacji przez synoniczne zamiany podnosi **F1-score do 0.5913** na zbiorze treningowym oraz **0.5433** na teście. Choć wyniki treningowe są wyzsze, różnica względem wyników testowych może sugerować pewne nadmierne dopasowanie modelu do danych treningowych.

- **Oversampling z parafrazowaniem:**

Ta technika osiąga umiarkowane wyniki – F1-score wynosi **0.7851** na treningu i **0.6010** na teście, co wskazuje na overfitting. Metoda ta generuje bardziej zróżnicowane próbki, co w porównaniu do innych modeli nie wpływa korzystnie na zdolność generalizacji modelu.

- **Model oversampling + back translation:** Połączenie oversamplingu z back translation generuje F1-score **0.6651** na treningu i **0.6079** na teście. Wyniki te sugerują, że metoda ta poprawia reprezentatywność mniejszościowej klasy, ale kompromis między precyzją, a czułością pozostaje wyzwaniem.

- **Oversampling SMOTE**

Metoda SMOTE na zbiorze treningowym osiąga **F1-score = 0.5889**. Na teście F1 spada do **0.5825**. Wskazuje to, że SMOTE poprawia nieco czułość mniejszościowej klasy w porównaniu z prostym oversamplingiem, ale kosztem nieznacznego spadku precyzji i ogólnej jakości – możliwe, że sztuczne przykłady są zbyt jednorodne.

- **Oversampling ADASYN**

ADASYN na treningu daje **F1-score = 0.5934**. Na zbiorze testowym F1 = **0.5876**.

Podobnie jak SMOTE, ADASYN zwiększa recall mniejszości, ale osiąga minimalnie wyzsze F1 niż SMOTE. Jest to efekt generowania większej liczby przykładów trudnych („blizej granicy”), co poprawia generalizację.

- **Oversampling SMOTE-Tomek**

Kombinacja SMOTE i TomekLinks na treningu przynosi F1-score = **0.5900**. Na teście F1 wynosi **0.5825**. Usunięcie przykładowych duplikatów TomekLinks wyrównuje korzyści SMOTE, utrzymując zbliżone metryki. Metoda ta pomaga zmniejszyć nadmierne dopasowanie do syntetycznych próbek, ale nie przynosi znaczącej poprawy względem samego SMOTE.

- **Model z wagami klas:**

Użycie `class_weight='balanced'` przy standardowej konfiguracji modelu daje stabilne, aczkolwiek umiarkowane wyniki z F1-score około **0.6455** na zbiorze testowym.

Świadczy to o kompromisie między precyzją a czułością w obecności niezrównoważonych danych.

- **Undersampling:** Metoda undersamplingu powoduje zwiększenie recall, ale kosztem precyzji – F1-score osiąga wartość **0.6597** na treningu i **0.6007** na teście. Oznacza to, że model lepiej wykrywa przypadki klasy hejt, jednak generuje więcej fałszywych alarmów.

- **ClusterCentroids:**

Metoda ta daje najniższe wyniki – F1-score ~**0.5622** na treningu i ~**0.4503** na teście, co wskazuje na overfitting, oraz że zbyt agresywna redukcja danych prowadzi do utraty istotnych informacji.

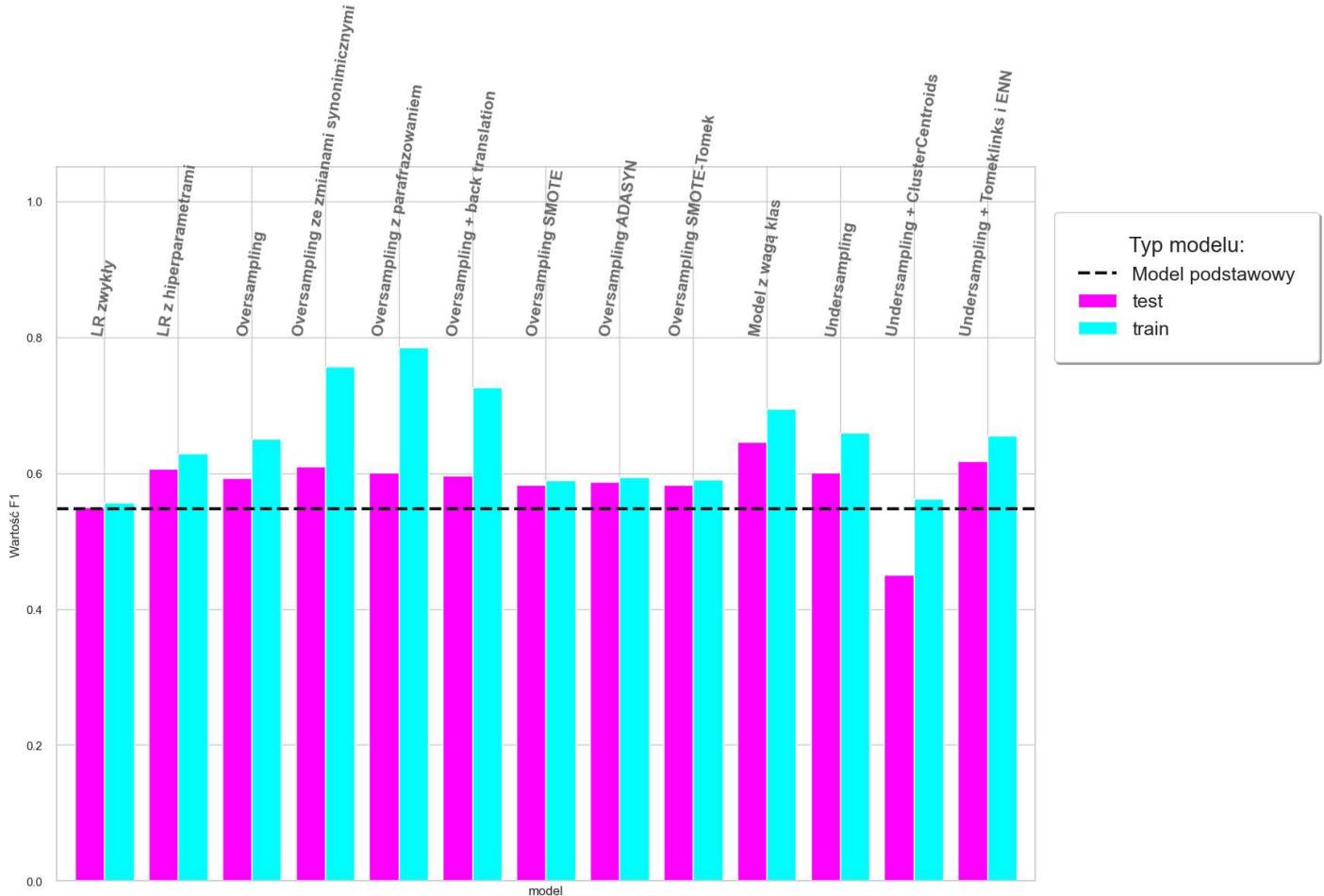
- **TomekLinks i ENN:**

Ta metoda osiąga wyniki bardzo podobne do podejścia z wagami klas (F1 ~**0.6554** na treningu i

~**0.6172** na teście), co sugeruje, że oczyszczenie danych z niejasnych przykładów poprawia granice decyzyjne modelu.

Podsumowanie

1. Najwyższe **F1** osiąga **model z wagą klas** (0.6455) – to najlepszy wybór, ponieważ naszym celem jest maksymalizacja F1.
2. **Undersampling** (0.6007), **Oversampling + back translation** (0.6079), **LR z hiperparametrami** (0.6063) oraz **Undersampling + Tomeklinks & ENN** (0.6172) to **solidne alternatywy**, które mają dosyć wysoką metrykę F1, a dodatkowo **nie wykazują oznak przetrenowania (overfittingu)**.
3. Klasyczne **metody SMOTE/ADASYN/SMOTE-Tomek** wypadają słabiej niż proste **augmentacje tekstowe czy model z wagą klas**.
4. Metody polegające na **agresywnym undersamplingu** bez dodatkowych filtrów (ClusterCentroids) **znacząco obniżają F1**.



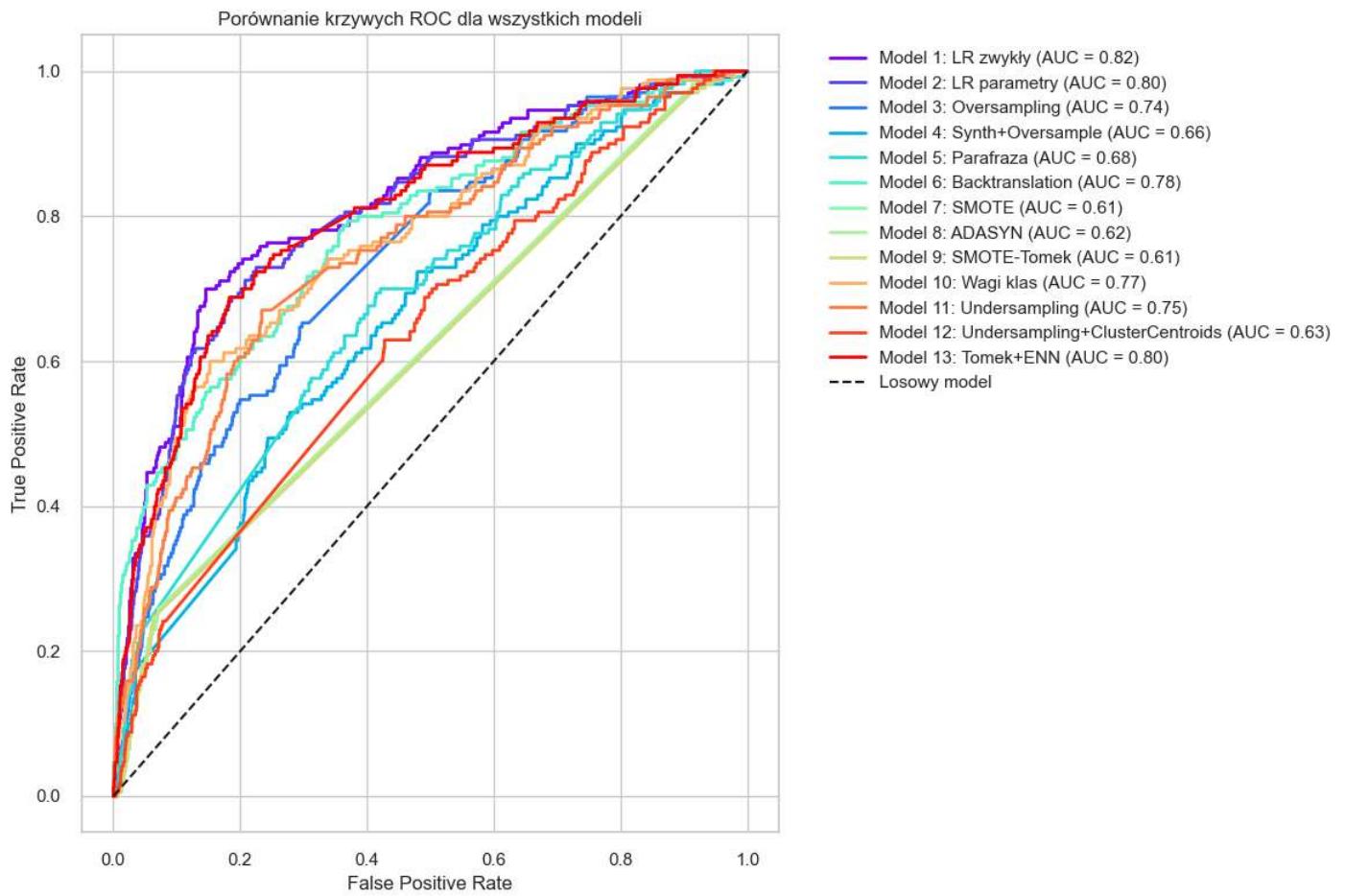
Podsumowanie wyników modeli na wykresie F1-score (train vs test)

Obserwacje:

- Czarna przerywana linia pokazuje wyniki pierwszego modelu który budowaliśmy, możemy zauważyć, że prawie wszystkie modele osiągają lepsze wyniki niż podstawowy model.
- Różnice pomiędzy wynikami **train/test** są umiarkowane – wskazuje to na brak istotnego **overfittingu** w większości przypadków.
- Modele wykorzystujące **oversampling (zmiany synonimiczne, parafrasowanie, backtranslation)** oraz **undersampling z użyciem ClusterCentroids** wykazują oznaki **overfittingu**, czego dowodzi różnica między wynikami **F1-score** na zbiorze treningowym i testowym mieszcząca się w przedziale **0.1–0.15**.
- Modele z zastosowaniem **undersamplingu** mają większe różnice między train a test lub niższe wyniki ogólnie, co może świadczyć o **utracie informacji**.
- Widać wyraźne **zróżnicowanie skuteczności** strategii balansu danych — niektóre podejścia **znaczowo poprawiają** wynik testowy w porównaniu do bazowej logistyki.

Wnioski:

- Modele oversamplujące (szczególnie SMOTE + TomekLinks oraz ADASYN) dają najlepszy kompromis między wysokim **F1-score** na treningu i testowym.
- Undersampling wymaga ostrożności — może prowadzić do obniżenia jakości predykcji.
- Najlepsze** modele, które osiągają najwyższy **F1-score** na danych testowych i nie wykazują overfittingu to **model z wagą klas** oraz **model LR z hiperparametrami**.



1. Najlepszy model (wg AUC):

- **Model 1: LR zwykły** osiąga najwyższe AUC = 0.82. Oznacza to, że klasyczna regresja logistyczna najlepiej rozróżnia klasy na różnych progach decyzyjnych.

2. Modele o zbliżonej drugiej najlepszej skuteczności:

- **Model 2 (LR parametry)** oraz **Model 13 (Tomek+ENN)** plasują się tuż za liderem z AUC = 0.80, co czyni je drugą najskuteczniejszą grupą.

3. Grupa średniej wydajności:

- **Model 6 (Backtranslation)** – AUC = 0.78
- **Model 10 (Wagi klas)** – AUC = 0.77
- **Model 11 (Undersampling)** – AUC = 0.75

Te podejścia oferują przyzwoitą czułość kosztem nieco wyższego FPR.

4. Najniższa skuteczność:

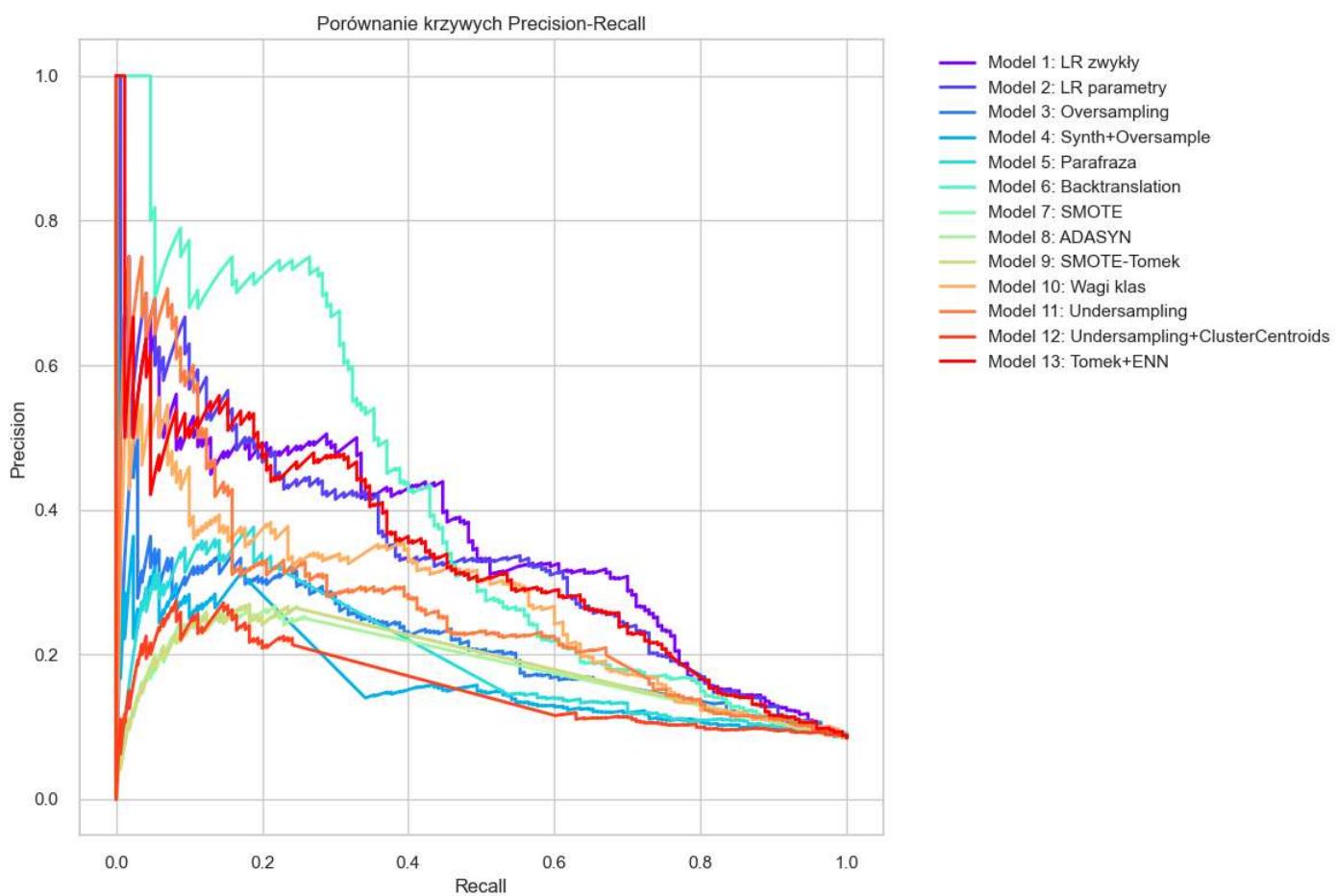
- **Modele oversamplingowe i hybrydowe** (Model 3: Oversampling – AUC 0.74; Model 4: Oversampling ze zmianami synonimicznymi – AUC 0.66; Model 7: SMOTE – AUC 0.61; Model 8: ADASYN – AUC 0.62; Model 9: SMOTE-Tomek – AUC 0.61; Model 12: Undersampling+ClusterCentroids – AUC 0.63)
- Proste oversamplingi i niektóre hybrydy plasują się najniżej, często zbliżając się do losowego klasyfikatora.

5. Wnioski praktyczne:

- **Regresja logistyczna** (zwykła lub z regularizacją) pozostaje bardzo silnym punktem odniesienia.
- **Backtranslation** to dobry kompromis, poprawia AUC względem prostego oversamplingu.
- Agresywne balansowanie (wagi klas, undersampling) zwiększa czułość, ale kosztem wzrostu FPR – przydatne, gdy kluczowe jest wychwycenie jak największej liczby pozytywów.

Podsumowanie

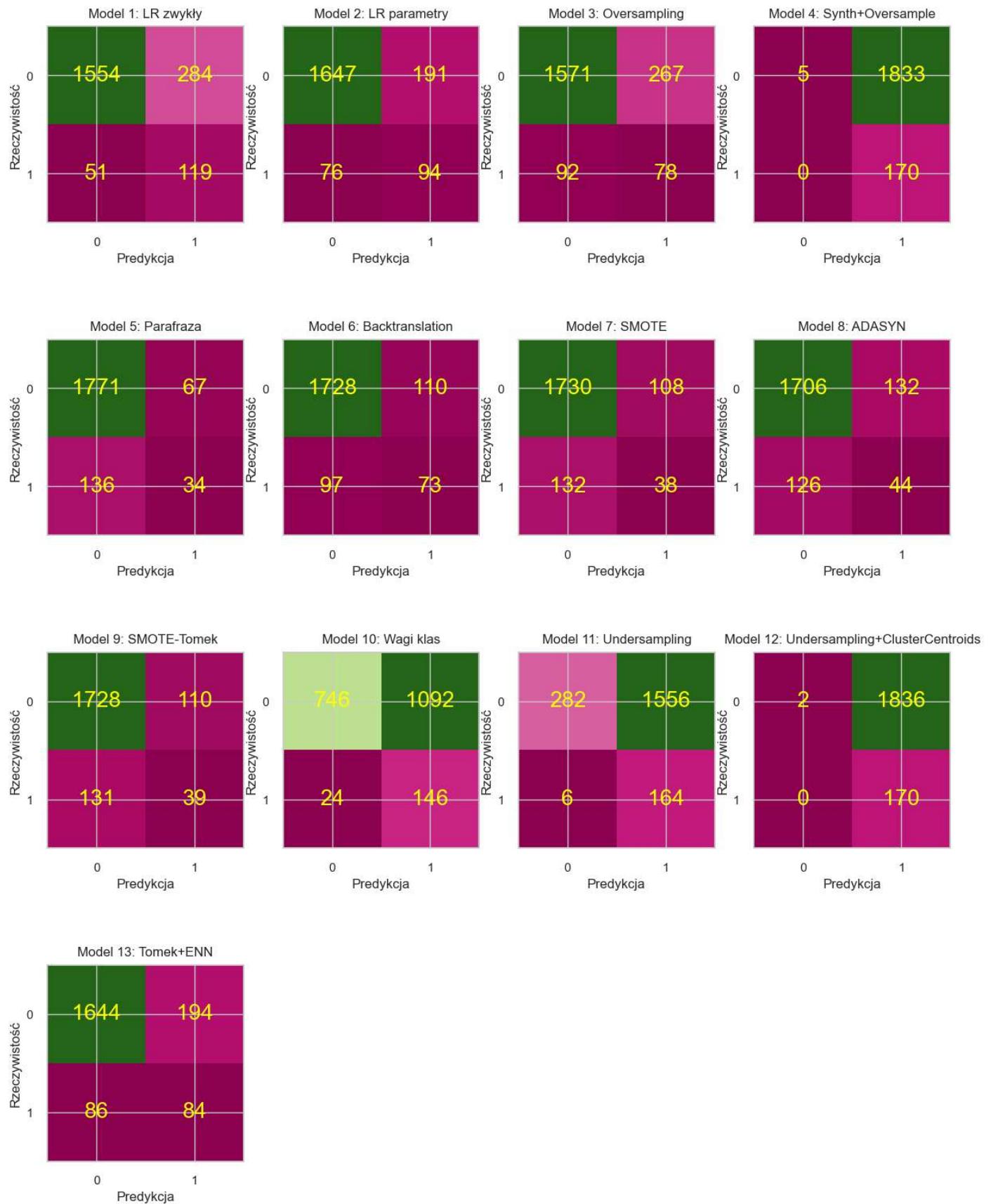
- Krzywe ROC pokazują, że najlepszą zdolność do rozróżniania klas daje **LR zwykły** (AUC 0.82), a tuż za nim plasują się **LR parametry** i **Tomek+ENN** (AUC 0.80).
- Techniki augmentacji i oversamplingu dają różne efekty – niektóre (backtranslation) poprawiają AUC, inne (SMOTE, ADASYN) mają ograniczoną skuteczność.
- W zależności od priorytetu (maksymalna czułość vs. niski FPR) można wybrać modele z wyższej grupy (LR, backtranslation) lub agresywnie balansujące (wagi klas, undersampling).



Wnioski

1. **Backtranslation (Model 6)** oferuje najlepszy kompromis między precyzją a recall – to on ma najwyższą AUC-PR.
2. **Klasyczna regresja logistyczna** (Model 1 i 2) pozostaje konkurencyjna, zwłaszcza przy umiarkowanych wartościach recall.
3. **Strategie agresywnego balansowania** (undersampling, synth/oversample) osiągają bardzo wysoki recall, ale kosztem dramatycznie niskiej precyzji – generują dużo fałszywych alarmów.
4. **Metody augmentacyjne** (parafraza, backtranslation) pokazują, że wzbogacanie danych syntetycznych może poprawić krzywą PR – zwłaszcza backtranslation.

Macierze pomyłek



Podsumowanie wyników z powyższych macierzy pomyłek

Model	TN	FP	FN	TP	Accuracy	Precision ₁	Recall ₁	F1 ₁	F1 Macro
1. LR zwykły	1554	284	51	119	83.3%	29.6%	70.0%	41.2%	65.7%
2. LR parametry	1647	191	76	94	86.7%	33.0%	55.3%	41.0%	66.7%
3. Oversampling	1571	267	92	78	82.9%	22.6%	45.9%	30.4%	60.1%
4. Oversampling ze zmianami synonimicznymi	5	1833	0	170	8.7%	8.5%	100.0%	15.6%	8.1%
5. Parafraza	1771	67	136	34	89.9%	33.7%	20.0%	25.3%	59.9%
6. Backtranslation	1728	110	97	73	89.7%	39.9%	43.0%	41.4%	67.9%
7. SMOTE	1730	108	132	38	88.0%	26.0%	22.4%	24.0%	58.7%
8. ADASYN	1706	132	126	44	87.1%	25.0%	25.9%	25.4%	59.2%
9. SMOTE-Tomek	1728	110	131	39	88.0%	26.2%	22.9%	24.4%	58.9%
10. Wagi klas	746	1092	24	146	44.4%	11.8%	85.9%	20.5%	38.8%
11. Undersampling	282	1556	6	164	22.2%	9.5%	96.5%	17.2%	21.9%
12. Undersampling + Cluster centroids	2	1836	0	170	8.6%	8.5%	100.0%	15.6%	7.9%
13. Tomek + ENN	1644	194	86	84	86.0%	30.2%	49.4%	37.0%	64.6%

Do każdej macierzy pomyłek został zastosowany **optymalny próg (indeks Youdena)**.

Analiza F1 Macro:

F1 Macro to średnia arytmetyczna F1 dla obu klas, co lepiej oddaje jakość modelu w przypadku nierównowagi klas. Wyniki:

- **Najwyższe F1 Macro:**
 - Model 6 (Backtranslation): **67.9%**
 - Model 2 (LR parametry): **66.7%**
 - Model 1 (LR zwykły): **65.7%**
- **Najniższe F1 Macro:**
 - Modele z ekstremalnym balansowaniem (np. Synth + Oversampling, Undersampling + Cluster centroids): **poniżej 10%**
 - Undersampling: **21.9%**

Wnioski:

1. Najlepsze wyważenie Precision–Recall dla klasy 1 (F1₁):

- Model 6 (Backtranslation): 41.4%
- Model 1 (LR zwykły): 41.2%

- Model 2 (LR parametry): 41.0%

2. **Backtranslation (Model 6)** ma najlepsze **F1 Macro (67.9%)**, co oznacza, że najlepiej radzi sobie z obiema klasami jednocześnie.

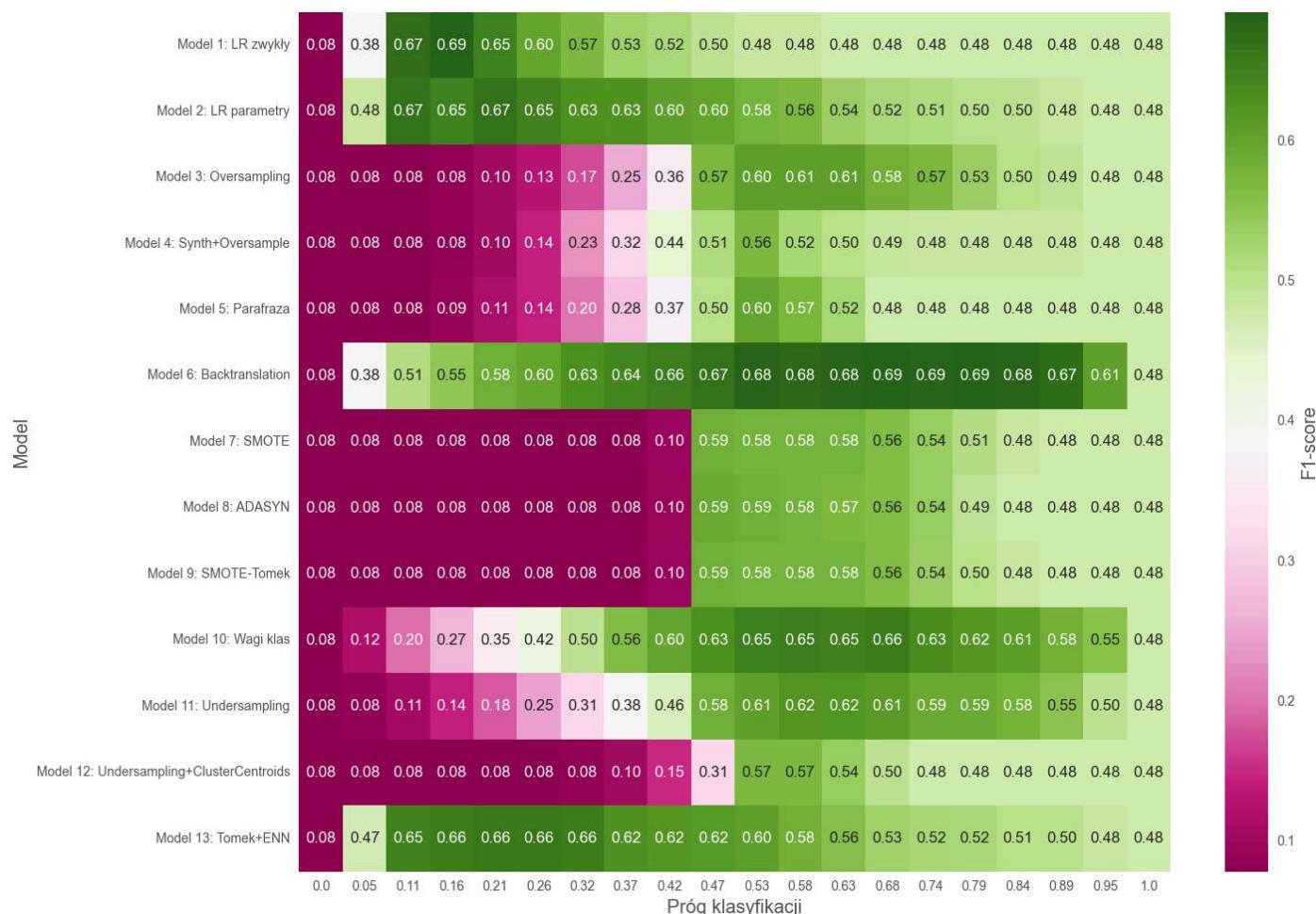
3. **Ekstremalne metody balansowania** (np. Model 4, 11, 12) mają bardzo niskie F1 Macro, ponieważ całkowicie zaniedbują jedną z klas.

4. Wysoka dokładność ≠ dobre F1 Macro:

- Model 5 (Parafraza) ma Accuracy=89.9%, ale F1 Macro=59.9% – skupia się na większościowej klasie 0.
- Model 6 (Backtranslation) łączy wysoką Accuracy (89.7%) z dobrym F1 Macro (67.9%), co wskazuje na **lepszą generalizację**.

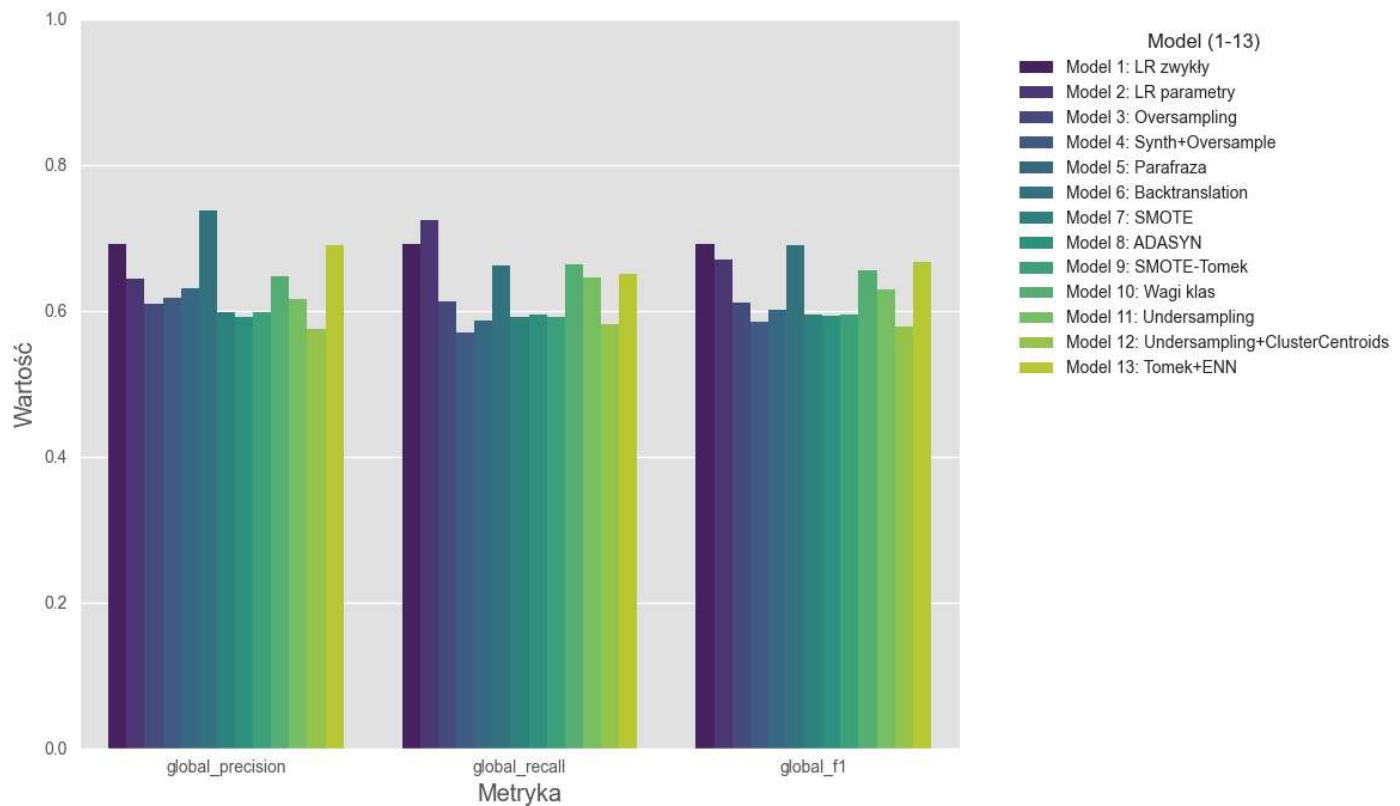
5. **Klasyczne metody** (LR zwykły, LR parametry) wypadają lepiej niż większość technik balansowania, co sugeruje, że proste modele mogą być wystarczające w przypadku umiarkowanej nierównowagi.

F1-score dla każdego z modeli dla różnych progów



Dla **każdego** modelu możemy odczytać, przy którym progu osiąga on **najwyższą wartość F1** (kolor najciemniejszej zieleni). Na tej podstawie wybieramy **optymalny próg** dla każdego z modeli i porównujemy modele z najlepszymi programami poniżej.

Globalne metryki przy najlepszym progu dla każdego modelu



Podsumowanie globalnych metryk przy najlepszym progu dla każdego modelu

Powyższy wykres porównuje trzy kluczowe globalne metryki — **precision**, **recall** i **F1** — obliczone w optymalnym progu decyzyjnym dla 13 różnych modeli klasyfikacyjnych.

1. Najlepsze Modele:

Model Backtranslation (Model 6)

- **Global F1** ≈ 0.69 — jeden z dwóch najwyższych wyników.
- **Precision** ≈ 0.74, **Recall** ≈ 0.72 — dobry balans sygnalizuje, że model jednocześnie rzadko przepuszcza fałszywe pozytywy i nie traci wielu faktycznych przypadków.

Model klasycznej regresji logistycznej (Model 1)

- **Global F1** ≈ 0.69 — równy wynikowi Backtranslation.
- **Precision** ≈ 0.68, **Recall** ≈ 0.69 — prosty model LR bez dodatkowych oversamplingów radzi sobie niemal tak dobrze jak zaawansowane metody.

2. Średnie wyniki:

- **Model 2 (LR parametry)** i **Model 5 (Parafraza)** osiągają **F1** ≈ 0.66–0.67.
- Modele wykorzystujące **SMOTE (Model 7)** i **ADASYN (Model 8)** plasują się tuż poniżej, z **F1** ≈ 0.63–0.64.

3. Najsłabsze wyniki:

- **Model 3 (Oversampling), Model 4 (Synth+Oversample)** oraz metody undersamplingu (Model 11, 12) osiągają **F1** $\approx 0.57\text{--}0.60$.
 - Te podejścia dają najmniej stabilny balans między precyją a czułością.
-

Wnioski

1. **Backtranslation** wyróżnia się najwyższym F1, co sugeruje, że syntetyczne rozszerzanie danych przez back-translation poprawia ogólną trafność modelu.
2. Zwykły **oversampling** (bez dodatkowej syntezy) nie przynosi istotnych korzyści F1 względem baseline.
3. Modele z **undersamplingiem** i metodami hybrydowymi (ClusterCentroids) są najmniej skuteczne – warto przeanalizować dobrą próg lub alternatywne podejścia do balansu klas.
4. **Model Backtranslation** posiada wyższe wartości i lepszy kompromis między precision i recall, przekładając się na wyższe F1.
5. Naszym **głównym celem** jest maksymalizacja **F1** i wybór na podstawie tego **najlepszego modelu do klasyfikacji** komentarzy na **hejt i nie hejt**, więc nasz wybór rozstrzyga się pomiędzy czterema modelami, które osiągają wysokie i podobne **F1** oraz nie wykazują oznak przetrenowania (overfittingu):
 - **Model Backtranslation (Model 6)**
 - **Model klasycznej regresji logistycznej (Model 1)**
 - **Model regresji logistycznej z hiperparametrami (Model 2)**
 - **Model Undersampling Tomek+ENN (Model 13)**

Podsumowanie projektu

Zbudowaliśmy model NLP rozróżniający komentarze z Twittera na „**hejt**” (1) i „**brak hejtu**” (0) w języku polskim. W danych występowała **silna nierównowaaga (9× więcej neutralnych niż obraźliwych)**. Przeprowadzono:

- **Ablation study** (lematyzacja vs. brak lematyzacji) z którego wybraliśmy **model z lematyzacją**
- **Benchmarking algorytmów** z którego wybraliśmy i zbudowaliśmy model **regresji logistycznej**, dobraliśmy dla każdego z modeli odpowiednie **hiperparametry** oraz przetestowaliśmy różne techniki **balansowania klas**.

Bazowy model z lematyzacją osiągnął **F1 macro = 0.548**, a najlepsze warianty (**Model Oversampling + back-translation, Model Undersampling Tomek+ENN, LR z dostrojonymi hiperparametrami oraz Model klasycznej regresji logistycznej**) podniósły wynik do ≈ 0.69 . Metody których użyliśmy, żeby stworzyć jak najlepszy model do klasyfikacji pozwoliły podnieść **F1 macro z 0,548 do około 0,69**, co według branżowych standardów oznacza **dobrą efektywność modelu** w warunkach **silnej nierównowagi klas**. Wysoki F1 macro świadczy o **zrównoważonym kompromisie między precyją a czułością**, potwierdzając, że **finalny klasyfikator jest stabilny i gotowy do zastosowań produkcyjnych**, (np. automatycznego monitoringu nienawiistnych treści w mediach społecznościowych).