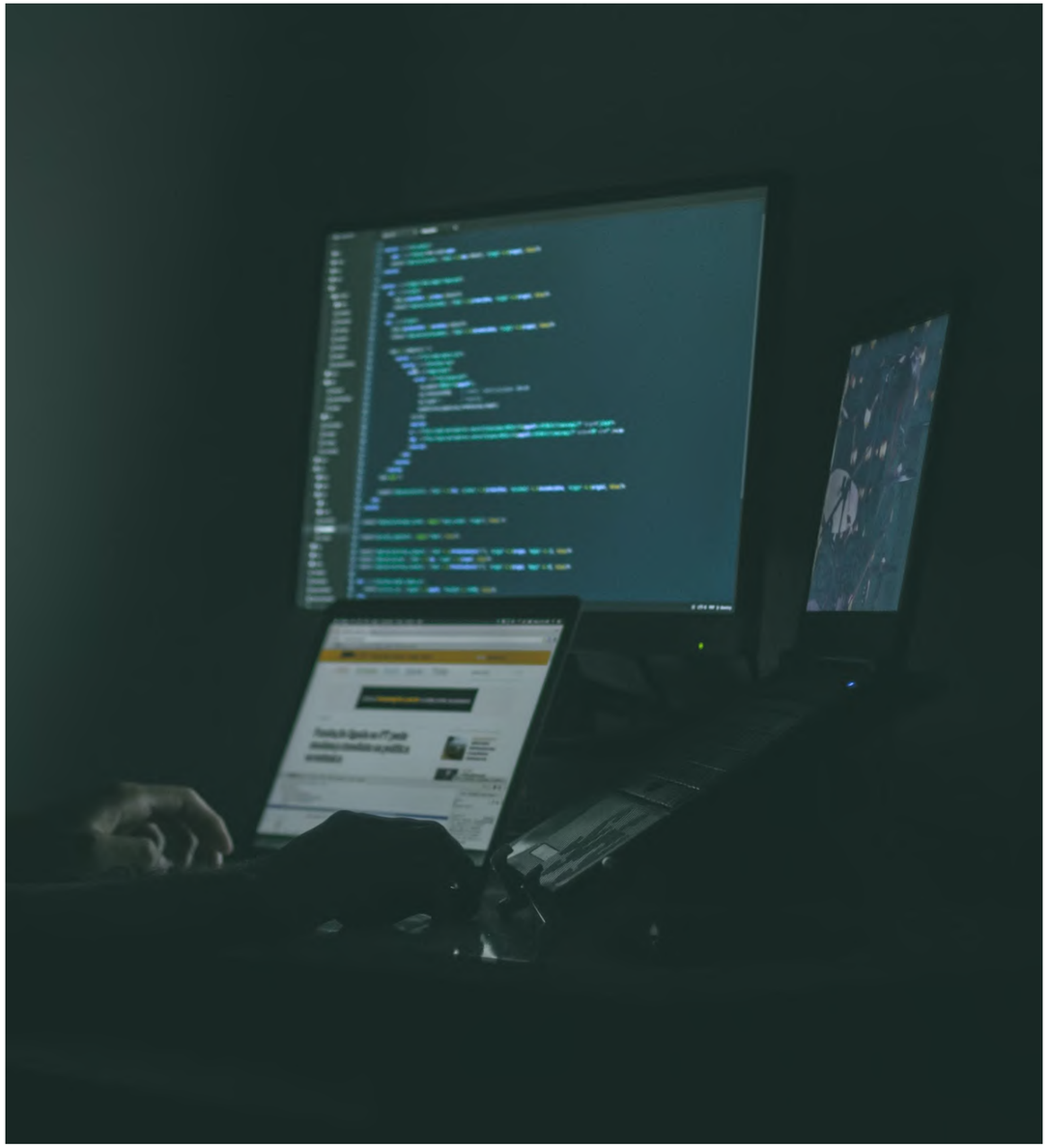


Klasyfikacja szkodliwych treści

Analiza i klasyfikacja tekstu w oparciu o metody przetwarzania języka naturalnego

Agenda

1 Wprowadzenie

2 Analiza tekstu

3 Klasyfikacja szkodliwych treści

4 Rezultaty i wnioski

Wprowadzenie

Sekcja skupia się na określeniu podjętej problematyki i opisie potencjalnego rozwiązania.
Przedstawione zostaną pokrótce wykorzystane metody i techniki oraz dostępny zbiór danych.

Opis podjętej problematyki

01

Problem

Cyberprzemoc, phishing, szkodliwe i krzywdzące treści stanowią **coraz częstszą** i niepokojącą tendencję w mediach społecznościowych oraz komunikacji online. To zjawisko stwarza poważne wyzwania, gdyż **klientom grozi utrata danych i środków finansowych.**

02

Rozwiązanie

Wykrywanie i **reagowanie** na te zagrożenia, poprzez klasyfikację szkodliwych treści, stanowi potencjalne rozwiązanie, które uchroni klientów. Przy zadaniach klasyfikacji tekstu przetwarzanie języka naturalnego (**NLP**) w połączeniu z uczeniem maszynowym (**ML**) odgrywa kluczową rolę.

03

Metody i narzędzia

W celu stworzenia skutecznego narzędzia, wykorzystano typowe techniki NLP, takie jak **n-gramy** i **TF-IDF**, oraz modele uczenia maszynowego, takie jak **SVM**, **NB** oraz **HerBERT**. Implementacja tych rozwiązań została przeprowadzona w ogólnodostępnym języku programowania **Python**.

04

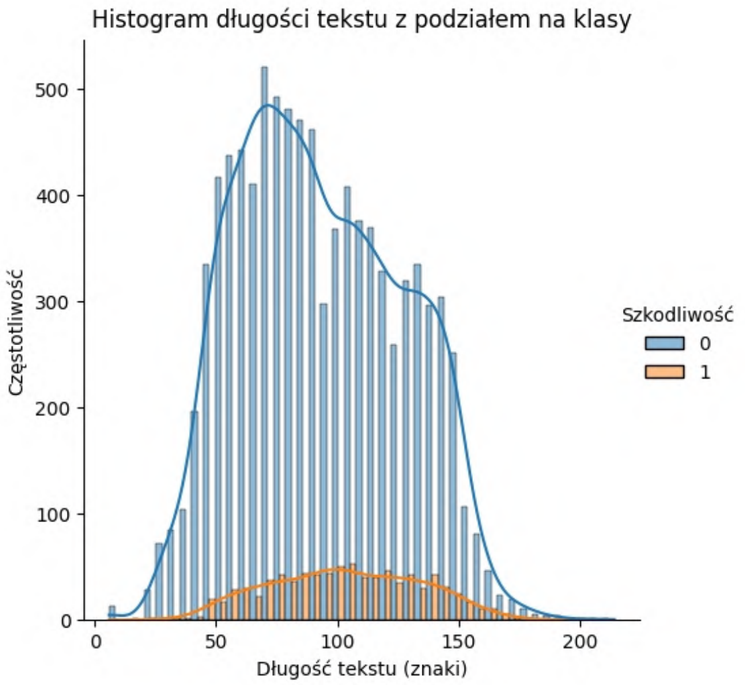
Dane

Narzędzie stworzono w oparciu o powszechny zbiór danych od kampanii ewaluacyjnej dla narzędzi **NLP PolEval 2019.** Zbiór zawiera **neutralne** oraz **szkodliwe** treści pisane przez użytkowników (tweety). Obejmują one cyberprzemoc, mowę nienawiści i powiązane zjawiska.

Analiza tekstu

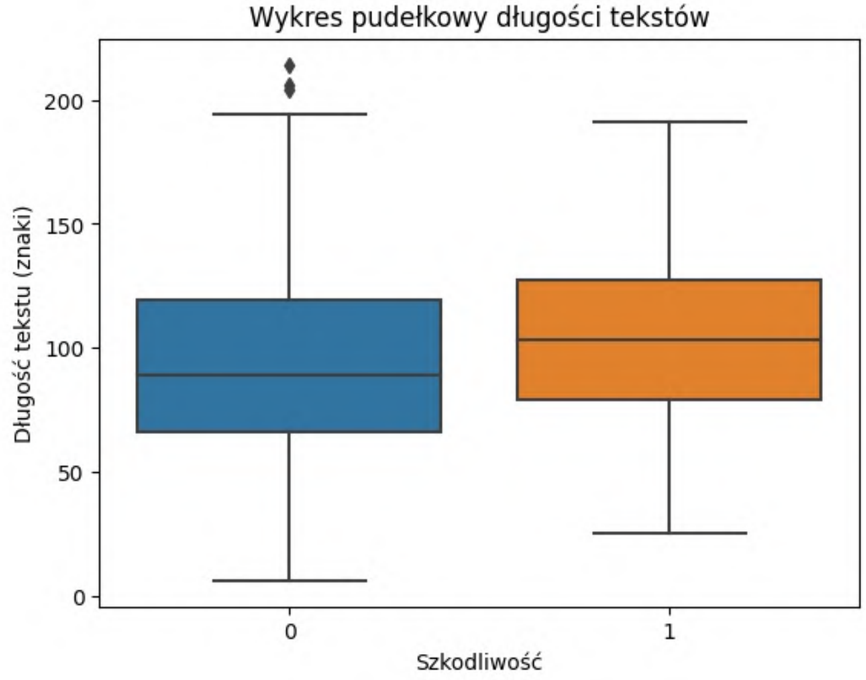
Sekcja skupia się na badaniu zawartości tweetów poprzez badanie długości sekwencji, występowania konkretnych cech w tekstach oraz analizy modeli n-gram.

Analiza długości tekstów



Zróżnicowane rozkłady

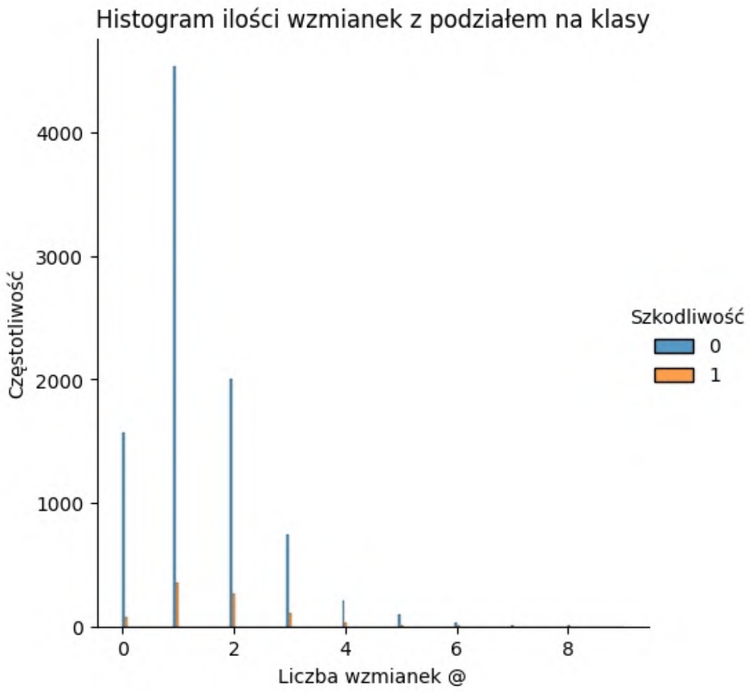
Rozkład tweetów szkodliwych jest bardziej **spłaszczony** z **mniejszą koncentracją**. Z kolei treści nieszkodliwe posiadają rozkład **skoncentrowany wysmukły**. Obydwa są względnie **symetryczne**.



Występujące wartości odstające

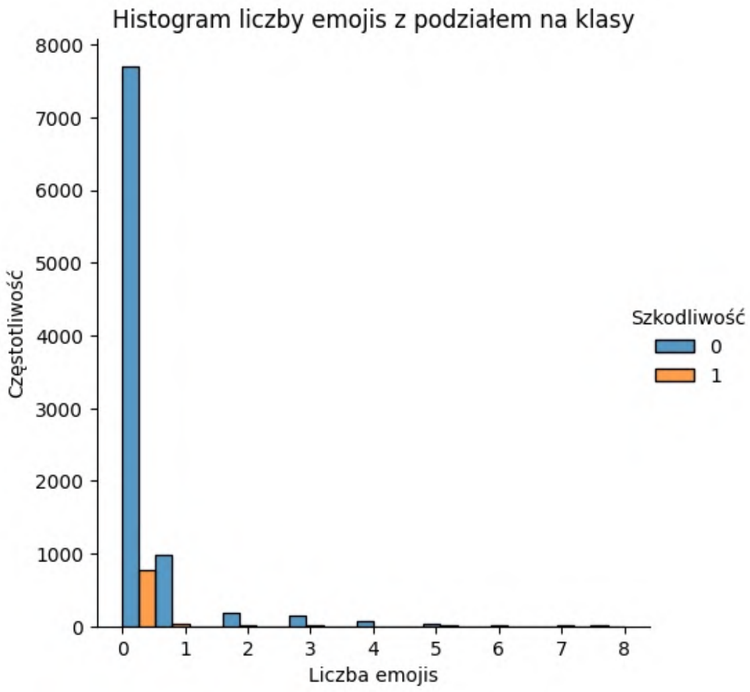
Średnia długość tekstów nieszkodliwych jest **mniejsza**, a wąsy są nieco **dłuższe** w porównaniu do szkodliwych. Ponadto można wśród nich zauważyć **obserwacje odstające**.

Analiza ukrytych cech w tekście



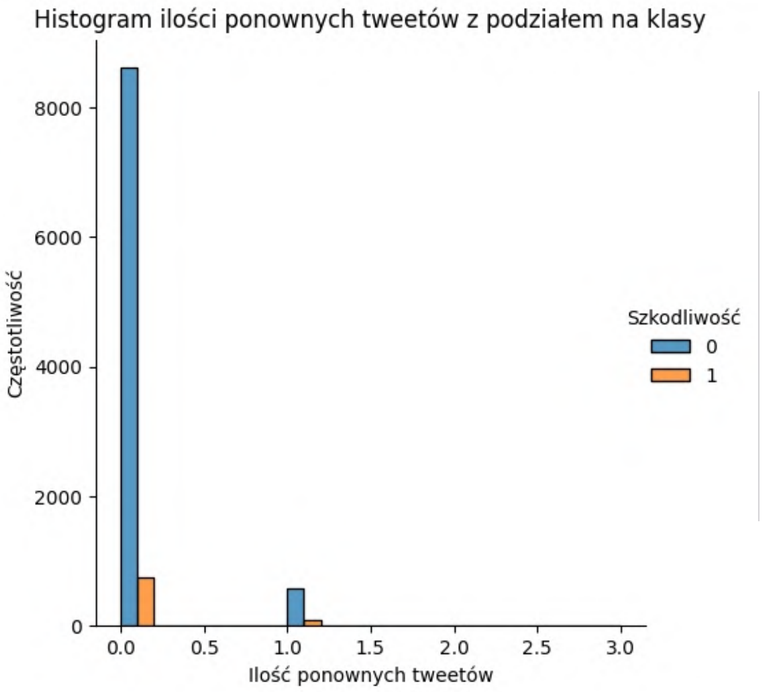
Ilość oznaczeń

Wykres częstości występowania wzmianek w tweetach **nie wykazuje** żadnej konkretnej właściwości. Różnica w częstotliwości wynika z **niezrównoważenia klas**. Występują także **obserwacje odstające.**



Ilość emotikonów

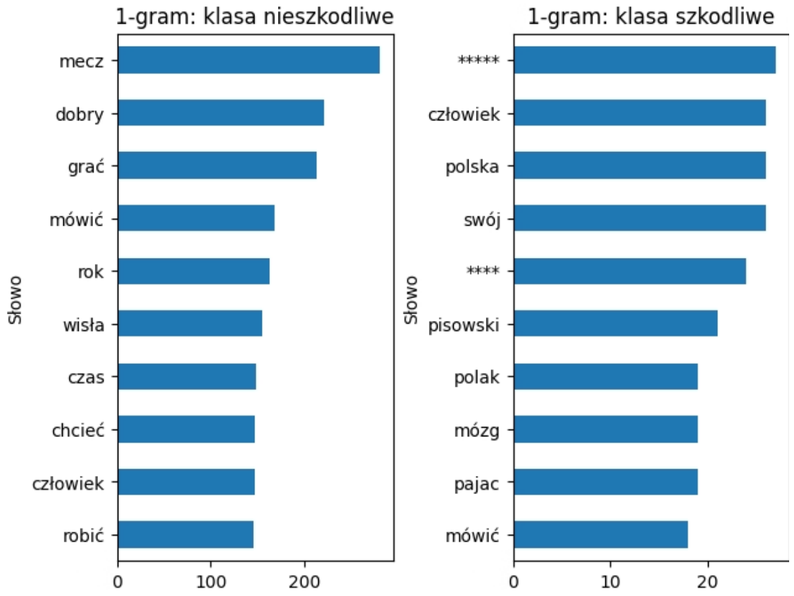
Wykres częstości występowania ikonek emoji w tekstach **ukazuje** pewną potencjalnie istotną cechę. W tekstach nieszkodliwych emotikony występują **częściej** oraz w większych ilościach. Z kolei dla tweetów szkodliwych emotikony emoji przeważnie **nie występują**.



Ilość tweetów wtórnych

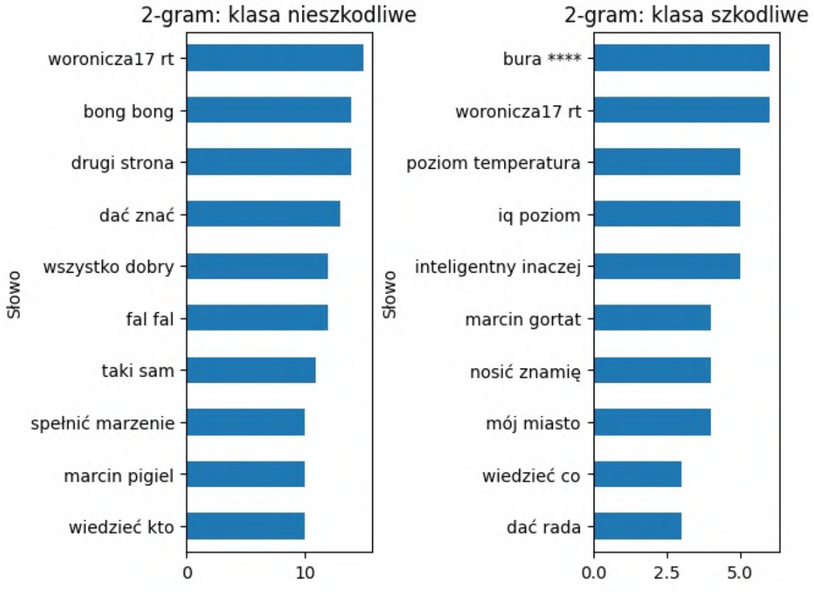
Histogram ilości ponownych tweetów z podziałem na klasy **nie implikuje** konkretnych wniosków. Obydwie klasy zawierają retweety z naturalną proporcją większościową tweetów nieszkodliwych wynikającą z **niezbalansowanego** zbioru danych.

Analiza modeli n-gram



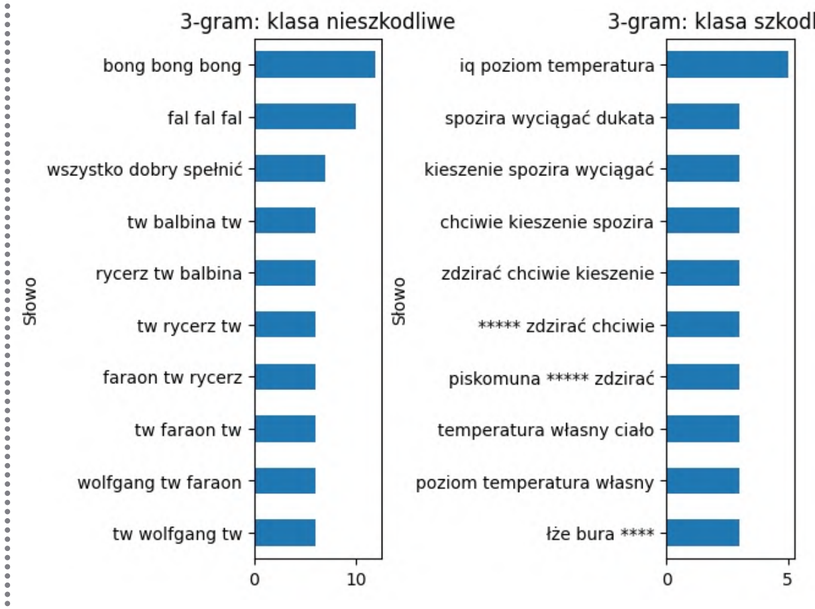
Uni-gram

W tekstach szkodliwych jednymi z najczęstszych słów są **wulgaryzmy**. Ponadto występują nawiązania do narodu oraz polityki. Występują części wspólne.



Bi-gram

Jednym z najczęstszych bigramów jest **program telewizyjny** nadawany w telewizji publicznej. W tekstach szkodliwych w dalszej części najczęstszy jest **wulgaryzm**. Występują też frazy nawiązujące do poziomu inteligencji.



Tri-gram

Brak części wspólnej. Treści szkodliwe głównie wyrażają **wulgaryzmy** oraz **negatywnie nacechowane słowa** jak chciwość. Są też nawiązania polityczne.

Klasyfikacja szkodliwych treści

Sekcja skupia się na procesie budowania modelu do klasyfikacji tekstów o negatywnym charakterze. Wykorzystane zostały metody takie jak TF-IDF, EDA, SMOTE, klasyczne modele typu Naive Bayes, SVM oraz modele uczenia głębokiego typu HerBERT. Ewaluacja odbyła się z wykorzystaniem metryk klasyfikacji (ACC, Precision, Recall, F1) oraz krzywej ROC AUC.

Wektoryzacja i nadpróbkowanie

Wektoryzacja to proces przekształcenia słów lub dokumentów na wektory liczbowe. Zastosowano metodę **TF-IDF** służącą obliczaniu ważności słów w danym dokumencie w oparciu o częstotliwość ich występowania (TF) oraz odwrotną częstotliwość ich występowania we wszystkich dokumentach w zbiorze (IDF).

Z racji niezbalansowanego zbioru danych (mała ilość tekstów szkodliwych) zastosowano **nadpróbkowanie**, które wyrównuje poziom klas poprzez dołączenie nowych obserwacji z klasy mniejszościowej. Wykorzystano metody typu **EDA** (Easy Data Augmentation) polegające na syntetycznym stworzeniu nowej próbki danej klasy poprzez proste transformacje na tekście oraz **SMOTE**, która wyrównuje poziom klas w zbiorze poprzez losowanie obserwacji z klasy mniejszościowej wraz z jego sąsiadem tworząc ich kombinację wypukłą.

```
# Set variables with features and label
X_tr = train.text
y_tr = train.label

# Initialize TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Fit and transform training features
X_tf = vectorizer.fit_transform(X_tr)

# Transform test features
X_test_res = vectorizer.transform(test.text)

# Initialize SMOTE technique at given seed
sm = SMOTE(random_state=42)

# Fit and transform training features and labels
X_res, y_res = sm.fit_resample(X_tf, y_tr)
```

```python
# Set parameters values
estimator = Pipeline([
        ('tfidf', TfidfVectorizer()),
        ('svm', SVC()),
    ])

# Specify grid with parameters
param_grid = {
    'tfidf__ngram_range': [(1, 1), (1, 2), (1, 3)],
    'tfidf__norm': ['l2'],
    'tfidf__smooth_idf': [True],
    'tfidf__sublinear_tf': [False],
    'svm__C': [0.1, 1, 10],
    'svm__kernel': ['linear', 'rbf'],
}

# Perform CV
svm_model, svm_results, grid_search = cm.perform_cross_validation(X, y, estimator, param_grid)

cv_train_results, cv_val_results = {}, {}
for k in svm_results.keys():
    metric = k.split('_')[-1]
    if 'mean_train_' in k:
        cv_train_results[metric] = np.mean(svm_results[k])
    if 'mean_test_' in k:
        cv_val_results[metric] = np.mean(svm_results[k])

print("Train CV results:\n", cv_train_results)
print("Validation CV results:\n", cv_val_results)

# Get predictions
y_pred = svm_model.predict(test.text)
y_test = test.label

# Evaluate results
scores_svm = cm.score(y_test, y_pred)
print("Test dataset results:\n", scores_svm)

# Save model
dump(svm_model, 'models/svm_model.joblib')
```

Modele klasyczne

Wytrenowane zostały dwa bazowe modele uczenia maszynowego dla zadania klasyfikacji: klasyfikator Naiwnego Bayesa (**NB**) oraz maszyna wektorów nośnych (**SVM**). Połączono przy tym **różne kombinacje** względem nadpróbkowania. Trening odbył się z **dostrojeniem hiperparametrów** modelu stosując **3-stopniową walidację krzyżową**. Po lewej widoczny fragment kodu dla treningu modelu SVM w oparciu o EDA.

Naiwny klasyfikator Bayesa to **prosty** i efektywny algorytm klasyfikacji, który opiera się na zastosowaniu **reguły Bayesa** z założeniem o **niezależności cech.** Przypisuje nową obserwację do odpowiedniej klasy na podstawie prawdopodobieństwa wystąpienia jej cech w każdej z klas.

Maszyna wektorów nośnych (SVM) to algorytm uczenia maszynowego, który służy m.in. do klasyfikacji. Wyznacza **optymalną hiperpłaszczyznę** w przestrzeni wielowymiarowej w celu jak najlepszego **oddzielenia danych** należących do różnych klas.

Modele uczenia głębokiego

Wytrenowano także model uczenia głębokiego **HerBERT**. Jest to model typu BERT czyli dwukierunkowy **enkoder** reprezentacji **transformatora**. Modele oparte na architekturze transformatorów w ostatnim czasie są modelami typu **SOTA** w dziedzinie NLP. HerBERT jest wstępnie przetrenowanym modelem na polskim korpusie tekstowym. Wyjściem modelu jest zakodowana treść dokumentów w postaci wektora nazywana **embeddingiem**. Dzięki treningowi zachowuje koncept znaczeniowy wyrazów w odróżnieniu do podstawowych technik wektoryzacji. Może służyć do **wyszukiwania semantycznego**.

Model ten **dostrojono** na dostępnych danych treści szkodliwych i neutralnych. Na warstwę wyjściową nałożono funkcję **sigmoidalną** poprzedzoną warstwą typu **dropout** w celu **regularyzacji**. Po prawej stronie znajduje się fragment kodu z treningu modelu wraz z przyjętymi wartościami współczynnika uczenia, wielkości partii i liczby epok.

```
# Set number of epochs, batch size and learning rate
EPOCHS = 5
BATCH_SIZE = 4
LR = 2e-05

# Initialize HerBERT model
model = cm.HerBertForSequenceClassification(num_classes=1, dropout_rate=0.5)
model_custom_name = 'herbert_model'

# Start model training
results, model = cm.train_model(
    model = model,
    train_data = df_train,
    val_data = df_val,
    learning_rate = LR,
    epochs = EPOCHS,
    batch_size= BATCH_SIZE,
    custom_model_name=model_custom_name
    )
results['model'] = model_custom_name

# Evaluate model on test dataset
test_res = cm.evaluate(
    model = model,
    test_data = df_test
    )
test_res['model'] = model_custom_name

# Save model
checkpoint = {
'model': cm.HerBertForSequenceClassification(num_classes=1, dropout_rate=0.5),
'state_dict': model.state_dict()
}
torch.save(checkpoint, 'models/' + model_custom_name + '.pth')
```

Rezultaty i wnioski

Sekcja skupia się na zaprezentowaniu uzyskanych rezultatów oraz wyciągniętych wniosków, w tym biznesowych.

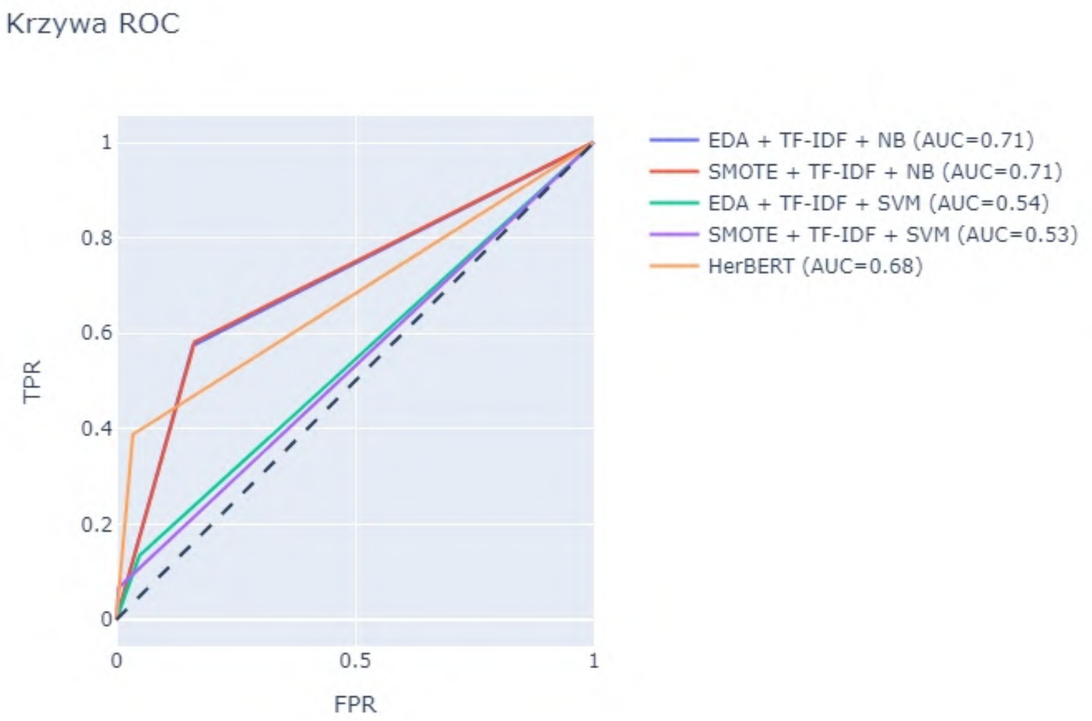
Rezultaty i rekomendacja

Najlepsze rezultaty osiągnęły modele **NB** w obu kombinacjach nadpróbkowania oraz **HerBERT**. W przypadku NB, preferowany jest wybór techniki **EDA** z racji na **prostotę** i **interpretowalność** syntetycznie wygenerowanych nowych próbek.

HerBERT osiągnął **najwyższą wartość wyniku F1** (będącą główną metryką oceny w kompetencji PolEval 2019). W porównaniu do modelu **NB ma wyższą wartość precyzji**, ale **niższą wartość recall**. Oznacza to, że **NB skuteczniej odnajduje treści szkodliwe kosztem precyzji**, a **HerBERT wykrywa te treści precyzyjniej, ale w mniejszej ilości**. Model **NB** uzyskał także **wyższy wynik AUC**.

Wychodząc z założenia, że **cenniejsze jest wykrycie treści szkodliwych** kosztem precyzji, zalecany jest wybór modelu NB. Ponadto, wyniki modelu **łatwo uzasadnić**, gdyż opiera się on na jednym z podstawowych twierdzeń rachunku prawdopodobieństwa.

Model	ACC	Precision	Recall	F1
EDA + TF-IDF + NB	**0.793**	**0.339**	**0.575**	**0.427**
SMOTE + TF-IDF + NB	0.791	0.338	0.582	0.427
EDA + TF-IDF + SVM	0.842	0.3	0.134	0.186
SMOTE + TF-IDF + SVM	0.871	0.692	0.067	0.122
HerBERT	**0.888**	**0.634**	**0.388**	**0.481**



Wnioski

SVM potrzebuje syntetycznych próbek wysokiej jakości

SVM znany jest z dużej skuteczności w przypadku danych wielowymiarowych oraz pracy na niewielkich próbkach jak w tym przypadku (za sprawą **sztuczki jądra**). Mimo to, w eksperymencie osiągnął **najgorsze wyniki**, najpewniej za sprawą nadpróbkowania. Zbiór danych wymagał wytworzenia sporej ilości sztucznych obserwacji co wprowadziło **szum** w przestrzeni wielowymiarowej cech i **nadmierne dopasowanie** modelu do danych treningowych (**over-fitting**).

HerBERT potrafi generalizować już na niewielkiej próbce

HerBERT jest modelem wstępnie przetrenowanym na polskim korpusie danych tekstowych. Wymaga jedynie **dostrojenia** na nowych rozpatrywanych danych, a eksperyment pokazuje, że **nie jest wymagana ich olbrzymia ilość**. Co więcej, **nie muszą być one najlepszej jakości** (jak dla SVM), gdyż w przypadku modeli DL cechy z danych są **wydobywane na etapie uczenia**. Sytuacja taka występuje jednak, gdy nowe dane są podobnej natury co dane, na których został wytrenowany.

Klasyfikator NB nie jest zależny od metody nadpróbkowania

Klasyfikator Naiwnego Bayesa osiągnął takie same wyniki dla obu metod nadpróbkowania co nasuwa wniosek, iż **nie mają one istotnego wpływu na uzyskane rezultaty** przy użyciu tego modelu. W obu technikach przestrzeń wektorowa cech wejściowych została uzupełniona o **relatywnie zbliżone próbki syntetyczne** (SMOTE tworzy kombinacje, a EDA składa się przetransformowanych danych początkowych) co może wyjaśniać tą sytuację. Niemniej jednak w celu potwierdzenia należałoby **przeprowadzić eksperyment na szerszej ilości technik** balansowania danych.

Zastosowania biznesowe

01

Skuteczne zabezpieczenie klientów

Wykorzystanie modeli NLP w połączeniu z uczeniem maszynowym, takich jak Naiwny Bayes i HerBERT, umożliwia instytucjom **efektywne wykrywanie szkodliwych treści**. Przeprowadzony eksperyment można **rozszerzyć o nowe dane** w celu **wykrywania cyberprzemocy i phishingu**, co pozwoli na **szybkie reagowanie** na zagrożenia i skuteczne **zabezpieczenie użytkowników** przed utratą danych i środków finansowych.

02

Minimalizacja ryzyka reputacyjnego

Klasyfikacja szkodliwych treści przy użyciu zaawansowanych technik NLP i modeli ML pozwala instytucjom na **monitorowanie i identyfikację** potencjalnych **zagrożeń dla ich reputacji**. Wcześniejsze wykrycie i skuteczna reakcja na negatywne treści (przykładowo w mediach społecznościowych) pozwoli **minimalizować ryzyko szkód dla wizerunku.**

03

Doskonalenie interakcji z użytkownikami

Wykorzystanie modeli klasyfikacyjnych pozwala na **analizę treści generowanych przez użytkowników**, co może dostarczyć cennych informacji o ich preferencjach i potrzebach. Instytucje mogą wykorzystać te dane do **personalizacji usług**, **zwiększenia zaangażowania** użytkowników oraz **poprawy jakości obsługi**, co może przyczynić się do wzmocnienia relacji z użytkownikami i budowy lojalności do marki.

Referencje

1. Kod dostępny w repozytorium: https://github.com/szpwski/nlp_poleval_text_classification
2. Zbiór danych PolEval 2019 z zadania 6. klasyfikacji tekstu. http://2019.poleval.pl/index.php/tasks/task6
3. Ogrodniczuk M, Kobyliński Ł.: *Proceedings of the PolEval 2019 Workshop*, Instytut Podstaw Informatyki PAN (2019).
4. Padurariu C, Breaban M. E.: *Dealing with Data Imbalance in Text Classification*, Procedia Computer Science (2019).
5. Wei J, Zou K.: *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*, Association for Computational Linguistics (2019).
6. Murphy K.: *Probabilistic Machine Learning: An Introduction*, The MIT Press (2022).
7. Mroczkowski R, Rybak P, Wróblewska A, Gawlik I.: *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*, Association for Computational Linguistics (2021).