

# NLP

## Przetwarzanie języka naturalnego Dokumentacja końcowa

Paulina Podpora (303750)

Igor Kraszewski (310164)

Patrycja Wysocka (306022)

Styczeń 2022

### 1 Definicja problemu

Celem projektu jest pochylenie się nad problemem klasyfikacji postów zamieszczanych na platformie Twitter za pomocą metod NLP. Główna analiza przeprowadzona zostanie na zbiorze zawierającym posty dotyczące zdarzeń czy też wypadków, gdzie celem będzie nauczenie algorytmu wykrywania informacji o rzeczywistych katastrofach. W tym celu wykonana zostanie podstawowa eksploracyjna analiza danych razem z ich wizualizacją, aby zapoznać się z charakterystyką wpisów. Po przetwarzaniu danych tekstowych stworzone zostaną modele predyktacyjne na podstawie kilku wybranych algorytmów a następnie wyniki działania przygotowanych modeli zostaną porównane przy zastosowaniu odpowiednich metryk ewaluacyjnych. Zbiory danych na których będzie wykonywany projekt to:

- Suspicious Communication on Social Platforms(1)
- Depression: Twitter Dataset(2)
- Disaster Tweets(3)

### 2 Studia literaturowe

W czasach powszechnego dostępu do mediów społecznościowych analiza umieszczanych w nich przez szerokie grono użytkowników treści jest coraz częściej wybieraną przez badaczy metodą poszukiwania odpowiedzi na najistotniejsze pytania. Opracowanie zagadnień takich jak analiza sentymentów(5) użytkowników wokół spraw jak epidemia Covid-19(6) lub postrzeganie społeczności LGBT(8), przebieg katastrof na podstawie postów użytkowników(7) czy wykrywanie chorób takich jak anemia(9) lub zaburzenia odżywiania to tylko niektóre z całego morza zastosowań, jakie mogą mieć takie analizy.

Platformą najczęściej wybieraną przez badaczy do takich analiz jest Twitter. Wśród najważniejszych powodów tych wyborów możemy umieścić powszechność tego medium zarówno wśród zwykłych użytkowników, jak i szeroko rozumianych uczestników życia publicznego, takich jak dziennikarze, politycy czy najwyższej rangi agencje czy instytucje, dla których jest podstawową platformą społecznościową, na której zamieszczają najważniejsze komunikaty i komentarze. Ta specyfika sprawia, że Twitter jest bardzo czuły na ważne wydarzenia w życiu społecznym, które w czasie niemal rzeczywistym generują wysyp wpisów milionów użytkowników. Jak wynika z

danych z lutego 2021 roku, na platformie zbierane są dane pochodzące od ponad 330 milionów użytkowników generujących łącznie ponad bilion postów dziennie(4).

Operacyjnalizacja większości wymienionych wcześniej zagadnień sprowadza się do rozważenia odpowiedniego problemu klasyfikacji(10)(11). Nic więc dziwnego, że najczęściej testowanymi i porównywanymi przy tych problemach algorytmami są klasyfikatory takie jak Naiwny Bayes, lasy losowe, drzewa decyzyjne, SVM i regresja logistyczna(12).

Tu jednak badacze często stają przed typowym problemem, który w realiach postów ograniczonych przez 280 znaków i często językiem potocznym, niepoprawnych gramatycznie, pisanych z użyciem skrótów i emotikonów staje się jeszcze wyraźniejszy. Problemem wstępnej eksploracji danych często jest główną przeszkodą, z którą muszą sobie poradzić. Tradycyjne metody dobrze działające dla danych takich jak artykuły z gazet często w tym specyficznyim środowisku cechują się bardzo słabą skutecznością, dlatego niezbędnym stało się dostosowanie ich do warunków Twittera(14). Tak powstał na przykład jeden z popularniejszych pakietów - TwitterNLP(13).

Powyższą konstrukcję dobrze widać na przykładzie problemu detekcji katastrof. Analiza wpisów ma odpowiedzieć na binarnie postawione pytanie o to, czy użytkownicy piszą o prawdziwej katastrofie czy nie (kiedy mogą na przykład używać języka metaforycznego lub po prostu mówić od rzeczy bądź roszciewać fake newsy). Jest to istotny problem, bo skuteczny algorytm wykrywania wpisów o odpowiednim charakterze mógłby pomóc w szybszej reakcji odpowiednich służb(15). Podchodzić do niego można skorzystać z różnych binarnych klasyfikatorów, ale zawsze taką analizę należałoby zacząć od solidnego, dostosowanego do zadania preprocessingu. Zbudowanie odpowiedniego modelu dotyczącego problemu detekcji katastrof będzie głównym zadaniem naszego projektu.

### 3 Opis rozwiązania/algorytmu i implementacja

#### 3.1 Preprocessing

W zadaniu wykonano następujące kroki w ramach preprocessingu:

- Usunięcie linków
- Usunięcie referencji do innych użytkowników
- Usunięcie hashtagów
- Usunięcie znaków interpunkcyjnych
- Usunięcie wielkich liter z tekstu
- Translacja i ekstrakcja emotikonów
- Analiza sentymentu\*
- Spellcheck
- Lemmatyzacja
- Usunięcie stopwords

\*Analiza sentymentu zostaje zrobiona w tym etapie, ponieważ biblioteka TextBlob, której będziemy używać nie potrzebuje aż tak dokładnie przeprocessowanych danych.

##### 3.1.1 Usunięcie linków

Z każdego tweeta zostały usunięte wszelkie linki pojawiające się w nim, ale żeby nie tracić informacji jaką dany link ze sobą niesie, do ramki danych zostały dodane dwie nowe kolumny: jedna z linkiem usuniętym z tweeta, a druga informująca ile linków było zawartych w tweecie.

### **3.1.2 Usunięcie referencji do innych użytkowników**

Jeżeli w danych tweecie będzie odniesienie do innego użytkownika, bądź będzie to reweet, zostanie to usunięte z samego wpisu. Aby nie tracić tych informacji zostaną stworzone nowe kolumny: jedna z informacją czy dany wpis jest retweetem, a druga z informacją do kogo odnosił się dany wpis, kogo autor oznaczył.

### **3.1.3 Usunięcie hashtagów**

Z każdego tweeta hashtagi zostały usunięte, został jedynie sam ich tekst (który potem może być chociażby sprawdzone spellcheckiem, czy zmieniony lematyzacją), natomiast sama informacją jaką niósł hashtag została też zapisana w osobnej kolumnie.

### **3.1.4 Usunięcie znaków interpunkcyjnych**

Z każdego tweetu zostały usunięte wszelkie znaki interpunkcyjne.

### **3.1.5 Usunięcie wielkich liter z tekstu**

Każda wielka litera w tweecie została skonwertowana na małą.

### **3.1.6 Translacja i ekstrakcja emotikonów**

Z każdego wpisu emotikony zostały przetłumaczone korzystając z biblioteki emoji. Np. dana emotka prezentująca śmiech zostanie przetłumaczona na 'laughing on the floor'. Aby nie tracić tych danych emotki znajdujące się w tekście zostaną zapisane do kolumny.

### **3.1.7 Analiza sentymenu**

Przy pomocy biblioteki TextBlob została przeprowadzona analiza sentymenu danego wpisu. Zostały dodane nowe kolumny: polarity - wskaźnik określający wydźwięk tweetu w skali od -1 do 1. Wartości w przedziale [-1, 0] są odbierane jako negatywne (im bliżej -1 tym bardziej negatywny), 0 jako neutralne, (0; 1] jako pozytywne. Kolejną kolumną jaka powstała jest sentiment, w którym będzie słowne określenie, czy tweet był: pozytywny, neutralny czy negatywny. Będzie ona użyta do statystyk. Został dodany także subjectivity - wskaźnik określający jak bardzo subiektywny był tweet w przedziale [0;1].

### **3.1.8 Spellcheck**

Przy pomocy słowników z biblioteki spylls.hunspell każde słowo zostało sprawdzone pod względem poprawności zapisu. Jeżeli słowo nie znalazło się w słowniku było ono zamieniane na poprawne wedle sugestii biblioteki. W przypadku gdy biblioteka nie będzie w stanie poradzić sobie z danym słowem to zostaje ono w tweecie, a na stdout jest ono wypisane, tak, żebyśmy mieli kontrolę nad tym co się dzieje w trakcie działania programu.

### **3.1.9 Lemmatyzacja**

Przy pomocy word\_tokenize z biblioteki NLTK oraz WordNetLemmatizer została przeprowadzona lematyzacja słów zawartych w tweetach.

### **3.1.10 Usunięcie stopwords**

Przy pomocy korpusu z biblioteki NLTK tweety zostały wyczyszczone ze wszelkich stopwords. Rozwiążanie będzie polegało na porównaniu kilku algorytmów do klasyfikacji danych tekstowych z Twitter'a. Rozwiązywany problem będzie problemem klasyfikacji binarnej.

## **3.2 Algorytmy**

W zadaniu porównane zostaną algorytmy:

- Regresja logistyczna
- Maszyna wektorów nośnych (SVM)
- Naiwny wielomianowy klasyfikator Bayesa
- K najbliższych sąsiadów
- Drzewo decyzyjne
- Las losowy

Użyjemy implementacji powyższych algorytmów z wykorzystaniem biblioteki Python'a - scikit-learn.

### **3.2.1 Regresja logistyczna**

Jest to model regresji stosowany gdy zmienna może przyjmować tylko dwie wartości. Opiera się ona na specyficznym sposobie wyrażania prawdopodobieństwa, które nazywane jest szansą – stosunek prawdopodobieństwa sukcesu do prawdopodobieństwa porażki. Sam model regresji logistycznej po prostu modeluje prawdopodobieństwo wyniku na podstawie danych wejściowych – nie przeprowadza klasyfikacji statystycznej (nie jest klasyfikatorem) Natomiast można go użyć do stworzenia klasyfikatora poprzez wybranie odpowiedniej granicy i klasyfikując dane wejściowe z prawdopodobieństwem większym niż granica jako jedna klasa, poniżej granicy jako druga; jest to powszechny sposób tworzenia klasyfikatora binarnego.

### **3.2.2 Maszyna wektorów nośnych (SVM)**

Jest to nieprobabilistyczny, liniowy klasyfikator binarny. Dodatkowo dzięki użyciu "kernel trick" może skutecznie działać w przypadku klasyfikacji nieliniowej. Działa na zasadzie wyznaczenia w czasie uczenia hiperpłaszczyzny rozdzielającej, z jak największym marginesem, dwie klasy.

### **3.2.3 Naiwny wielomianowy klasyfikator Bayesa**

Klasyfikator probabilistyczny oparty na założeniu niezależności atrybutów. Jego model prawdopodobieństwa jest modelem warunkowym, opartym o twierdzenie Bayesa. Model wielomianowy opisuje prawdopodobieństwo liczby wystąpień poszczególnych n-gramów w zaobserwowanym dokumencie.

### **3.2.4 K najbliższych sąsiadów**

Algorytm działający na zasadzie porównania argumentów danej obserwacji z wartościami tych argumentów w zbiorze treningowym, znalezienia k najbliższych tej obserwacji sąsiadów w zbiorze uczącym (k jest hiperparametrem algorytmu), a następnie uśrednienia wartości wyjścia tych sąsiadów w celu predykcji wyjścia dla analizowanej obserwacji. Algorytm sprawdza się szczególnie gdy zależność między argumentami, a wyjściem jest złożona.

### 3.2.5 Drzewo decyzyjne

Jest to wszechstronny algorytm uczenia maszynowego nadający się zarówno do klasyfikacji jak i regresji. Algorytm jest w stanie dopasować bardzo złożone zestawy danych. Działa on na zasadzie wyodrębniania wiedzy z zestawu przykładów. Drzewo składa się z węzłów będącymi poszczególnymi atrybutami, gałęzi będących wartościami tych atrybutów i liści tworzących poszczególne decyzje.

### 3.2.6 Las losowy

Algorytm lasu losowego jest zespołowym algorytmem uczenia maszynowego - polega na konstruowaniu wielu drzew decyzyjnych podczas uczenia i generowania na wyjściu klasy, która najczęściej występuje dla poszczególnych drzew. Algorytm ten poprawia tendencje drzew decyzyjnych do nadmiarowego dopasowywania się do danych treningowych. Każde drzewo trenowane jest na losowym podzbiorze cech (atrybutów).

### 3.2.7 Implementacja

Po wykonaniu preporcessingu wyciągnięto z danych kolumny "text", "target", "links", "is\_retweet", "references", "emojis", "hashtags", "polarity", "subjectivity", a dla zbioru **Mental Health** dodatkowo "followers", "friends", "favourites", "retweets". Elementy z kolumny "hashtags" i "emojis" były listą słów, a więc zamienione zostały one na słowa. Kolumny "links" i "references" zostały zamienione za zliczenia ilości linków oraz referencji do innych osób. Kolumna polarity przyjmowała wartości ujemne co powodowało brak możliwości użycia algorytmu MultinomialNB, a więc jej wartości zostały przeskalowane przy pomocy MinMaxScaler'a do wartości z przedziału (0, 1). Przeskalowano również wartości z reszty kolumn, które były liczbowe. Następnie wartości z kolumn "text", "hashtags" i "emojis" wykonano zamienienie danych w macierz częstości wystąpień termów dla termów o liczbie wystąpień większej niż 5 przy pomocy funkcji sklearn TfidfVectorizer i ColumnTransformer. Następnie została ona połączona z wspomnianymi wcześniej kolumnami, a następnie dane zostały podzielone na zbiory testowe i treningowe (wykonane w funkcji "read\_df()").

Do trenowania modeli została stworzona klasa "ModelsTraining" z metodami "grid\_search\_fit()" oraz "train\_models()". Pierwsza z nich odpowiada za wytrenowanie pojedyńczego modelu przy pomocy funkcji GridSearchCV dla odpowiedniego algorytmu i podanych hiperparametrów, które przechowywane są w słowniku, a następnie zapisanie go przy pomocy biblioteki pickle do formatu .pkl. Druga z tych metod wykorzystuje pierwszą do wytrenowania modeli dla wszystkich algorytmów.

Została również stworzona klasa "ModelsPredict" mająca metody "predict\_train()" oraz "predict\_test()", które wykonują przidykcje dla danego modelu na odpowiednim zbiorze.

Klasa "ModelsEval" posiada metodę "eval\_metrics()", która wylicza odpowiednie statystyki dla predykcji, a także metodę "conf\_matrix()" służącą do tworzenia macierzy pomyłek.

Ostatnią klasą jest klasa "ModelsComparison", która łączy funkcjonalności pozostałych klas implementując metodę "train\_all()", która pozwala na wytrenowanie wszystkich modeli, oraz metodę "compare()", która dla każdego stworzonego modelu dokonuje predykcji i wypisuje na terminal wyniki dla zbioru treningowego oraz testowego.

Skrypt "main.py" uruchamia całość programu. W pętli po nazwach zbiorów danych wykonywane ich wczytywanie i przerabianie za pomocą funkcji "read\_df()", a następnie dane są trenowane i porównywane.

## 4 Technologie

Technologie użyte przez zespół to:

- język programowania - Python 3.10.\*:
  - pandas
  - NLTK
  - TextBlob
  - emoji
  - spylls.hunspell
  - WordNetLemmtizer
  - re
  - matplotlib
  - seaborn
  - PyTorch
  - scikit-learn
  - numpy
- Eksperymenty, wykresy - Jupyter Notebook
- System kontorli wersji - GitHub

## 5 Instrukcja obsługi

### 5.1 Pobranie bibliotek

Przed uruchomieniem programu należy upewnić się, że posiada się wszystkie potrzebne biblioteki, które znajdują się w pliku `requirements.txt`, a także pobrać niezbędne pakiety do biblioteki `nltk`, które znajdują się w pliku `script.sh`. Należy więc w konsoli wywołać kolejno:

```
pip install -r requirements.txt  
./script.sh
```

### 5.2 Preprocessing

W ramach preprocessingu została utworzona klasa 'Preprocess' odpowiedzialna za wszystko. Są w niej podane jako zmienne globalne ścieżki do plików `.csv`, z których są pobierane dane. Aby przeprocesować ramki należy utworzyć instancję klasy `preprocess`, jako argument podać ścieżkę do pliku (bądź zmienną globalną), a następnie wywołać metodę `preprocess`. Automatycznie ona na koniec zapisze obrobione ramki danych do pliku `.json`. Przykład:

```
mental_health = Preprocess(MENTAL_HELATH)  
mental_health.preprocess()
```

Przykłady wywołania preprocessingu znajdują się w pliku `preprocessing_notebook.ipynb`

### 5.3 Modele

Po pobraniu przepreprocessowanych danych i umieszczeniu ich w folderze `/dane` obok skryptów Python'a należy pobrać przetrenowane modele i rozpakować je do folderu `/modele`, który również powinien znaleźć się w tym samym miejscu co reszta. W przypadku chęci przetrenowania modeli ponownie wystarczy pozbyć się modeli z folderu `/modele`, a uruchomiony skrypt przetrenuje je na nowo i zapisze w postaci pliku `.pkl`. Mając modele w wspomnianym folderze skrypt pominie etap trenowania i wyłącznie wykona predykcje na zbiorach treningowym i testowym, a następnie dokona ewaluacji przez podanie odpowiednich miar oraz stworzenie macierzy pomyłek. Elementy te oprócz pojawienniu się w terminalu oraz na ekranie, zapiszą się również do odpowiednich dla nich folderów (należy stworzyć folder `/scores` na pliki tekstowe z wartościami miar dla przetestowanych modeli).

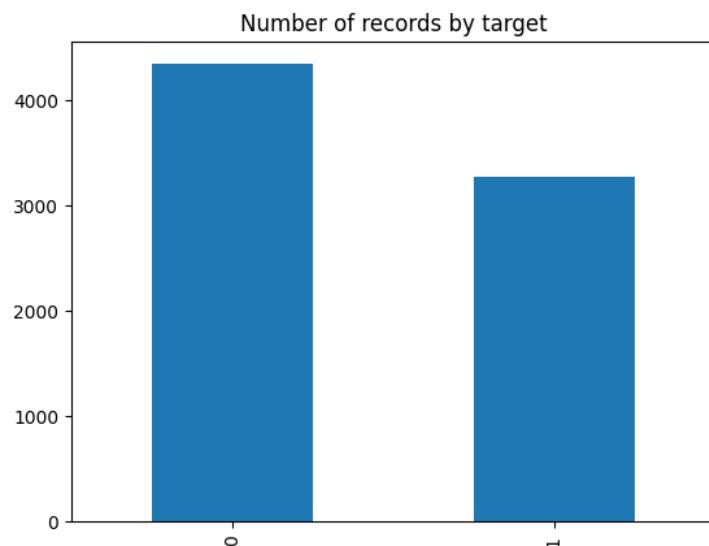
Aby uruchomić cały program wystarczy uruchomić skrypt "main.py", który wykona wszystkie czynności dla naszych danych.

## 6 Testy

### 6.1 Disaster Tweets

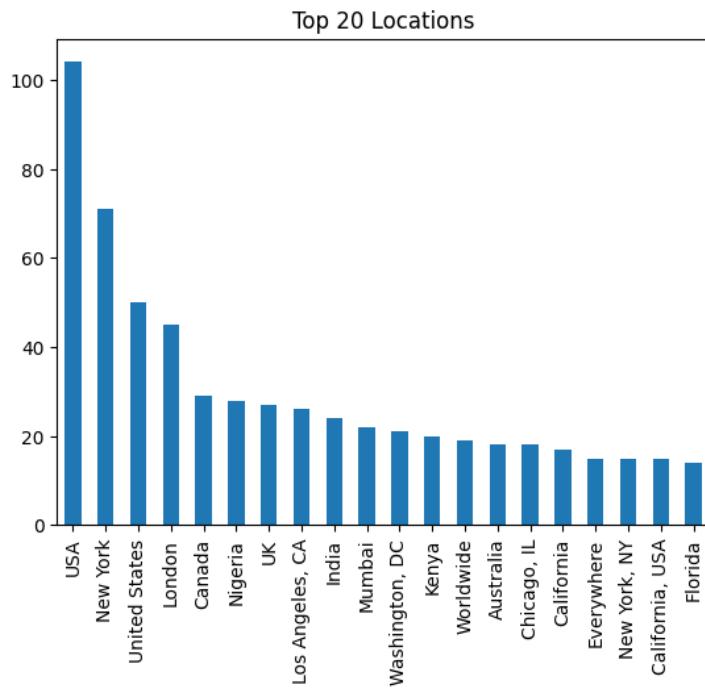
#### 6.1.1 EDA - Exploratory Data Analysis

Przed utworzeniem modeli, wykonano analizy przygotowanych po preprocessingu danych. Zbiór "Disaster Tweets" stworzony został do przygotowywania modeli klasyfikujących czy dany post jest informacją o rzeczywistej katastrofie czy też tylko używa słownictwa z takimi kojarzonego. Zbiór zawiera tekst posta, informację o lokalizacji z której został opublikowany oraz słowa kluczowe, przez które wpis został zakwalifikowany do możliwie informujących o rzeczywistych wydarzeniach. Kolumna `target` informuje o przypisaniu wpisu do jednej z grup, gdzie 1 oznacza informację o katastrofie a 0 jej brak. Zauważono, że wpisów pierwszego typu jest w zbiorze znaczaco mniej (Rysunek 1).



Rysunek 1: Ilość wpisów w zależności od klasyfikacji

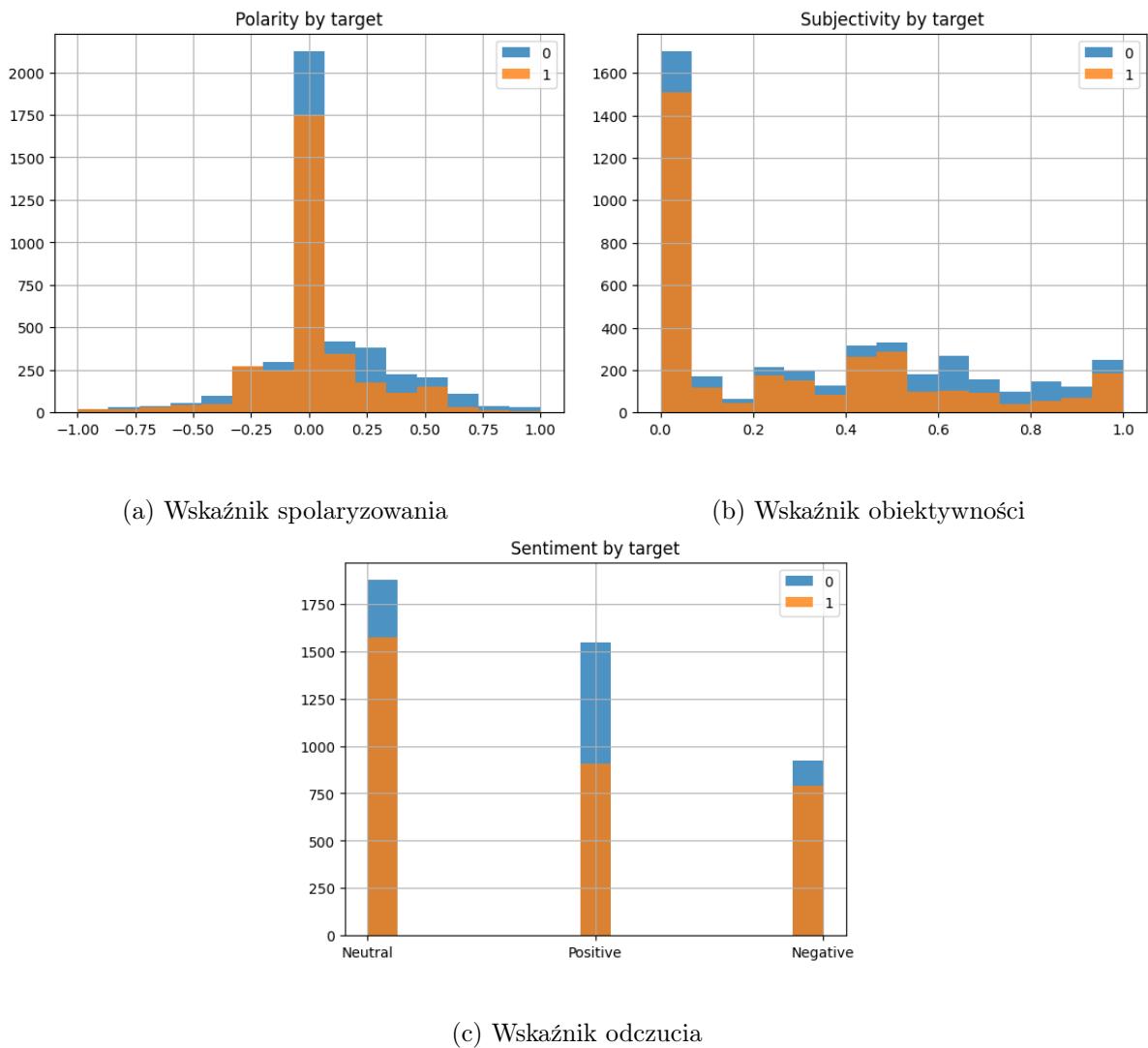
Sprawdzono z jakich lokalizacji najczęściej pojawiają się wpisy (Rysunek 2). Widać tutaj dużą rozbieżność typu podawania informacji. Większość wpisów pochodzi z terenu USA, ale przez brak unifikacji sposobu podawania informacji nie da się uzyskać bardziej szczegółowych informacji bez dodatkowej obróbki danych. Dodatkowo, możliwość wybrania przez użytkownika miejsca



Rysunek 2: Najczęściej występujące lokalizacje

"wszędzie"(Everyware) lub "na całym świecie"(Wordwide) sprawia, że mało prawdopodobna jest przydatność tej informacji przy tworzeniu modelu.

Zbadano zależność wskaźników związanych z analizą odczuć w zależności od klasyfikacji wpisu (Rysunek 3). Widoczne jest, że w zbiorze zawierającym wartość 1 w kolumnie *target* jest delikatnie więcej wpisów z ujemną wartością wskaźnika *polarity*, jest tam podobna ilość wpisów pozytywnych i negatywnych. W drugim ze zbiorów zauważalna jest duża przewaga wpisów pozytywnych oraz tych o dodatnim wskaźniku *polarity*, w porównaniu do poprzedniego przypadku widoczna jest również większa reprezentacja wpisów subiektywnych, co oznacza więcej emocjonalnych informacji niż takich faktów. Zgadza się to ze spodziewanym efektem - posty niedotyczące katastrof a zawierające jedynie słowa kluczowe z nimi kojarzone często będą miały pozytywny wydźwięk.

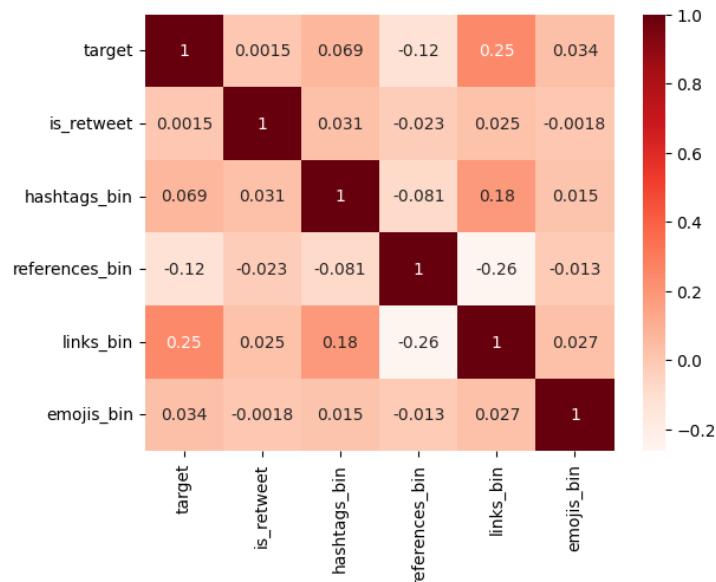


Rysunek 3: Wskaźniki związane z analizą sentymentu w zależności od klasyfikacji

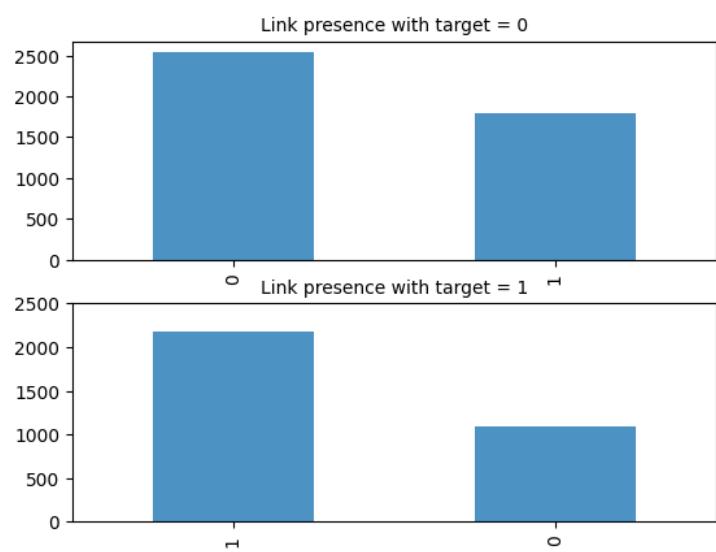
Policzono korelacje między kolumnami zawierającymi wartości binarne. Kolumny z końcówkami *-bin* pokazują, czy dany wpis zawiera element z danej kategorii. Na podstawie wyników przygotowano wykres typu heatmap obrazujący te zależności (Rysunek 4). Zauważono dzięki temu, że najbardziej skorelowana z końcową klasyfikacją posta jest obecność linku. Sprawdzono tę zależność i przedstawiono na Rysunku 5. Widać na nim, że w przypadku wpisów uznanych za informujące o tragedii jest dwa razy więcej wpisów z linkami niż tych bez, natomiast w drugim przypadku jest odwrotnie. Można spodziewać się, że często przy wpisach opisujących istotne wydarzenia, osoby publikujące posty dołączają linki do stron informujących o zdarzeniu czy możliwościach pomocy osobom poszkodowanym. Często przy wpisach zawierających linki zdarzają się też hashtagi, natomiast jest mniejsza szansa na zauważenie przy nich referencji.

Aby przyjrzeć się dokładniej treściom postów, zebrano wszystkie występujące słowa i przeanalizowano te najczęściej występujące. 100 najczęściej występujących zebrano w postaci mapy słów (Rysunek 6) natomiast 20 z nich przedstawiono na wykresie informującym o dokładnej liczbie wystąpień (Rysunek 7). Najczęściej występujące słowa w zbiorze to: fire, news oraz video. Bardzo dużo z kolejnych słów nacechowane jest negatywnie i odnosi się do wypadków, np: disaster, emergency, police czy storm.

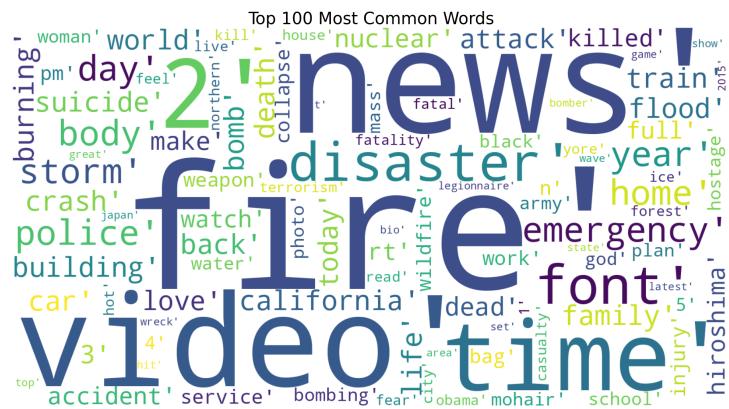
Przygotowano 2 oddzielne zbiory wszystkich wykorzystanych słów dla różnych klasyfikacji końcowych. Otrzymano znaczco różniące się mapy słów (Rysunek 8) oraz wykresy częstotliwości występowania najpopularniejszych słów (Rysunek 9). W przypadku zbioru zawierającego



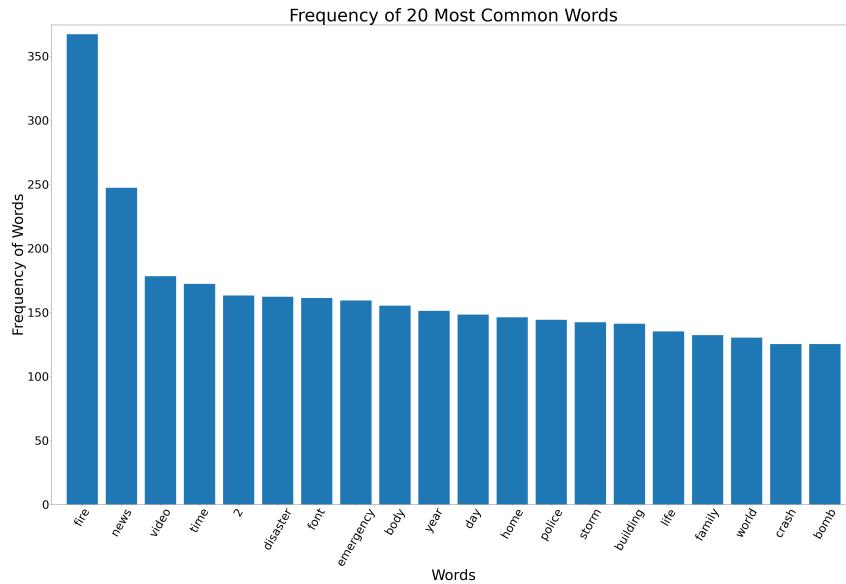
Rysunek 4: Heatmap - wykres korelacji wskaźników



Rysunek 5: Obecność linków z zależnością od klasyfikacji



Rysunek 6: Mapa słów dla całego zbioru

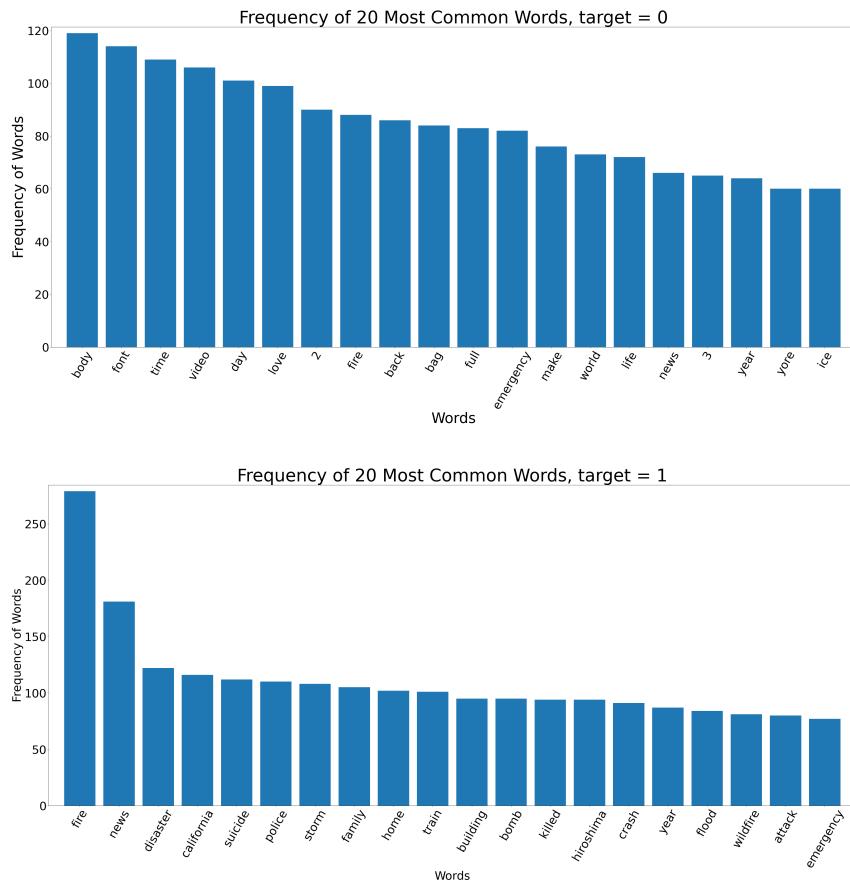


Rysunek 7: Najczęściej występujące słowa

faktyczne informacje o katastrofach dominują słowa z nimi związane, kojarzone raczej negatywnie, podczas gdy w drugim zbiorze wśród najczęściej występujących słów znajdują się bardziej neutralne wyrażenia. Aby zobaczyć tę różnicę wystarczy porównać trzy najczęściej występujące słowa, dla pierwszego przypadku są to: fire news oraz disaster, natomiast dla drugiego: body, font i time.



Rysunek 8: Mapy słów w zależności od klasyfikacji



Rysunek 9: Najczęściej występujące słowa w zależności od klasyfikacji

### 6.1.2 Testy przygotowanych modeli

Po zakończonej analizie, rozpoczęto testy przygotowanych modeli. Wyniki opisujące dokładność modelu, miarę F1, precyzję oraz czułość przedstawiono w Tabeli 1.

Tabela 1: Porównanie stworzonych modeli do klasyfikacji

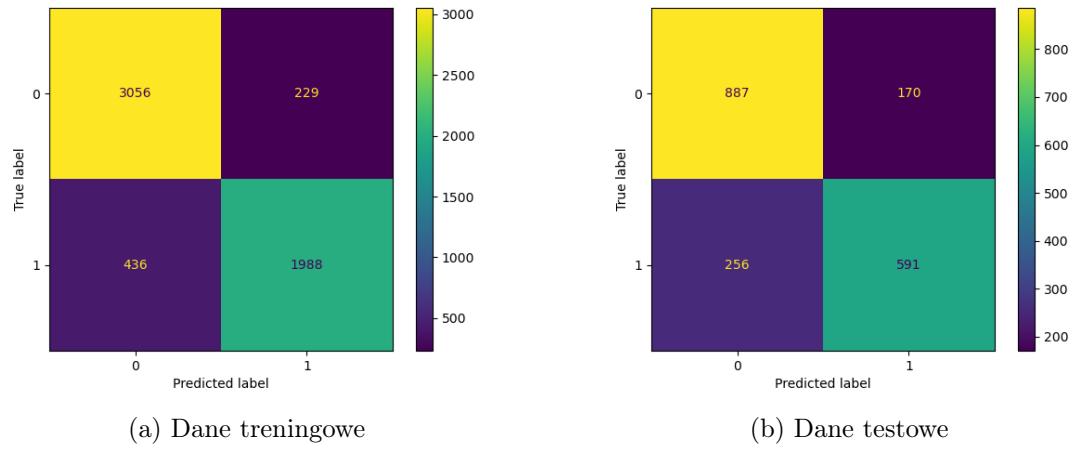
Algorytm	Dane	Dokładność	Miara F1	Precyzja	Czułość
Regresja Logistyczna	Treningowe	0,88	0,86	0,90	0,82
	Testowe	0,78	0,74	0,78	0,70
SVM	Treningowe	0,85	0,81	0,89	0,74
	Testowe	0,79	0,73	0,82	0,66
Naiwny klasyfikator Bayesa	Treningowe	0,83	0,79	0,85	0,73
	Testowe	0,79	0,74	0,82	0,68
K najbliższych sąsiadów	Treningowe	0,78	0,71	0,80	0,64
	Testowe	0,66	0,56	0,68	0,47
Drzewo decyzyjne	Treningowe	0,93	0,90	1,0	0,83
	Testowe	0,74	0,69	0,75	0,63
Las losowy	Treningowe	1,0	1,0	1,0	1,0
	Testowe	0,78	0,74	0,80	0,68

Jako, że ten zbiór danych jest niebalansowany i zawiera więcej obiektów jednej niż drugiej klasy, porównanie będzie dokonywane ze skupieniem się na wartości miary F1 zamiast na dokładności. Na podstawie powyższej tabeli widać, że na zbiorze testowym najlepiej radzą sobie naiwny klasyfikator Bayesa, Regresja Logistyczna oraz Las Losowy. W przypadku tego ostatniego, widoczny jest bardzo duży overfitting - metoda niemal idealnie sklasyfikowała zbiór treningowy dając o wiele słabsze wyniki na zbiorze testowym. Najgorzej poradziła sobie metoda K najbliższych sąsiadów osiągając wartość miary F1 na poziomie wyłącznie 0,56. W przypadku drzewa decyzyjnego, mimo bardzo dobrego wyniku klasyfikacji na zbiorze treningowym, model jest drugim najsłabiej wypadającym przypadkiem.

Widoczna jest znaczna różnica pomiędzy miarami dla zbioru treningowego i testowego dla wszystkich algorytmów poza Naiwnym Klasyfikatorem Bayesa, gdzie ta różnica jest niewielka. Innym przypadkiem, gdzie różnica między zbiorami jest niewielka jest przypadek algorytmu SVM. Model wykazał tam niższą niż w najlepszych przypadkach miarę FI dla zbioru testowego, mimo że charakteryzował się wysoką dokładnością i precyją. Niska była tam wartość czułości, przez co model nie jest uznawany za jeden z najlepiej sprawdzających się w tym przypadku. Największa czułość pojawiła się w modelu Regresji Logistycznej, natomiast największa precyza weszła w modelach SVM oraz Naiwnym klasyfikatorze Bayesa.

Osiągnięty najlepszy wynik na podstawie miary F1 to 0,74 dla modeli Regresji Logistycznej, Naiwnego Klasyfikatora Bayesa i Lasu Losowego.

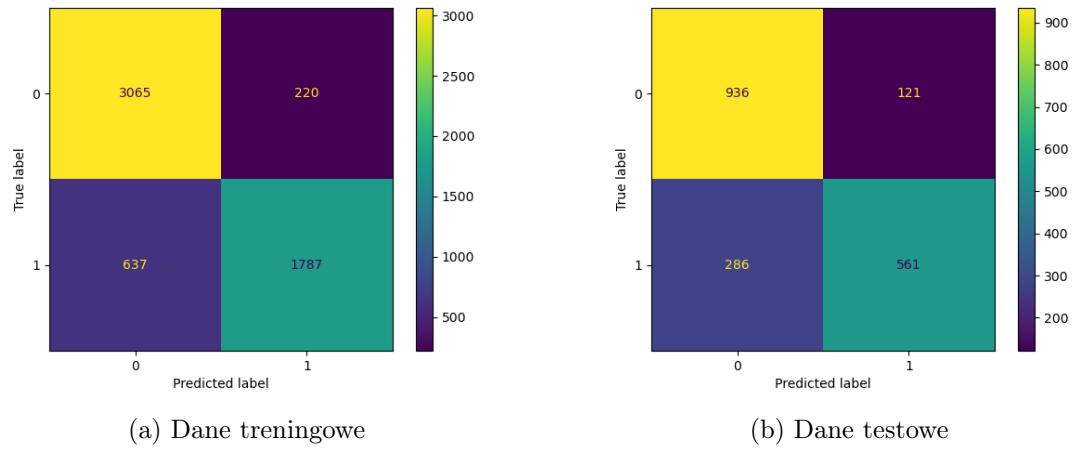
### 6.1.3 Regresja Logistyczna



Rysunek 10: Macierz pomyłek dla regresji logistycznej

W przypadku danych treningowych widoczne jest, że model daje mniej fałszywie pozytywnych wyników, niż fałszywie negatywnych w obu zbiorach, danych treningowych i testowych, może to wynikać z przewagi elementów klasy 0 w zbiorze danych. Jest to model charakteryzujący się największą czułością dla danych testowych co widoczne jest w liczbie fałszywie negatywnych klasyfikacji w porównaniu do prawdziwie pozytywnych.

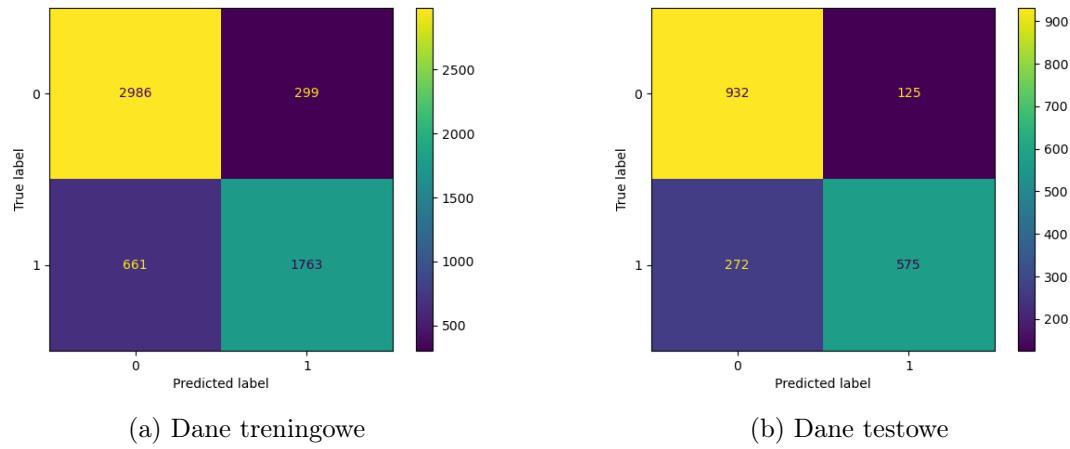
### 6.1.4 SVM



Rysunek 11: Macierz pomyłek dla maszyny wektorów nośnych

Dla klasyfikatora SVM, dla zbioru treningowego widoczne jest prawie trzykrotnie więcej obiektów przyporządkowanych fałszywie negatywnie, niż fałszywie pozytywnie dla danych treningowych. Dysproporcja ta zmniejsza się przy danych testowych, gdzie znajduje się ich jedynie dwa razy więcej. Macierz pomyłek odzwierciedla dużą różnicę między precyzyją a czułością modelu.

### 6.1.5 Naiwny klasyfikator Bayesa



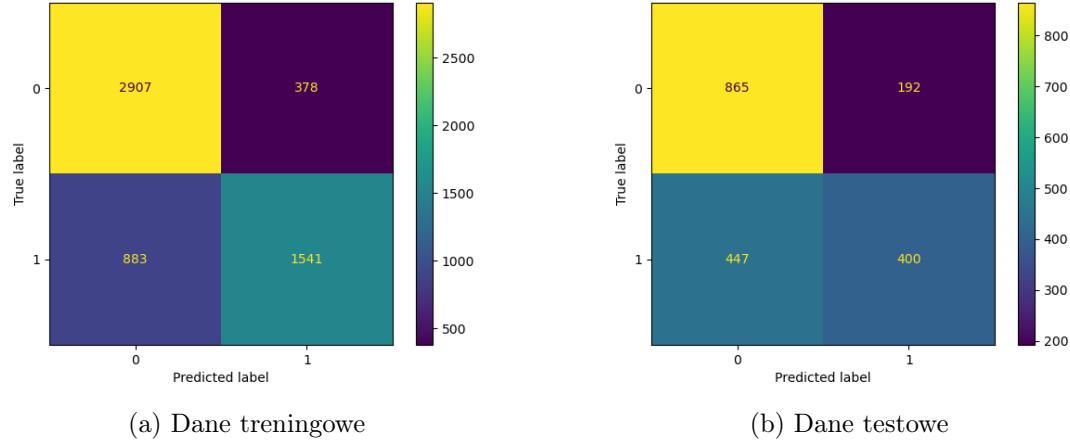
(a) Dane treningowe

(b) Dane testowe

Rysunek 12: Macierz pomyłek dla naiwnego klasyfikatora Bayesa

W przypadku naiwnego klasyfikatora Bayesa widoczna jest znacznie większa liczba fałszywie zaklasyfikowanych wpisów niż w poprzednich przypadkach na danych treningowych. Delikatnie mniejsza jest natomiast różnica między fałszywie pozytywnymi a fałszywie negatywnymi przypisiami w zbiorze danych testowych.

### 6.1.6 K najbliższych sąsiadów



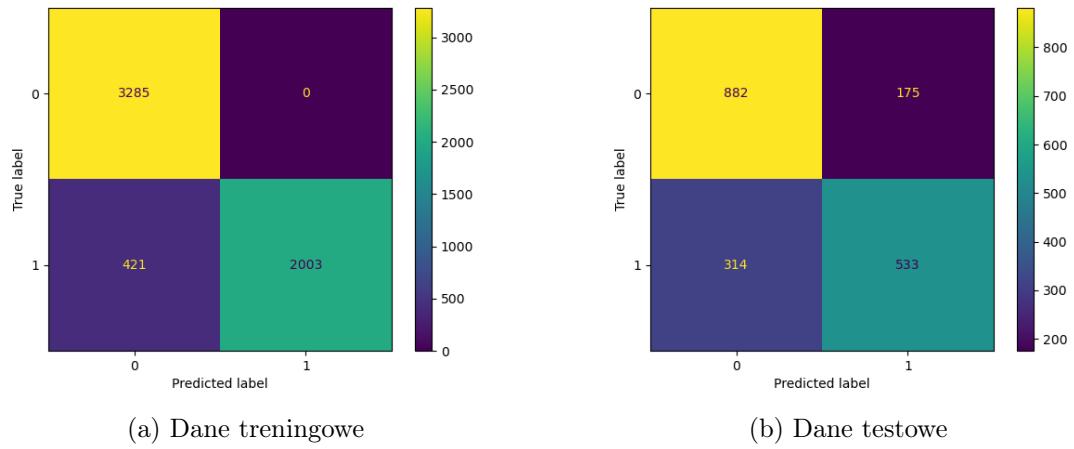
(a) Dane treningowe

(b) Dane testowe

Rysunek 13: Macierz pomyłek dla metody k najbliższych sąsiadów

Dla modelu, który wypadł najgorzej, również jest widoczna wspominana tendencja dysproporcji między fałszywymi przypisiami, których jest w tym przypadku bardzo dużo. W przypadku zbioru testowego widać, że fałszywie negatywne przewidywanie wystąpiło więcej razy niż prawdziwie pozytywne, tych pierwszych jest ponad 1.5 raza więcej niż w poprzednich przypadkach.

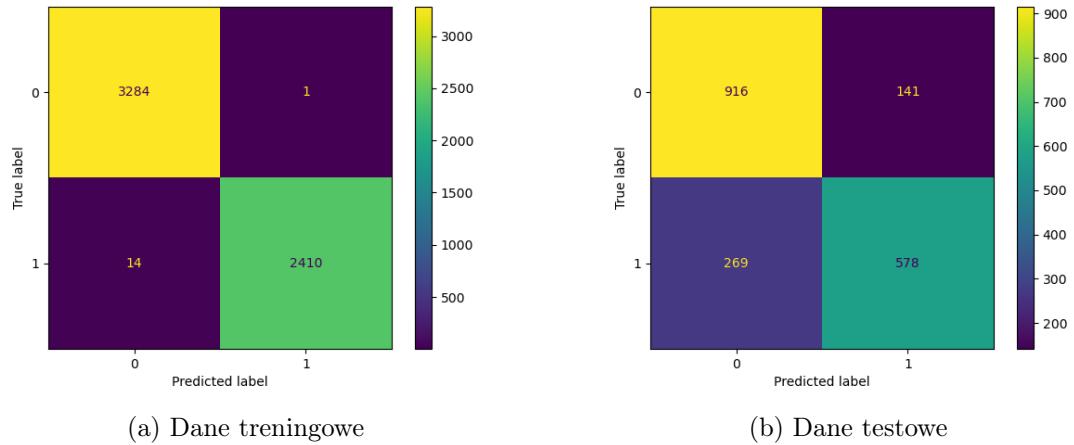
### 6.1.7 Drzewo decyzyjne



Rysunek 14: Macierz pomyłek dla drzewa decyzyjnego

Dla drzewa decyzyjnego na zbiorze treningowym rzucająca się w oczy jest zerowa ilość fałszywie pozytywnych przypisań, tych fałszywie negatywnych jest również w porównaniu do innych modeli niewiele. Sytuacja zmienia się niestety na danych testowych, gdzie fałszywie negatywnych przypisań jest wiele.

### 6.1.8 Las losowy



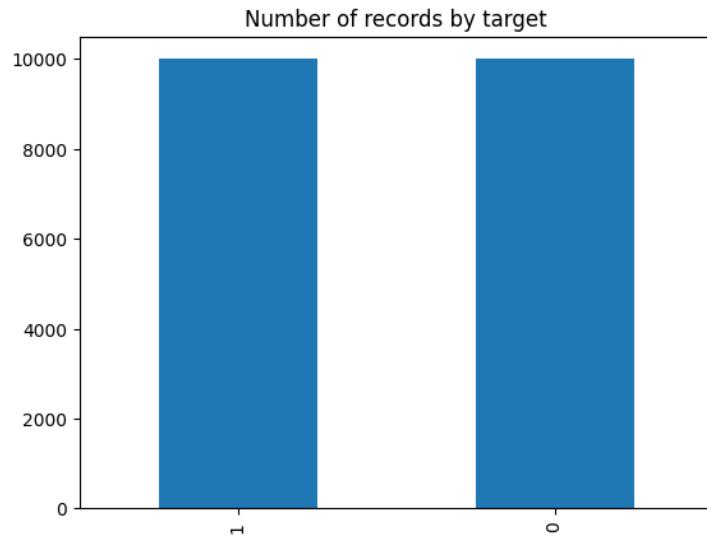
Rysunek 15: Macierz pomyłek dla lasu losowego

Las losowy na zbiorze treningowym działa prawie perfekcyjnie. Tylko 1 element sklasyfikował fałszywie pozytywnie i tylko 14 fałszywie negatywnie. Jednak w przypadku zbioru testowego nie wygląda to już tak dobrze choć wciąż bardzo dobrze względem reszty modeli. Model gorzej radzi sobie z nadawaniem klasy 1 niż 0, ponieważ stosunek FN do FP to ponad 2.

## 6.2 Depression: Twitter Dataset

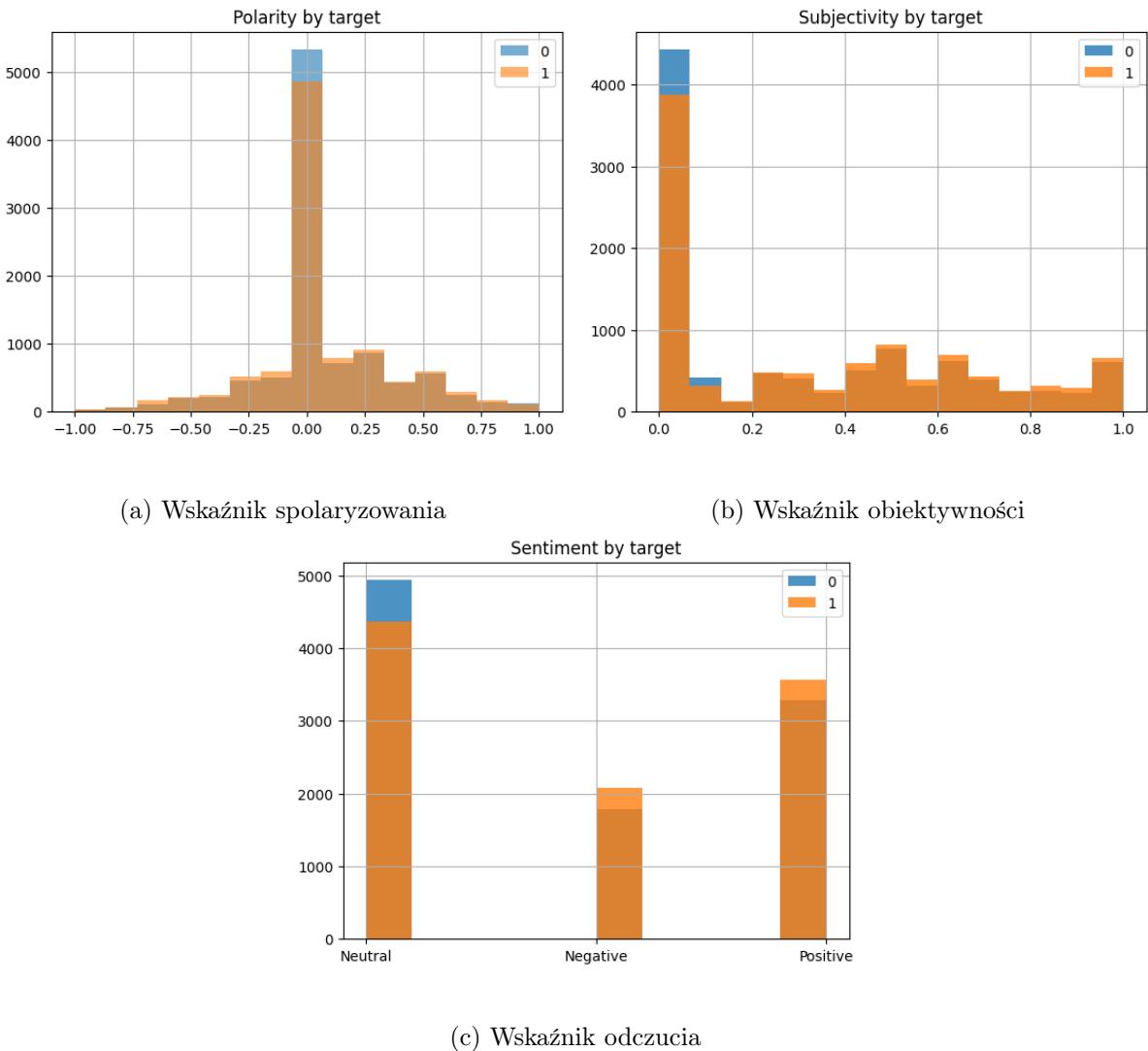
### 6.2.1 EDA - Exploratory Data Analysis

Zbiór "Depression: Twitter Dataset" stworzony został do przygotowywania modeli klasyfikujących czy dany post został napisany przez osobę cierpiącą na depresję. Zbiór zawiera tekst posta, datę jego powstania, id użytkownika oraz liczbę obserwujących, przyjaciół, ulubionych i opublikowanych statusów. Kolumna *target* informuje o przypisaniu wpisu do jednej z grup, gdzie 1 oznacza informację o chorobie psychicznej autora a 0 jej brak. Zauważono, że zbiór jest dobrze zbilansowany i występuje ta sama liczba wpisów dla obydwu grup (Rysunek 16).



Rysunek 16: Ilość wpisów w zależności od klasyfikacji

Zbadano zależność wskaźników związanych z analizą odczuć w zależności od klasyfikacji wpisu (Rysunek 17). W zbiorze zawierającym wartość 0 w kolumnie *target* widać więcej nienacechowanych emocjonalnie wypowiedzi w stosunku do drugiej grupy. Przewaga widoczna jest przy słupkach z wartościami 0 przy wskaźniku *polarity* oraz *subjectivity* a także przy wskaźniku opisującym ogólne odczucie jako neutralne. Nie jest to jednak duża różnica, co ponownie pokazuje dobre zbilansowanie zbioru.

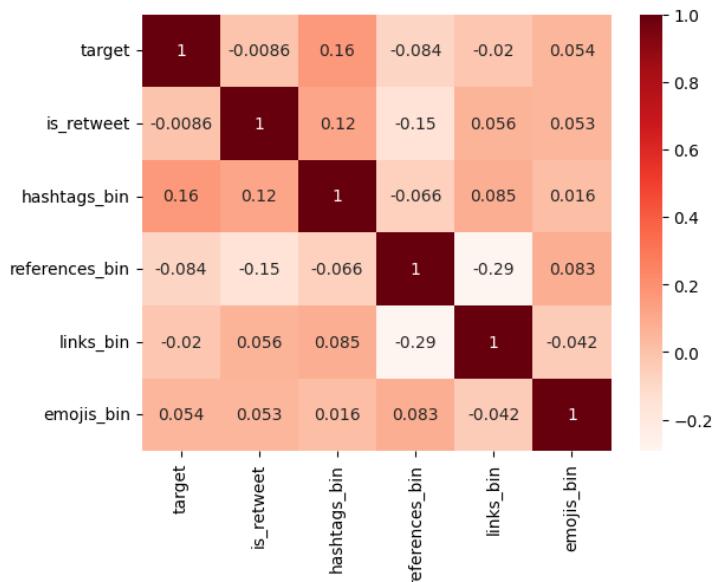


Rysunek 17: Wskaźniki związane z analizą sentymentu w zależności od klasyfikacji

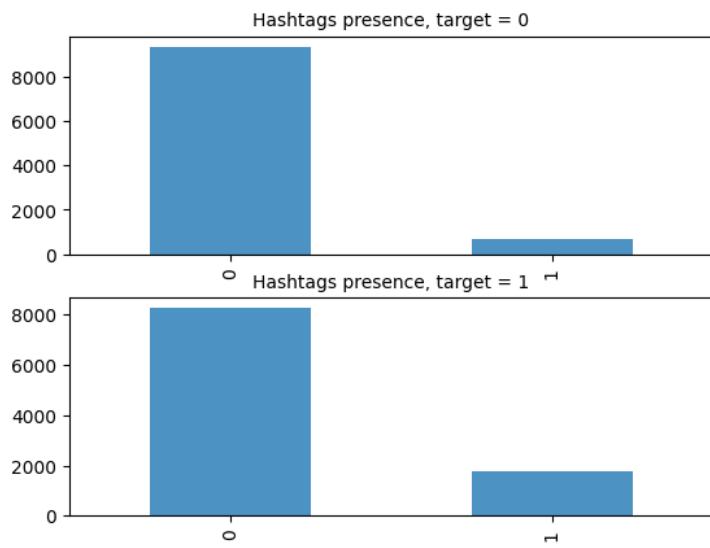
Policzono korelacje między kolumnami zawierającymi wartości binarne. Na podstawie wyników przygotowano wykres typu heatmap obrazujący te zależności (Rysunek 18). W przypadku tego zbioru nie zaobserwowano dużych korelacji związanych z wartością pola *target*, największa z nich występowała między końcową klasyfikacją posta a obecnością hashtags. Sporządzono wykres, na którym pokazano tę zależność (Rysunek 19). Widoczne jest, że w przypadku sklasyfikowanych pozytywnie postów występuje więcej postów z hashtagiem (około 20%) niż w przeciwnym przypadku (około 10%). Zaobserwowało znaczącą ujemną korelację między obecnością odniesienia a obecnością linku oraz między obecnością odniesienia a polem *is\_retweet*.

Tak samo jak w przypadku poprzedniego zbioru, zebrano wszystkie występujące słowa i przeanalizowano te najczęściej występujące. 100 najczęściej występujących zebrano w postaci mapy słów (Rysunek 21) natomiast 20 z nich przedstawiono na wykresie informującym o dokładniej liczbie wystąpień (Rysunek 22). Najczęściej pojawiającym się w zbiorze słowem jest *depression*. Wiele z innych pojawiających się słów odnosi się do nie zawsze negatywnych przeżywanych emocji jak na przykład: *love*, *feel*, *happy*, pojawia się też kilka mówiących o doświadczeniach osób z chorobami psychicznymi: *mental*, *overcome*, *treatment*. Drugim najczęściej występującym słowem jest *font*, co w rzeczywistości oznacza słowo *don't* i powstało w wyniku działania biblioteki służącej do sprawdzania poprawności słów.

Dla głębszej analizy występujących słów, przygotowano 2 oddzielne zbiory wszystkich wykorzystanych słów dla różnych klasyfikacji końcowych. W wyniku tego, otrzymywano mapy słów

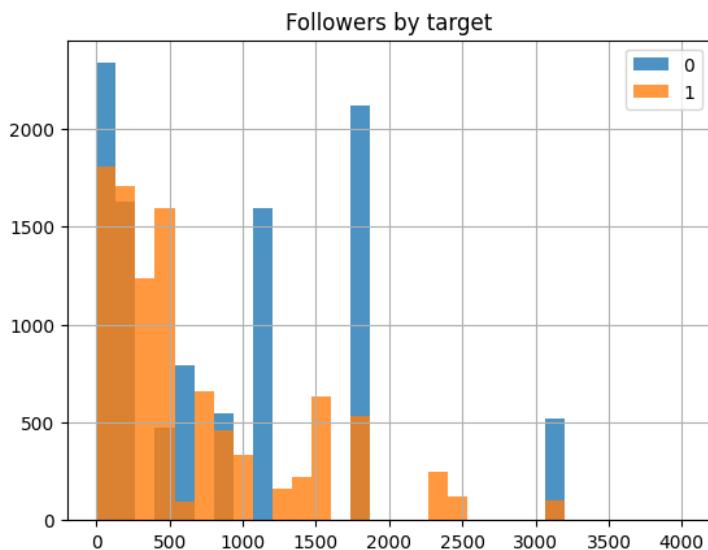


Rysunek 18: Heatmap - wykres korelacji wskaźników

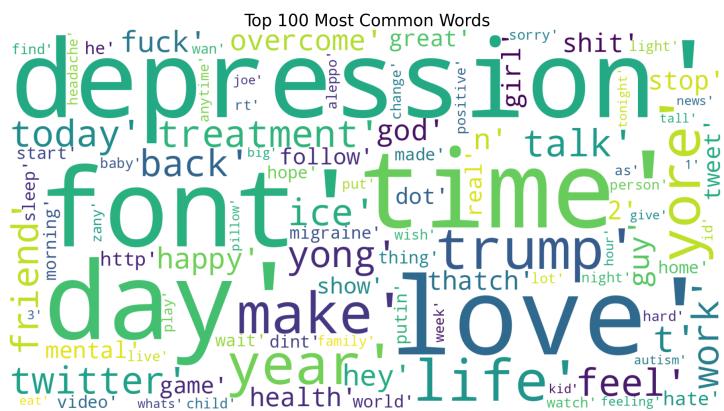


Rysunek 19: Obecność hashagów z zależnością od klasyfikacji

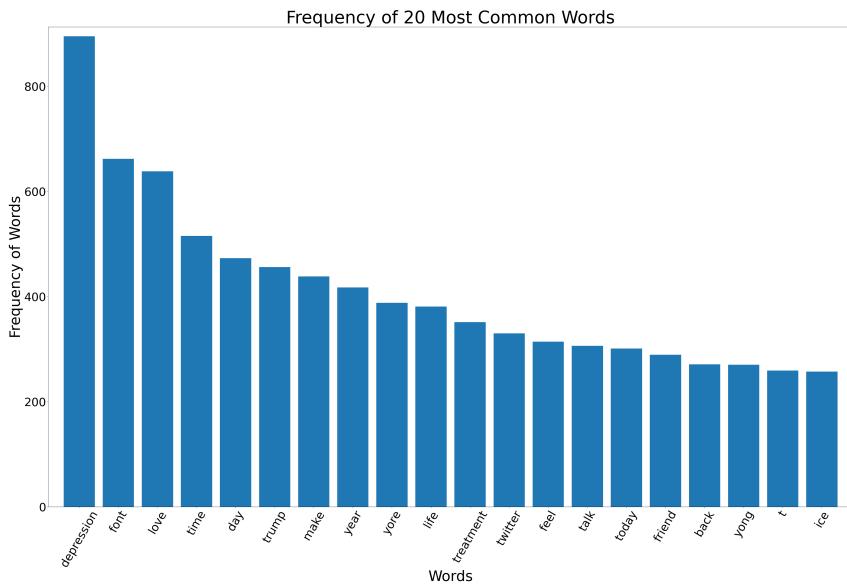
(Rysunek 23) oraz wykresy częstotliwości występowania najpopularniejszych słów (Rysunek 24). Dla zbioru postów pisanych przez osoby cierpiące na problemy psychiczne widoczna jest większa ilość związanych z tym wyrażeń: depression, treatment, overcome, health. Wiele spośród 100 najpopularniejszych słów w tym zbiorze wiąże się z bólem. Bardziej neutralne słowa znajdują się w przeciwnym zbiorze. Występowanie słowa *trump* jako najczęściej tam występującego może wiązać się z negatywnymi odczuciami jakie kojarzą się osobom piszącym posty z byłym prezydentem USA.



Rysunek 20: Liczba obserwujących z zależnością od klasyfikacji



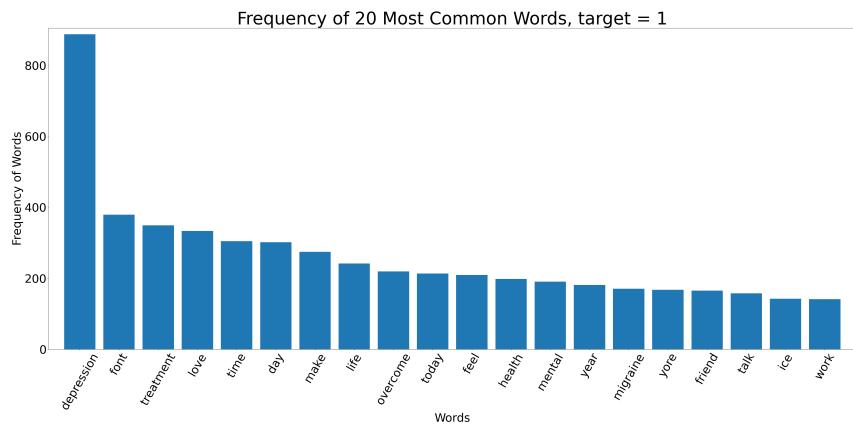
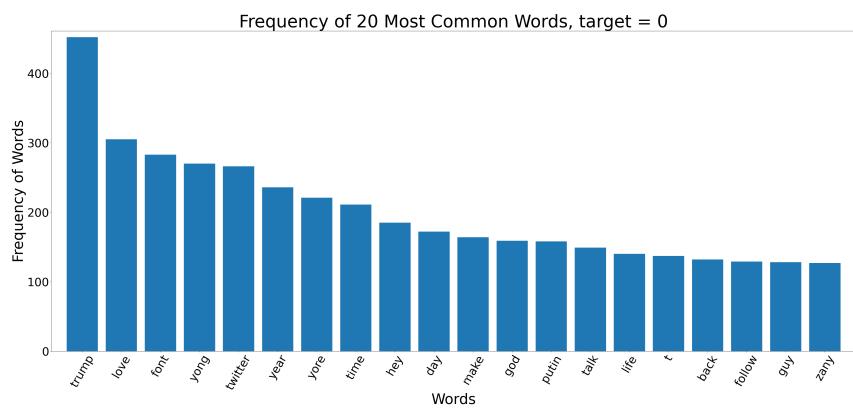
Rysunek 21: Mapa słów dla całego zbioru



Rysunek 22: Najczęściej występujące słowa



Rysunek 23: Mapy słów w zależności od klasyfikacji



Rysunek 24: Najczęściej występujące słowa w zależności od klasyfikacji

### 6.2.2 Testy przygotowanych modeli

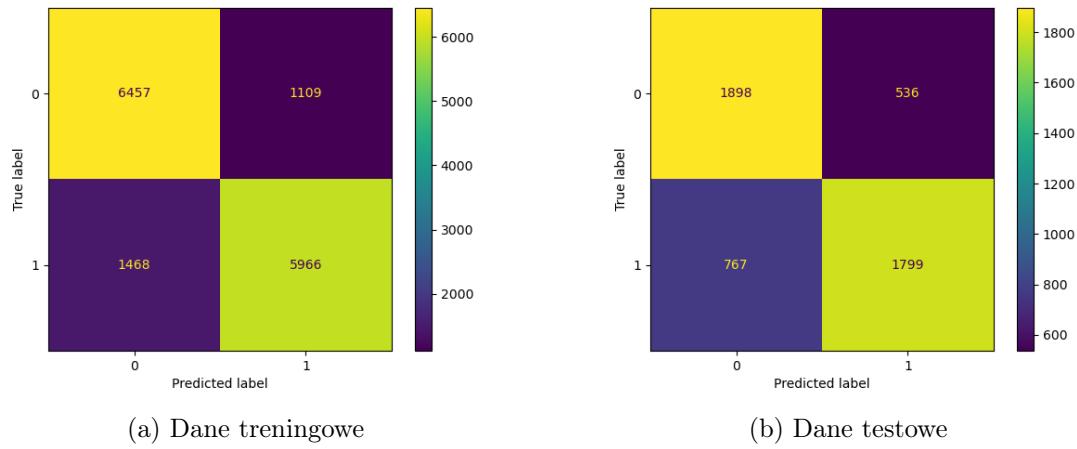
Wykonano testy przygotowanych modeli. Wyniki opisujące dokładność modelu, miarę F1, precyzję oraz czułość przedstawiono w Tabeli 2.

Tabela 2: Porównanie stworzonych modeli do klasyfikacji

Algorytm	Dane	Dokładność	Miara F1	Precyzja	Czułość
Regresja Logistyczna	Treningowe	0,83	0,82	0,84	0,80
	Testowe	0,74	0,74	0,77	0,70
SVM	Treningowe	0,80	0,80	0,83	0,77
	Testowe	0,75	0,75	0,78	0,72
Naiwny klasyfikator Bayesa	Treningowe	0,80	0,79	0,82	0,77
	Testowe	0,75	0,75	0,77	0,72
K najbliższych sąsiadów	Treningowe	0,79	0,77	0,87	0,69
	Testowe	0,60	0,53	0,66	0,44
Drzewo decyzyjne	Treningowe	0,73	0,64	0,99	0,46
	Testowe	0,63	0,52	0,82	0,38
Las losowy	Treningowe	0,99	0,99	1,0	0,99
	Testowe	0,74	0,74	0,76	0,72

W przypadku tego zbioru, wyniki na danych testowych są najlepsze i prawie identyczne dla dwóch z przygotowanych modeli: SVM i Naiwnego klasyfikatora Bayesa. Dokładność i miara F1 osiągają tam wartość 0,75. Algorytmy różnią się precyzją, przyjmując większą wartość tego wskaźnika dla modelu SVM. Niewiele gorsze, bo o jedynie jedną setną, są algorytmy implementujące Regresję Logistyczną oraz Las Losowy. Dużo wyższe parametry otrzymano badając modele na danych treningowych, nie są to jednak tak duże różnice jak przy poprzednim zbiorze. Dla algorytmu Lasu Losowego na danych treningowych otrzymano wartości precyzji oraz miary F1 równe aż 0,99. Ponownie algorytm ten wykazuje bardzo duże dopasowanie na tym zbiorze podczas gdy przy testach daje rezultaty podobne do innych algorytmów. Najgorszym z algorytmów ponownie okazał się model K najbliższych sąsiadów, gdzie dokładność i precyzja są na najniższym poziomie. Drzewo decyzyjne w tym przypadku charakteryzuje się największą precyzją na zbiorze testowym ale ze względu na bardzo niską czułość (0,38) ma najwyższy wynik przy wartości miary F1.

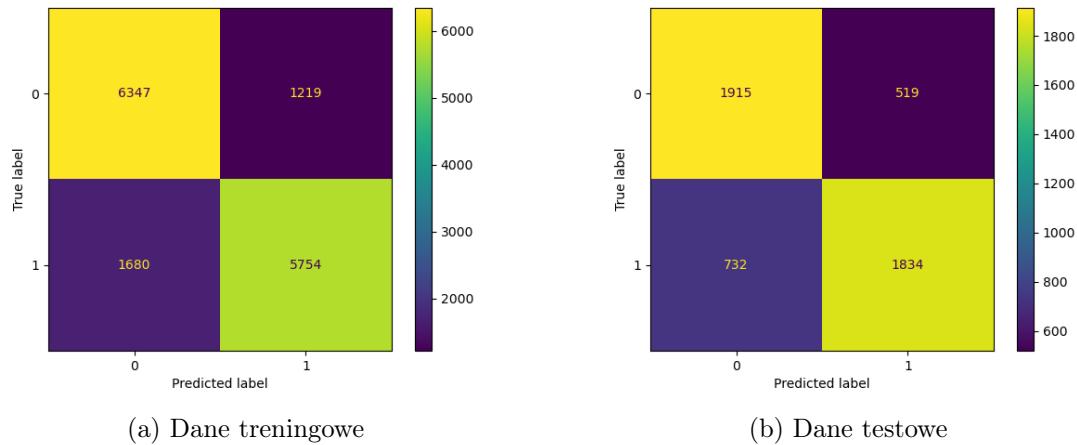
### 6.2.3 Regresja Logistyczna



Rysunek 25: Macierz pomyłek dla regresji logistycznej

W obu zbiorach występuje przewaga w ilości fałszywie negatywnych do fałszywie pozytywnych przypisań, jest ona jednak nie tak duża jak w przypadku zbioru "Disaster Tweets", gdzie przewaga była dwu- lub trzykrotna. Zbiór jest dobrze zbalansowany, zawiera taką samą ilość elementów klasy 0 co 1, stąd różnice nie są duże.

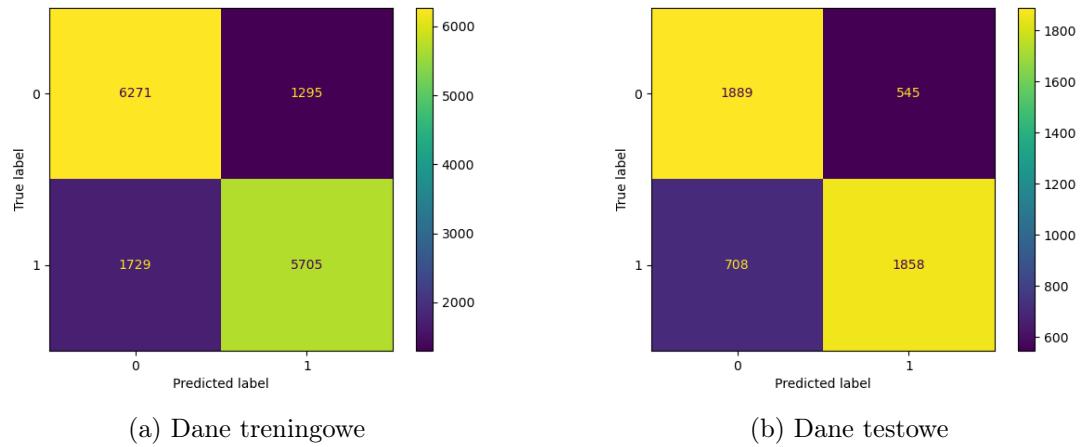
### 6.2.4 SVM



Rysunek 26: Macierz pomyłek dla maszyny wektorów nośnych

Dla klasyfikatora SVM, widoczna jest gorsza praca modelu dla danych treningowych niż w przypadku Regresji Logistycznej. Różnica między ilością fałszywie pozytywnych i fałszywie negatywnych przypisań dla zbioru treningowego jest większa, tak samo jak ich suma. Dla zbioru danych testowych widoczna jest już lepsza praca klasyfikatora SVM, pokazująca się z mniejszej sumie błędnie sklasyfikowanych wpisów.

### 6.2.5 Naiwny klasyfikator Bayesa



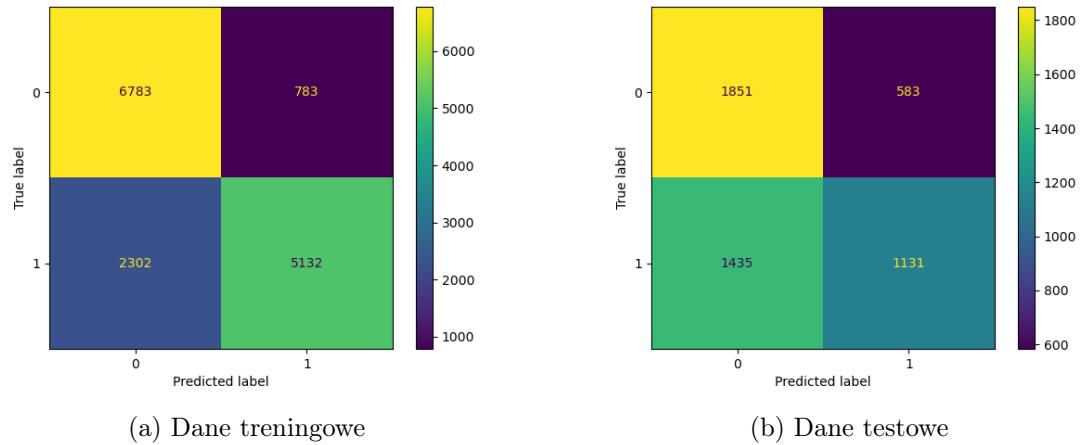
(a) Dane treningowe

(b) Dane testowe

Rysunek 27: Macierz pomyłek dla naiwnego klasyfikatora Bayesa

W przypadku naiwnego klasyfikatora Bayesa znowu widoczna jest słabsza praca klasyfikatora na zbiorze danych treningowych, jest tu bardzo dużo fałszywie negatywnych przypisań. Przy danych testowych, liczba fałszywie negatywnych przypisań jest z kolei najmniejsza wśród wszystkich testowanych modeli.

### 6.2.6 K najbliższych sąsiadów



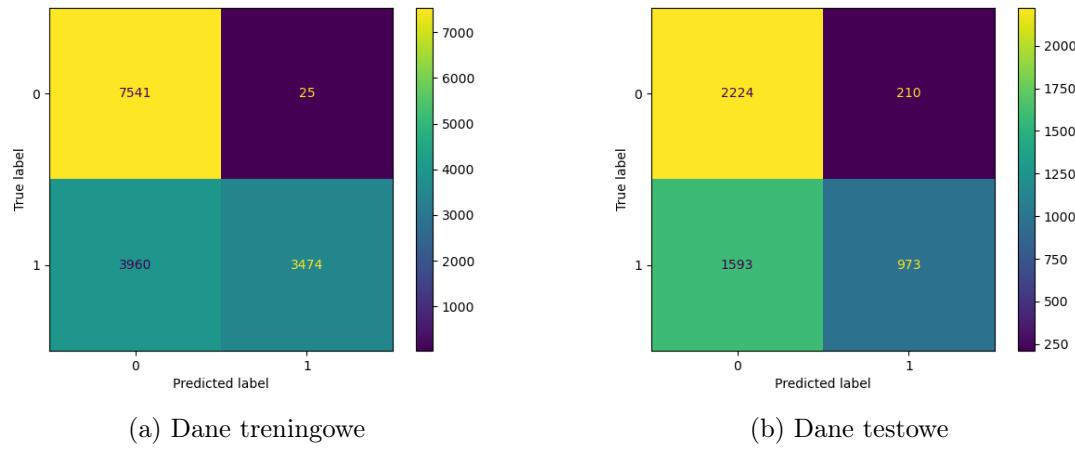
(a) Dane treningowe

(b) Dane testowe

Rysunek 28: Macierz pomyłek dla metody k najbliższych sąsiadów

Model wykorzystujący metodę k najbliższych sąsiadów jest modelem który ponownie wypadł najsłabiej w przypadku danych testowych - widać tutaj bardzo dużą liczbę błędnie zaklasyfikowanych wpisów. Dla zbioru treningowego również widoczna jest bardzo duża dysproporcja między fałszywie przypisanymi wpisami, tych fałszywie negatywnych jest aż 2302. Przy danych testowych, danych zaklasyfikowanych jako fałszywie negatywne jest o wiele więcej niż tych zaklasyfikowanych prawdziwie pozytywnie, co wpływa na wartość czułości poniżej 0.5.

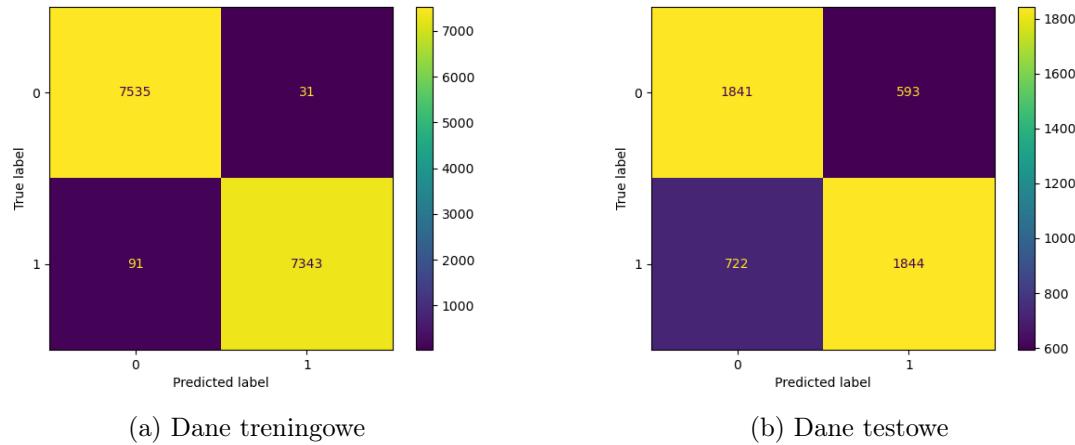
### 6.2.7 Drzewo decyzyjne



Rysunek 29: Macierz pomyłek dla drzewa decyzyjnego

Dla drzewa decyzyjnego na zbiorze treningowym widoczna jest bardzo duża liczba fałszywie negatywnych przypisań w obydwu zbiorach. Bardzo niska ilość fałszywie pozytywnych przypisań sugeruje, że model cechuje się tendencją do przypisywania większości wpisów do klasy 0. Zaklasyfikowanych w ten sposób przez model wpisów jest około 11500 natomiast tych zaklasyfikowanych jako 1 jest 3500. Jest to najgorzej wypadający na danych treningowych model, działa natomiast lepiej niż algorytm K najbliższych sąsiadów dla danych testowych.

### 6.2.8 Las losowy



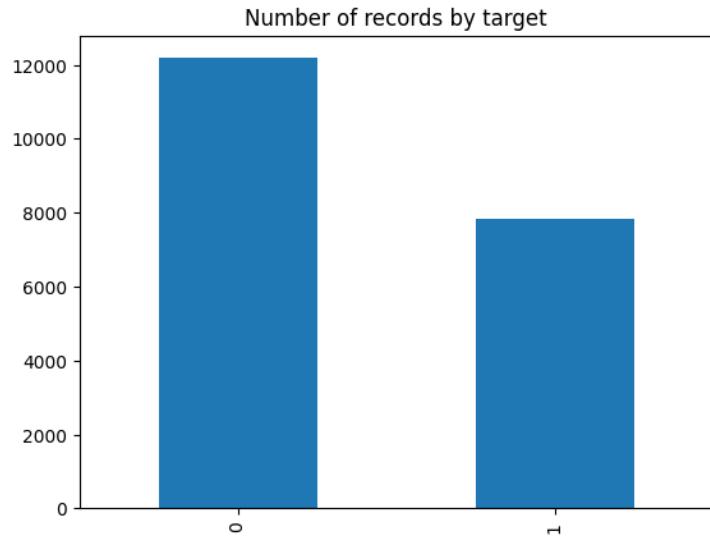
Rysunek 30: Macierz pomyłek dla lasu losowego

Las losowy na zbiorze treningowym działa bardzo dobrze. Jedynie 31 elementów sklasyfikowane zostało fałszywie pozytywnie, większa, choć dalej niewielka liczba wpisów została sklasyfikowana jako fałszywie negatywne (91). Jednak w przypadku zbioru testowego nie wygląda to już tak dobrze, choć wciąż jest to zadowalający względem reszty modeli. Przy danych testowych nie dostrzegalna jest już tak duża różnica między fałszywie pozytywnymi a fałszywie negatywnymi przypisiami.

## 6.3 Suspicious Communication on Social Platforms

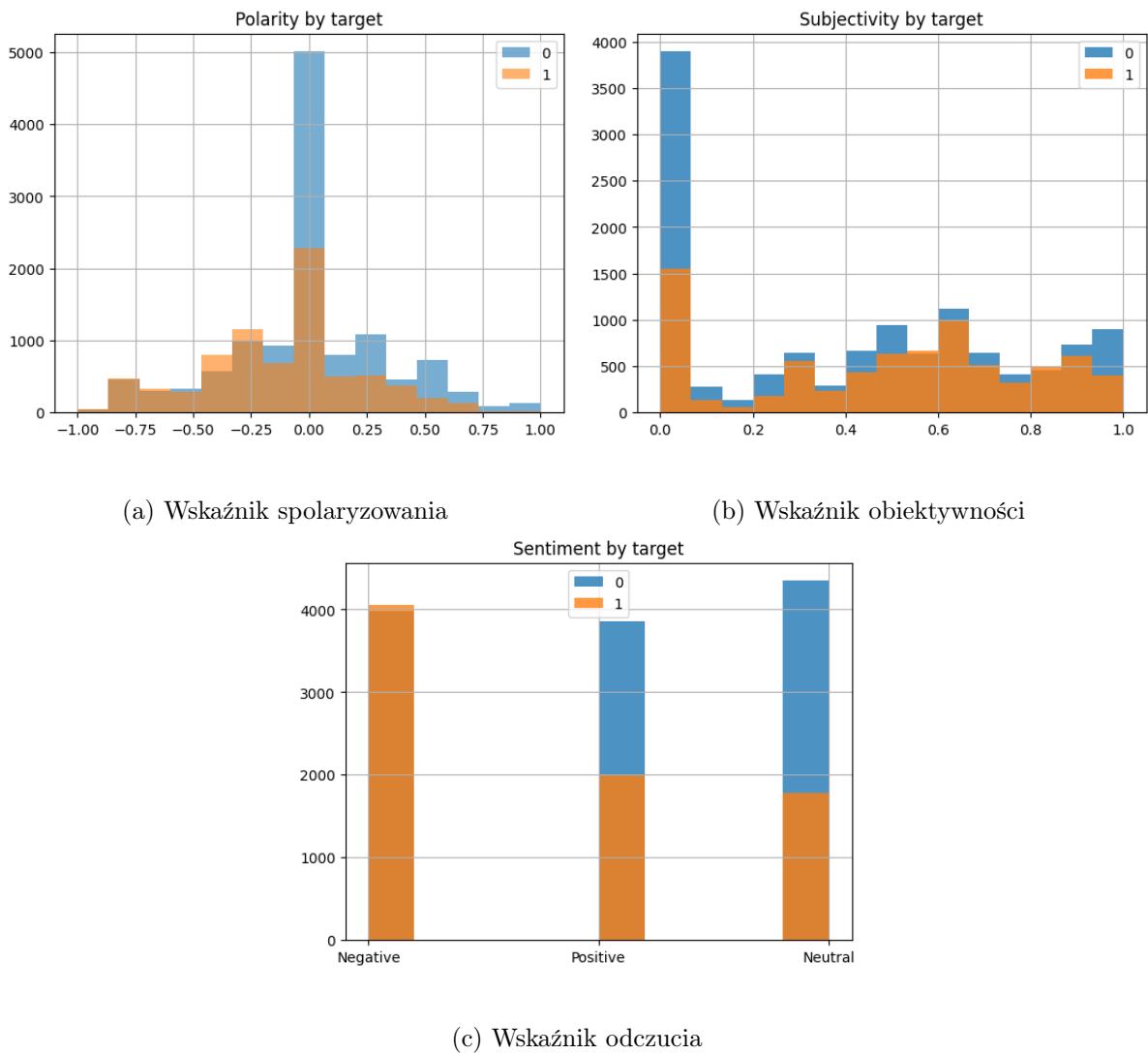
### 6.3.1 EDA - Exploratory Data Analysis

Zbiór "Suspicious Communication on Social Platforms" wykorzystywany jest do uczenia modelu rozpoznawania, czy dany wpis jest podejrzany, tzn czy jest obraźliwy, zawierający mowę nienawiści. Zbiór jedynie treść posta oraz przypisanie do grupy podejrzanych lub nie wpisów. W zbiorze występuje więcej zbiorów oznaczonych w kolumnie *target* jako 0 niż tych z 1 (Rysunek 31).

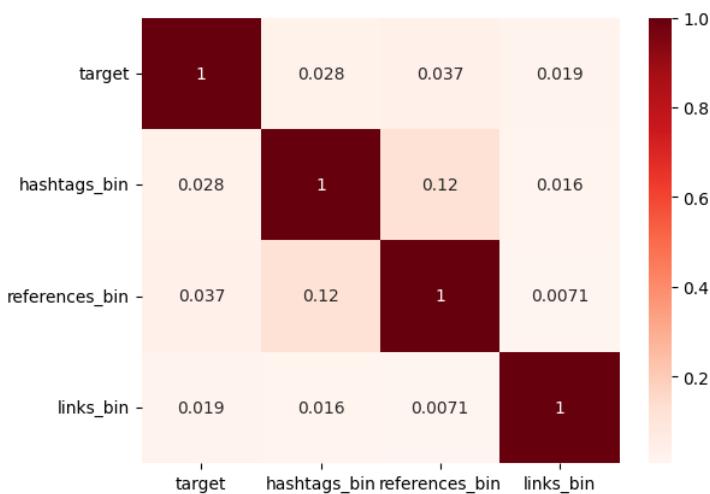


Rysunek 31: Ilość wpisów w zależności od klasyfikacji

Dla posiadanych danych zbadano zależność wskaźników związanych z analizą odczuć w zależności od klasyfikacji wpisu (Rysunek 32). Przy wartości *polarity* widać różnicę w rozkładach dla różnych podgrup. Przy wpisach niesklasyfikowanych jako podejrzane widać bardziej równomierny rozkład na dodatnie i ujemne wartości niż w drugim przypadku, gdzie dominujące są wartości ujemne. Tę zależność widać również przy wskaźniku *sentiment* gdzie w zbiorze z podejrzanymi wpisami dominują posty oznaczone jako negatywne, natomiast w przeciwnym znajduje się podobna ilość zaklasyfikowanych negatywnie pozytywnie i neutralnie z delikatną przewagą wpisów neutralnych.



Rysunek 32: Wskaźniki związane z analizą sentymentu w zależności od klasyfikacji

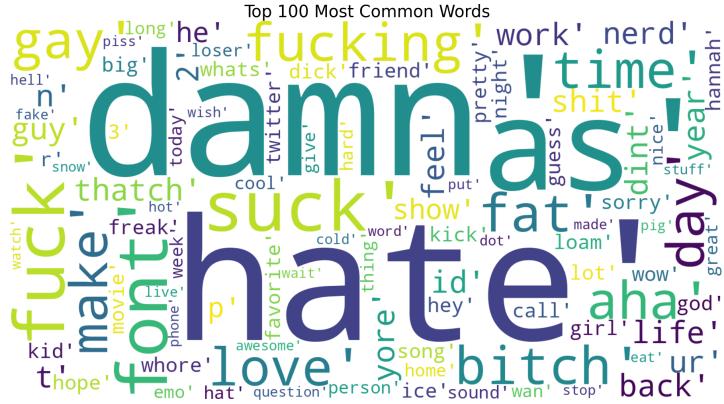


Rysunek 33: Heatmap - wykres korelacji wskaźników

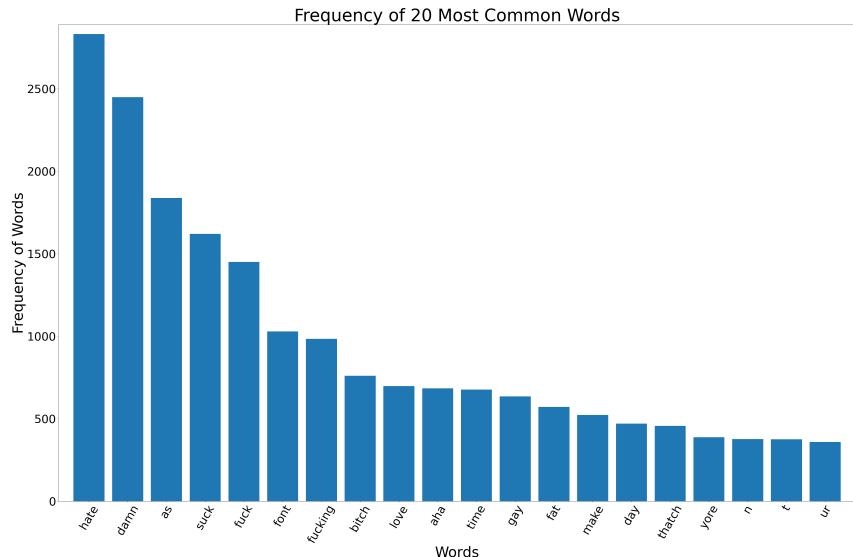
Policzono korelacje między kolumnami zawierającymi wartości binarne. Na podstawie wyni-

ków przygotowano wykres typu heatmap obrazujący te zależności (Rysunek 33). Ze względu na brak emotikon oraz postów typu *retweet* pominięto te wartości. Nie zauważono dużych korelacji między wartościami, największą z nich jest wartość 0.12 mówiąca o związku między obecnością hashtaga a odnośnika. Korelacje związane z klasyfikacją mają wartość poniżej 0.05 co czyni je pomijalnymi.

Ponownie zebrano wszystkie występujące słowa i przeanalizowano te najczęściej występujące. Wyniki przedstawiono w postaci mapy słów (Rysunek 34) oraz na wykresie informującym o dokładniej liczbie wystąpień (Rysunek 35). Najczęściej pojawiające się słowa mają często negatywny wydźwięk i mogłyby być uznane za obraźliwe, co widoczne jest szczególnie na mapie słów. Słowo występujące największą ilość razy - *hate*, pojawia się ponad 2800 razy.



Rysunek 34: Mapa słów dla całego zbioru

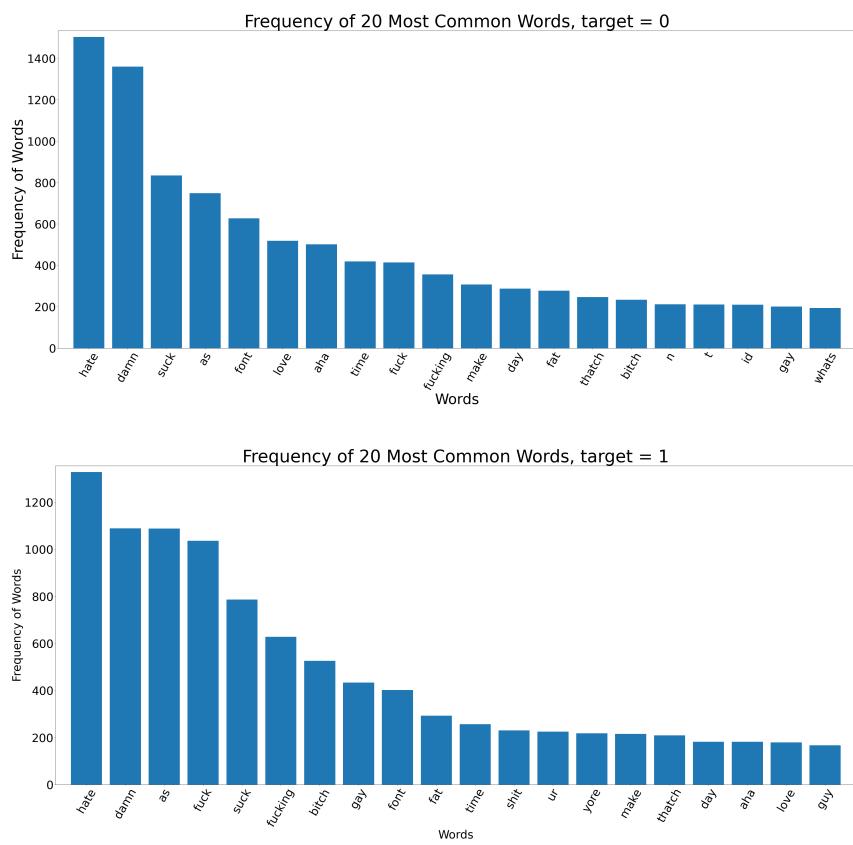


Rysunek 35: Najczęściej występujące słowa

Podzielono dane na 2 zbiory i ptrzymano mapy słów (Rysunek 36) oraz wykresy częstotliwości występowania najpopularniejszych słów (Rysunek 37) w zależności od klasyfikacji wpisów. W obydwu zbiorach najczęściej pojawiające się słowa są prawie takie same, pięć najpopularniejszych w obu zbiorach słów pokrywa się. Ta zależność sugeruje trudność w przygotowaniu modelu, oznacza, że bardzo istotny będzie kontekst użycia brzmiących obraźliwie wyrażeń.



Rysunek 36: Mapy słów w zależności od klasyfikacji



Rysunek 37: Najczęściej występujące słowa w zależności od klasyfikacji

### 6.3.2 Testy przygotowanych modeli

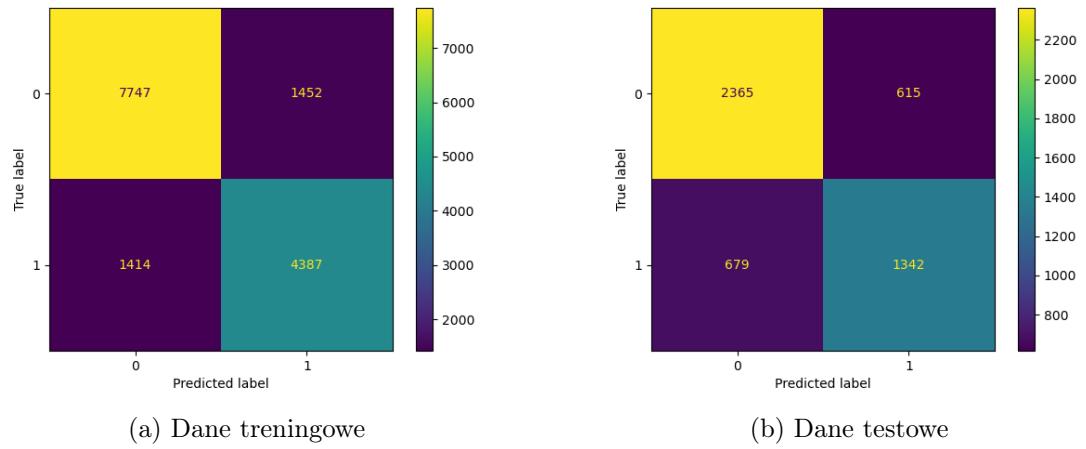
Po zakończonej analizie, rozpoczęto testy przygotowanych modeli. Wyniki opisujące dokładność modelu, miarę F1, precyzję oraz czułość przedstawiono w Tabeli 3.

Tabela 3: Porównanie stworzonych modeli do klasyfikacji

Algorytm	Dane	Dokładność	Miara F1	Precyzja	Czułość
Regresja Logistyczna	Treningowe	0,81	0,75	0,75	0,76
	Testowe	0,74	0,67	0,69	0,66
SVM	Treningowe	0,81	0,75	0,75	0,75
	Testowe	0,74	0,67	0,68	0,67
Naiwny klasyfikator Bayesa	Treningowe	0,75	0,62	0,77	0,52
	Testowe	0,70	0,54	0,70	0,44
K najbliższych sąsiadów	Treningowe	0,92	0,89	0,89	0,90
	Testowe	0,78	0,69	0,78	0,61
Drzewo decyzyjne	Treningowe	0,99	0,99	0,99	0,98
	Testowe	0,81	0,78	0,75	0,82
Las losowy	Treningowe	0,98	0,99	0,98	0,99
	Testowe	0,89	0,87	0,84	0,89

Dla tego zbioru, najlepsze parametry przy danych treningowych i testowych uzyskano dla algorytmu Lasu Losowego - jest to znacząco wyróżniający się model. Podobnie jak w poprzednich przypadkach, dla danych treningowych uzyskał on bliskie 1 wyniki, natomiast przy zbiorze testowym dobre działanie modelu utrzymało się, dając bardzo dobrą dokładność na poziomie 0,89. W przypadku Drzewa Decyzyjnego, które poprzednio również dawało niezbyt dobre wyniki widoczna jest znacząca poprawa dla tych danych - jest on drugim najlepiej działającym algorytmem. Na zbiorze treningowym algorytmy Drzewa Decyzyjnego oraz Lasu Losowego dają niemal identyczne wyniki, na zbiorze testowym natomiast oba wyróżniają się, poza wysoką dokładnością, większą czułością niż precyzją. Zaskakująco słabo wypadł tutaj algorytm wykorzystujący Naiwny klasyfikator Bayesa, w przeciwieństwie do poprzednich zbiorów gdzie dawał jedne z najlepszych wyników, tutaj daje najsłabsze, szczególnie biorąc pod uwagę czułość. Modele Regresji Logistycznej i SVM dają podobne do tych osiągniętych w poprzednich zbiorach wartości dokładności, dużo tracą jednak gdy spojrzy się na miarę F1. Algorytm K najbliższych dał dużo lepsze od spodziewanych wyniki, osiągając dokładność na poziomie 0,78. Ze względu na specyfikę zbioru, mimo znacznie większej ilości danych w klasie 0 niż tych w klasie 1. generalne wyniki były najlepsze, osiągając najmniejszy overfitting.

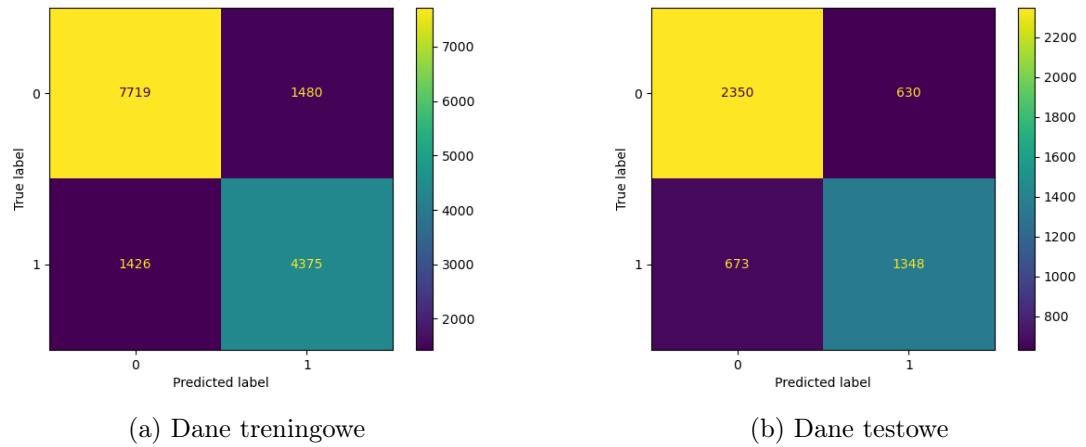
### 6.3.3 Regresja Logistyczna



Rysunek 38: Macierz pomyłek dla regresji logistycznej

W przypadku modelu Regresji Logistycznej, widoczna jest prawie taka sama ilość danych sklasyfikowanych jako fałszywie pozytywne i fałszywie negatywne, mimo większej wielkości klasy 0. Z tego powodu, precyza i czułość są na podobnym poziomie, mniejszym jednak niż dokładność.

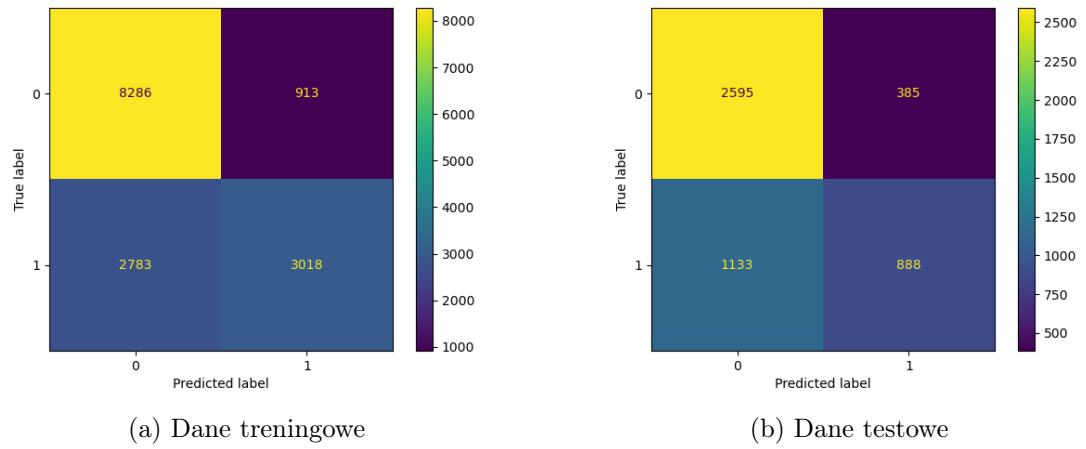
### 6.3.4 SVM



Rysunek 39: Macierz pomyłek dla maszyny wektorów nośnych

Dla tego modelu wyniki są bardzo podobne do Regresji Logistycznej. Proporcja między fałszywie pozytywnymi i negatywnymi klasyfikacjami jest bardzo podobna, delikatnie mniejsza jednak przy zbiorze testowym, co przekłada się na jeszcze mniejszą różnicę między precyza i czułością.

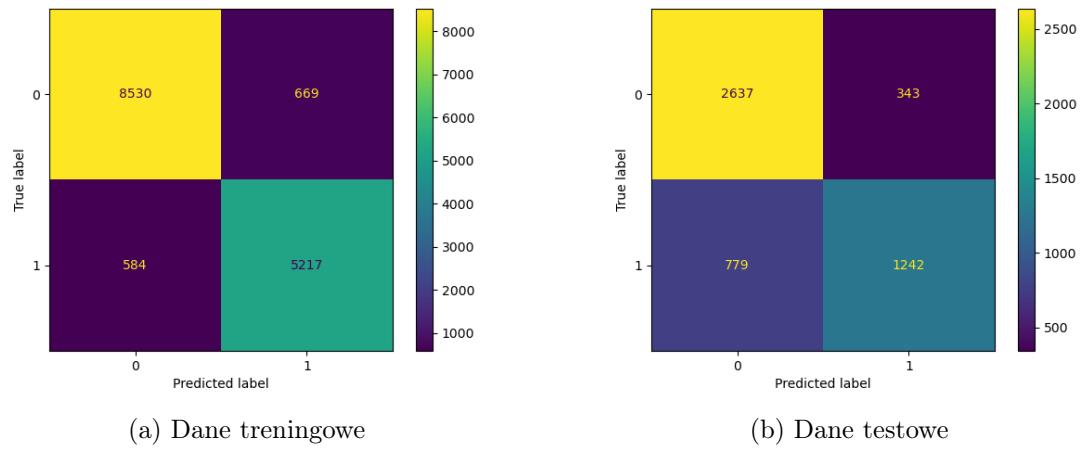
### 6.3.5 Naiwny klasyfikator Bayesa



Rysunek 40: Macierz pomyłek dla naiwnego klasyfikatora Bayesa

Naiwny klasyfikator Bayesa w tym przypadku daje bardzo słabe wyniki. Widoczna jest bardzo zbliżona liczba fałszywie negatywnych i prawdziwie pozytywnych przypisań w przypadku zbioru treningowego, na zbiorze testowym zachodzi wręcz przewaga tych pierwszych nad drugimi. Model charakteryzuje się przez to bardzo niską czułością, przypisując nieproporcjonalnie dużo elementów do klasy 0.

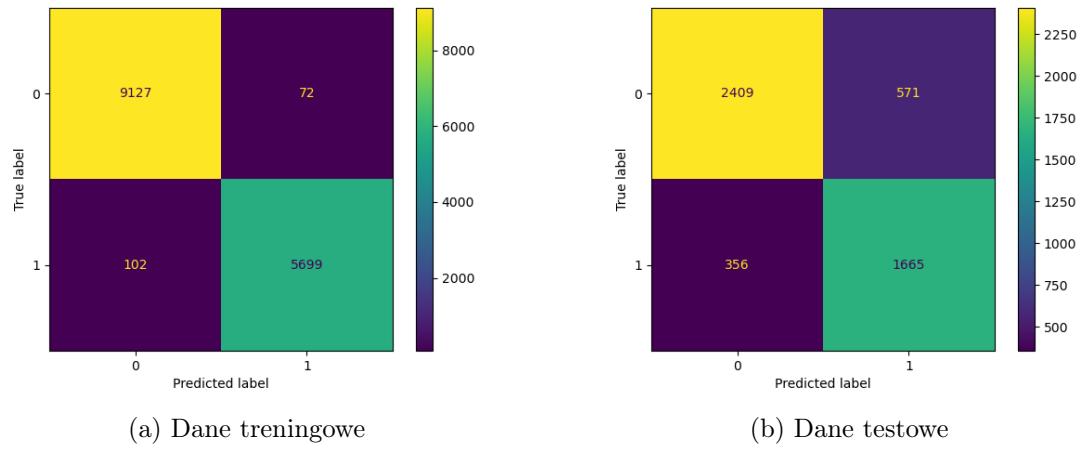
### 6.3.6 K najbliższych sąsiadów



Rysunek 41: Macierz pomyłek dla metody k najbliższych sąsiadów

Widoczna jest duża, jednak zbliżona ilość fałszywie pozytywnych i negatywnych przypisań w przypadku danych treningowych. Na danych testowych ponad dwukrotnie więcej jest elementów fałszywie negatywnych. W porównaniu do poprzednich algorytmów, niewiele jest wartości fałszywie pozytywnych a suma fałszywie przypisanych elementów jest mniejsza, dzięki czemu odnotowuje się większą dokładność modelu.

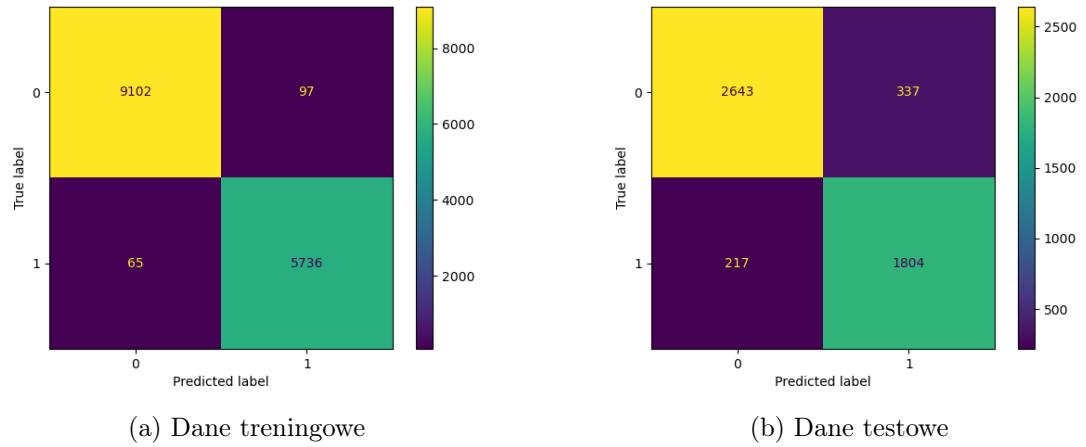
### 6.3.7 Drzewo decyzyjne



Rysunek 42: Macierz pomyłek dla drzewa decyzyjnego

Dla algorytmu Drzewa Losowego, wyniki na danych treningowych są bardzo dobre, widać bardzo niewielką liczbę fałszywych przypisań - jedynie 72 fałszywie pozytywnych i 102 fałszywie negatywnych. Dla zbioru testowego wyniki pogarszają się, jednak widoczna jest przewaga fałszywie pozytywnych nad fałszywie negatywnymi elementami. Oznacza to, że model był bardziej odporny na większą liczebność klasy 0.

### 6.3.8 Las losowy



Rysunek 43: Macierz pomyłek dla lasu losowego

Zdecydowanie najlepiej działającym modelem był ten, wykorzystujący algorytm Lasu Losowego. Na zbiorze treningowym jest jedynie około 150 fałszywych przypisań. Dla zbioru testowego te wartości również są niskie, jednak gorsze niż dla zbioru treningowego. Różnica między fałszywie pozytywnymi i negatywnymi przypisami dla danych testowych nie jest duża, widoczna jest jednak przewaga tych pierwszych, co ponownie świadczy o odporności modelu na większą liczebność jednej z klas. Największa jest liczba poprawnych przypisań - jest ich aż 89%.

## 7 Wnioski

Najlepsze wyniki dla osiągnięto dla ostatniego zbioru danych (Suspicious Communication). Mimo różnic w wielkościach klas, oprócz najlepszego wyniku otrzymano tam najmniejszy overfitting. Wynika to prawdopodobnie z tego, że widoczna była znaczna przewaga sentymentów negatywnych (dużo negatywnych wypowiedzi) nad pozytywnymi dla danych klasyfikowanych jako zawierające podejrzane wypowiedzi. Jest to również jedyny zbiór który nie zawierał dodatkowych informacji poza samymi treściami wpisów - niewiele było użyć emotikon, linków czy postów typu retweet, które w pozostałych zbiorach mogły mieć nieproporcjonalnie duży wpływ na końcową klasyfikację. Zbiór ten nie zawierał również innych kolumn związanych z na przykład lokalizacją (zbiór Disaster) lub ilością obserwujących osób lub opublikowanych dotychczas postów (zbiór Mental Health).

Kolejną zauważalną różnicą między pierwszymi zbiorami a ostatnim widoczna jest w najczęściej występujących słowach. W pierwszych zbiorach mapy słów przygotowane dla danych przypisanych do klasy 1 i 0 znaczaco różnią się od siebie, podczas gdy w ostatnim najczęściej występujące słowa w obu przypadkach są prawie takie same. Ze względu na wykorzystanie wektoryzacji TFIDF (Term Frequency, Inverse Document Frequency), czyli ważenia częstością termów, najczęściej występujące słowa mają mniejszą wagę niż w przypadku obliczania zwykłej częstości termów. Dobrze pasuje to do specyfiki trzeciego zbioru, ponieważ wynik klasyfikacji zależał tu bardziej od mniej popularnych słów i kontekstów niż od samych, pojedynczo występujących wyrażeń.

Skupiając się na samych algorytmach których użyto do przygotowania modeli, można zauważyć że dla pierwszych dwóch zbiorów najlepsze rezultaty dawał algorytm SVM i Naiwnego Bayesa, w czołówce był też algorytm Lasu Losowego. Dla ostatniego zbioru najlepszy okazał się Las Losowy i Drzewo Decyzyjne. Algorytm Lasu Losowego dla każdego z przypadków wykazał się najlepszymi wynikami dla zbiorów treningowych. Las Losowy był jednak również algorytmem, który uczył się najdłużej.

Algorytm K wspólnych sąsiadów okazał się najsłabszy przy pierwszych dwóch zbiorach natomiast dawał dość dobre wyniki w przypadku trzeciego.

## 8 Link do repozytorium

[https://github.com/kraszor/NLP-Group\\_Project](https://github.com/kraszor/NLP-Group_Project)

## 9 Referencje

- [1] Suspicious Communication on Social Platforms [kaggle.com](https://www.kaggle.com/c/suspicious-communication-on-social-platforms)
- [2] Depression: Twitter Dataset + Feature Extraction [kaggle.com](https://www.kaggle.com/c/depression-detection-tweets)
- [3] Natural Language Processing with Disaster Tweets [kaggle.com](https://www.kaggle.com/c/nlp-with-disaster-tweets)
- [4] Dean, B. (2021). Twitter usage and growth statistics: How many people use Twitter in 2021, Retrieved from Backlinko website: [Https://backlinko.com/twitter-users](https://backlinko.com/twitter-users)
- [5] Jain A. P., and P. Dandannavar. (2016) "Application of machine learning techniques to sentiment analysis."2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT): 628-632
- [6] Sangamesh Hosgurmath, Vishwanath Petli, V.K. Jalihal, An omicron variant tweeter sentiment analysis using NLP technique, Global Transitions Proceedings, Volume 3, Issue 1, 2022,215-219, ISSN 2666-285X
- [7] A., Ahmadalipour, A., Abbaszadeh, P., Moradkhani, H. (2020). Leveraging machine learning for predicting flash flood damage in the Southeast US. Environ. Res. Lett., 15, 024011.

- [8] Veny Amilia Fitri, Rachmadita Andreswari, Muhammad Azani Hasibuan, Sentiment Analysis of Social Media Twitter with Case of Anti LGBT Campaign in Indonesia using Naïve Bayes, Decision Tree, and Random Forest Algorithm, Procedia Computer Science 161 (2019) 765–772
- [9] Samer Muthana Sarsam, Hosam Al-Samarraie, Ahmed Ibrahim Alzahrani, Abdul Samad Shibli, A non-invasive machine learning mechanism for early disease recognition on Twitter: The case of anemia, Artificial Intelligence in Medicine, Volume 134, 2022, 102428, ISSN 0933-3657,
- [10] Tirta Hema Jaya Hidayat, Yova Ruldeviyani, Achmad Rizki Aditama, Gusti Raditia Madya, Ade Wija Nugraha, Muhammad Wijaya Adisaputra, Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier, Procedia Computer Science 197,2022, 660-667,ISSN 1877-0509,
- [11] Shuai, Q., Y. Huang, L. Jin, and L. Pang. (2018) "Sentiment Analysis on Chinese Hotel Reviews with Doc2Vec and Classifiers."2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC): 1171-1174.
- [12] Tomas PRANCKEVIČIUS, Virginijus MARCINKKEVIČIUS, Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification Baltic J. Modern Computing, Vol. 5 (2017), No. 2, 221-232
- [13] Tweet NLP, <http://www.cs.cmu.edu/~ark/TweetNLP/>.
- [14] Dharini Ramachandran, R Parvathi, Analysis of Twitter Specific Preprocessing Technique for Tweets, Procedia Computer Science,Volume 165,2019,245-251, ISSN 1877-0509
- [15] Karimiziarani M., Jafarzadegan K., Abbaszadeh P., Shao W., Moradkhani H. Hazard risk awareness and disaster management: Extracting the information content of twitter data, Sustainable Cities and Society 77,2022, 103577