

# Rozpoznawanie wiadomości dotyczących katastrof

Warsztaty z technik uczenia maszynowego

Jakub Rymuza      Karol Nowiński

1 czerwca 2022

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
<b>2</b>	<b>Podział pracy</b>	<b>3</b>
<b>3</b>	<b>Wykorzystywane narzędzia</b>	<b>3</b>
<b>4</b>	<b>Użyte dane</b>	<b>3</b>
<b>5</b>	<b>Analiza danych</b>	<b>4</b>
5.1	Podstawowe informacje . . . . .	4
5.2	Długość tweetów . . . . .	4
5.3	Skład danych treningowych . . . . .	5
5.4	Słowa kluczowe . . . . .	6
<b>6</b>	<b>Przygotowanie danych</b>	<b>7</b>
6.1	Czyszczenie danych . . . . .	7
6.2	Tokenizacja . . . . .	8
6.3	Stemizacja . . . . .	8
6.4	Lematyzacja . . . . .	9
6.5	Wektoryzacja . . . . .	9
6.6	Wyniki . . . . .	9
<b>7</b>	<b>Modele</b>	<b>10</b>
<b>8</b>	<b>Wnioski</b>	<b>13</b>
<b>9</b>	<b>Bibliografia</b>	<b>14</b>

# 1 Wstęp

Projekt polega na przetwarzaniu wiadomości z serwisu Twitter w celu ustalenia, które z nich mówią o katastrofach. Może to pomóc m.in. we wcześniejszym rozpoznawaniu sytuacji zagrażających życiu. Może to też pomóc służbom w ocenie skali zagrożenia zdarzeniem.

## 2 Podział pracy

W związku z tym, że grupa składa się z dwóch osób, zdecydowaliśmy, że nie będziemy dzielić pracy na poszczególne role. Praca wykonywana będzie w większości wspólnie, pracując wspólnie nad rozwiązaniem problemu.

## 3 Wykorzystywane narzędzia

Projekt opiera się na procesowaniu języka naturalnego za pomocą technik uczenia maszynowego. Projekt zostanie wykonany w języku Python. W szczególności wykorzystane zostaną biblioteki *NumPy* oraz *Pandas* z wykorzystaniem notatników *Jupyter*.

## 4 Użyte dane

W projekcie wykorzystano dwa zbiory danych z [1]:

- zbiór treningowy *train.csv*, który zostanie wykorzystany do wytrenowania modelu do rozwiązywania problemu. Zawiera on 7613 rekordów. Rekordy zawierają następujące pola:
  - *id* - identyfikator rekordu,
  - *keyword* - słowo kluczowe uprzednio wyciągnięte z wiadomości - wszystkie słowa kluczowe dotyczą katastrof, są to słowa takie jak na przykład "crash", "earthquake" i tym podobne. To pole jest opcjonalne, tzn. nie każdy rekord je zawiera,
  - *location* - lokalizacją z której wiadomość została wysłana. Podobnie jak *keyword*, jest to pole opcjonalne,
  - *text* - najważniejsza pole - zawiera treść wiadomości,

- *target* - wartość logiczna stwierdzająca czy dana wiadomość mówi o katastrofie czy nie.
- zbiór testowy *test.csv* - zbiór na którym model będzie testowany. Zawiera on 3263 rekordów. Rekordy wyglądają tak samo, jak w przypadku zbioru treningowe, za wyjątkiem oczywiście braku pola *target*, którego obliczenie jest celem projektu.

## 5 Analiza danych

### 5.1 Podstawowe informacje

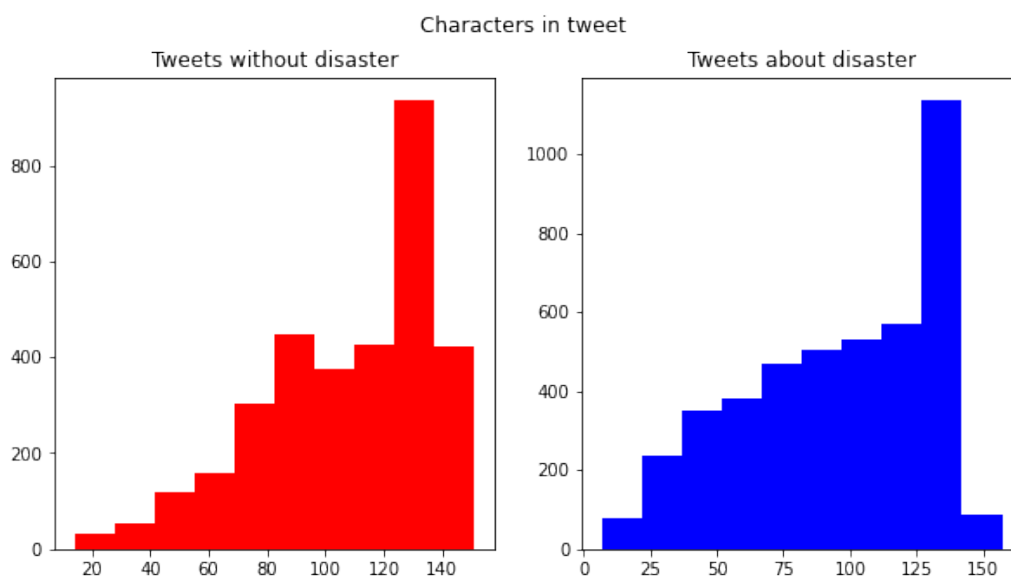
W zbiorze *train* składającym się z 7613 rekordów, tylko 5080 posiada podaną lokację, a 7552 posiada podane słowo kluczowe (*keyword*).

Natomiast w zbiorze *test* na 3263 rekordów, tylko 2158 posiada podaną lokację, a 3237 posiada podany *keyword*.

Wynika stąd, że w obu zbiorach prawie wszystkie rekordy mają podane słowo kluczowe, natomiast tylko około dwie trzecie z nich mają podaną lokację.

### 5.2 Długość tweetów

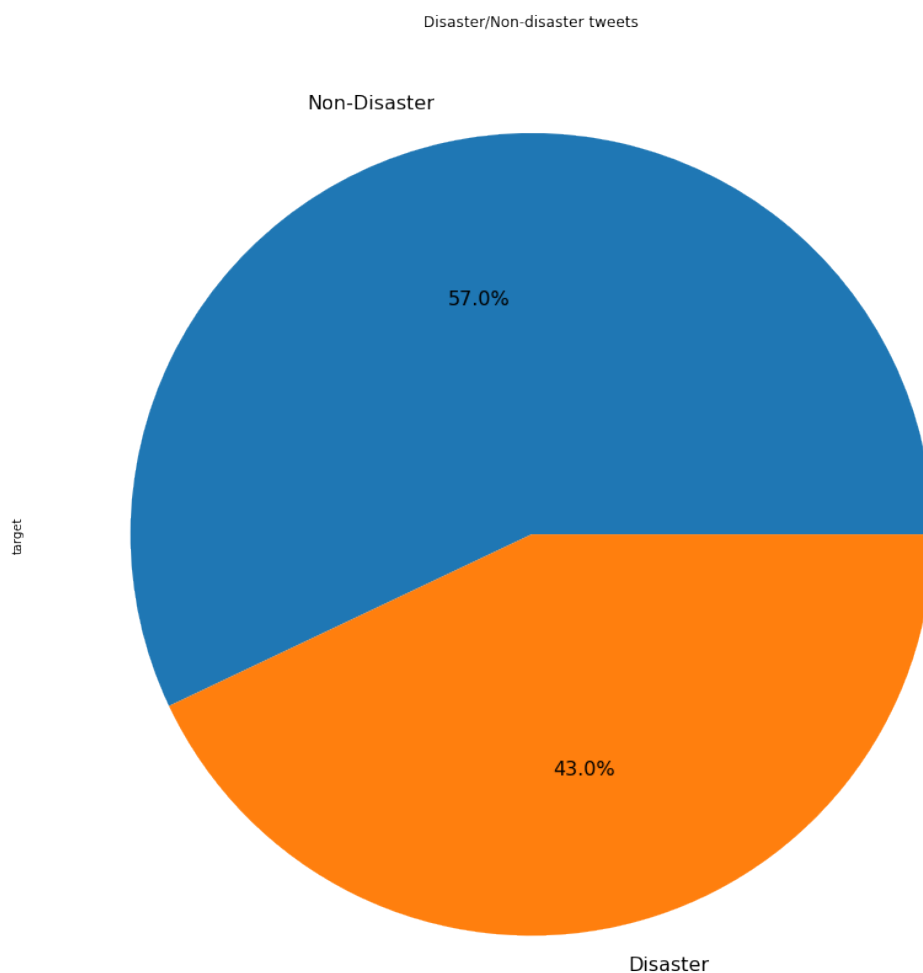
Na rysunku 1 umieszczono histogramy liczby tweetów w danym przedziale długości w zależności od tego czy mówią o katastrofie czy nie dla zbioru *train*. Można zauważyć, tweety o katastrofie częściej są krótkie lub średniej długości, natomiast rzadko są bardzo długie. Natomiast w obu przypadkach pik występuje dla długości około 120 znaków.



Rysunek 1: Znaki w tweet'cie

### 5.3 Skład danych treningowych

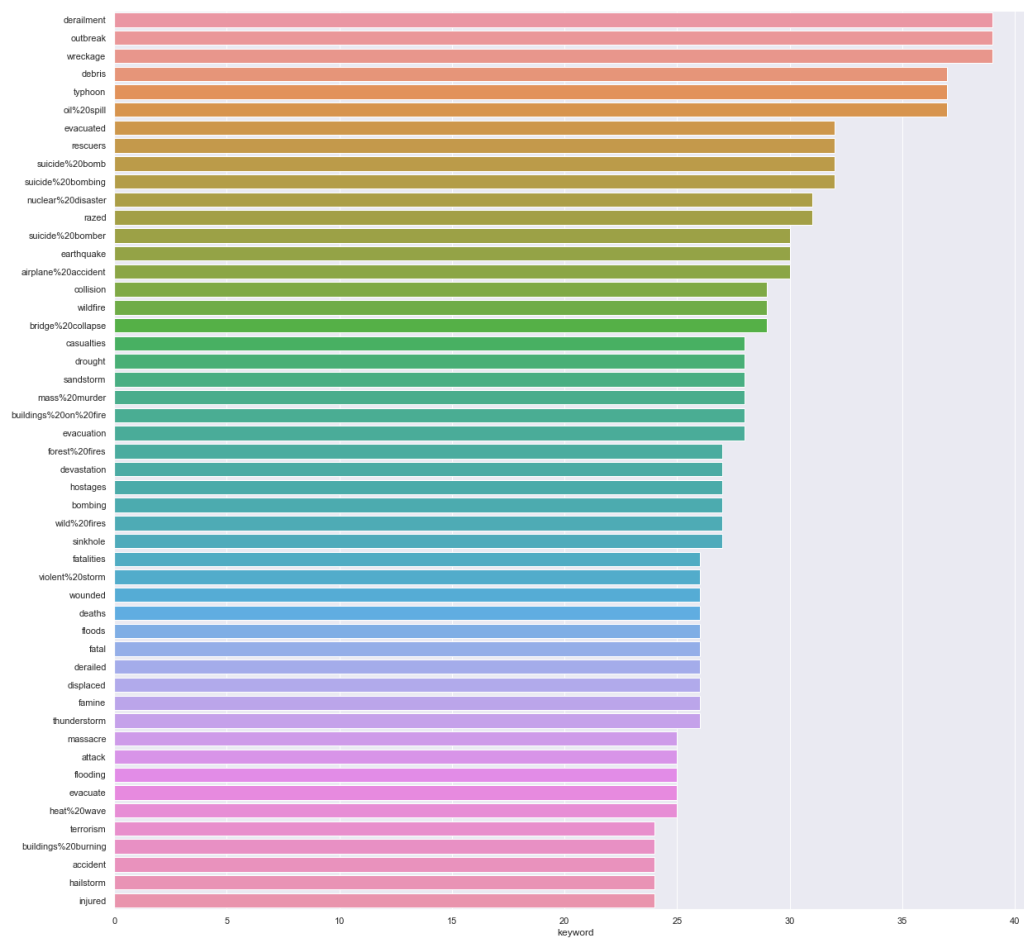
Na rysunku 2 zaznaczono skład zbioru *train* ze względu na ilość tweetów o katastrofach. Jak widać dane są rozmieszczone dość równomiernie, z niewielką przewagą tweetów niemówiących o katastrofach. Taki skład danych treningowych umożliwia dobre wytrenowanie modelu.



Rysunek 2: Stosunek tweetów o katastrofach i nie o katastrofach

## 5.4 Słowa kluczowe

Słowa kluczowe mogą pełnić ważną rolę przy klasyfikacji tweetu. Na rysunku 3 przedstawiono popularność poszczególnych słów kluczowych w zbiorze *train*. Jak widać najpopularniejsze słowa kluczowe to *derailment*, *outbreak* i *wreckage*.



Rysunek 3: Popularność słów kluczowych

## 6 Przygotowanie danych

### 6.1 Czyszczenie danych

W celu poprawienia treningu modelu przeprowadzono następujące operacji "czyszczące" dane:

- Usunięcie z tekstu kodów - "%20" (kod ASCII spacji) przekształcono na spację, zaś "&" przekształcono na słowo "and".
- Przekształcenie całego tekstu do małych liter. Dzięki temu słowa *fire* i

*Fire* będą traktowane jako to samo słowo, a nie dwa różne.

- Usunięcie linków z danych - nasz model nie będzie w stanie wchodzić w linki i ich analizować, więc tylko utrudniałyby one niepotrzebnie proces uczenia.
- Poprawienie błędów konwersji w danych - w używanych danych w niektórych miejscach można znaleźć ciągi `%20` (kod ASCII spacji) zamiast znaku spacji. Może to spowodować, że na przykład fraza *building%20on%20fire* będzie traktowana jako jedno słowo i nie rozpozna że chodzi tam o coś związanego z ogniem.
- Rozwinięcie skrótów - na przykład zamiana *asap* na *as soon as possible*.
- Usunięcie tzw. *stopwords*. Są to słowa typu *the* czy *is*, które wnoszą niewiele informacji do treści i mogą zostać pominięte przy treningu modelu.

Po "wyczyszczeniu" danych we wcześniejszej części, kolejnym krokiem mającym przygotować dane do efektywnego wykorzystania przez modele były: tokenizacja, stemizacja, lematyzacja oraz wektoryzacja.

## 6.2 Tokenizacja

Krok ten polega na podziale tekstu (ciągu słów) na tablicę pojedynczych słów. Wykorzystano dwie metody tokenizacji:

- Regexp Tokenization - ten typ tokenizacji przy podziale na słowa, wyrzuca wszelkie znaki interpunkcyjne.
- Treebank Tokenization - ten typ tokenizacji zachowuje wszystkie znaki interpunkcyjne.

## 6.3 Stemizacja

Stemizacja (ang. *steeming*) usuwa ze słów końcówki, zachowując jedynie tzw. temat wyrazu (ang. *stem*). Temat wyrazu to część wspólna wszystkich wyrazów z danej rodziny. Na przykład tematem słowa "residents" jest "resid", a słowa "asked" jest "ask".



## 6.4 Lematyzacja

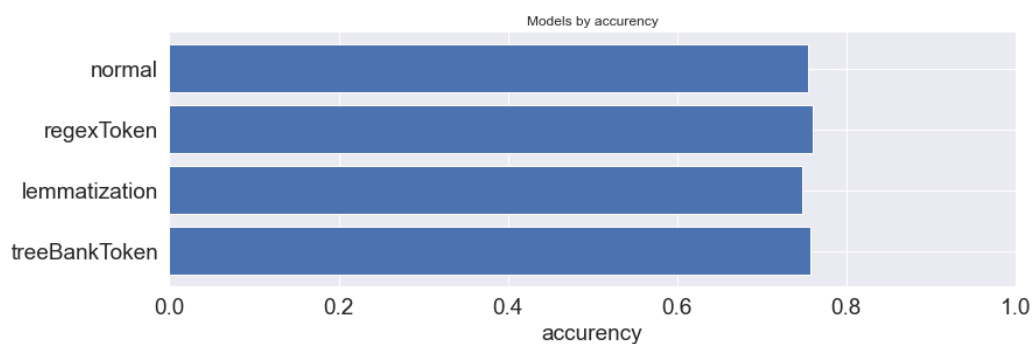
Lematyzacja (ang. lemmatization) przekształca słowa do podstawowej, "słownikowej" formy. Na przykład "is" oraz "are" są przekształcane do "be", zaś "cars" do "car".

## 6.5 Wektoryzacja

Wektoryzacja to technika przekształcająca ciąg słów (tokenów) na tablicę liczb, którą wykorzystują modele. Wybrany typ wektoryzacji to wektoryzacja typu CountVectorizer - zlicza on po prostu ilość tokenów danego typu i zapisuje w rzadkiej macierzy. Innym typem wektoryzacji jest wektoryzacja TF-IDF, która bierze pod uwagę to jak rzadki jest dany token w tekście. Ta metoda jednak nie przyniosła lepszych wyników, a nawet je pogorszyło. Mogło to wynikać z wcześniejszych faz przygotowania danych (przede wszystkim usunięcie tzw. stop words).

## 6.6 Wyniki

Poniższy wykres przedstawia porównanie różnych metod tokenizacji, stemizacji oraz lematyzacji po użyciu ich na modelu klasyfikatora drzewa decyzyjnego. Jak widać zyski z zastosowania tych metod są dla naszych danych znikome, a wręcz mogą nieco pogarszać wyniki. Z tego względu w dalszej części będziemy rozważać wyniki dla których nie zastosowano lematyzacji ani stemizacji.

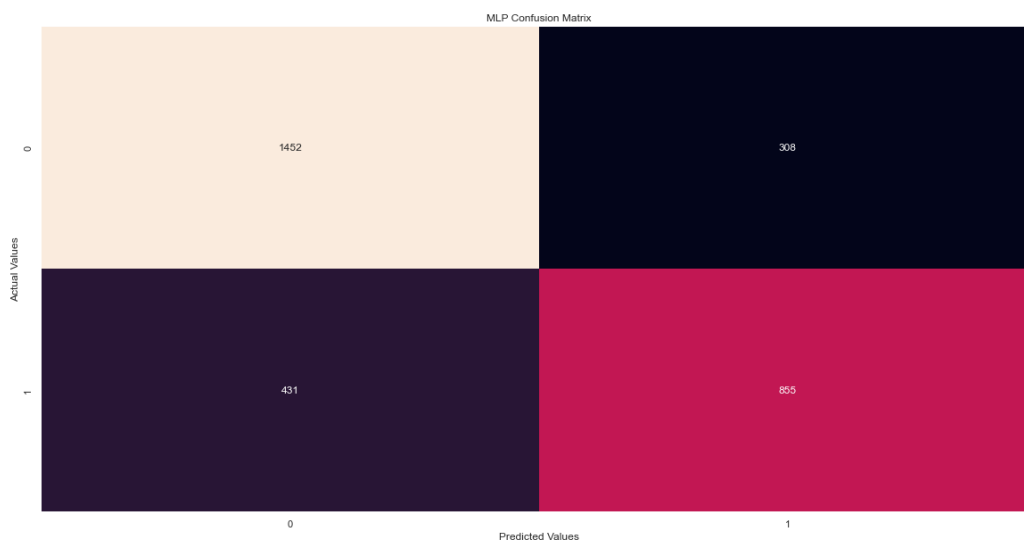


Rysunek 4: Porównanie metod tokenizacji, stemizacji oraz lematyzacji

## 7 Modele

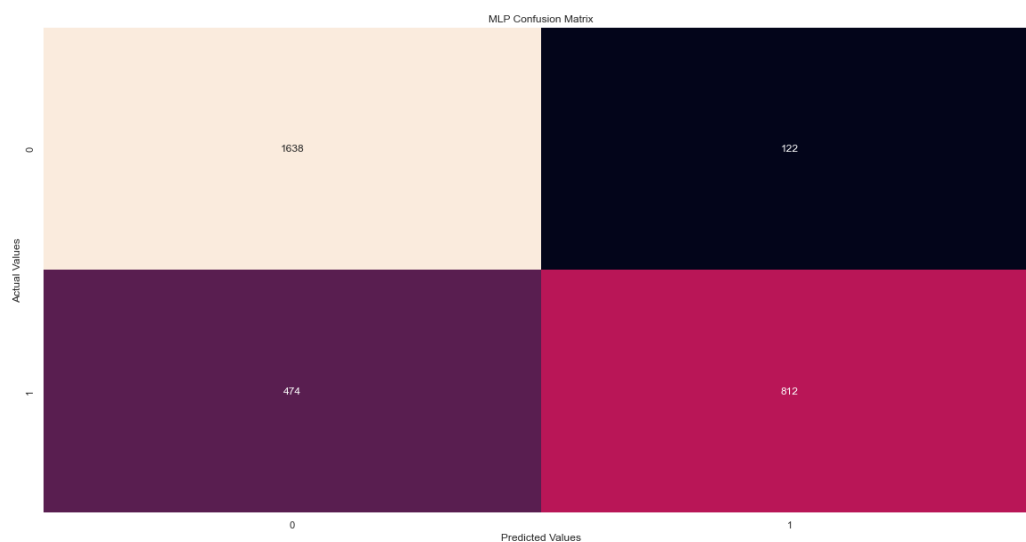
Po przygotowaniu danych, użyto różnych modeli w celu sprawdzeniu, który z nich jest najlepszy. Użyte modele wraz z dokładnością (accuracy) oraz tablicami pomyłek (confusion matrices):

- Klasyfikator drzewa decyzyjnego (decision tree classifier) - dokładność - 75.73%



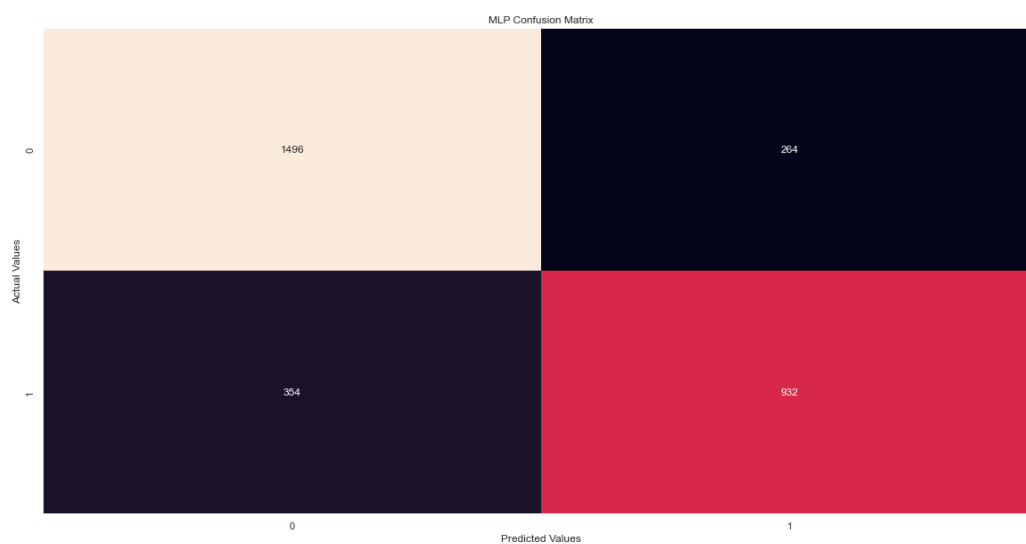
Rysunek 5: Tablica pomyłek dla klasyfikator drzewa decyzyjnego

- Naiwny klasyfikator bayesowski (Naive Bayes classifier) - dokładność - 80.43%



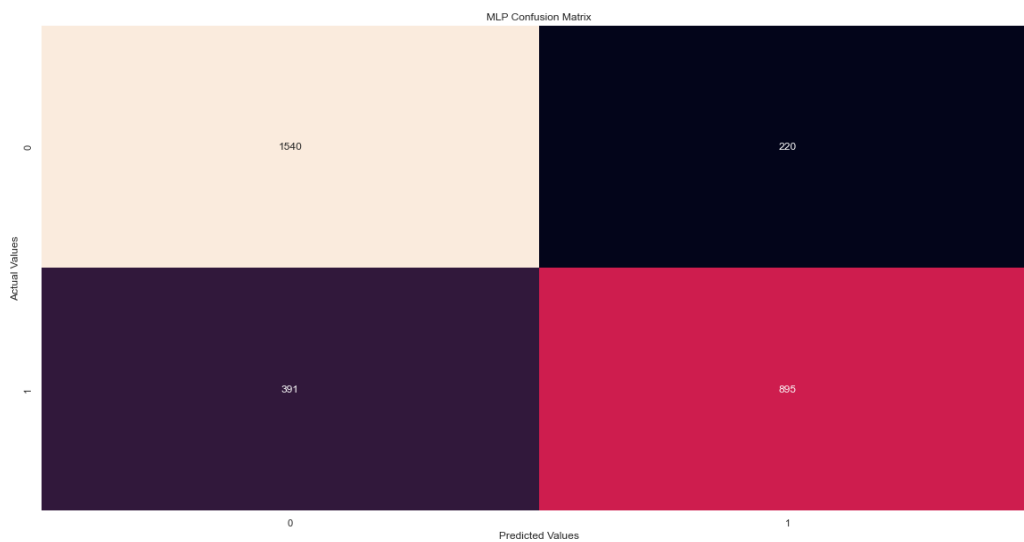
Rysunek 6: Tablica pomyłek dla naiwnego klasyfikatora bayesowskiego

- Wielomianowy naiwny klasyfikator bayesowski (multinomial naive Bayes classifier) - dokładność - 79.71%



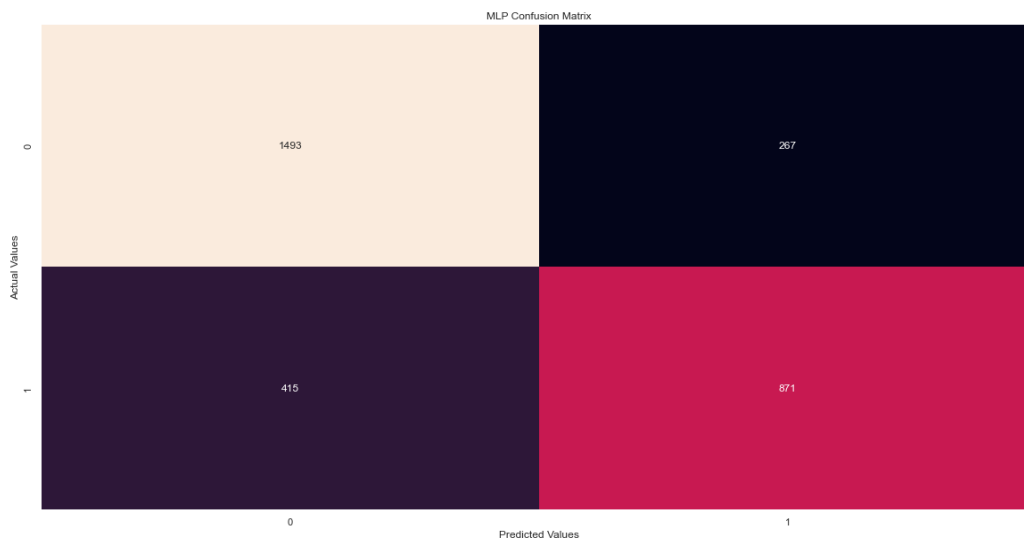
Rysunek 7: Tablica pomyłek dla wielomianowego naiwnego klasyfikatora bayesowskiego

- Klasyfikator regresji logistycznej (logistic regression classifier) - dokładność - 79.94%



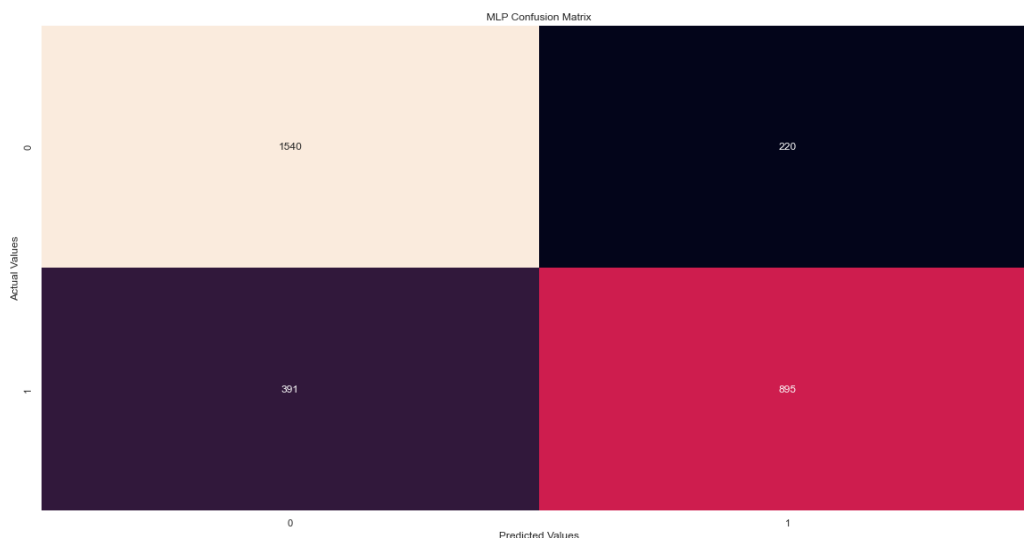
Rysunek 8: Tablica pomyłek dla klasyfikatora regresji logistycznej

- Klasyfikator regresji grzbietowej (ridge regression classifier) - dokładność - 77.61%



Rysunek 9: Tablica pomyłek dla klasyfikatora regresji grzbietowej

- Klasyfikator lasu losowego (random forest classifier) - dokładność - 78.92%

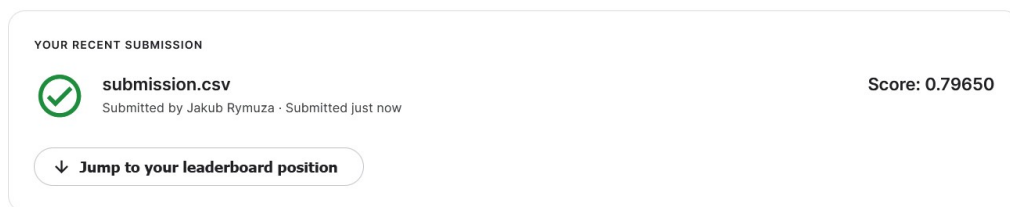


Rysunek 10: Tablica pomyłek dla klasyfikatora lasu losowego

## 8 Wnioski

W wyniku projektu udało się stworzyć system automatycznie klasyfikujący treść tweetów. W dzisiejszym świecie pełnym informacji jest to bardzo ważna funkcja. Wśród wykorzystanych modeli, najlepsze wyniki uzyskał naiwny klasyfikator bayesowski - powyżej 80%. Bardzo zbliżone wyniki uzyskał też klasyfikator regresji logistycznej oraz wielomianowy naiwny klasyfikator bayesowski. Dokładność powyżej 80% można w uczeniu maszynowym można już traktować jako dość wysoki, zatem cel projektu można uznać za zrealizowany.

Przy wysłaniu przesłania wyniku do Kaggle uzyskano następujący wynik:



Rysunek 11: Wyniki z Kaggle

Stemizacja i leinizacja dla danych użytych w projekcie nie dawała wymiernych korzyści, dlatego warto z niej zrezygnować w celu przyspieszenia procesu klasyfikacji.

## 9 Bibliografia

- [1] Treść zadania na stronie Kaggle - <https://www.kaggle.com/c/nlp-getting-started>
- [2] O. Duda, P. Hart, D. Stork - Pattern Classification, 2nd Edition
- [3] N. Indurkha, F. Damerau - Handbook of Natural Language Processing, 2nd Edition