

**Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej**

Projekt dyplomowy

*Rozpoznawanie tzw. „fake newsów”*

*na wąskim obszarze tematycznym*

*Recognition of so called „fake news”*

*on specified topic scope*

Autor: *Konrad Daniel Golemo*

Kierunek studiów: Automatyka i Robotyka

Opiekun pracy: *dr hab. inż. Jerzy Baranowski*

Kraków, 2022

**Spis treści**

[1. Wstęp 4](#_Toc91607912)

[1.1. Wprowadzenie 4](#_Toc91607913)

[1.2. Cel i zakres pracy 4](#_Toc91607914)

[1.3. Układ pracy 4](#_Toc91607915)

[2. Analiza aktualnego stanu wiedzy 5](#_Toc91607916)

[2.1. Uczenie maszynowe 5](#_Toc91607917)

[2.2. Przetwarzanie języka naturalnego 5](#_Toc91607918)

[2.3. Wyniki realizacji podobnych projektów 5](#_Toc91607919)

[2.3.1. Detekcja tzw. „fake newsów” w języku angielskim 5](#_Toc91607920)

[2.3.2. Detekcja tzw. „fake newsów” w języku niemieckim 8](#_Toc91607921)

[2.3.3. Detekcja tzw. „fake newsów” w języku hiszpańskim 9](#_Toc91607922)

[3. Realizacja projektu 10](#_Toc91607923)

[3.1. Wybór obszaru tematycznego i tworzenie zbiór danych 10](#_Toc91607924)

[3.2. Przetwarzanie zbioru danych 11](#_Toc91607925)

[3.3. Wybrane algorytmy przetwarzania języka naturalnego 11](#_Toc91607926)

[3.3.1. Naiwny klasyfikator bayesowski 11](#_Toc91607927)

[3.3.2. Maszyna wektorów nośnych 12](#_Toc91607928)

[3.3.3. Las losowy 13](#_Toc91607929)

[4. Rezultaty 14](#_Toc91607930)

[4.1. Testy jednostkowe 14](#_Toc91607931)

[4.2. Ewaluacja algorytmów 14](#_Toc91607932)

[5. Podsumowanie 15](#_Toc91607933)

[5.1. Zrealizowane założenia projektowe 15](#_Toc91607934)

[5.2. Wnioski 15](#_Toc91607935)

[5.3. Kierunki dalszego rozwoju 15](#_Toc91607936)

[6. Bibliografia 16](#_Toc91607937)

# 1. Wstęp

## 1.1. Wprowadzenie

Zjawisko tzw. „fake newsów” dotyczy wielu tematów i dziedzin, m.in. wiadomości z zakresu polityki, technologii, historii czy zdrowia. Znaczny wzrost zainteresowania tym zagadnieniem nastąpił po wyborach prezydenckich w Stanach Zjednoczonych w 2016 roku [1] [2]. Dalej o tematyce fake newsów…

## 1.2. Cel i zakres pracy

Rozbudowanie tego co poniżej…

Celem pracy jest przetestowanie podstawowych metod klasyfikacji tekstu do wykrywania tzw. „fake newsów” pochodzących z pewnego wybranego wąskiego obszaru.

Od opracowanego oprogramowania oczekuje się:

- wykorzystania systemu kontroli wersji (git, najchętniej Github)

- pokrycia projektu testami jednostkowymi i wykorzystania mechanizmu Continuous Integration (np. Travis CI)

- zapewnienia dokumentacji (dowolne narzędzie)

- minimalnego front endu, np. w postaci aplikacji webowej w Dashu opublikowanej na Heroku.

- kod po zakończeniu pracy ma być publiczny, licencja dowolna

## 1.3. Układ pracy

Bardziej rozbudowany spis treści – opis tego, co znajduje się w każdym rozdziale

# 2. Analiza aktualnego stanu wiedzy

## 2.1. Uczenie maszynowe

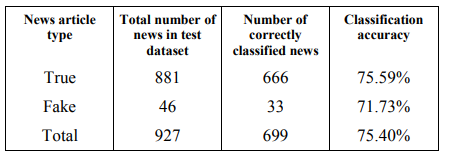
## 2.2. Przetwarzanie języka naturalnego

## 2.3. Wyniki realizacji podobnych projektów

### 2.3.1. Detekcja tzw. „fake newsów” w języku angielskim

W publikacji [3] wykorzystano zbiór danych zgromadzony przez serwis BuzzFeed News, zawierający 2282 artykułów o tematyce politycznej przypisanych ręcznie do jednej z czterech kategorii: „głównie prawda”, „głównie fałsz”, „mieszanina prawdy i fałszu” oraz „brak treści merytorycznej”. Podczas implementacji klasyfikatora zredukowano liczebność zestawu poprzez odrzucenie artykułów nienależących do grupy „głównie prawda” lub „głownie fałsz” oraz przypadków wadliwych – rekordów niezawierających żadnego tekstu. Finalnie otrzymano zbiór zawierający łącznie 1771 artykułów, spośród których ok. 5% stanowiły tzw. „fake newsy”.

Do wykrywania tekstów zawierających nieprawdziwe informacje wykorzystano naiwny klasyfikator bayesowski. Dane podzielono na trzy zestawy: zbiór uczący, walidacyjny i testowy. W zaproponowanej procedurze klasyfikacji iterowano po każdym wyrazie w analizowanym artykule, a o przydzieleniu do jednej z dwóch grup decydowało prawdopodobieństwo warunkowe wyznaczone na podstawie zawartych w tekście pewne konkretnych wyrazów. Po wytrenowaniu klasyfikatora i dostrojeniu jego parametrów przystąpiono do testów, w wyniku których uzyskano ok. 75% dokładności klasyfikacji (Tabela X)

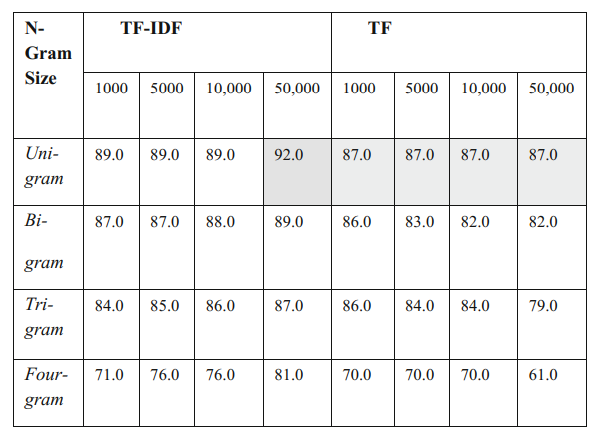


W celu poprawy działania klasyfikatora zaproponowano m.in. usunięcie z artykułów wyrazów należących do stop listy, wykorzystanie tzw. „stemmingu” (doprowadzenie wyrazów do ich podstawowych form poprzez usunięcie formantu [4]), czy wyznaczanie prawdopodobieństwa nie na pojedynczych wyrazach, lecz na ich grupie.

W pracy [5] zgromadzono 25 200 artykułów o tematyce politycznej, spośród których połowa stanowią wiado mości prawdziwe, a długość każdego z tekstów jest nie mniejsza niż 200 znaków. Dane podzielono pomiędzy zbiór treningowy (80%) i testowy (20%). Do określenia przestrzeni cech użyto modelu -gramowego, a następnie zredukowano liczbę tych cech przy pomocy metody TF (ang. *term frequency*) oraz TFIDF (ang. *term frequency - inverse document frequency*).

Klasyfikację przeprowadzono za pomocą metody stochastycznego gradientu (ang. *Stochastic Gradient Descent,* SGD), maszyny wektorów nośnych (ang. *Support Vector Machine, SVM*), liniowej maszyny wektorów nośnych (ang. Linear *Support Vector Machine, LSVM*), algorytmu -najbliższych sąsiadów (ang. *K-Nearest Neighbour, KNN*) oraz drzewa decyzyjnego (ang. *Decision Tree, DT*). W trakcie trenowania algorytmów wykorzystano 5-krotny sprawdzian krzyżowy (ang. *k-fold cross-validation*).

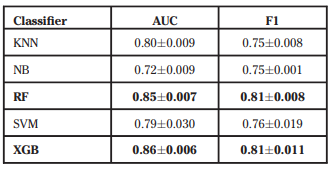
W eksperymencie porównano obie metody ekstrakcji cech przy zmiennym rozmiarze N-gramu dla każdego z zastosowanych algorytmów klasyfikacji. Największą dokładność na poziomie 92% osiągnięto dla metody TFIDF z -gramem o rozmiarze 1 i algorytmem LSVM (Tabela X).



W artykule [6] wykorzystano ten sam zbiór danych, co w pozycji [3] (2282 tekstów politycznych), lecz zaproponowano inną metodę podziału na treści wiarygodne i nieprawdziwe. Odrzucono wiadomości oznaczone etykietą „brak treści merytorycznej”, a te należące do grup „głównie fałsz” i „mieszanina prawdy i fałszu” połączono i potraktowano jako teksty fałszywe (342 pozycje) . Pozostałe, należące do klasy „głównie prawda”, przydzielono do kategorii artykułów prawdziwych (1666 pozycje).

Klasyfikacji dokonano poprzez zastosowanie następujących algorytmów: -najbliższych sąsiadów (ang. *K-Nearest Neighbour, KNN*), naiwny klasyfikator bayesowski, las losowy (ang. *Random Forest, RF*), maszynę wektorów nośnych z radialną funkcją bazową (ang. *Support Vector Machine with RBF kernel, SVM*) oraz tzw. „XGBoost”. Cechy artykułów wyznaczono ręcznie, wyróżniając trzy kategorie: cechy tekstu (m.in. właściwości językowe, leksykalne czy semantyczne), cechy źródła danych (np. uprzedzenia polityczne czy wiarygodność) i cechy otoczenia (takie jak liczba interakcji użytkowników z postem).

Ewaluację klasyfikacji przeprowadzono za pomocą współczynnika -score oraz obliczonego pola pod krzywą ROC (ang. *Receiver Operating Characteristic*), obrazującą relacje między czułością i specyficznością modelu. Najlepsze rezultaty uzyskano dla algorytmów lasu losowego i tzw. „XGBoost” (Tabela X).



[Porównanie]

Opis różnic, jeśli zastosowano te same algorytmy to z czego może wynikać różnica w wynikach?

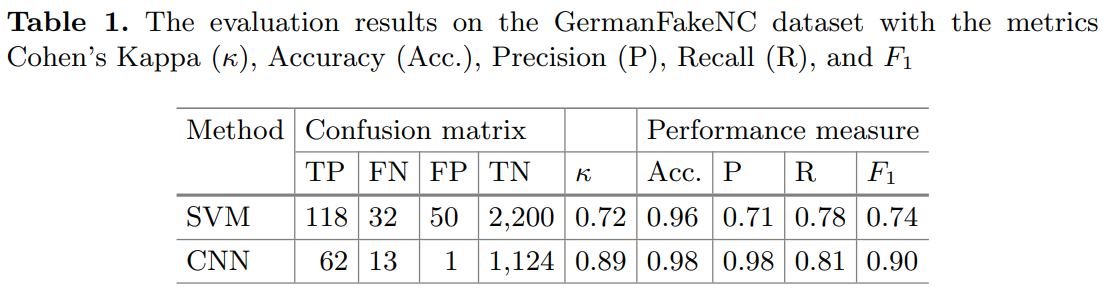
### 2.3.2. Detekcja tzw. „fake newsów” w języku niemieckim

W artykule [7] zgromadzono zbiór danych zawierający 490 artykułów przekazujących nieprawdziwe informacje oraz 4500 wiadomości opublikowanych przez wiarygodnych wydawców głównego nurtu. Zastosowano dwie metody klasyfikacji: jedną opierającą się na maszynie wektorów nośnych (ang. *Support Vector Machine, SVM*), drugą na konwolucyjnej sieci neuronowej (ang. *Convolutional Neural Network, CNN*). Do wytrenowania klasyfikatorów wykorzystano teksty zawierające co najmniej 200 słów, co zredukowało liczbę tzw. „fake newsów” z 490 do 300.

W przypadku klasyfikacji z wykorzystaniem algorytmu SVM połowę zbioru artykułów wykorzystano do treningu, drugą połowę przeznaczając na testy. Na podstawie zestawu danych, za pomocą metody TFIDF (ang. *term frequency - inverse document frequency*) określono wagę słów w tekście i wyznaczono odpowiadające im wektory cech.

W ramach klasyfikatora opierającego się na CNN dane podzielono na trzy zestawy – treningowy (50% zbioru), walidacyjny (25%) i testowy (25%), a w celu wytrenowania architektury sieci neuronowej wykorzystano wcześniej przygotowane tzw. „word embeddings” – wektorową reprezentację słów, w której wyrazy o podobnym znaczeniu znajdują się blisko siebie.

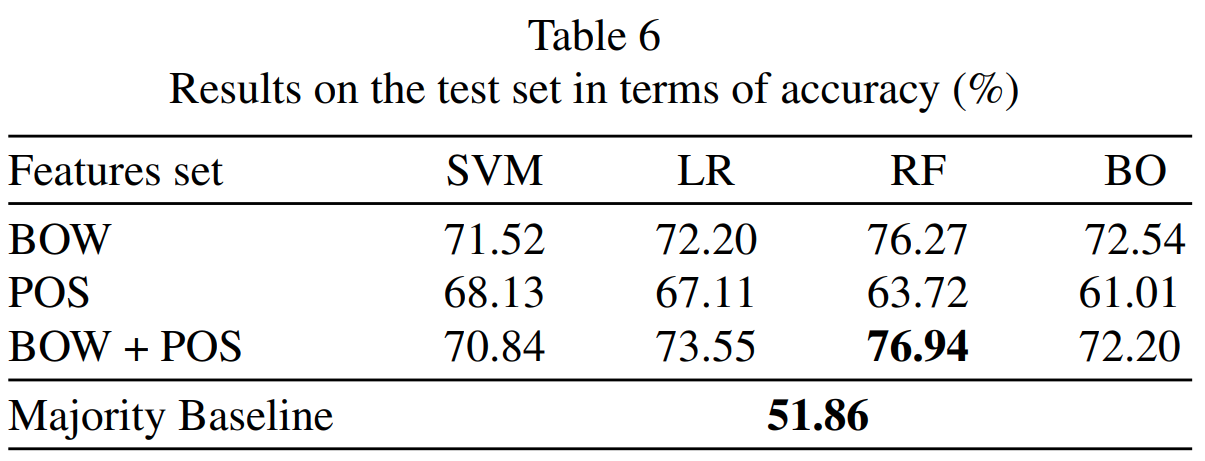
W wyniku ewaluacji za pomocą współczynnika -score, dla klasyfikatora z algorytmem SVM otrzymano , a dla metody wykorzystującej CNN uzyskano . Pełny raport zawarty jest w Tabeli X.



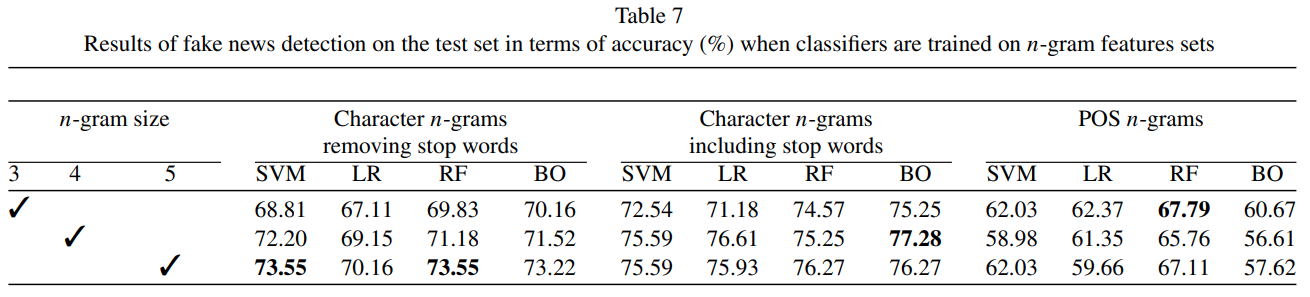
### 2.3.3. Detekcja tzw. „fake newsów” w języku hiszpańskim

W pracy [8] zebrano łącznie 971 artykułów z 9 kategorii tematycznych, w tym 491 oznaczonych jako prawdziwe i 480 jako fałszywe, które następnie podzielono na zestawy treningowe (70% zbioru) i testowe (30%). Określono cztery modele reprezentacji cech: tzw. „bag-of-words”, gramatyczne tagowanie (ang. *grammatical tagging, part-of-speech tagging, POS tagging*), znakowe -gramy oraz POS -gramy i cztery metody klasyfikacji: maszynę wektorów nośnych, regresję logistyczną (ang. *logistic regression*), las losowy (ang. *random forest*) oraz tzw. „boosting”.

Porównano działanie klasyfikatorów trenowanych na zbiorach indywidualnych cech („bag-of-words”, gramatyczne tagowanie oraz połącznie obu), rezultaty w postaci procentowej wartości dokładności zamieszczono w Tabeli X.



Zestawiono także modele bazujące na reprezentacji cech za pomocą -gramów o rozmiarach od 3 do 5. W przypadku znakowych -gramów rozróżniono dwa scenariusze – jeden zakładający usuwanie wyrazów należących do stop listy (ang. *stop words*), drugi pozostawianie ich.



W publikacji lepsze rezultaty uzyskano w przypadku korzystania z reprezentacji -gramowej (77,58% dokładności dla znakowego -gramu o rozmiarze 4 i metodzie „boosting” i 76,94% dokładności dla kombinacji „bag-of-words” z gramatycznym tagowaniem i algorytmu lasu losowego).

# 3. Realizacja projektu

## 3.1. Wybór obszaru tematycznego i tworzenie zbiór danych

Na początku roku 2020 w związku z pandemią COVID-19 zaczęto obserwować znaczną liczbę tzw. „fake newsów” podważających istnienie wirusa czy jego śmiertelność [9]. W uwagi na mnogość wiadomości dotyczących koronawirusa, zarówno prawdziwych, jak i fałszywych, w niniejszej pracy zdecydowano się skupić właśnie na tym obszarze tematycznym.

Do stworzenia korpusu mogącego posłużyć jako dane w procesie klasyfikacji, konieczne było zgromadzenie odpowiedniej liczby tekstów zawierających prawdziwe albo fałszywe informacje. W tym celu wykorzystano tzw. „web scraping”, czyli technikę pozyskiwania informacji z witryn internetowych [10].

Nieprawdziwe wiadomości pobrano z portalu fakehunter.pap.pl [11]. Na jego łamach, na podstawie wiarygodnych źródeł, weryfikowana jest poprawność informacji przesłanych przez użytkowników. Przeanalizowane doniesienia są następnie publikowane w formie artykułów zawierających werdykt („Fake News” albo „Prawda”) oraz raport eksperta wraz ze zdjęciem całości zgłoszonego tekstu lub jego fragmentu, a także link do źródła wiadomości.

Jako źródło wiarygodnych wiadomości przyjęto serwis informacyjny firmy Termedia [12] publikujący artykuły o tematyce medycznej.

Do dokończenia…

## 3.2. Przetwarzanie zbioru danych

### 3.2.1. Oczyszczanie danych

### 3.2.2. Ekstrakcja cech tekstu

## 3.3. Wybrane algorytmy klasyfikacji

### 3.3.1. Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski (ang. *Naive Bayes Classifier*) to probabilistyczna metoda klasyfikacji, której zadaniem jest wyznacznie zgodnie z twierdzeniem Bayesa najbardziej prawdopodobnej klasy dla danego wektora zmiennych. Istotnym uproszczeniem znacznie usprawniającym proces uczenia (zwłaszcza przy dużej liczbie zmiennych) jest założenie o wzajemnej niezależność predyktorów, umożliwiające wyznaczanie parametrów dla każdej ze zmiennych oddzielnie. [13]

Zgodnie z [14] prawdopodobieństwo przynależności wektora cech tekstu do klasy jest równe:

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

Uwzględniając założenie o niezależności predyktorów:

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

Zgodnie z [15], z uwagi na fakt, że mianownik jest stały, reguła klasyfikacji określona jest zależnością:

|  |  |  |
| --- | --- | --- |
|  |  | (1.3) |

a więc najbardziej odpowiednią klasę dla danego wektora cech tekstu stanowi:

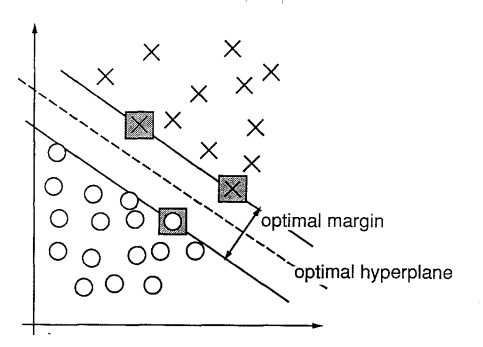
|  |  |  |
| --- | --- | --- |
|  |  | (1.4) |

gdzie wartości częstości występowania klasy w zbiorze danych () oraz częstości występowania cechy w tekstach należących do klasy () są szacowane na podstawie zestawu danych uczących.

W naiwnym klasyfikatorze bayesowskim z biblioteki scikit-learn, wykorzystanym w realizacji projektu, wartość jest estymowana z uwzględnieniem wygładzania addytywnego, co ma na celu wyeliminowanie zerowania się tego składnika w przypadkach, gdy dana para cecha-klasa nie wystąpiła w zbiorze treningowym. [16]

### 3.3.2. Liniowa maszyna wektorów nośnych

Liniowa maszyna wektorów nośnych (ang. *Linear* *Support Vector Machine, LSVM*) to algorytm stosowany w problemach klasyfikacji binarnej, którego celem jest określenie hiperpłaszczyzny rozdzielającej zestaw danych pomiędzy dwie klasy. Wymiar tej hiperpłaszczyzny zależy od liczebności zbioru cech, na podstawie którego będzie ona wyznaczana. Margines wokół granicy (funkcji) decyzyjnej maksymalizowany jest na podstawie par wektorów cech i odpowiadających im etykiet . [17].



**Rysunek 3.1.** Przykład funkcji decyzyjnej dla danych liniowo separowalnych,   
szarym kolorem oznaczono wektory nośne. [18]

Zgodnie z [18] zestaw danych uczących , jest liniowo separowalny, jeśli nierówność:

|  |  |  |
| --- | --- | --- |
|  | , | (2.1) |

jest spełniona dla wszystkich jego elementów.

Wektory, dla których to tzw. wektory nośne, na podstawie których w przestrzeni cech konstruowana jest optymalna hiperpłaszczyzna określona zależnością:

|  |  |  |
| --- | --- | --- |
|  | , | (2.2) |

rozdzielająca dane uczące z największym możliwym marginesem, opisanym wzorem:

|  |  |  |
| --- | --- | --- |
|  | . | (2.3) |

Wyznaczanie hiperpłaszczyzny (2.2) sprowadza się do rozwiązania problemu programowania kwadratowego – minimalizacji iloczynu przy ograniczeniach (2.1).

Maszynia wektorów nośnych z biblioteki scikit-learn, wykorzystana w realizacji projektu, przy obliczaniu współczynników hiperpłaszczyzny uwzględnia dodatkowo funkcję straty typu hinge loss, kontrolującą wpływ przypadków predykcji ze zbyt małym marginesem lub błędnej. [19]

### 3.3.3. Las losowy

Las losowy (ang. *Random Forest, RF*) to metoda uczenia maszynowego działająca w oparciu o grupę drzew decyzyjnych (ang. *Decision Tree*).

Zgodnie z [20], algorytm drzewa decyzyjnego dokonuje rekurencyjnego, binarnego podziału przestrzeni cech według poszczególnych zmiennych objaśniających (predyktorów), aż do momentu spełnienia kryterium zatrzymania. W rezultacie otrzymywane są niepodzielone węzły, zwane końcowymi, na podstawie których określana jest predykcja algorytmu. Do zalet drzew decyzyjnych należą m.in.:

* możliwość stosowania w problemach klasyfikacji i regresji,
* możliwość opisania zależności między zmiennymi objaśniającymi,
* odporność na braki danych, odstające wartości i predyktory bez znaczenia.

Istotną wadę drzew decyzyjnych stanowi m.in. niestabilność (nieznaczna zmiana w przestrzeni zmiennych skutkuje innym drzewem), a także mniejsza dokładność względem nowszych metod (np. sieci neuronowych).

W celu poprawienia rezultatów, drzewa, z których składa się las losowy, tworzone są na podstawie wylosowanego ze zwracaniem podzbioru dostępnych danych, a podział w węzłach tych drzew odbywa się z wykorzystaniem pewnej wylosowanej liczby predyktorów – poprzez wybór najlepszego podziału dla tego podzbioru zmiennych objaśniających. W przypadku klasyfikacji prognozę algorytmu stanowi wynik głosowania wśród drzew (klasa wybrana przez większość), a przy wyznaczaniu wartości liczbowej zmiennej objaśnianej w regresji, średnia wyników uzyskanych przez poszczególne drzewa. [21]

W modelu lasu losowego z biblioteki scikit-learn, wykorzystanym w realizacji projektu, prognozę stanowi średnia predykcji wszystkich drzew, na podstawie której określana jest przynależność do klasy, a nie klasa wybrana przez większość. [22]

## 3.4. Model

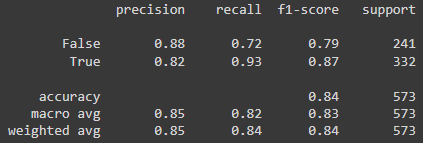
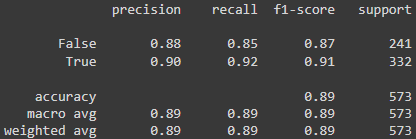
## 3.5. Aplikacja

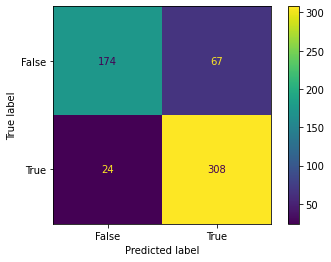
# 4. Rezultaty

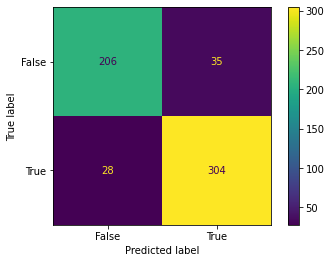
## 4.1. Testy jednostkowe

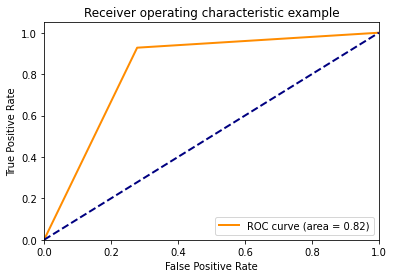
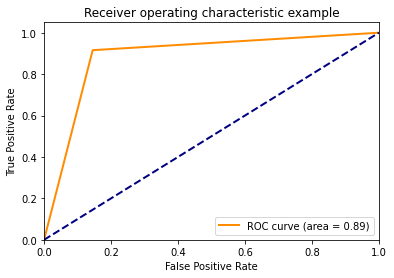
## 4.2. Ewaluacja algorytmów

Przykładowe wyniki









# 5. Podsumowanie

## 5.1. Zrealizowane założenia projektowe

## 5.2. Wnioski

## 5.3. Kierunki dalszego rozwoju

# 6. Bibliografia

x

|  |  |
| --- | --- |
| 1. | Gelfert A. Fake News: A Definition. Informal Logic. 2018 Marzec: p. 84-117. |
| 2. | Quandt T, Frischlich L, Boberg S, Schatto-Eckrodt T. Fake News. Münster: Westfalski Uniwersytet Wilhelma w Münsterze; 2019. |
| 3. | Granik M, Mesyura V. Fake news detection using naive Bayes classifier. In IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON); 2017; Kijów. p. 900-903. |
| 4. | Jurafsky D, Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 2nd ed.: Prentice Hall; 2008. |
| 5. | Ahmed H, Traore I, Saad S. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments; 2017; Vancouver. p. 127-138. |
| 6. | Reis JCS, Correia A, Murai F, Veloso A, Benevenuto F. Supervised Learning for Fake News Detection. IEEE Intelligent Systems. 2019 Marzec-Kwiecień: p. 76-81. |
| 7. | Vogel I, Jiang P. Fake News Detection with the New German Dataset “GermanFakeNC”. In Digital Libraries for Open Knowledge; 2019; Oslo. p. 288-295. |
| 8. | Sidorov G, Moreno J, Adorno HG, Posadas-Durán JP. Detection of fake news in a new corpus for the Spanish language. Journal of Intelligent & Fuzzy Systems. 2019 Maj; 36(5): p. 4869-4876. |
| 9. | van der Linden S, Roozenbeek J, Compton J. Inoculating Against Fake News About COVID-19. Frontiers in Psychology. 2020 Październik. |
| 10. | Diouf R, Sall O, Birregah B, Bousso M, Sarr EN, Bousso M, et al. Web Scraping: State-of-the-Art and Areas of Application. In IEEE International Conference on Big Data; 2019; Los Angeles. p. 6040-6042. |
| 11. | Fake Hunter. [Online]. [dostęp: 12 grudzień 2021]. <https://fakehunter.pap.pl>. |
| 12. | Termedia. [Online]. [dostęp: 12 grudzień 2021]. <https://www.termedia.pl>. |
| 13. | McCallum A, Nigam K. A Comparison of Event Models for Naive Bayes Text Classification. 2001 Maj. |
| 14. | Zhang H. The Optimality of Naive Bayes. In Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference; 2004; Floryda. |
| 15. | Raghavan P, Schütze H, Manning C. Introduction to Information Retrieval Cambridge: Cambridge University Press; 2009. |
| 16. | Dokumentacja naiwnego klasyfikatora bayesowskiego (scikit-learn). [Online]. [dostęp: 18 Grudzień 2021]. <https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes>. |
| 17. | Boser B, Guyon I, Vapnik V. A Training Algorithm for Optimal Margin Classifier. Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory. 1992: p. 144-152. |
| 18. | Cortes C, Vapnik V. Support-vector networks. Machine Learning. 1995: p. 273-297. |
| 19. | Dokumentacja liniowej maszyny wektorów nośnych (scikit-learn). [Online]. [dostęp: 20 Grudzień 2021]. <https://scikit-learn.org/stable/modules/svm.html#id15>. |
| 20. | Cutler A, Cutler DR, Stevens JR. Random Forests. Machine Learning. 2011 Styczeń: p. 157-176. |
| 21. | Breiman L. Random Forests. Machine Learning. 2001 Październik: p. 5-32. |
| 22. | Dokumentacja algorytmu lasu losowego (scikit-learn). [Online]. [dostęp: 24 grudzień 2021]. <https://scikit-learn.org/stable/modules/ensemble.html#forest>. |
| 23. | Dumais S, Osuna E, Platt J, Schölkopf B. Support vector machines. IEEE Intelligent Systems and their Applications. 1998: p. 18-28. |

x