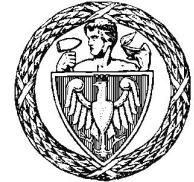


**Politechnika Warszawska**

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Sterowania i Elektroniki Przemysłowej

# Praca dyplomowa inżynierska

na kierunku Informatyka Stosowana

w specjalności Inżynieria danych i multimedia

Platforma do detekcji niewiarygodnych informacji w internecie

Maciej Czarkowski

numer albumu 292810

Miłosz Wójcik

numer albumu 292923

promotor  
dr inż. Krzysztof Hryniów

WARSZAWA 2022



## **Platforma do detekcji niewiarygodnych informacji w internecie**

### **Streszczenie**

To jest streszczenie. To jest trochę za krótkie jako że powinno zająć całą stronę.

**Słowa kluczowe:**



# **Platform for detecting unreliable information on the Internet**

## **Abstract**

This is abstract. This one is a little too short as it should occupy the whole page.

### **Keywords:**



,załącznik nr 8 do zarządzenia nr 42 /2020 Rektora PW  
annex no 8 to regulation no. 42 /2020 of the WUT Rector

Logo PW  
WUT logo

**Politechnika Warszawska**  
*Warsaw University of Technology*

Maciej Czarkowski.....

imię i nazwisko studenta  
*name and surname of the student*  
292810.....

numer albumu  
*student record book number*  
Informatyka Stosowana.....

kierunek studiów  
*field of study*

.....Warszawa.....

miejscowość i data  
*place and date*

## OŚWIADCZENIE *DECLARATION*

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

*Under the penalty of perjury, I hereby certify that I wrote my diploma thesis on my own, under the guidance of the thesis supervisor.*

Jednocześnie oświadczam, że:

*I also declare that:*

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2019 r. , poz. 1231 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,  
*this diploma thesis does not constitute infringement of copyright following the act of 4 February 1994 on copyright and related rights (Journal of Acts of 2019, item 1231 with further amendments) or personal rights protected under the civil law*
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,  
*the diploma thesis does not contain data or information acquired in an illegal way,*
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,  
*the diploma thesis has never been the basis of any other official proceedings leading to the award of diplomas or professional degrees,*
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,  
*all information included in the diploma thesis, derived from printed and electronic sources, has been documented with relevant references in the literature section,*
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.  
*I am aware of the regulations at Warsaw University of Technology on management of copyright and related rights, industrial property rights and commercialisation.*

Oświadczam, że treść pracy dyplomowej w wersji drukowanej oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

*I certify that the content of the printed version of the diploma thesis and the content of the diploma thesis in the Archive of Diploma Theses (APD module) of the USOS system are identical.*

.....  
czytelny podpis studenta  
*legible signature of the student “.*



,załącznik nr 8 do zarządzenia nr 42 /2020 Rektora PW  
annex no 8 to regulation no. 42 /2020 of the WUT Rector

Logo PW  
WUT logo

**Politechnika Warszawska**  
*Warsaw University of Technology*

Miłosz Wójcik.....

imię i nazwisko studenta  
*name and surname of the student*  
292923.....

numer albumu  
*student record book number*  
Informatyka Stosowana.....

kierunek studiów  
*field of study*

.....Warszawa.....

miejscowość i data  
*place and date*

## OŚWIADCZENIE *DECLARATION*

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

*Under the penalty of perjury, I hereby certify that I wrote my diploma thesis on my own, under the guidance of the thesis supervisor.*

Jednocześnie oświadczam, że:

*I also declare that:*

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2019 r. , poz. 1231 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,  
*this diploma thesis does not constitute infringement of copyright following the act of 4 February 1994 on copyright and related rights (Journal of Acts of 2019, item 1231 with further amendments) or personal rights protected under the civil law*
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,  
*the diploma thesis does not contain data or information acquired in an illegal way,*
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,  
*the diploma thesis has never been the basis of any other official proceedings leading to the award of diplomas or professional degrees,*
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,  
*all information included in the diploma thesis, derived from printed and electronic sources, has been documented with relevant references in the literature section,*
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.  
*I am aware of the regulations at Warsaw University of Technology on management of copyright and related rights, industrial property rights and commercialisation.*

Oświadczam, że treść pracy dyplomowej w wersji drukowanej oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

*I certify that the content of the printed version of the diploma thesis and the content of the diploma thesis in the Archive of Diploma Theses (APD module) of the USOS system are identical.*

.....  
czytelny podpis studenta  
*legible signature of the student “.*



# Spis treści

<b>1 Wstęp</b>	<b>1</b>
1.1 Motywacja . . . . .	1
1.2 Dane statystyczne . . . . .	3
1.3 Pobudki do działania w ramach analizowanego tematu . . . . .	5
1.4 Podział pracy . . . . .	6
1.4.1 Zarys podziału obowiązków . . . . .	6
1.4.2 Systematyka i kontrola postępów . . . . .	6
1.4.3 Narzędzia organizacji pracy . . . . .	6
<b>2 Analiza wstępna</b>	<b>9</b>
2.1 Założenia oraz oczekiwane efekty . . . . .	9
2.2 Problematyka zagadnienia . . . . .	10
2.3 Istniejące rozwiązania . . . . .	11
2.3.1 Google Safe Browsing . . . . .	12
2.3.2 Facebook SimSearchNet++ . . . . .	12
2.3.3 Logically . . . . .	12
2.4 Tradycyjne serwisy weryfikujące . . . . .	13
2.4.1 Snopes.com . . . . .	13
2.4.2 FactCheck.org . . . . .	13
2.5 Źródła danych . . . . .	13
2.6 Pierwsze propozycje realizacji . . . . .	14
<b>3 Specyfikacja funkcjonalna</b>	<b>15</b>
3.1 Wymagania . . . . .	15
3.2 Diagram przypadków użycia . . . . .	16
3.3 Opis dostępnych funkcjonalności . . . . .	16
3.3.1 Analiza dynamiczna strony z podanego adresu URL pod kątem jej budowy . . . . .	16
3.3.2 Klasyfikacja strony z podanego adresu URL w kategoriach prawdopodobieństwa prawdziwości informacji . . . . .	17
3.3.3 Możliwość przeglądania bazy danych przeanalizowanych stron . . . . .	17

3.3.4	Możliwość zapoznania się ze statystykami dotyczącymi danych uczących	17
3.3.5	Możliwość zapoznania się z wykorzystanymi algorytmami oraz procesem uczenia sieci neuronowej . . . . .	18
3.4	Widoki aplikacji . . . . .	18
3.4.1	Analiza podanego URL . . . . .	18
3.4.2	Zebrane dane . . . . .	19
3.4.3	Statystyki . . . . .	20
3.4.4	Baza wiedzy . . . . .	22
3.5	Format danych wejściowych . . . . .	22
<b>4</b>	<b>Specyfikacja implementacyjna</b>	<b>23</b>
4.1	Stos technologiczny . . . . .	23
4.1.1	Przetwarzanie po stronie serwera (backend) . . . . .	23
4.1.2	Warstwa interfejsu użytkownika (frontend) . . . . .	25
4.1.3	Narzędzia pomocnicze . . . . .	25
4.2	Interfejs API . . . . .	26
4.3	Opis przechowywanych danych . . . . .	26
4.4	Parametry uruchomieniowe programu . . . . .	31
<b>5</b>	<b>Realizacja aplikacji</b>	<b>33</b>
5.1	Wykorzystany sprzęt . . . . .	33
5.2	Aplikacja serwerowa (backend) . . . . .	33
5.2.1	Pozyskiwanie kodu strony . . . . .	33
5.2.2	Webscrapping . . . . .	34
5.2.3	Utworzzone narzędzia pomocnicze . . . . .	40
5.3	Interfejs graficzny (frontend) . . . . .	40
<b>6</b>	<b>Analiza danych i sieć neuronowa</b>	<b>43</b>
6.1	Wybrany model sieci neuronowej . . . . .	43
6.2	Przetwarzanie wstępne . . . . .	44
6.3	Analiza eksploracyjna danych . . . . .	46
6.4	Proces uczenia sieci . . . . .	59
6.4.1	Podział danych . . . . .	59
6.4.2	Metody oceny modeli . . . . .	59
6.4.3	Modyfikacje w celu poprawy wyników . . . . .	59
6.4.4	Wybór najefektywniejszych rozwiązań . . . . .	59
6.5	Przetwarzanie wyników . . . . .	59
6.6	Wizualizacja wyników . . . . .	59

<b>7 Uzyskane wyniki</b>	<b>61</b>
<b>8 Wnioski oraz podsumowanie</b>	<b>63</b>
<b>Bibliografia</b>	<b>65</b>
<b>Wykaz skrótów i symboli</b>	<b>67</b>
<b>Spis rysunków</b>	<b>69</b>
<b>Spis tabel</b>	<b>71</b>



# Rozdział 1

## Wstęp

### 1.1 Motywacja

Wraz z wejściem w nowe millenium, w nową erę technologiczną weszła ludzka codzienność. Gdy Dr Joseph Carl Robnett Licklider, jeden z ojców sukcesu internetu i osoba w dużym stopniu odpowiedzialna za jego rozwój, w 1959r. sformułowała podstawowe założenia działania wielkiej sieci komputerowej, która w sposób zdecentralizowany pozwoli na komunikację pomiędzy urządzeniami, dostrzegano w nich szansę na postawienie kroku w przód w wyścigu zimnowojennym ze Związkami Radzieckim. Jego praca *Man-Computer Symbiosis* [17] stała się fundamentem i ukierunkowaniem rozwoju rozproszonej sieci, której zastosowanie wyszło daleko poza rysowany wówczas horyzont możliwości widocznych wraz z jej rozwojem.

Jak wszyscy wiemy, ARPANET<sup>1</sup>, który służyć miał głównie do celów militarnych, był preludium do stworzenia sieci, która jest naszym codziennym towarzyszem. Powstanie wyszukiwarki Google na przełomie tysiącleci, w 1998r.<sup>2</sup>, możemy określić jako całkiem symboliczny kamień milowy w rozwoju internetu, z którego „dobroci” korzystamy praktycznie każdego dnia. Internet przyniósł ze sobą wiele możliwości, a wraz z nimi pojawiły się różnorodne pomysły na ich wykorzystanie.

Klasyczne zastosowania internetu takie jak możliwość komunikowania się, składowania dokumentów, w często niezwykle odległych lokalizacjach, możliwości zakupów oraz prowadzenia bankowości są udogodnieniami, które na codzień umożliwiają nam komfortowe wykonywanie wielu czynności. Razem z nimi kroczą informacje, przekazywane pomiędzy szerokimi gronami osób fakty, które są dla nas jak szósty, prawie doskonały zmysł – pozwalają zobaczyć to, czego nie możemy zobaczyć na własne oczy czy usłyszeć będąc świadkiem wydarzeń. Internet przejął

---

<sup>1</sup><https://pl.wikipedia.org/wiki/ARPANET>

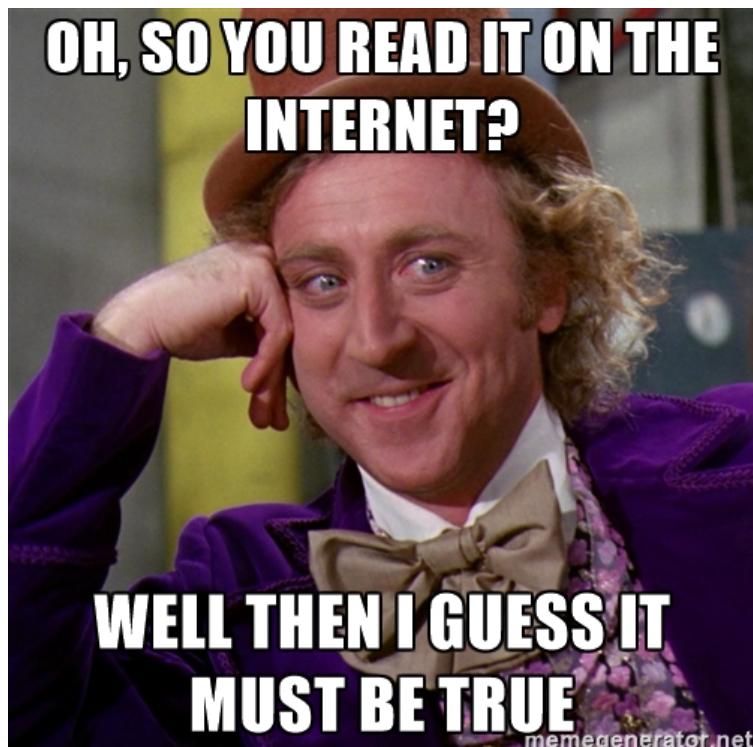
<sup>2</sup><https://pl.wikipedia.org/wiki/Google>

inicjatywę rozprzestrzeniania się informacji od klasycznych, znanych wcześniej rozwiązań jak radio, telewizja czy gazety.

Wszystkie znane wcześniej źródła informacji znalazły swoje miejsce w internecie, gdzie możliwości udostępniania informacji są prawie nieograniczone. Przekazywanie informacji i opiniotwórczość opuściło ciasno domknięty oligopol niewielu przedsiębiorstw odpowiedzialnych za dystrybucję informacji, a stało się dostępne dla wszystkich użytkowników internetu, którzy w prosty sposób mogą udostępniać wiadomości oraz dzielić się komentarzami na ich temat [4].

Olbrzymi wybór źródeł ciągnie jednak ze sobą konieczność ich weryfikacji. Niezwykle często napotykane przez nas artykuły są w pełni nakierowane na pozyskanie od nas pewnych informacji bądź też wprowadzenie nas w dezinformację i błędne przekonanie dotyczące różnych faktów. Funkcja propagandowa, która jest jedną z domen internetu, przybrała miarę narzędzia do szerzenia opinii i faktów przyjaznych dla ich autorów, często odbiegających od rzeczywistości [16].

Nasza praca ma za zadanie stworzenie narzędzia, które niczym sito pozwoli na weryfikację wiarygodności źródeł, na które natrafiamy. Dzięki uzyskaniu wstępnej informacji na temat danego źródła, użytkownik mógłby spojrzeć na analizowany tekst z wyczuloną uwagą, dzięki czemu nieprawdziwe informacje z podejrzanej źródła nie byłyby przyjęte jako prawdziwe.



**Rysunek 1.** Ironiczne odniesienie do informacji umieszczanych w internecie  
Źródło: [11]

Mem przedstawiony na Rysunku 5 z góry wskazuje na ironiczne zabarwienie zawartego na nim tekstu. Zgodnie z wspomnianymi przez nas dobrymi praktykami, informacje znalezione w internecie powinniśmy traktować z dozą dystansu i racjonalnego podejścia, a osoby, które są ślepo zapatrzone w każdą zauważoną informację, mogą się później natknąć na nieciekawą prawdę.

Samo pojęcie fakenews, oznaczające w tłumaczeniu na język polski po prostu fałszywą informację[16]. Nabrąło ono szczególnie na znaczeniu na przestrzeni ostatnich kilku lat, a największej popularności nabrąło przy okazji wyborów prezydenckich w Stanach Zjednoczonych w 2020r. oraz podczas pandemii COVID-19 i podążającej za nią kampanii szczepień.

## 1.2 Dane statystyczne

W tym miejscu chcielibyśmy poświęcić uwagę dokonanym już analizom na temat istoty fakenewsów i ich potencjalnego wpływu na nie tylko sprawy życia codziennego, ale również na sprawy polityczne i inne istotne kwestie ogólnego interesu. O ile fakenewsy w świecie celebrytów czy świata mediów nie są zazwyczaj szkodliwe dla całego społeczeństwa (choć oczywiście nie należy ich deprecjonować ze względu na często krzywdzący charakter dla osób będących przedmiotem pomówień), o tyle te, które mają wpływać na duże ruchy społeczne oraz mogące być powodem wzniecenia niepokojów w różnych regionach świata, powinny być poddane szczególnej uwadze i możliwie szybkiej eliminacji.

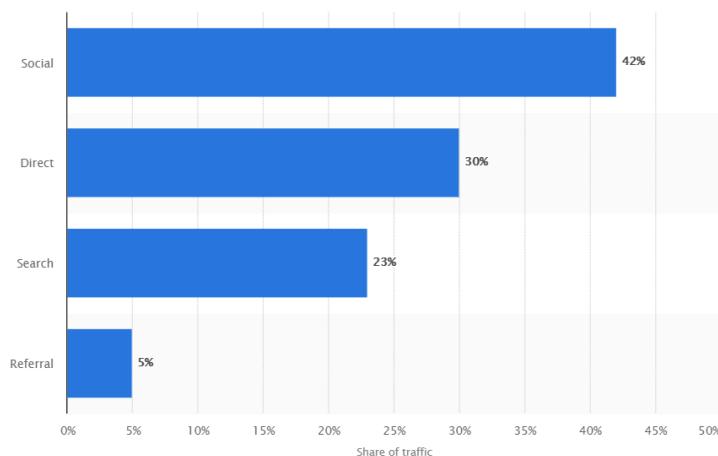
Zebrane przez nas informacje, zwizualizowane na wykresach, pozwalają wysnuwać wnioski, które mogą iść wraz z adaptacją fałszywych informacji wśród ludzi i rozprzestrzenianiem się ich wśród coraz to większych grup.

### Źródła fakenewsów

Wykres 2 wyraźnie wskazuje, gdzie fałszywe informacje rozprzestrzeniają się najpierw.

Analizując go, możemy zauważyć, że fakenewsy najlepiej „czują się” w mediach społecznościowych. Dzieje się tak za sprawą największej dynamiki tego rodzaju środka przekazu, gdzie informacje wymieniane są pomiędzy kolejnymi użytkownikami w niezwykle szybkim tempie, a działanie tzw. poczty pantoflowej tylko potęguje to zjawisko. Wskazują na to naukowcy [14] oraz działania zespołu Facebook'a, który wprowadził algorytmy promujące posty znajomych ponad te zawierające informacje, które przekazane w formie niezbyt treściwego posta mogą mieć charakter mylący.[2]

Można powiedzieć, że serwisy takie jak Facebook czy Twitter są idealnym inkubatorem dla fałszywych informacji, gdzie mogą spokojnie się rozwijać a nawet mutować w coraz to groźniejsze.



Rysunek 2. Rozprzestrzenianie się fałszywych informacji w sieci

Źródło: [3]

W tym miejscu należy zwrócić uwagę na pewną istotną zależność, która bezpośrednio nawiązuje i jest przyczyną powyższego stanu zaprezentowanego na wykresie. Informacje przesyłane przez szeroko pojęte social media mają swoje źródło w artykułach znalezionych na różnych stronach w internecie, często na stronach o charakterze pseudonaukowym. W ten sposób budowana jest wokół nich otoczka prawdy, która utwardza się wraz z przekazywaniem danej informacji w coraz to szerszym gronie. Media społecznościowe są tu przekaźnikiem, który wykorzystuje informacje, które swoje źródło mają w zupełnie innym miejscu.

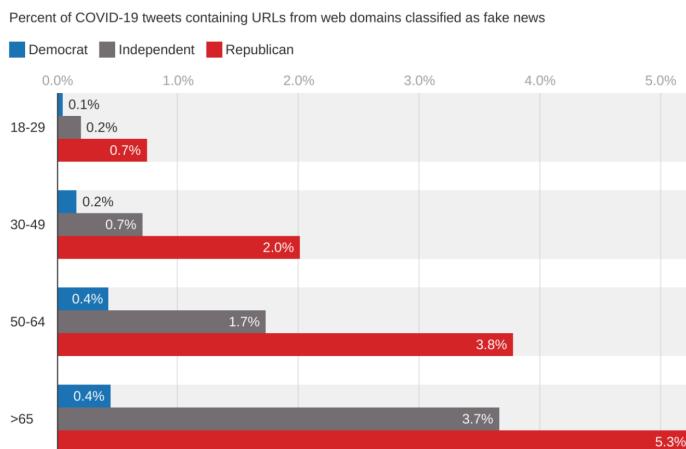
To właśnie dlatego postanowiliśmy zwrócić naszą analizę do źródeł, czyli stron internetowych, gdzie najczęściej zaczyna się cykl życiowy fakenewsów. Eliminacja, a przynajmniej wskaźówki pozwalające na częściowe wykluczenie platform potencjalnie skłonnych do powielania fałszywych informacji, które są zapewniane przez naszą platformę, staje w tym miejscu na straży źródła informacji.

### Charakter manipulacyjny

Wspomniane wcześniej wydarzenia ostatnich lat, które z pewnością poruszyły olbrzymie grupy społeczne i dotyczyły wielu osób, czyli pandemia COVID-19 oraz wybory w Stanach Zjednoczonych, na które, z racji na charakter USA jako supermocarstwa o znaczących wpływach, zwróciło oczy wiele osób stały się potencjalnym środowiskiem do wykorzystania ze złymi intencjami, w celu manipulacji innych grup we własnym interesie.

Wykres z rysunku 3 wskazuje, z podziałem na grupy wiekowe, osoby najbardziej podatne na przeglądanie fałszywych informacji jak i konkretne struktury polityczne, które przyczyniały się do rozpowszechnienia tychże informacji. Analiza danych z serwisu Twitter wykazała, że

### 1.3. Pobudki do działania w ramach analizowanego tematu



**Rysunek 3.** Udział źródeł uznanych sklasyfikowanych jako fake news w tweetach poszczególnych frakcji politycznych związanych z pandemią COVID-19

Źródło: [15, s. 7]

duża liczba tweetów osób starszych oraz, w odniesieniu do wyborów prezydenckich w Stanach Zjednoczonych, reprezentantów środowiska republikanów, była skłonna do rozpowszechniania informacji ze źródeł, które zostały sklasyfikowane jako potencjalnie fałszywe.

Ukazana statystyka pozwala na wnioskowanie, iż wśród republikanów fałszywe informacje dotyczące pandemii rozchodziły się znaczco szybciej niż po drugiej stronie obozu politycznego, a co za tym idzie mogły być powodem niepokojów społecznych, co miało pewne odzwierciedlenie w protestach odbywających się w terminach bliskich samym wyborom prezydenckim. Uistatnia to rolę propagandową fałszywych informacji oraz wskazuje najbardziej podatne środowiska (osoby w podeszłym wieku), które są mniej skłonne do weryfikacji przeszukiwanych źródeł. Nasza platforma pozwala w tym miejscu na szersze wykorzystanie, gdzie każdy, przed rozpowszechnieniem danej informacji, mógłby ją zweryfikować z jej wykorzystaniem.

## 1.3 Pobudki do działania w ramach analizowanego tematu

Podjęliśmy się stworzenia tego typu platformy nie tylko ze względu na konieczność realizacji pracy dyplomowej czy zrealizowania pewnego wyzwania programistycznego, ale również z pragmatyzmu, który nakazuje nam otrzymywane dane poddawać analizie, która pozwoli wydobyć z nich najważniejsze, wiarygodne elementy. W przypadku skutecznego działania naszego rozwiązania moglibyśmy je opublikować, a korzystający z niego użytkownicy zyskali by bazę, na podstawie której mogliby weryfikować poprawność przeglądanych danych. W ten

sposób platforma zyskałaby charakter opiniotwórczy, co już na starcie stawia przed nią duże wyzwanie.

## **1.4 Podział pracy**

### **1.4.1 Zarys podziału obowiązków**

### **1.4.2 Systematyka i kontrola postępów**

Realizując tak obszerny projekt staraliśmy się na tym aby praca była realizowana systematycznie, w kolejnych iteracjach. Do tego celu idealnie stworzone są metodyki zwinne wytwarzania oprogramowania, które pozwalają na efektywną realizację zadań mając na uwadze możliwe dynamiczne zmiany w wymaganiach. Każdą kolejną iterację, nazywaną Sprintem wieńczyliśmy spotkaniami z promotorem, podczas których mogliśmy zrobić retrospekcję wykonanych zadań oraz zdefiniować te na najbliższy sprint.

Nasza praca nie była zorganizowana w systemie czysto SCRUMowym, ze względu na pozostałe obowiązki związane ze studiami, które uniemożliwiałyby nam pewne podstawowe dla metodyki SCRUM kwestie, jak codzienne *stand upy*. Z tego też powodu zdecydowaliśmy się na zastosowanie pewnej hybrydy metodyki SCRUM oraz Kanban, w której to dzielimy naszą pracę na sprints, w których to dostarczamy pewne funkcjonalności, a pracę organizujemy na tablicach z zadaniami w kolejnych stanach, jednakże pozwalamy sobie na pewną elastyczność w zakresie spotkań – zarówno z Product Ownerem (promotorem) jak i pomiędzy członkami zespołu deweloperskiego.

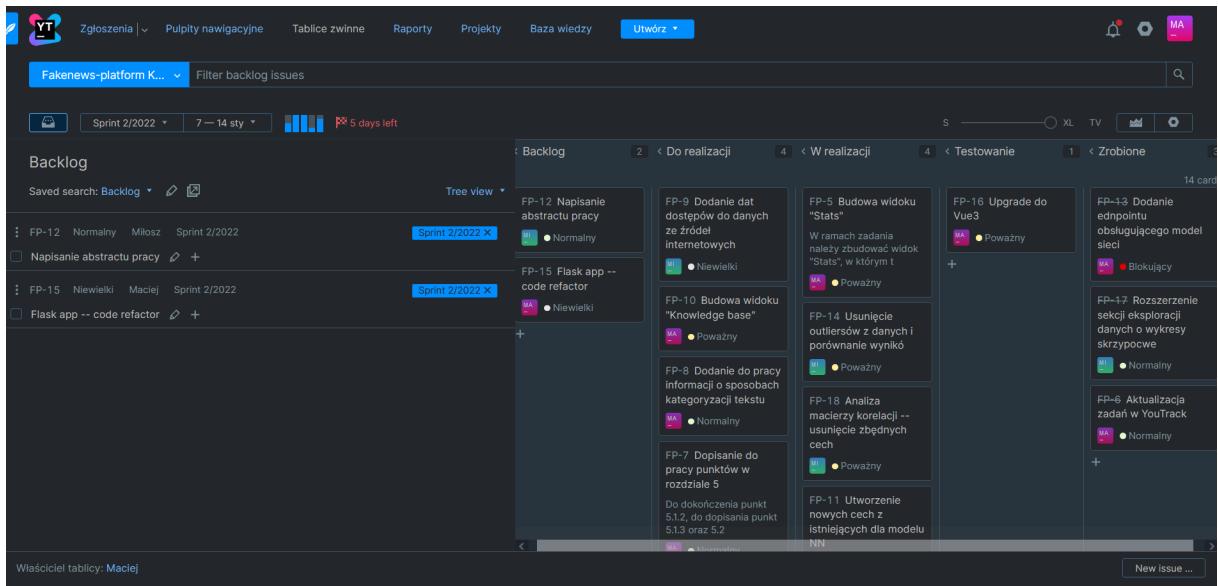
### **1.4.3 Narzędzia organizacji pracy**

#### **JetBrains YouTrack**

Podczas pracy w zespole niezbędnym jest użycie narzędzi pozwalających na proste przekazywanie informacji o podziale i postępie prac nad projektem. Zdecydowaliśmy się na wybór rozwiązania firmy *JetBrains*, ponieważ poznaliśmy je w trakcie studiów, a także ze względu na obfitą bibliotekę dodatków wspomagających wysiłek włożony w zarządzanie czasem i projektami. Przyjazny interfejs i tablice o charakterze drag & drop<sup>3</sup> pozwalają na sprawne tworzenie i przesuwanie zadań na tablicy kanbanowej.

---

<sup>3</sup>przeciągnij i upuść



**Rysunek 4.** Tablica Kanban w usłudze Youtrack  
 Źródło: autorskie (zrzut ekranu)

## Github

W celu umożliwienia kontroli wersji projektu oraz spójności pomiędzy prowadzonymi równolegle działaniami członków zespołu, skloniliśmy się do znanego nam, oraz najczęściej wykorzystywanego rozwiązania dla systemu kontroli wersji Git, jakim jest serwis Github. Stworzenie repozytorium projektu pozwoliło nam na lepszą organizację pracy i dało możliwość do oddzielenia wytwarzania kolejnych funkcjonalności przy jednoczesnej możliwości kontrolowania historycznych efektów prac.

## Neptune

Do sprawniejszego zarządzania pracami nad systemami uczenia maszynowego wykorzystaliśmy narzędzie *neptune.ai* - portal narzędzia. Umożliwia ono sprawną współpracę przy projektowaniu sieci neuronowych poprzez zapewnianie jednolitej platformy do przechowywania, wyświetlania, porównywania i organizacji wyników uzyskanych przez sieci. Skróciło to czas potrzebny do podzielenia się wynikami pracy między członkami zespołu.



# Rozdział 2

## Analiza wstępna

### 2.1 Założenia oraz oczekiwane efekty

Budując pierwsze założenia dotyczące projektu postanowiliśmy skupić się na wykorzystaniu możliwości sztucznej inteligencji, do znajomości której wprowadziły nas zajęcia realizowane na Wydziale Elektrycznym Politechniki Warszawskiej. Było to uwarunkowane nie tylko realizacją specjalizacji *Inżynieria i analiza danych*, ale również chęcią rozwoju w kierunku nowych technologii, jakim niewątpliwie jest uczenie maszynowe oraz tworzenie sztucznych sieci neuronowych.

Z racji na obszerność i wieloaspektowość pracy, postanowiliśmy również wykorzystywać dostępne technologie w ramach analizy danych, które połączone trybami w całość pozwoliły na stworzenie efektywnie pracującej maszyny, łączącej zalety wielu narzędzi.

W ramach realizacji projektu chcieliśmy stworzyć platformę, która pozwoli użytkownikom na weryfikację wiadomości znalezionych w internecie. Platforma ta miałyby być stworzona w formie aplikacji webowej, która w intuicyjny i przyjazny wizualnie sposób opisywałaby podane na wejście źródło jako godne zaufania lub uznane jako potencjalne źródło fakenewsów.

Postanowiliśmy, że klasyfikacja przekazanych w formie linków do danych stron nie będzie binarna (zero-jedynkowa), aby arbitralnie nie wskazywać czy dana strona jest rzetelna czy nie. Wynika to z niedoskonałości sztucznej inteligencji, która pomimo dużej skuteczności potrafi zupełnie błędnie sklasyfikować pewne informacje. Dobrym przykładem sytuacji, gdzie percepcja sieci neuronowych nie byłaby najlepszym wyznacznikiem, jest zestawienie zdjęć zaprezentowane poniżej.

W naszym przypadku błędne sklasyfikowanie artykułu lub strony jako niezaufanej i nierzetelnej byłoby bardzo krzywdzące dla jej autora, dlatego też zdecydowaliśmy, że nasza platforma będzie wyciągała jedynie wskazówki dotyczące strony, na które warto zwrócić uwagę przy jej analizie. Forma *tagowania* stron pozwoli na pozostawienie czynnika ludzkiego, który

przy udziale informacji zwróconych przez rozwiązania sztucznej inteligencji może pozwolić na pełną i poprawną klasyfikację danego źródła przez użytkownika.

Niezwyczajnym założeniem jest stworzenie rozwiązania działającego dla tekstu w języku angielskim. Dzięki temu mogliśmy skupić się na tworzeniu rozwiązania ogólnodostępnego, dostępnego praktycznie dla każdego. W ten sposób sieci neuronowe uniknęły problemu z tłumaczeniem, zaś już istniejące rozwiązania pozwoliły na przyśpieszenie prac. Opracowanie rozwiązania działającego dla języka polskiego wymagałoby implementacji algorytmów przetwarzania języka naturalnego, co byłoby innowacją na rodzimym rynku technologicznym.

## 2.2 Problematyka zagadnienia

Analizowane zagadnienie jest niezwyczajne rozbudowane i wieloaspektowe. Co za tym idzie, odpowiednia analiza i stworzenie dobrego modelu wymaga prawidłowego doboru cech wykorzystanych do nauki sieci neuronowej. Rozbudowanie problemu wynika głównie z konieczności dokonania analizy składniowej pełnego kodu HTML strony, zawierającego jednocześnie wiele skryptów języka JavaScript oraz stylów CSS, które wykorzystywane są przez przeglądarki do obsługi interakcji z użytkownikiem i nadawania stronom odpowiedniego wyglądu. Odfiltrowanie tego typu elementów jest nietywialne ze względu na brak stałego schematu do budowania



**Rysunek 5.** Sytuacja problematyczna dla rozwiązań z zakresu sztucznej inteligencji  
Źródło: [6]

strony. Każda witryna cechuje się swoją architekturą, a co za tym idzie nie ma możliwości generalizacji ekstrakcji tekstów na stronie do analizy.

Opisane wyzwania wymagają podjęcia dużych prac w zakresie analizy kodu strony, z którego należy odpowiednio wyizolować teksty do analizy. Sama pomyślna ekstrakcja tekstów widocznych na stronie nie może być jednak uznana za pełen sukces, gdyż nie każdy tekst widoczny w przeglądarce jest elementem będącym częścią artykułu.

Inną kwestią jest analiza semantyczna tekstu – działania w tym zakresie są niezwykle ograniczone. Każdy tekst cechuje się własną strukturą składniową, a co za tym idzie, nie jest możliwa ocena prawdziwości zawartych w artykule informacji na podstawie słów kluczowych. W związku z opisanymi powyżej cechami tekstów, połączenie ich elementów w potencjalnie weryfikowalne mapy typu klucz-wartość jest zagadnieniem niemożliwym do zrealizowania w sposób rzetelny i poprawny. Efektem tego stanu rzeczy jest brak możliwości weryfikacji prawdziwości poszczególnych elementów zdań. Warto w tym miejscu dodać, że nawet posiadając poprawnie ustrukturyzowane dane wyciągnięte z artykułu (np. "*James Bond date of birth*": "12.10.1982") należałoby parsować strony uzyskane jako wyniki wyszukiwania po kluczach w popularnych wyszukiwarkach, w celu dokonania ponownej analizy tekstu i poszukiwania dopasowania na zaufanych stronach. Tego typu działanie czyniłoby to zagadnienie nie tylko niezwykle trudnym programistycznie, ale również obliczeniowo – analiza składniowa każdej ze stron zajęłaby dużo czasu ze względu na konieczność ich pełnego renderowania wraz z obecnym na nich JavaScriptem.

Trudności z analizą samego tekstu danego artykułu skłaniają ku próbom analiz całego kodu strony. Rozwiązanie to, choć skomplikowane, może wskazać dobre praktyki programistyczne wykorzystywane przez rzetelne serwisy, które jako priorytet traktują przejrzysty odbiór artykułu, a nie obrzucenie użytkownika materiałami marketingowymi czy elementami noszącymi znamiona *phishingu*.

## 2.3 Istniejące rozwiązania

Decydując się na realizację tej tematyki mieliśmy świadomość, że istnieją rozwiązania w tym zakresie. Pozwalają one jednak na analizę zazwyczaj mocno ograniczoną, dotyczącą odseparowanych elementów, które możemy znaleźć na stronie internetowej, a nie witryny jako całości. Postanowiliśmy opisać kilka z najpopularniejszych rozwiązań, które były wskazówkami i niekiedy przewodnikami przy tworzeniu naszej platformy.

### 2.3.1 Google Safe Browsing

Jednym z istniejących rozwiązań, z których skorzystaliśmy jest Google Safe Browsing. Jest to darmowa usługa od firmy Google, oceniająca bezpieczeństwo strony, do której link zostanie podany. Safe Browsing analizuje link pod względem trzech kategorii – złośliwego oprogramowania w kodzie strony, analizy pobieranych plików po kliknięciu w link i ataków typu *Phishing*<sup>1</sup>.

Usługa w dużej mierze skupia się na chronieniu użytkownika korzystającego z produktów Google w czasie rzeczywistym. Można jednak skorzystać z API usługi, aby po wysłaniu zapytania z linkiem, otrzymać odpowiedź zawierającą opisane poszczególne zagrożenia. Planujemy wykorzystać to narzędzie do analizy linków występujących na stronach z wiadomościami, a wyniki tej analizy będą wpływać na późniejszą ocenę danej wiadomości.

### 2.3.2 Facebook SimSearchNet++

W ramach walki z obecną pandemią choroby Covid-19 koncern Facebook (obecnie Meta), odpowiadający m.in. za portale Facebook i Instagram, stworzył model SimSearchNet++. Służy on do rozpoznawania obrazów będących wariacjami oryginalnego obrazu. Poprzez użycie frameworku GAN – Generative Adversarial Network – jest w stanie znajdować zmodyfikowane (przyjęte, przepisane w formacie mema, obrócone) obrazy szerzące dezinformacje. Uzyskują to poprzez porównywanie nowoudostępnianych treści do istniejącej biblioteki zawierającej treści zakwalifikowane jako fałszywe. W ten sposób poprzez jednorazowy fact-check są w stanie flagować każdą kolejną kopię tej informacji i ostrzegać użytkowników przed fakenewsami [5].

### 2.3.3 Logically

Kolejnym rozwiązaniem wykorzystującym technologie uczenia maszynowego do walki z fałszywymi informacjami opracowała firma *Logically*. Od 2017<sup>2</sup> oferują usługi wykrywania *fake news*, zarówno na duże zlecenia, jak i dla pojedyńczych użytkowników prywatnych. Ich głównym odbiorcą są duże portale społecznościowe, a także fora, które za opłatą mogą weryfikować wszystkie linki publikowane przez ich użytkowników, aby zapobiec rozprzestrzeniania się fałszywych wiadomości<sup>3</sup>. Ponadto oferują darmową wtyczkę do przeglądarki Chrome<sup>4</sup>, która analizuje tekst na wyświetlanej stronie i oznacza potencjalne oszustwa.

---

<sup>1</sup>Oszustwo polegające na upozorowaniu wiadomości lub strony internetowej na wzór treści od zaufanych źródeł, celem uzyskania danych ofiary[18]

<sup>2</sup>Według strony firmowej, <https://www.logically.ai/about>, dostęp na dzień 06.01.2022

<sup>3</sup>Według <https://www.logically.ai/logically-intelligence>, dostęp na dzień 06.01.2022

<sup>4</sup>Według <https://www.logically.ai/products/browser-extension>, dane z 06.01.2022

## 2.4 Tradycyjne serwisy weryfikujące

Oprócz automatycznych systemów, niezwykle ważnym elementem walki z fałszywymi informacjami są serwisy zajmujące się weryfikacją wiadomości zamieszczanych w internecie. W dużej mierze działają one na zasadzie *fact-checkingu*<sup>5</sup>.

### 2.4.1 Snopes.com

Jednym z największych serwisów zajmujących się zamieszczaniem informacji o *fake news* jest *Snopes.com*[19, s. 285]. Został on założony w 1994 roku przez Davida Mikkelsona, który wraz ze swoją żoną znajdowali i weryfikowali popularne artykuły<sup>6</sup>. Przez ponad 27 lat serwis regularnie sprawdza wszelkie podejrzane informacje, jednocześnie zachowując przejrzystość finansową<sup>7</sup>. Dzięki temu jego reputacja jest niezachwiana podejrzeniami o korupcję.

### 2.4.2 FactCheck.org

Wiele *fake news* wywodzi się z politki, często sami rządzący kłamią lub nie mówią całej prawdy. Aby walczyć z tym problemem, dziennikarz Brooks Jackson założył w 2003 roku portal FactCheck.org, aby niwelować zamieszanie wynikające z oszustw polityków w USA<sup>8</sup>. Redakcja zajmuje się między innymi weryfikacją wypowiedzi w trakcie debat politycznych, 'łańcuszkami'<sup>9</sup> na portalach społecznościowych, wiadomościami o Covid-19 i szczepieniach<sup>10</sup>.

## 2.5 Źródła danych

Aby poprawnie oceniać czy dany artykuł jest fałszywy, należy ocenić jego treść, a także wszystkie dodatkowe elementy występujące na stronie. Do nauki sieci neuronowych potrzeba natomiast odpowiednio dużej bazy wiedzy, aby po przećwiczeniu była w stanie odpowiednio kategoryzować nowo wprowadzane strony. W tym celu przed projektem sieci zebraliśmy ręcznie kilkaset linków do stron z wiadomościami, zarówno zaufanych jak i tych budzących wątpliwości co do autentyczności przedstawianych tam informacji. Każda witryna została oflagowana jako prawdziwa lub fałszywa, a następnie opisana kilkunastoma parametrami.

<sup>5</sup>Proces weryfikacji informacji poprzez sprawdzenie faktów w wiarygodnych źródłach[13]

<sup>6</sup>Dane ze strony <https://www.snopes.com/about/>, dostęp na dzień 06.01.2022

<sup>7</sup>Dane publikowane na stronie <https://www.snopes.com/disclosures/>, dostęp na dzień 06.01.2022

<sup>8</sup>Opis założyciela z artykułu "Is this a great job, or what?", <https://www.factcheck.org/2003/12/is-this-a-great-job-or-what/>, dostęp na dzień 06.01.2022

<sup>9</sup>Wiadomości o treści zachęcającej do dalszego rozsyłania jej do kolejnych odbiorców, często fałszywe

<sup>10</sup>Według <https://www.factcheck.org/a-guide-to-our-coronavirus-coverage/>, dostęp na dzień 06.01.2022

Ponadto, do celów analizy tekstu wykorzystaliśmy również bazę słów uznawanych za nieprzyzwoite. Była ona niezwykle przydatna przy próbie oceny jakości tekstu artykułu. Duża liczba przekleństw sugeruje niższy poziom dziennikarstwa, a fałszywe wiadomości w większości przypadków zaczynają swój cykl życia na niszowych portalach. Baza danych została pobrana z portalu *Free Web Headers*<sup>11</sup>.

## 2.6 Pierwsze propozycje realizacji

. Poszukując wskazówek odpowiednio adresujących zagadnienia poruszone przez nas w podrozdziale *Problematyka zagadnienia* postanowiliśmy odnieść się do istniejących artykułów i prac naukowych powstały w analizowanym temacie. Jednym z nich jest artykuł powstały w 2018r. na jedną z konferencji poruszającej zagadnienia uczenia maszynowego. Monther Aldawairi i Ali Alwahedi w swojej pracy [9] zwracają uwagę na zdefiniowane przez nas wcześniej wyzwania dotyczące detekcji fałszywych informacji, sugerując jednocześnie kilka elementów, na które należy zwrócić uwagę przy analizie tekstu ze strony. Zgodnie z ich analizą, zbyt długie tytuły artykułów, duża liczba słów napisanych wielkimi literami, znaków wykrzyknienia czy zapytania może wskazywać na potencjalną fałszywość informacji.

Wykorzystując dużą wartość powyższej pracy, nie sposób nie zauważyc pominięcia w niej samej ekstrakcji tekstów artykułu ze stron, która sklasyfikowana została jako jedno z większych wyzwań w naszej pracy. W związku z tym elementem niezbędnym do rozpoczęcia prac było odnalezienie narzędzi, które odpowiednio pozwolą na analizę składniową stron – zarówno jeśli chodzi o budowę struktury języka HTML jak i CSS i JavaScript. Eksploracja źródeł internetowych pozwoliła nam na znalezienie odpowiednich do tego celu narzędzi, których wykorzystanie opisujemy dokładniej w dalszej części pracy (specyfikacji implementacyjnej i opisie realizacji).

---

<sup>11</sup>[https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/#google\\_vignette](https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/#google_vignette), dostęp na dzień 2021.10.15

# Rozdział 3

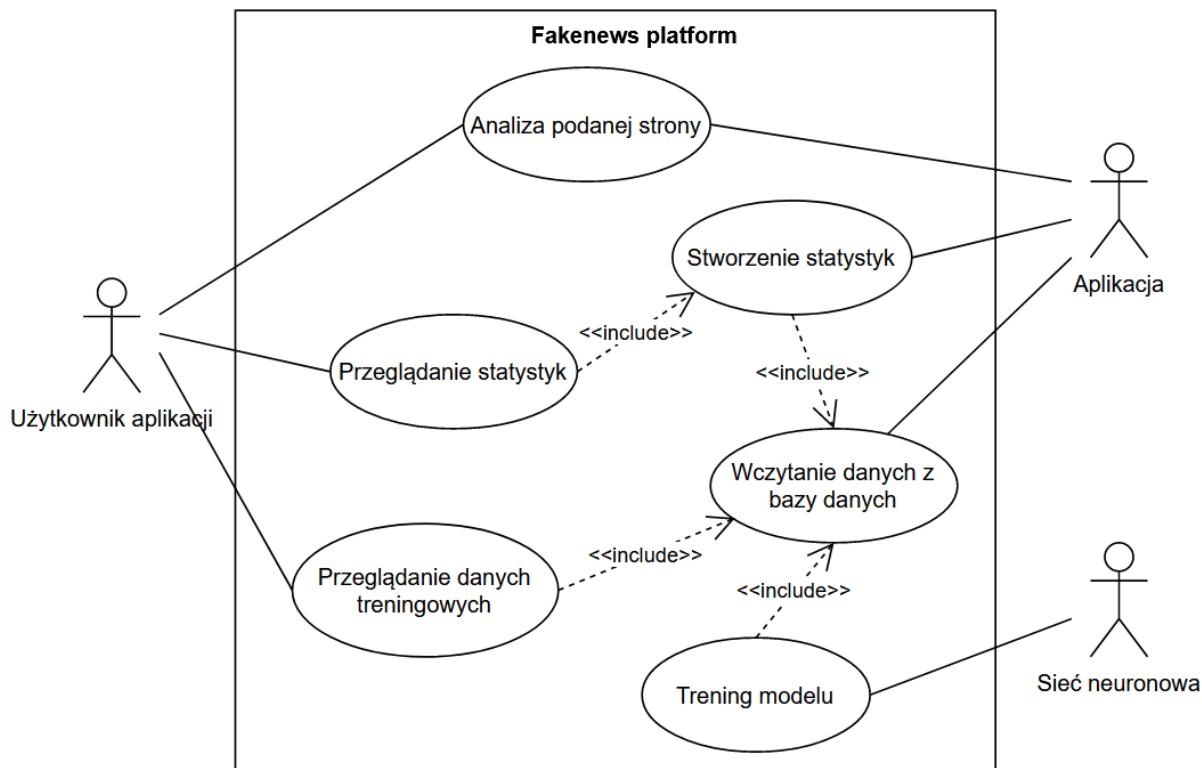
## Specyfikacja funkcjonalna

### 3.1 Wymagania

Istotą aplikacji *Fakenews platform* jest umożliwienie użytkownikowi uzyskanie wyniku wiarygodności dla wybranej przez niego strony. Głównym wymaganiem jest wyliczenie tego współczynnika na podstawie analizy kodu strony z wykorzystaniem narzędzi sztucznej inteligencji. Oprócz wartości wynikowej użytkownik powinien mieć możliwość zapoznania się z danymi ze zbioru treningowego, aby zrozumieć i spróbować zauważać pewne cechy, które łączące cechy o wysokiej ocenie.

Dodatkowo, w ramach aplikacji webowej, użytkownik powinien mieć możliwość przeanalizowania statystyk dotyczących poszczególnych cech. W celu lepszego zrozumienia sposobu realizacji projektu, udostępniona powinna również zostać baza wiedzy na temat zastosowanych rozwiązań, a w szczególności dotycząca algorytmów sieci neuronowych.

## 3.2 Diagram przypadków użycia



**Rysunek 6.** Diagram przypadków użycia aplikacji

Źródło: autorskie, diagram wykonany przy pomocy <https://app.diagrams.net/>

## 3.3 Opis dostępnych funkcjonalności

### 3.3.1 Analiza dynamiczna strony z podanego adresu URL pod kątem jej budowy

Użytkownik ma możliwość wprowadzenia wybranego adresu URL w zakładce startowej aplikacji. Po wysłaniu żądania analizy strony do serwera zwrócone zostaną dane strony wynikające z analizy dynamicznie renderowanej strony. Są to dane o strukturze analogicznej do tych znajdujących się w bazie danych — użytkownik ma możliwość zapoznania się ze statystykami dotyczącymi analizowanej strony takimi jak:

- liczby wybranych tagów HTML na stronie;
- dane dotyczące witryn do których odwołuje się analizowana strona;
- dane dotyczące stylowania z wykorzystaniem języka CSS;

- dane odnośnie interpretowanych na stronie skryptów języka JavaScript;
- informacje dotyczące zawartych na stronie zdjęć i przeprowadzonej dla nich analizy sentymetu.

### **3.3.2 Klasyfikacja strony z podanego adresu URL w kategoriach prawdopodobieństwa prawdziwości informacji**

Oprócz danych statystycznych dotyczących strony opisanych w przypadku poprzedniego wymagania, użytkownik w ramach informacji zwrotnej otrzymuje również informację o sklasyfikowaniu strony przez sieć neuronową. Wynik zawarty w zakresie  $<-1, 1>$  pozwala na zapoznanie się z efektami analizy zadanej strony przez sztuczną inteligencję. W ten sposób odbiorca uzyskuje информацию czy wybrana przez niego witryna jest oceniana jako rzetelna i zawierająca prawdziwe informacje czy też może jest klasyfikowana jako źródło podejrzanych lub fałszywych wiadomości.

### **3.3.3 Możliwość przeglądania bazy danych przeanalizowanych stron**

Odbiorca aplikacji ma możliwość zapoznania się z danymi, które zostały wykorzystane w procesie nauki modelu. Są to pierwotne dane statystyczne, uzyskane przy początkowej analizie strony (w tej sekcji nie zawieramy cech, które tworzymy podczas procesu nauki sieci). Wgląd w dane pozwala na zapoznanie się z informacjami, które są pozyskiwane z analizowanych stron w celu dalszego przetwarzania i ich klasyfikacji. Dzięki tej funkcjonalności użytkownik końcowy może również zgłębić ciekawe dane odnośnie stron internetowych i ich budowy (dzięki umieszczeniu w tabeli wartości dotyczących wykorzystywanych stylów, skryptów czy też odwołań do serwisów zewnętrznych). Możliwe jest sortowanie danych, co pozwala na uszeregowanie stron internetowych zgodnie z ich naturalnym porządkiem dla danej cechy.

### **3.3.4 Możliwość zapoznania się ze statystykami dotyczącymi danych uczących**

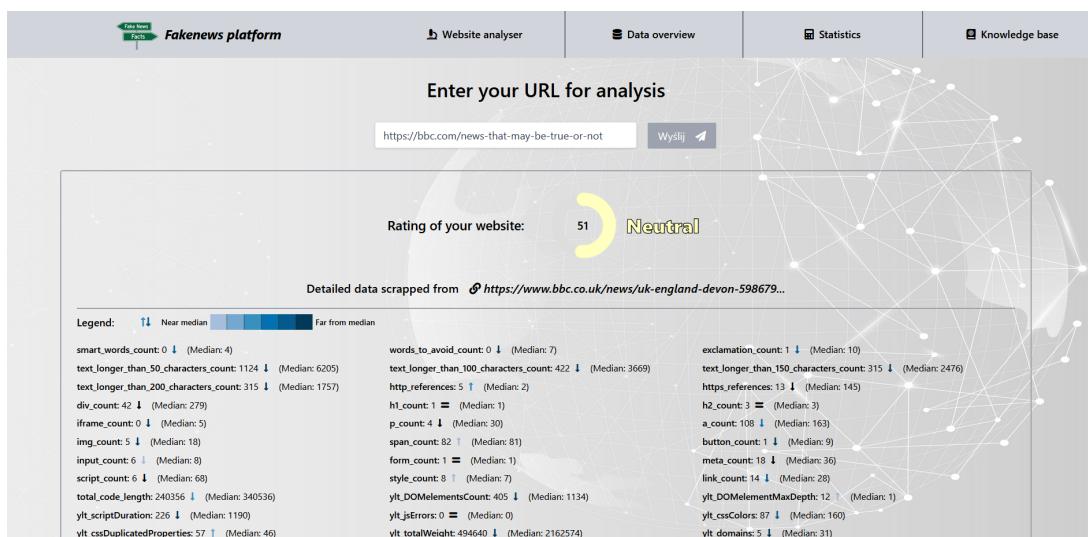
Jednym z modułów aplikacji jest również sekcja statystyk opisowych, w której odbiorca może zgłębić zestaw statystyk odnoszących się do poszczególnych cech. Dla wybranej przez niego cechy prezentowane są wartości skrajne, kwartyle i podstawowe miary zmienności (bardziej szczegółowe rozwinięcie dostępne jest przy opisie widoku „Statistics”[10]).

### 3.3.5 Możliwość zapoznania się z wykorzystanymi algorytmami oraz procesem uczenia sieci neuronowej

Użytkownik ma możliwość zapoznania się z teorią dotyczącą wykorzystywanych mechanizmów oraz narzędzi przy tworzeniu sieci neuronowych jak i procesem uczenia sieci wraz z wizualizacją efektów w kolejnych krokach. Może on zgłębić opisy wykorzystywanych algorytmów, ich uwarunkowania oraz możliwości. Oprócz informacji technicznych odnośnie algorytmu sieci neuronowej ma on również wgląd w technologie, narzędzia, pakiety wykorzystane przy tworzeniu aplikacji. Są to informacje analogiczne do tych zawartych w naszej pracy inżynierskiej, jednakże przekazane „w pigułce”, aby dać użytkownikowi możliwość ogólnego poglądu na mechanizmy działania aplikacji, głównie po stronie serwera, do której nie ma on dostępu.

## 3.4 Widoki aplikacji

### 3.4.1 Analiza podanego URL



Rysunek 7. Widok startowy, pozwalający na analizę wybranej strony

Źródło: autorskie (zrzut ekranu)

Zakładka „Website Analyzer”, będąca jednocześnie stroną startową aplikacji webowej, w bezpośredni sposób wykorzystuje „serce” naszego projektu, jakim jest wyuczony sieci neuronowej. Po wpisaniu, w widocznym na widoku polem tekstowym, wybranego adresu url strony do analizy, wysypane zostaje żądanie do serwera, które inicjuje analizę *webscrapingową* strony, a następnie zebrane dane przekazuje do warstwy sieci neuronowej, która to determinuje ocenę strony pod kątem zawierania wiarygodnych informacji. Przetworzone żądanie zwraca

odpowiedź z oceną wiarygodności strony, która na widoku ukazywana jest w postaci interaktywnego koła, ładującego się w części proporcjonalnej do uzyskanej oceny. Wraz z widocznym w kole wynikiem liczbowym, wyświetlany jest kategoryczny opis rezultatu. W celu lepszej wizualizacji wyniku, do wyświetlenia rezultatu zastosowaliśmy sekwencyjną skalę kolorów, która zaczerpnięta została ze strony Colorbrewer<sup>1</sup>. Jest to źródło niezwykle popularne przy tworzeniu wizualizacji danych, które zawiera wiele propozycji skali kolorów, co może być pomocne przy wizualizacjach kartograficznych, ale również, tak jak w naszym przypadku, przy chęci przedstawienia sentymentu nawiązującego do wizualizowanego wyniku. Wybrana przez nas skala kolorów przyporządkowuje kolory czerwone stronom o niskim wyniku wiarygodności, poprzez żółte dla wyników neutralnych, aż po nasyconą zieloną dla stron o największej wiarygodności.

Oprócz samej oceny strony, w odpowiedzi z serwera otrzymujemy również dane z procesu webscrappingu, które wyświetlane są poniżej, w sekcji "Detailed data". W ramach tej sekcji zaprezentowane zostały wartości dla poszczególnych cech, istotnych przy analizie strony. W celu lepszej wizualizacji wyników, również w tym miejscu zdecydowaliśmy się na zastosowanie skali kolorów i zaprezentowanie odchyлеń wyników od mediany określonej dla całego zbioru uczącego. W tym celu określiliśmy dla danych kwartyle, ale z podziałem na 12 części. 6-stopniowa skala kolorów, uzyskana również przy pomocy narzędzia Colorbrewer, wraz z strzałkami w górę lub w dół, pozwoliła na czytelne wskazanie czy dana wartość jest większa czy mniejsza od mediany oraz jak bardzo od niej odstaje. Wybrane zostały kolory o neutralnym zabarwieniu emocjonalnym (odcienie niebieskiego), co miało na celu nie wprowadzanie użytkownika w błąd, że wyższa lub niższa wartość dla danej kolumny jest czymś pożądanym bądź nie – w zależności od cechy jest to zmienne.

#### 3.4.2 Zebrane dane

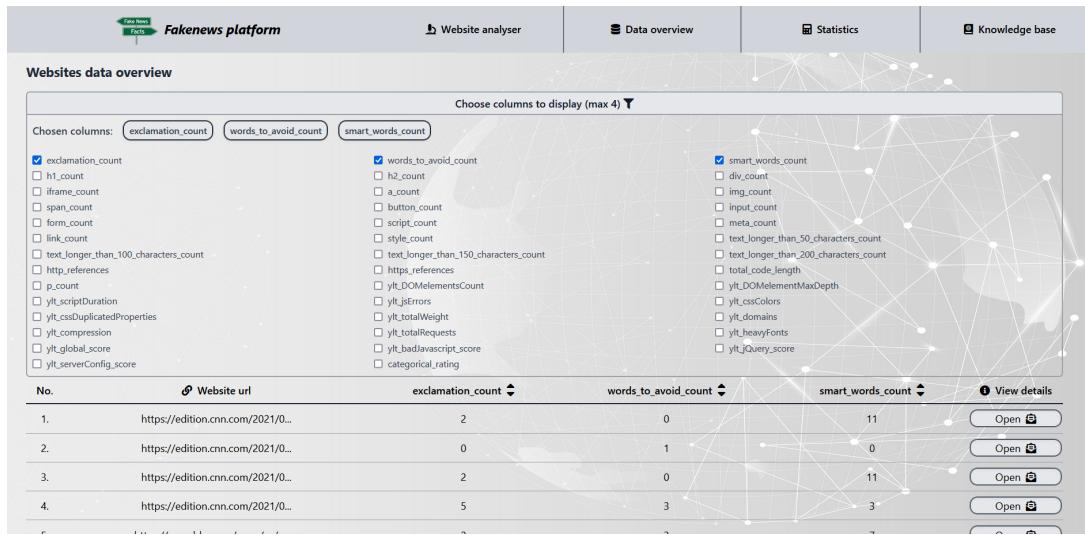
W ramach zakładki „Data overview” (pol. „Przegląd danych”) z bazy danych, za pośrednictwem serwera, pozyskiwana jest pełna tabela danych wykorzystanych do procesu uczenia sieci neuronowej. Widok ten ma umożliwić użytkownikowi zapoznanie się z pełnym zestawem danych, które posłużyły do uczenia modelu.

Z racji na niezwykle dużą liczbę kolumn określających kolejne cechy stron (ich liczba to 40), nie mogliśmy zdecydować się na wyświetlenie wszystkich danych w jednej tabeli – byłoby to bardzo nieczytelne. W związku z tym postanowiliśmy stworzyć zestaw filtrów, przy pomocy których można wybrać maksymalnie 4 kolumny, których dane wyświetcone zostaną w tabeli. Jedyną stałą i nieusuwalną kolumną jest adres URL, który pozwala na identyfikację strony.

---

<sup>1</sup><https://colorbrewer2.org/>, dostęp na dzień 2021.12.18

### Rozdział 3. Specyfikacja funkcjonalna



**Rysunek 8.** Widok aplikacji pozwalający na przeglądanie danych wykorzystanych do nauki modelu sieci neuronowej

Źródło: autorskie (zrzut ekranu)

Inną funkcjonalnością, zaimplementowaną na potrzeby zwiększenia możliwości analizy danych przez użytkownika, są dodane dla każdej kolumny wskaźniki sortujące. Po wybraniu odpowiednio strzałki w dół lub w góre istnieje możliwość posortowania jednej z wybranych kolumn w kolejności rosnącej lub malejącej odpowiednio dla danej cechy.

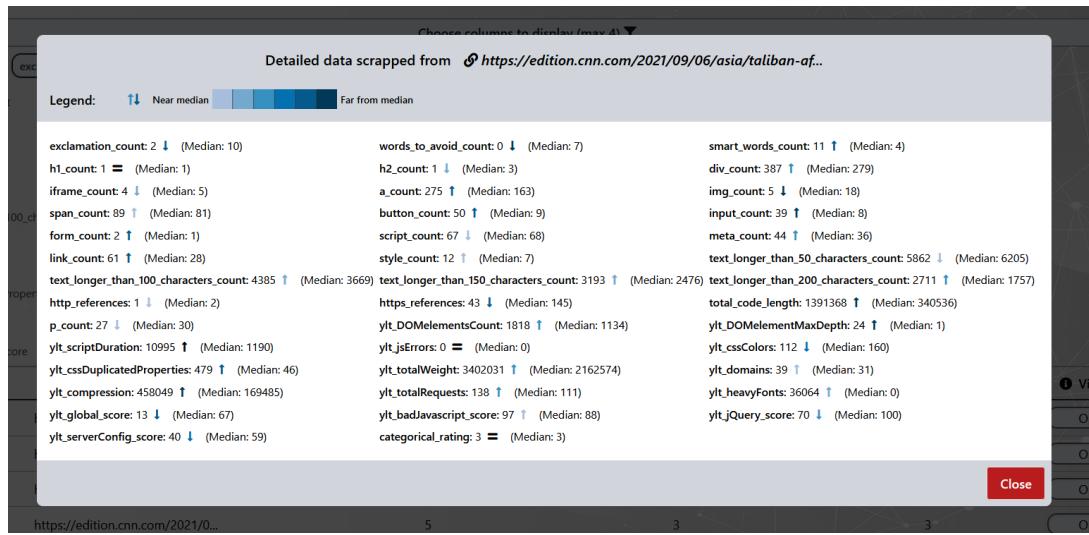
Dodatkowo, w celu umożliwienia przeglądu wartości dla wszystkich kolumn, udostępniliśmy widok szczegółowy w formie modala, otwieranego przy pomocy przycisku „Open” w ostatniej kolumnie tabeli (rysunek 9). Po naciśnięciu przycisku wyświetlane jest okno z zawartością podobną do tej z grafiki 7. Oprócz samych wartości dla kolejnych kolumn przedstawiona jest skala kolorów wraz ze strzałkami, które określają położenie wartości we wskazanym oddaleniu od mediany.

#### 3.4.3 Statystyki

Złożoność i wielkość danych wymusza obecność oddzielnej sekcji na statystyki dotyczące poszczególnych cech. W związku z tym faktem, w widoku „Statistics”[10] zaprezentowane są podstawowe, statystyczne dane opisowe dotyczące kolejnych cech. W celu wyświetlenia statystyk dla cechy, należy wybrać stosowną nazwę kolumny z menu kontekstowego po lewej stronie ekranu. W prawej części ukazane zostają kolejne wartości określające:

- 25%, czyli piwerszy kwartyl danych;
- 50% – medianę;
- 75% – trzeci kwartyl danych;

### 3.4. Widoki aplikacji



Rysunek 9. Modal ze szczegółowymi danymi dotyczącymi wybranej strony

Źródło: autorskie (zrzut ekranu)



Rysunek 10. Widok statystyk dla poszczególnych cech

Źródło: autorskie (zrzut ekranu)

- max – wartość maksymalną;
- min – wartość minimalną;
- std – odchylenie standardowe.

Oprócz wartości liczbowych w sekcji umieszczone zostaną wykresy dla poszczególnych cech, np. wykresy skrzypcowe, których wybór zdeterminowany jest faktem, iż, w porównaniu do np. wykresów pudełkowych, ten rodzaj oferuje możliwość prezentacji rozkładu zmienności i dystrybucji cechy, co może uwidoczyć chociażby wielomodalność rozkładu.

### 3.4.4 Baza wiedzy

W ramach widoku „Knowledge base” zaprezentowane i opisane zostały najistotniejsze rozwiązania zastosowane przy realizacji projektu. Opisane metody dotyczą głównie aspektu naukowego: analizy danych (webscrappingu) oraz procesu uczenia sieci neuronowej i wykorzystanych do tego algorytmów. Informacje zaprezentowane są możliwie najczęściej w sposób graficzny, co pozwala odbiorcy na prostą i treściwą analizę. W tej sekcji znajdują się również uwagi dotyczące podejmowanych działań w selekcji i transformacji cech oraz proste statystyki odnoszące się do każdego z poszczególnych kroków, takie jak:

- macierze pomyłek;
- macierze korelacji;
- histogramy cech;
- krzywe ROC;
- skuteczność sieci w kolejnych epokach.

Szczegóły techniczne dotyczące procesu implementacji pozostałych komponentów oraz wykorzystanych narzędzi opisuje specyfikacja implementacyjna, będąca częścią niniejszej pracy.

Tu będzie foto z bazy wiedzy jak zostanie dokonczona

## 3.5 Format danych wejściowych

W celu uzyskania oceny dotyczącej wskazanej strony, od strony użytkownika aplikacji webowej, należy wskazać jej adres URL, zgodny z podstawowymi wymaganiami<sup>2</sup> (np. <http://www.example.com/questions/3456/my-document>). Adres powinien zostać wprowadzony w pole tekstowe dostępne w widoku startowym „Website analyzer”[grafika 7].

---

<sup>2</sup>Wymagania i schematy adresu URL: <https://en.wikipedia.org/wiki/URL>, dostęp na dzień 2022.01.04

# Rozdział 4

## Specyfikacja implementacyjna

### 4.1 Stos technologiczny

#### 4.1.1 Przetwarzanie po stronie serwera (backend)

##### Biblioteka requests\_html

Jest to narzędzie wspomagające webscrapping, które odpowiada za dostęp do kodu wybranej strony. Poprzez obsługę JavaScriptu, biblioteka pozwala na renderowanie kodu nowoczesnych stron, których treść jest ładowana dynamicznie, poza podstawowym żądaniem GET inicjowanym poprzez wpisanie adresu w pasku przeglądarki. Z wykorzystaniem tego narzędzia możliwe jest łatwiejsze i szybsze uzyskanie faktycznego kodu strony internetowej, który udostępniany jest w formie artykułu użytkownikowi.

##### Beautiful Soup

*Beautiful Soup* to biblioteka w języku Python służąca do filtrowania plików HTML i XML celem ekstrakcji poszukiwanych danych. Została wykorzystana przy analizie kodu źródłowego stron, które zostały poddane analizie. W pracy wykorzystana została najnowsza wersja 4.9.3<sup>1</sup>, która współpracowała z wykorzystaną w całości projektu wersją języka *Python* 3.9.9.

Biblioteka ta potrafi przekształcić kod HTML do bardziej czytelnej wersji - funkcja *.prettify()*. Pozwala to na czytelne przedstawienie kodu w postaci zagnieżdzonej. Po takiej transformacji fragmenty tekstu są łatwo znajdowane jako wartości zmiennych w obiekcie przechowującym przekształcony kod HTML. W pracy nieocenioną wartość miało znajdowanie linków wewnętrz kodu. W ten sposób można zautomatyzować proces znajdowania linków i źródeł wewnętrz przeszukiwanej strony.

---

<sup>1</sup>Według danych z 17.06.2021

## Yellow Lab Tools

Jest to kolejne narzędzie, które, razem z Beautiful Soup, odpowiada za stworzenie cech dla modelu sieci neuronowej. Umożliwia ono przetestowanie czy strona spełnia dobre praktyki deweloperskie poprzez analizę możliwych problemów z kodem HTML, CSS, JS lub obrazami czy czcionkami. Poprzez wysłanie odpowiedniego żądania do serwera usługi, możliwe jest uzyskiwanie niezbędnych danych. Autorem rozwiązania jest Atanar Technologies<sup>2</sup>, firma działająca w zakresie rozwoju rozwiązań CDN, pozwalających na zwiększaniu i poprawianiu jakości dostępu do zawartości stron internetowych<sup>3</sup>.

## Flask

Flask to mikro framework języka Python, stworzony do projektowania aplikacji webowych<sup>4</sup>. Wykorzystaliśmy go w celu stworzenia warstwy logiki aplikacji. Dzięki zastosowaniu w nim silnika Jinja2 aplikacja zapewnia bezpieczeństwo przed atakami typu *Cross-site scripting*.

## Numpy i Pandas

Są to podstawowe narzędzia służące do sprawnego zarządzania danymi wykorzystywanyimi w analizie danych. Dzięki kompatybilności z bibliotekami ML oraz mnogości funkcji usprawniających działania matematyczne usprawniają one pracę przy trenowaniu pierwszych modeli sieci neuronowych, które stanowią punkt wyjścia do projektowania końcowych rozwiązań.

## Tensorflow i Keras

Keras to interfejs zaprojektowany przez François Cholleta, korzystający z rozwiązań oferowanych przez ekosystem Tensorflow, pozwalający na tworzenie modeli z wykorzystaniem predefiniowanych sieci neuronowych. Pozwolił nam na łatwe stworzenie prototypów sieci które wykorzystujemy w fazie projektowania naszego modelu. Modularność oferowana w tym interfejsie pozwala nam na stopniowe zwiększanie skomplikowania prototypów, jednocześnie zapewniając jasne komunikaty odnośnie wyników i występujących błędów.

## Keras Tuner

Nieocenioną pomocą przy projektowaniu i testowaniu prototypów modelu jest framework Keras Tuner. Pozwala nam na automatyzację optymalizacji hiperparametów sieci. Korzystając z

---

<sup>2</sup><https://www.atanar.com/>

<sup>3</sup>CDN – istota działania: <https://www.akamai.com/our-thinking/cdn/what-is-a-cdn>, dostęp na dzień 2022.01.04

<sup>4</sup>Wg opisu twórcy - <https://palletsprojects.com/p/flask/>

wbudowanych algorytmów optymalizacji przeprowadza testy wielu sieci dąży do określonego przy tworzeniu środowiska testowego celu, w naszym przypadku jest to maksymalizacja precyzji klasyfikacji ocenianych stron.

### 4.1.2 Warstwa interfejsu użytkownika (frontend)

#### Vue JS

Vue JS jest popularnym oraz dynamicznie rozwijającym się frameworkm stworzonym do realizacji zarówno prostych, jak i bardzo rozbudowanych aplikacji webowych. Z jego funkcjonalności korzystają największe dostawcy i wiele przedsiębiorstw, z których możemy wyróżnić m.in. IBM, GitLab czy Adobe. Popularność frameworka wynika z szybkości implementacji projektów oraz oszczędności jeśli chodzi o wykorzystanie zasobów – Vue korzysta z wirtualnego DOMu (Document Object Development), czyli obiektowego modelu dokumentu, a co za tym idzie nie przeprowadza zmian na DOMie przeglądarki (na czym opiera się m.in. framework Angular, jedno z innych rozwiązań w zakresie rozwoju aplikacji webowych). Jest to świetna rozwiązanie do budowy rozwiązań typu Single Page Application, gdzie większość zapytań do serwera wykonywania jest asynchroniczne i odpowiada jedynie za pobranie danych do późniejszego wyświetlenia na stronie. Olbrzymią zaletą jest również skalowalność, która umożliwia wykorzystanie projektów tworzonych w Vue nie tylko jako kompletnych monolitycznych aplikacji, ale również jako komponentów możliwych do wykorzystania w innych aplikacjach.

### 4.1.3 Narzędzia pomocnicze

Niezbędną częścią naszej pracy było umiejętne korzystanie z istniejących rozwiązań, które umożliwiłyby ocję wykorzystanie celem usprawnienia pracy. Dzięki temu nasze zadanie jest skupione na szukaniu rozwiązania problemu *Fake News*. Dodatkowo dzięki obszernym dokumentacjom mogliśmy dostosować ich funkcjonalności wedle naszych potrzeb.

#### Chrome DevTools

Przed procesem pełnej automatyzacji niezbędnym było przeprowadzenie ręcznej analizy przykładowych stron, celem znalezienia anomalii w kodzie lub nieuczciwych skryptów i reklam na stronie z potencjalnym *Fake News*. W tym celu użyte zostało narzędzie Chrome DevTools, które służyło do analizy dowolnej ze stron w internecie. Korzystając z zapisu wszystkich zapytań wysyłanych przez przeglądarkę podczas włączania strony z informacjami udało się

określić konkretnych *SSP*, którzy są skłonni wyświetlać reklamy z gorszymi reklamami niż ogólnoprzyjętymi normami.

Chrome DevTools pozwala na pełen podgląd zarówno kodu źródłowego strony, jak i wszystkich dodatkowych kodów wykonywanych w tle otwieranej lub przeglądanej witryny. Dodatkowo, po włączeniu zakładki *Network* zapisuje wszystkie zapytania – zarówno pobierane, jak i wysyłane. Można tam sprawdzić, co było inicjatorem wysłania zapytania, a także treść takiego zapytania, a także czas przetwarzania takiego zapytania.

## 4.2 Interfejs API

W ramach komunikacji pomiędzy warstwą klienta (aplikacja frontendowa) oraz warstwą serwerową (aplikacja backendowa) stworzyliśmy prosty interfejs API (Application Programming Interface), który umożliwia wysyłanie żądań z przeglądarki klienta. W ten sposób zapewniliśmy możliwość pozyskiwania danych z bazy danych, za co odpowiada warstwa serwerowa aplikacji. Z uwagi na niewielką liczbę żądań niezbędnych do komunikacji obu warstw, w naszym przypadku wystarczające okazały się jedynie dwa, następujące endpointy (punkty końcowe usługi sieci Web):

- [POST] /search – żądanie wywołujące pełną analizę podanego w ciele zapytania adresu url. Oprócz informacji o stronie uzyskanymi w procesie webscrappingu, zapytanie zwraca informację o „współczynniku prawdziwości” strony, czyli parametrze uzyskanym po analizie strony przez wytrenowaną wcześniej sieć neuronową. Ciało zapytania powinno wyglądać następująco:

```
1 {
2     "url": "https://www.someurl.com/article"
3 }
```

---

**Listing 1.** Ciało żądania do analizy strony

- [GET] /fetchdata – żądanie pozwalające na pobranie wszystkich danych zebranych w bazie danych, czyli rekordów służących do treningu sieci neuronowej. Nie wymaga ono podawania żadnych parametrów, zwracane są wszystkie kolumny z wszystkich rekordów w bazie danych, dzięki czemu w pełni można się z nimi zapoznać z poziomu przeglądarki.

## 4.3 Opis przechowywanych danych

Dane, będące podstawą do przeprowadzenia analizy eksploracyjnej oraz treningu i testowania sieci neuronowych, zgromadzone zostały przez nas w pojedynczej tabeli bazy danych.

Wykorzystanym systemem zarządzania bazą danych był MySQL – znana i powszechnie stosowana usługa, która pozwala na proste i sprawne uzyskiwanie dostępu do danych oraz zapisywanie nowych rekordów. W ramach bazy danych i istniejącej w niej tabeli `websites` stworzyliśmy szereg kolumn, które służyły do przechowywania danych charakterystycznych dla poszczególnych cech analizowanych artykułów. Lista poniżej prezentuje ich nazwy oraz krótkie wyjaśnienie istotności dla analizy zawartości stron:

- `url` – jednoznacznie identyfikujący adres strony, będący primary key w tabeli;
- `exclamation_count` – liczba słów w tekście znajdującym się na stronie napisanych w całości wielkimi literami oraz wykrzykników i znaków zapytania, czyli charakterystycznych elementów dla treści nacechowanych emocjami, próbującymi bardzo bezpośrednio „dotknąć” odbiorcę poprzez wykrzyknięcia i stosowanie pytań;
- `words_to_avoid_count` – liczba słów znalezionych w tekście w języku angielskim, których należy unikać przy pisaniu profesjonalnych artykułów. Ich lista dołączona jest do pracy jako załącznik nr x;
- `smart_words_count` – znaleziona w tekście strony liczba słów, które uznawane są za wartościowe dla dobrych i profesjonalnych artykułów. Ich lista również dołączona jest do pracy jako załącznik nr x;
- `h1_count` – liczba tagów `h1` na stronie, będących tagami nagłówkowymi, wykorzystywany m.in. przy indeksowaniu stron w Google czy przy przetwarzaniu strony na wersję dla osób niewidomych;
- `h2_count` – liczba tagów `h2`, czyli niższych w hierarchii tagów, służących do oznaczania podsekcji strony. Wykorzystanie tego typu tagów jest mniej popularne w aplikacjach biznesowych, ale pełni istotną rolę w stronach z informacjami, gdzie możemy z wykorzystaniem tagów z kategorii heading uzyskać dobre indeksowanie w wyszukiwarkach z jednoczesnym utworzeniem odwołań do podsekcji w rezultatach wyszukiwania;

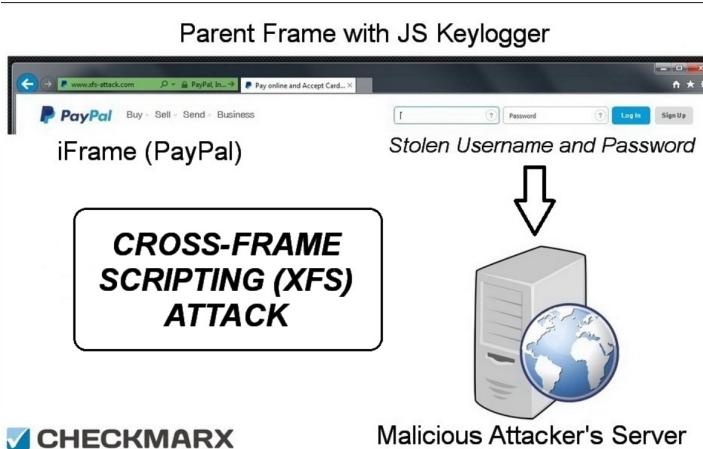


**Rysunek 11.** Efekt wykorzystania tagów typu heading w artykule zaprezentowany w wynikach wyszukiwania wyszukiwarki Google

Źródło: autorskie (zrzut ekranu ze strony <https://www.google.com/>)

- `div_count` – liczba podstawowych elementów języka HTML na stronie, grupujących i oddzielających kolejne sekcje i kontenery, wskazująca poniekąd na jej złożoność, niezwykle zależna od przyjętej architektury;

- `iframe_count` – liczba znaczników służących do tworzenia obszarów, na których mogą być osadzone inne dokumenty. Ten element jest niezwykle często wykorzystywany przez podmioty zajmujące się marketingiem internetowym, gdyż możliwe jest osadzenie w nim zewnętrznych materiałów takich jak reklamy czy po prostu innej strony. Jest on również wykorzystywany przez internetowych przestępcoów, którzy za pośrednictwem odpowiednie spreferowanego fragmentu strony stosują atak typu XFS, który ma za zadanie wyłudzenie danych od atakowanego poprzez stworzenie pozoru znajdującej się na bezpiecznej stronie, co prezentuje poniższa grafika;



Rysunek 12. Przykład strony realizującej atak z wykorzystaniem XSF

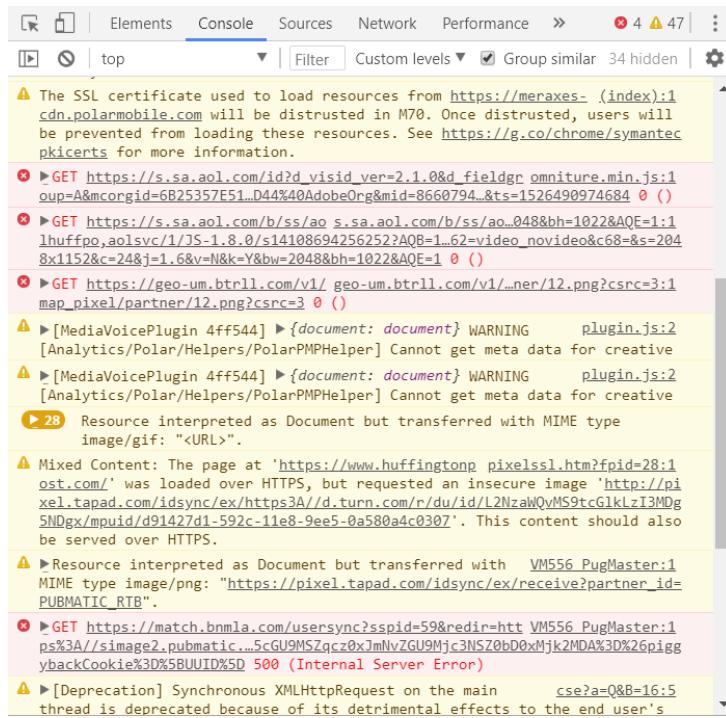
Źródło: [7]

- `a_count` – liczba odnośników do innych źródeł. Liczba tego typu elementów pozwala na oszacowanie jak często strona odnosi się do różnych źródeł poprzez klasyczne linki;
- `img_count` – liczba zdjęć na stronie;
- `span_count` – liczba tagów `span`, będących jednym z elementów infrastrukturalnych języka HTML, nie reprezentujących samym sobą żadnych treści;
- `button_count` – liczba klasycznych przycisków na stronie. W nowych infrastrukturach są one wykorzystywane coraz rzadziej (obsługę zdarzenia kliknięcia można zrealizować w dowolnym elemencie);
- `input_count` – liczba dostępnych pól interakcji z użytkownikiem. Element ten może być ukazany na stronie w formie pola tekstowego, ale również w formie przełączników typu toggle, radio czy checkboxów. Świadczą o interaktywności strony i możliwości jej komunikacji z użytkownikiem;
- `form_count` – liczba formularzy na stronie, szczególnie popularna w stronach o starszej architekturze, prosta forma interakcji z użytkownikiem, która pozwala na bezpośrednie wysyłanie żądań do serwera HTTP;

- `script_count` – liczba skryptów na stronie. Określa nam jak wiele różnych skryptów zostało zastosowanych do renderowania strony. Obsługa zdarzeń przy pomocy JavaScript daje wachlarz wielkich możliwości przy tworzeniu efektów na stronie, jednakże jest bardzo obciążająca dla ładowania strony;
- `meta_count` – liczba metadanych zaszytych w kodzie HTML. Elementy tego typu nie są wyświetlane na stronie, ale są niezwykle istotne dla elementów analiz maszynowych. Z informacji tych bardzo często korzystają silniki wyszukiwarek, przez co są one chętnie wykorzystywane przez strony z informacjami w celu poprawnej indeksacji i odpowiedniego pozycjonowania;
- `link_count` – liczba tagów definiujących powiązania pomiędzy bieżącym dokumentem a zewnętrznymi źródłami;
- `style_count` – liczba sekcji określających stylowanie na stronie;
- `text_longer_than_50_characters_length` – sumaryczna długość tekstów powyżej 50 znaków może świadczyć o stopniu rozbudowania wypowiedzi, jej zaawansowaniu i spójności, a co za tym idzie o jej wartości merytorycznej;
- `text_longer_than_100_characters_length` – kolumna analogiczna do opisanej wyżej, uwzględniająca jednak teksty o długości co najmniej 100 znaków;
- `text_longer_than_150_characters_length`;
- `text_longer_than_200_characters_length`;
- `http_references` – liczba odwołań do stron bez certyfikatu SSL. Jest to istotna informacja jeśli chodzi o wskazywanie na stronie odniesienia. Obecność odwołań do stron bez połączenia szyfrowanego powinna budzić zastanowienie i być ostrzeżeniem przed możliwie niebezpiecznym źródłem;
- `https_references` – w z opisaną wyżej cechą, liczba odwołań do źródeł z wgranyim ważnym certyfikatem SSL może być cenną informacją w ocenie jakości źródeł, z których korzysta analizowana strona;
- `total_code_length` – całkowita długość kodu strony, która pozwala ocenić złożoność strony, a co za tym idzie jej rozmiar oraz możliwość szybkiego ładowania w przeglądarce;
- `p_count` – liczba tagów służących do oznaczania bloków tekstu, wykorzystywana często do naturalnego separowania akapitów lub bloków tekstu;
- `ylt_DOMelementsCount` – sumaryczna liczba elementów DOMu, czyli obiektowego modelu dokumentu, która również pozwala ocenić złożoność strony;
- `ylt_DOMelementMaxDepth` – maksymalne zagnieżdżenie elementów na stronie, które, analogicznie do wspomnianej wyżej cechy, jest wyznacznikiem złożoności strony;
- `ylt_scriptDuration` – długość wykonywania skryptu JavaScript przy ładowaniu strony, która świadczy o jakości budowy strony i jej prędkości działania;

## Rozdział 4. Specyfikacja implementacyjna

- `ylt_jsErrors` – liczba błędów JavaScript, która jasno wskazuje na jakość strony i precyzyjność jej odpowiednią kompatybilność z przeglądarkami (strona powinna być zgodna z różnymi wersjami przeglądarek, a nie tylko z konkretnymi wersjami);



Rysunek 13. Przykładowe błędy JavaScript w konsoli przeglądarki

Źródło: [10]

- `ylt_cssColors` – liczba kolorów w stylowaniu z wykorzystaniem języka CSS. W stronach z informacjami istotne jest, aby liczba ta była ograniczona, co pozwala odbiorcy skupić się na analizowanym tekście artykułu;
- `ylt_cssDuplicatedProperties` – redundancja definicji określonych właściwości dla tagów (ich kilkukrotne definiowanie jest jedynie obciążeniem dla przeglądarki);
- `ylt_totalWeight` – całkowita waga strony, jest to kolejny wskaźnik pozwalający na ocenę jakości strony pod względem jej rozmiaru, a co za tym idzie określenie jej wydajności;
- `ylt_domains` – liczba odwołań do różnych domen. W przypadku stron z informacjami jest to istotne w kwestii różnorodności źródeł, z których korzysta strona, czy są one powiązane z tą samą witryną, czy też odnośniki prowadzą do innych stron, które często są jedynie materiałami marketingowymi, których zazwyczaj wolimy unikać przy pozyskiwaniu informacji z Internetu;

- `ylt_compression` – możliwość kompresji strony z zastosowaniem klasycznego algorytmu kompresji bezstratnej .zip, co jest kolejną cechą wskazującą na „ciężar” strony i jej łatwość ładowania, szczególnie istotną na urządzeniach mobilnych;
- `ylt_totalRequests` – liczba żądań HTTP realizowanych automatycznie przez stronę, czyli jeden z istniejszych wskaźników w analizie dynamicznej strony. Duża liczba żądań wiąże się zazwyczaj z odwołaniami do zewnętrznych źródeł, z których większość wiąże się zazwyczaj ze ściąganiem materiałów reklamowych ze źródeł zewnętrznych, które bardzo często odciągają uwagę odbiorcy i przeszkadzają w skupieniu się nad tekstem;
- `ylt_heavyFonts` – liczba „ciężkich” czcionek, czyli tych o dużym rozmiarze, będących również obciążeniem przy renderowaniu dla przeglądarki;
- `ylt_global_score` – wynik strony z serwisu Yellow Lab Tools, który pozwala na ocenę strony pod kątem wykorzystanego stylowania;
- `ylt_badJavascript_score` – ocena strony pod kątem wykorzystanych złych praktyk w języku JavaScript takich jak: wykorzystywanie funkcji `document.write`, stosowanie globalnych zmiennych oraz żądań synchronicznych, które spowalniają działanie strony, a także samej liczby błędów JavaScriptu przy renderowaniu strony;
- `ylt_jQuery_score` – wynik wyliczony na bazie liczby odwołań do biblioteki jQuery, która przez swoją dużą objętość poważnie spowalnia działanie strony i jest raczej reliktem przeszłości;
- `ylt_serverConfig_score` – wszystkie żądania, które strona wykonuje do serwerów HTTP cechują się pewnymi informacjami przekazywanymi w nagłówkach. Zgodnie ze sztuką bezpiecznego przesyłu danych, w tego typu nagłówkach nie powinny znajdować się informacje odnośnie serwera czy zastosowanych na nim technologii.

## **4.4 Parametry uruchomieniowe programu**

Uruchomienie aplikacji w sposób umożliwiający użytkownikowi poddanie analizie wybranego adresu URL wymaga wywołania komend odpowiadających za działania zarówno warstwy serwerowej jak i interfejsu graficznego. W celu uruchomienia aplikacji serwerowej, w katalogu głównym projektu, należy wywołać komendę `python main.py`. Kolejnym krokiem powinno być przejście do katalogu `frontend`, co możemy uczynić wpisując w konsolę `cd frontend`. Jest to katalog zawierający kod interfejsu graficznego użytkownika. W celu umożliwienia dostępu do aplikacji z poziomu przeglądarki internetowej należy wykonać komendę `npm run serve`. Następnie, należy przejść pod adres `http://localhost:8080`, gdzie dostępna będzie działająca aplikacja.



# Rozdział 5

## Realizacja aplikacji

### 5.1 Wykorzystany sprzęt

Realizując projekt z zakresu przetwarzania zbiorów danych i uczenia sieci neuronowej oraz emulowania lokalnie jądra przeglądarki, istotną kwestią jest sprzęt, na którym wykonywane są prace. Ma to wpływ nie tylko na czas wykonywania poszczególnych zadań, ale również na stabilność całej aplikacji. W naszym przypadku możliwości sprzętowe obejmowały 2 laptopy o następującej specyfikacji:

- Dell Latitude 5410, procesor Intel Core i7-10610U CPU @ 2.30 GHz, system operacyjny Windows 10 Pro, grafika Intel UHD Graphics 620, 16GB pamięci RAM;
- Dell Latitude 5580, procesor Intel Core i5-7200U CPU @ 2.50GHz, system operacyjny Microsoft Windows 10 Pro, grafika Intel HD Graphics 620, 16GB pamięci RAM.

### 5.2 Aplikacja serwerowa (backend)

#### 5.2.1 Pozyskiwanie kodu strony

Aby móc skutecznie zebrać dane niezbędne do analizy wybranego artykułu i późniejszego treningu sieci neuronowej, zadaniem priorytetowym okazało się samo pozyskanie danych. Wszelkie informacje, które chcieliśmy wykorzystać w naszym projekcie, możliwe były do uzyskania z jedynego, ale jednocześnie najbogatszego źródła, które zapewnia nam podany jako dana wejściowa adres URL, czyli kodu HTML strony. W tym miejscu musimy zwrócić uwagę na niezwykle istotny aspekt, który czyni to zadanie nietrywialnym. Istniejące obecnie strony są responsywne, interaktywne, a co za tym idzie, wiele kodu wykonywane jest podczas interakcji użytkownika z danym serwisem. Z tego powodu, faktyczna treść artykułu zaprezentowanego na danej stronie nie jest tożsama z treścią pobraną z wykorzystaniem

żądania GET, wykonanego poprzez wpisanie adresu strony w pasek wyszukiwania przeglądarki. Całą warstwę interaktywności obsługuje i zarządza język JavaScript, który obok statycznie renderowanego języka HTML, stanowi serce responsywności aplikacji webowej. Powoduje to duże różnice w faktycznie prezentowanym na stronie kodzie języka HTML, który budowany jest już po pobraniu strony żądaniem HTTP z serwera WWW. W związku z opisanyimi powyżej cechami obecnych stron internetowych musielibyśmy znaleźć rozwiązanie, które przed podaniem kodu analizie pozwala na zasymulowanie ładowania strony w przeglądarce. Do tego celu posłużyła nam, wspomniana w specyfikacji implementacyjnej, biblioteka requests\_html, która, z wykorzystaniem jądra przeglądarki Google Chrome (Chromium), bez interfejsu graficznego, pozwala programistycznie uzyskać kod strony, która wyświetlana jest użytkownikowi podczas czytania artykułu. W celu uzyskania poprawnego działania biblioteki, musielibyśmy utworzyć jej lokalną kopię (tzw. fork<sup>1</sup>, który następnie został przez nas zmodyfikowany pod działanie w systemie operacyjnym Windows. Po dokonaniu modyfikacji sama realizacja renderowania strony, sprowadziła się do poniższych linii kodu, które przypisują kod strony do jednej zmiennej, będącej podstawą dalszych analiz.

```
1 def get_raw_html(url):
2     session = requests_html.HTMLSession()
3     r = session.get(url)
4     r.html.render(sleep=2, keep_page=True, scrolldown=1, timeout=50)
5     html_raw = r.html.raw_html
6     session.close()
7     return html_raw
```

---

**Listing 2.** Funkcja renderująca kod HTML do analizy

Co istotne, wszystkie strony, które trafiały do zbioru treningowego sieci neuronowej, musiały być zweryfikowane pod kątem obecności treści nie będącej artykułem, np. modalą z zapytaniem o chęć zapisu do newslettera lub blokady płatnej treści (tzw. „paywall”[14]). Było to konieczne ze względu na fakt, że niektóre strony nawet pomimo wyrenderowania pełnej zawartości z wykorzystaniem JavaScript’u, zawierają podobne treści w pełni przesłaniające faktyczne treści artykułów, co potencjalnie obarczałoby błędem uczony model.

## 5.2.2 Webscrapping

Opisane w poprzedniej sekcji pozyskiwanie kodu strony było tylko prologiem do kolejnego kroku, czyli webscrapingu (tłumacząc na język polski „zdrapywania z sieci”). Proces ten

---

<sup>1</sup> „Fork a repo” – czym jest i jak wykonać operację fork: <https://docs.github.com/en/get-started/quickstart/fork-a-repo>

The threat posed by the Omicron variant has now come into sharper focus, with recent clinical data and laboratory studies lending support to early reports suggesting that it is milder but more transmissible than other variants of the new coronavirus.

“It spreads very, very fast, but it doesn’t appear to have the virulence or machismo to really pack as much of a wallop as the Alpha or Delta variants,” James Musser, chairman of Houston Methodist Hospital’s pathology and genomic medicine department and the leader of a new study of Omicron infections, said of the variant.

[TO READ THE FULL STORY](#)

[SUBSCRIBE](#)

[SIGN IN](#)

### THE WALL STREET JOURNAL.

Continue reading your article with  
a WSJ membership

[VIEW MEMBERSHIP OPTIONS](#)

**Rysunek 14.** Przykład blokady płatnej treści, uniemożliwiający prawidłową analizę treści strony  
Źródło: autorskie (zrzut ekranu)

zakładał odseparowanie, w ramach danych zawartych na stronie, danych użytecznych od tych zbędnych. Finalnym celem było uzupełnienie kolejnych kolumn w bazie danych, odwołujących się do co istotniejszych informacji dotyczących analizowanej strony. Opisane poniżej techniki definiują kroki, które podjęliśmy, aby możliwie najpoprawniej pozyskać niezbędne do uczenia sieci neuronowej dane.

### Wyszukiwanie widocznego tekstu

W celu możliwości poddania analizie kodu strony nie tylko jako całość, ale również jego istotniejszych elementów, konieczne było zastosowanie technik pozwalających na wybór jedynie interesujących nas elementów. Separacja tekstu z kodu całej strony wykonana została z pomocą przedstawionych na listingu 3 metod, które, analizując kolejne zawartości tagów na stronie, określają, czy dany element klasyfikujemy jako tekst. Połączona zawartość elementów uznanych za elementy widoczne na stronie jest łączona i zwieracana z metody `text_from_html` jako podstawa do dalszych analiz tekstu.

```
1 def tag_visible(element, min_length):  
2     words_to_ignore_file = open("resources/words_dictionaries/  
3         words_to_ignore.txt", 'r')  
4     words_to_ignore_list = words_to_ignore_file.read().split(",")
```

```
5     if element.parent.name in ['style', 'script', 'head', 'title', 'meta', '[document]']:
6         return False
7     if isinstance(element, Comment):
8         return False
9     if len(element) < min_length:
10        return False
11    if not isinstance(element, NavigableString):
12        return False
13    if len(element.parent.findChildren("a", recursive=True)) > 25:
14        return False
15    if any(word.upper() in element.upper() for word in
16          words_to_ignore_list):
17        return False
18
19
20 def text_from_html(soup, min_length):
21     texts = soup.findAll(text=True)
22     visible_texts = filter(lambda element: tag_visible(element,
23         min_length), texts)
24     return u" ".join(t.strip() for t in visible_texts)
```

**Listing 3.** Funkcje odpowiedzialne za wyizolowanie tekstu widocznego na stronie do dalszych analiz

Metoda `tag_visible`, przekazana jako argument do funkcji filtrującej kod strony, działa w naszym przypadku jako prosty system ekspertowy, który sprawdzając kolejne warunki, sprawdza czy zawartość analizowanego elementu może być uznana jako tekst widoczny na stronie. Wszystkie warunki mają swoje uzasadnienie w zastosowaniu, które należy w tym miejscu rozbić na czynniki pierwsze:

- `element.parent.name in ['style', 'script', 'head', 'title', 'meta', '[document]']` – sprawdzenie czy zawartość elementu nie znajduje się w drzewie obiektów HTML w jednym z tagów z pewnością nie będących wyświetlanymi na stronie;
- `isinstance(element, Comment)` – warunek określający czy element nie został sklasyfikowany przez BeautifulSoup jako komentarz;
- `len(element) < min_length` – test sprawdzający długość tekstu, pozwala on na odrzucenie elmentów zbyt krótkich przy tworzeniu pełnego tekstu artykułu, nie bierzemy w ten sposób pod uwagę np. nazw zakładek na stronie lub tytułów sekcji. Umożliwiliśmy w tym miejscu na specyfikację parametru minimalnej długości, dzięki czemu możliwe jest rozpatrzenie analizy wieloaspektowej, z określeniem różnych progów długości;

- `not isinstance(element, NavigableString)` – elementy widoczne na stronie są klasyfikowane przez BeautifulSoup jako obiekty klasy `NavigableString` (zdarza się zbyt duże generalizacje, co zdeterminowało utworzenie pozostałych warunków), dlatego też odrzucamy wszystkie elementy innych klas;
- `len(element.parent.findChildren("a", recursive=True)) > 25` – warunek sprawdzający, czy w ramach tego samego elementu nadrzędnego nie istnieje więcej niż 25 elementów typu a, co mogłoby sugerować trafienie na np. menu nawigacyjne, które samo w sobie nie jest treścią artykułu do analizy;
- `any(word.upper() in element.upper() for word in words_to_ignore_list)` – początkowe testy wielokrotnie klasyfikowały jako tekst widoczny na stronie treści polityki prywatności, które, jako element uwarunkowany prawnie i obecny na większości stron w podobnej formie, nie są istotne dla samej analizy treści. Z tego powodu stworzyliśmy słownik słów, które są charakterystyczne dla tego typu treści, a elementy zawierające co najmniej jedno ze słów ignorowanych są odrzucane. Plik ze stworzonym słownikiem dostępny jest jako załącznik X.

### Analiza odseparowanej treści

Uzyskana z metody `text_from_html` treść artykułu wykorzystywana jest przez metodę `get_text_related_data`, która odpowiada za uzupełnienie kolumn w bazie danych z informacjami powiązanymi z samym tekstem artykułu, czyli, opisane wcześniej:

- `smart_words_count`;
- `words_to_avoid_count`;
- `exclamation_count`;
- `text_longer_than_50_characters_count`;
- `text_longer_than_100_characters_count`;
- `text_longer_than_150_characters_count`;
- `text_longer_than_200_characters_count`.

Cechy powiązane z długością tekstu odwołują się do wywołań funkcji `text_from_html` z różnymi parametrami, które pozwalają na wyznaczenie długości całkowitej tekstu utworzonego z odseparowanych fragmentów o minimalnej określonej długości. Wartości `smart_words_count` oraz `words_to_avoid_count` odwołują się do plików słownikowych i przy pomocy poniższego fragmentu kodu zliczają słowa w tekście.

```

1 smart_words_list = smart_words_file.read().split(",")
2 words_to_avoid_list = words_to_avoid_file.read().split(",")
3
4 text_concat = text_from_html(soup, 30)

```

```
5
6 smart_words_count = sum(list(word in smart_words_list for word in
7     text_concat.split()))
8 words_to_avoid_count = sum(list(word in words_to_avoid_list for word
9     in text_concat.split()))
scrapping_results_dictionary['smart_words_count'] = smart_words_count
scrapping_results_dictionary['words_to_avoid_count'] =
    words_to_avoid_count
```

**Listing 4.** Fragment kodu odpowiedzialny za określenie wartości powiązanych ze słownikami

Liczba wystąpień wyrażeń *wykrzyknikowych* (`exclamation_count`), czyli tych napisanych wielkimi literami bądź zawierających znaki zapytania i wykrzykniki, pozyskana została z wykorzystaniem wyrażenia regularnego, które wyszukuje w tekście stosownych dopasowań. Wykorzystuje ona pakiet `re`, pozwalający na dokonywanie różnych operacji z wykorzystaniem wyrażeń regularnych.

```
1 scrapping_results_dictionary['exclamation_count'] =
2     len(re.findall(r"(\b[A-Z]{4,}\b|[?!])", text_concat))
```

**Listing 5.** Kod wyznaczający liczbę wyrażeń *wykrzyknikowych*

## Analiza składni kodu HTML

W celu poprawnego policzenia liczby wystąpień poszczególnych tagów HTML w kodzie strony wykorzystaliśmy bibliotekę Beautiful Soup, która udostępnia metodę `find_all`. Wykorzystanie tej metody w pętli iterującej tablicę z nazwami kolejnych tagów pozwoliło na sprawne stworzenie dużej liczby kolumn danych przydatnych do nauki modelu sieci neuronowej.

```
1 for tag in tag_array:
2     soup_count = soup.find_all(tag)
3     column_name = tag + "_count"
4     scrapping_results_dictionary[column_name] = len(soup_count)
```

**Listing 6.** Pętla tworząca kolejne kolumny dla liczby wystąpień tagów HTML w kodzie strony

Obliczając liczbę odwołań do stron wykorzystujących szyfrowany protokół HTTPS oraz nieszyfrowany HTTP musieliśmy stworzyliśmy hybrydowe rozwiązanie, korzystające jednocześnie z pakietu `re` jak i biblioteki Beautiful Soup.

```
1 def get_websites_references_data(scrapping_results_dictionary, soup):
2     scrapping_results_dictionary['http_references'] = len(
3         soup.find_all(href=is_referencing_http))
```

```

4     scrapping_results_dictionary['https_references'] = len(
5         soup.find_all(href=is_referencing_https))
6
7
8 def is_referencing_http(href):
9     return href and re.compile("http://").search(href)
10
11
12 def is_referencing_https(href):
13     return href and re.compile("https://").search(href)

```

**Listing 7.** Metody określające liczbę odwołań do stron korzystających z protokołu HTTP i szyfrowanego HTTPS

### Rola narzędzia Yellow Lab Tools

Wykorzystanie Yellow Lab Tools (dalej YLT) pozwoliło na uwszechstronnie zbioru danych poprzez dywersyfikację źródeł i metodyki pozyskiwania danych. Poprzez wysyłanie żądań do serwera YLT uzyskiwalismy dodatkowe informacje na temat jakości stylów CSS oraz skryptów języka JavaScript obecnych na stronie, co stało się kolejnym elementem istotnym w tworzeniu sieci neuronowej. Żądania wysyłane były przy użyciu pakietu języka Python `requests`. Po pierwszym wysłaniu żądania należało odpytywać serwer YLT o gotowość wyników. W celu ograniczenia obciążenia serwera ograniczyliśmy częstotliwość wysyłania żądań, wstrzymując wątek aplikacji na kilka sekund po każdym z nich.

Pomimo tego, przy testach aplikacji, mając na uwadze liczbę potrzebnych do zgromadzenia danych, nasze urządzenia zostały zablokowane, co uniemożliwiło nam gromadzenie większej liczby danych. W odpowiedzi trafiały do nas kody błędu 429 (*Too many requests* – przekroczona liczba żądań). W związku z tym, zastosowaliśmy pewnego rodzaju obejście problemu, jakim jest wykorzystanie serwerów proxy, które maskują faktyczne pochodzenie żądania, prowadząc zapytanie przez inne serwery HTTP. Skuteczność tego rozwiązania również nie była niestety zadowalająca, co wiązało się z wykorzystaniem darmowych serwerów proxy (np. [Geo Node Free Proxy List<sup>2</sup>](https://geonode.com/free-proxy-list/)), których działanie pozostawia wiele do życzenia.

Kolejnym zrealizowanym usprawnieniem, która miała zażegnać problem blokady w serwisie YLT, było zbudowanie lokalnej instancji narzędzia z wykorzystaniem środowiska *dockerowego*<sup>3</sup>, zainstalowanej na lokalnej maszynie WSL<sup>4</sup>. Niestety, pomimo zakończonego pomyślnie procesu

<sup>2</sup><https://geonode.com/free-proxy-list/>, dostęp na dzień 2022.01.03

<sup>3</sup>Docker service overview (przegląd usług dockerowych): <https://docs.docker.com/get-started/overview/>, dostęp na dzień 2022.01.05

<sup>4</sup>Windows Subsystems for Linux

wdrożenia kontenera z aplikacją na lokalne środowisko i wysyłania do niego zapytań, nie udało się uzyskać wystarczającej efektywności, co wynikało głównie z wydajnością, wykorzystywanego przez YLT pakietu, `phantomas.js`, służącego do generowania kodu strony dla dalszych analiz (narzędzie analogiczne do wykorzystanego przez nas `requests_html`).

Finalnie, po wielu próbach i długotrwałym pozyskiwaniu danych zamiennie z wykorzystaniem opisanych metod, wystosowaliśmy prośbę do autora YLT o wygenerowanie unikalnego klucza API, który pozwolił naszej aplikacji identyfikować się jako zaufana na serwerze, a co za tym idzie, nasze zapytania nie były blokowane, co uprościło dalsze prace.

### 5.2.3 Utworzzone narzędzia pomocnicze

Proces pozyskiwania danych był jednym z większych wyzwań podczas realizacji pracy. Wykorzystanie wielu skomplikowanych pakietów, korzystających z wdrażanych lokalnie rozwiązań, jąder przeglądarki, w duży sposób obniżało stabilność pełnej aplikacji. W związku z tym, musielismy rozważyć i zaimplementować narzędzia usprawniające proces pozyskiwania danych. Adresując najistotniejsze problemy skłoniliśmy się ku stworzeniu:

- skryptu `updateData.py`, który umożliwił testy i poprawę skuteczności działania metod dotyczących analizy tekstu, wykorzystując strony, które zostały już wcześniej przeprocesowane przez mniej stabilne rozwiązania;
- metody `predict_from_mock`, która dla zapytania o analizę wybranej przez użytkownika końcowego strony, przekazywała sieci neuronowej wyniki analizy strony odczytane z pliku `.json` (jeden z uzyskanych wcześniej wyników);
- metody `predict_random`, zwracającej losową ocenę strony, co pozwoliło na prostsze testy i rozwój interfejsu użytkownika (zdecydowanie skrócony czas na odpowiedź backendu).

## 5.3 Interfejs graficzny (frontend)

Tworzenie interfejsu aplikacji z użytkownikiem było elementem niezbędnym do nadania projektowi pełnego kształtu. Przy jego realizacji kierowaliśmy się głównymi zasadami dyktowanymi przez framework Vue, który pozwolił nam na rozwinięcie aplikacji typu SPA (*Single Page Application*), która do komunikacji z serwerem wykorzystuje głównie żądania asynchroniczne. Aplikacja została podzielona na szereg komponentów, z których „korzeniem” był komponent `App.vue`, zawierający menu nawigacyjne i osadzający komponenty właściwe dla wybranych przez użytkownika widoków. W wielu miejscach, dane otrzymywane z serwera musiały być odpowiednio filtrowane, tak, aby nie wyświetlać zbędnych informacji. Adresy URL powinny być odpowiednio sformatowane, a kolory adekwatne do wartości dla danych cech. W celu spełnienia

### *5.3. Interfejs graficzny (frontend)*

---

tych wymagań stworzyliśmy, oprócz samych widoków w HTML, szereg funkcji pomocniczych, które zadeklarowane globalnie mogły być używane w wielu komponentach.



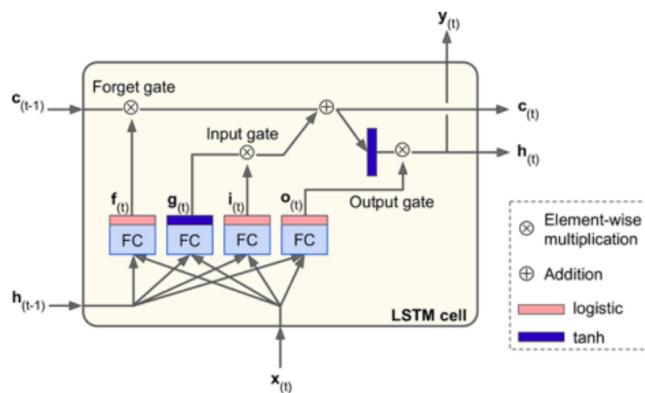
# Rozdział 6

## Analiza danych i sieć neuronowa

### 6.1 Wybrany model sieci neuronowej

Pierwotnie wybraną i docelową architekturą modelu sieci neuronowej platformy jest LSTM – Long short-term memory. Jest ona architekturą rekurencyjnej sieci neuronowej która niweluje problem zanikania gradientów w przypadku długich łańcuchów powiązań. Jest to niezwykle istotne ze względu na złożoność problemu, a także docelowe, długotrwałe uczenie modelu.

Podstawą każdej rekurencyjnej sieci neuronowej są moduły sieci neuronowych przekazujące informacje do przyszłych obliczeń. LSTM w odróżnieniu od podstawowych RNN zawiera w każdym module nie jedną powłokę sieci neuronowej, a cztery – wpływające na siebie nawzajem.



**Rysunek 15.** Architektura LSTM  
 Źródło: [12]

Pierwszą informacją przekazywaną do modułu LSTM jest stan komórki. Moduł przyjmuje stan komórki z poprzedniego modułu i umożliwia dodawanie lub usuwanie informacji z niego. Za obsługę tych operacji są odpowiedzialne specjalne struktury – bramki. W zależności od wartości

otrzymanej przez warstwę ‘sigmoid’ bramka określa jak wiele z danych przechodzących przez nią należy usunąć (bramka zapomnienia) lub dodać (bramka wejściowa).

Dodawanie danych odbywa się poprzez powiązanie danych wejściowych w danej chwili z danymi wynikowymi z chwili poprzedniej. Takie dane są równolegle wysyłane do dwóch warstw – jedna odpowiada za przekształcenie ‘tanh’, zaś druga kontroluje bramkę zapomnienia wyniku pierwszej. Po usunięciu odpowiedniej części danych z warstwy ‘tanh’, dane są dalej dodawane do stanu komórki.

Taki stan komórki jest gotowy do przekazania do modułu kolejnej chwili czasowej i równolegle przekształcany jest przez funkcję tanh i ostatniej bramki zapomnienia, a po usunięciu są przedstawiane jako wynik danej chwili czasowej w sieci neuronowej.

## 6.2 Przetwarzanie wstępne

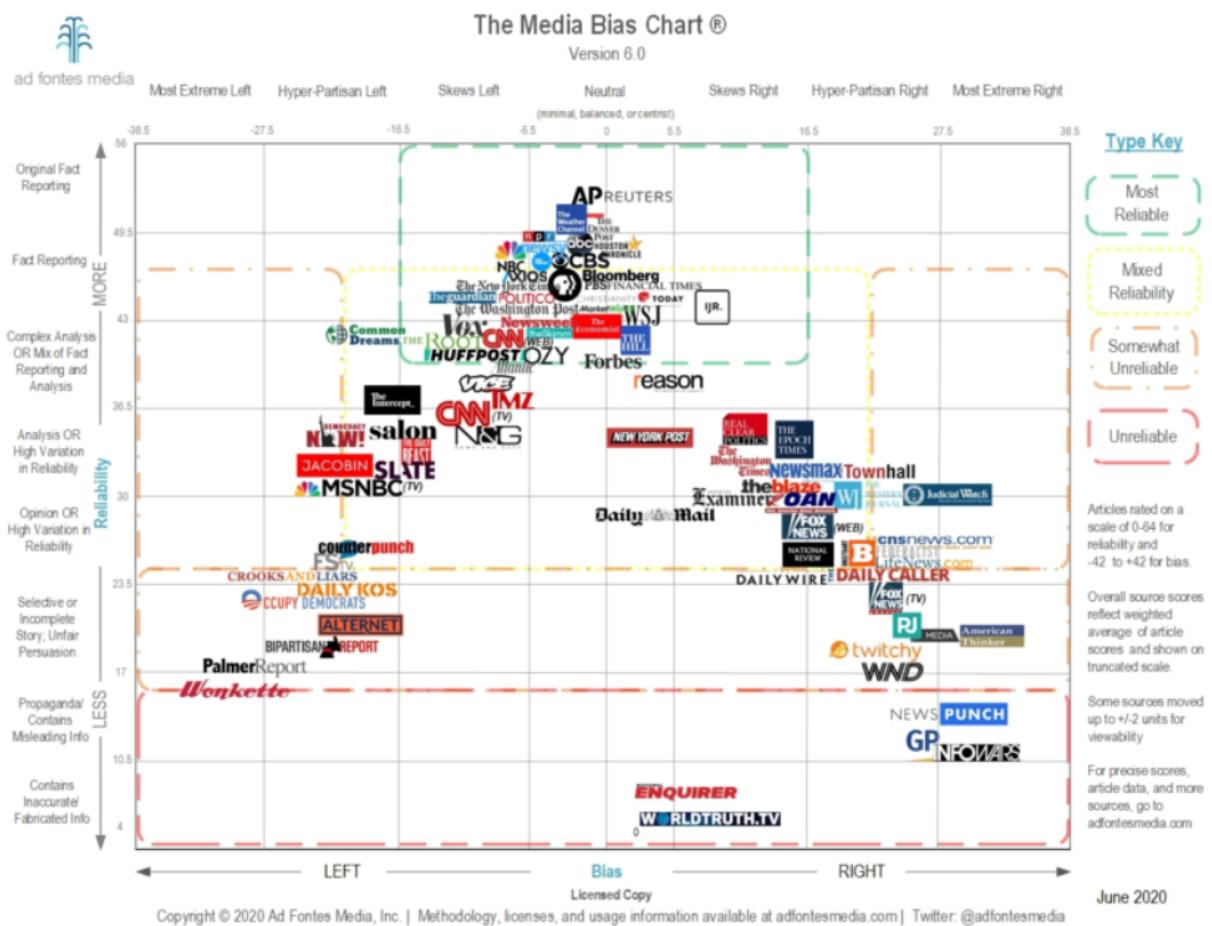
Podczas przetwarzania wstępnie danych uczących zastosowaliśmy dwa podejścia do klasyfikacji - klasyfikacja binarna i klasyfikacja w skali  $<1, 5>$ . W tym celu cała źródłowa baza danych została ręcznie oceniona:

- w pierwszym przypadku:
  - 0 – treść prawdziwa,
  - 1 – treść fałszywa,
- w drugim przypadku:
  - wartości mniejsze od 0 – treść fałszywa,
  - większe od 0 – treść prawdziwa
  - bliskie 3 (o odchyleniu  $\pm 0.7$ ) – treść sporna, trudna w ocenie lub nieznacznie nacechowana stronniczością.

Proces klasyfikacji stron został przeprowadzony w oparciu o dostępne źródła w zakresie badań wiarygodności i stronniczości źródeł i portali informacyjnych. Jednym z nich był *Media Bias Chart* [rysunek 16], diagram opracowany przez organizację użytku publicznego Ad Fontes Media<sup>1</sup>.

---

<sup>1</sup>Ad Fontes Media official web page: <https://adfontesmedia.com/>, dostęp na dzień 2022.01.07

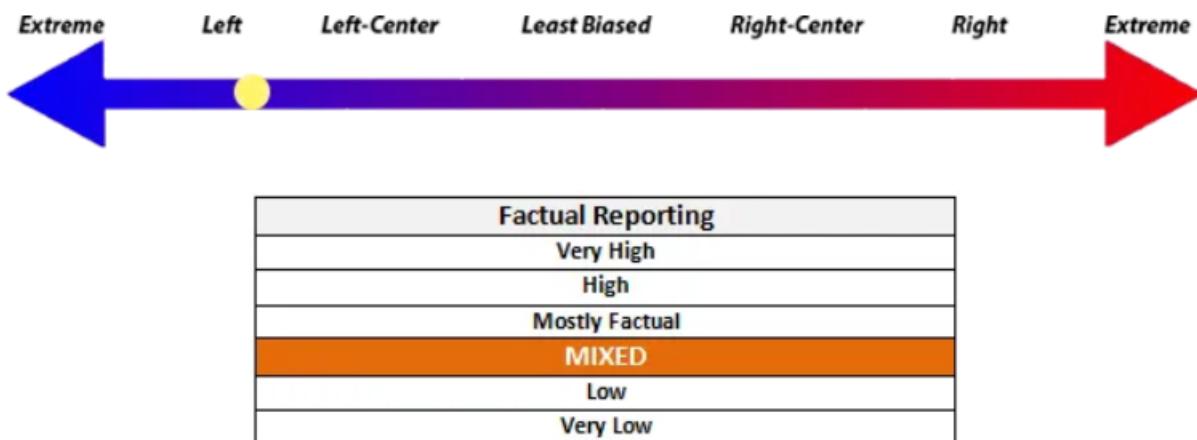


Rysunek 16. Diagram stroniczości popularnych w Stanach Zjednoczonych mediów  
 Źródło: [1]

Innym wykorzystanym narzędziem był serwis Media Bias/Fact Check<sup>2</sup>, który dla podanych przez użytkownika stron (nazw serwisów, a nie poszczególnych artykułów), ocenia ich stroniczość i umieszcza wybraną stronę na skali rozpinającej się pomiędzy wartościami *Far left* (źródło o stroniczości skrajnie lewicowej) oraz *Far right* (źródło o nacechowaniu skrajnie prawicowym). Określa on również jakość informacji przekazywanych przez daną instytucję/serwis, plasując je w kategorycznej skali o tytule *Factual reporting* (Merytoryczność) z wartościami skrajnymi *Very high* (bardzo wysoka) oraz *Very low* (bardzo niska). Wykorzystuje do tego metodologię opierającą się na ręcznych ocenach źródeł przez zespół analityków pod kierownictwem redaktora Dave'a M. Van Zandta. Pomimo faktu, że nie jest to narzędzie stosujące podejście naukowe i przetwarzanie maszynowe, jest ono szeroko cytowane, co

<sup>2</sup><https://mediabiasfactcheck.com/>, dostęp na dzień 2022.01.06

podkreśla w swoim artykule Daniel Funke z Poynter Institute [8], czyli niedochodowej szkoły dziennikarskiej i organizacji badawczej na rzecz wiarygodności informacji w sieci<sup>3</sup>.



Rysunek 17. Ocena wybranego serwisu uzyskana przy pomocy narzędzia Media Bias/Fact Check

Źródło: autorskie (zrzut ekranu ze strony <https://mediabiasfactcheck.com/>)

### 6.3 Analiza eksploracyjna danych

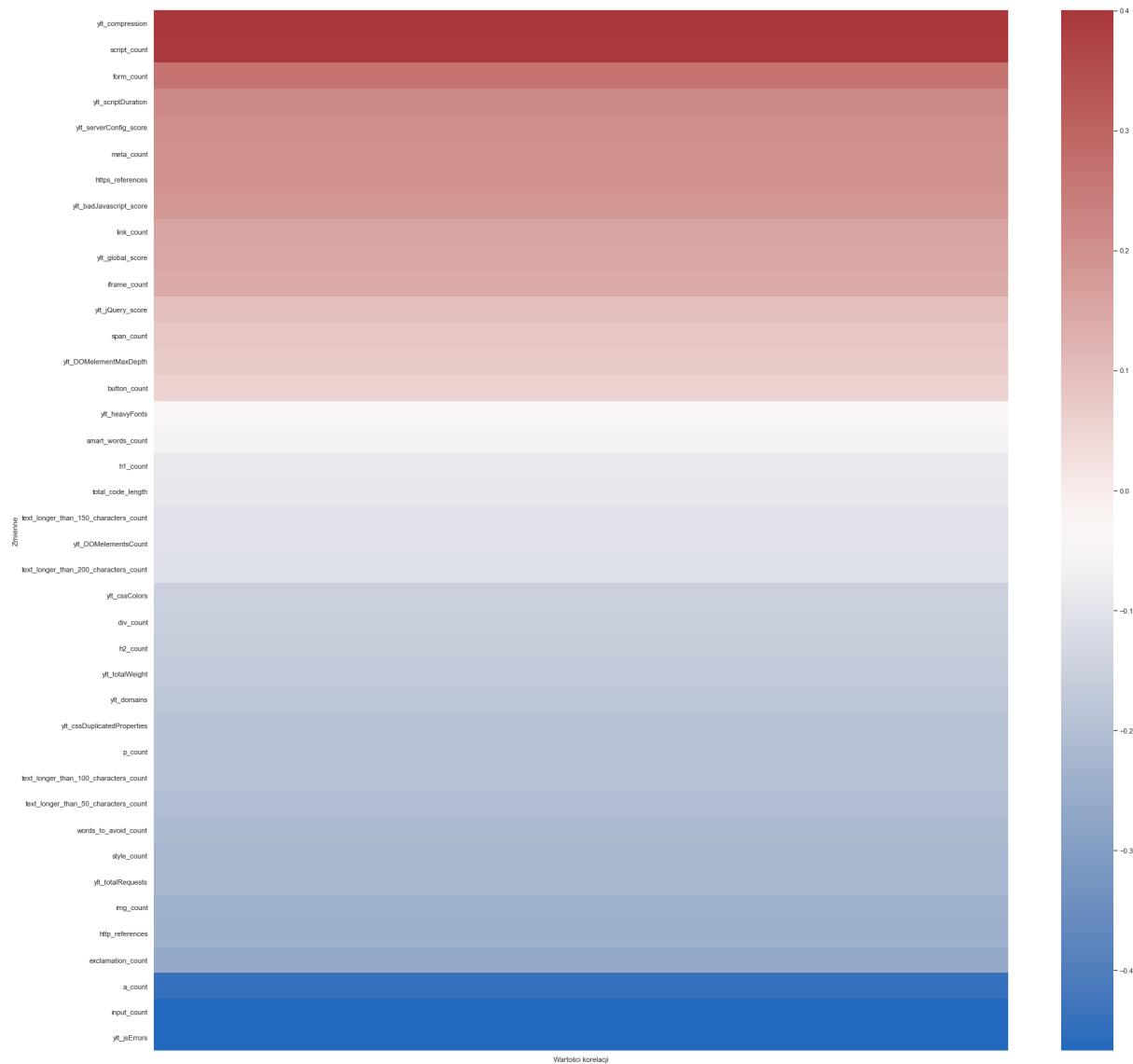
Po oznaczeniu wszystkich rekordów zajęliśmy się analizą eksploracyjną. Ze względu na sposób uzyskiwania danych, zbiór danych jest kompletny, co pozwoliło na rzetelne odczytywanie wskaźników.

W pierwszej kolejności sprawdziliśmy korelację zmiennych z wartościami binarnymi klasyfikacji. Najwyższymi wartościami okazały się zmienne 'ylt\_jsErrors', 'input\_count', 'a\_count', 'ylt\_compression', wartość bezwzględna współczynnika korelacji dla każdej z nich wynosiła powyżej 0.4 - dla pierwszych trzech była to wartość ujemna, dla ostatniej dodatnia.

---

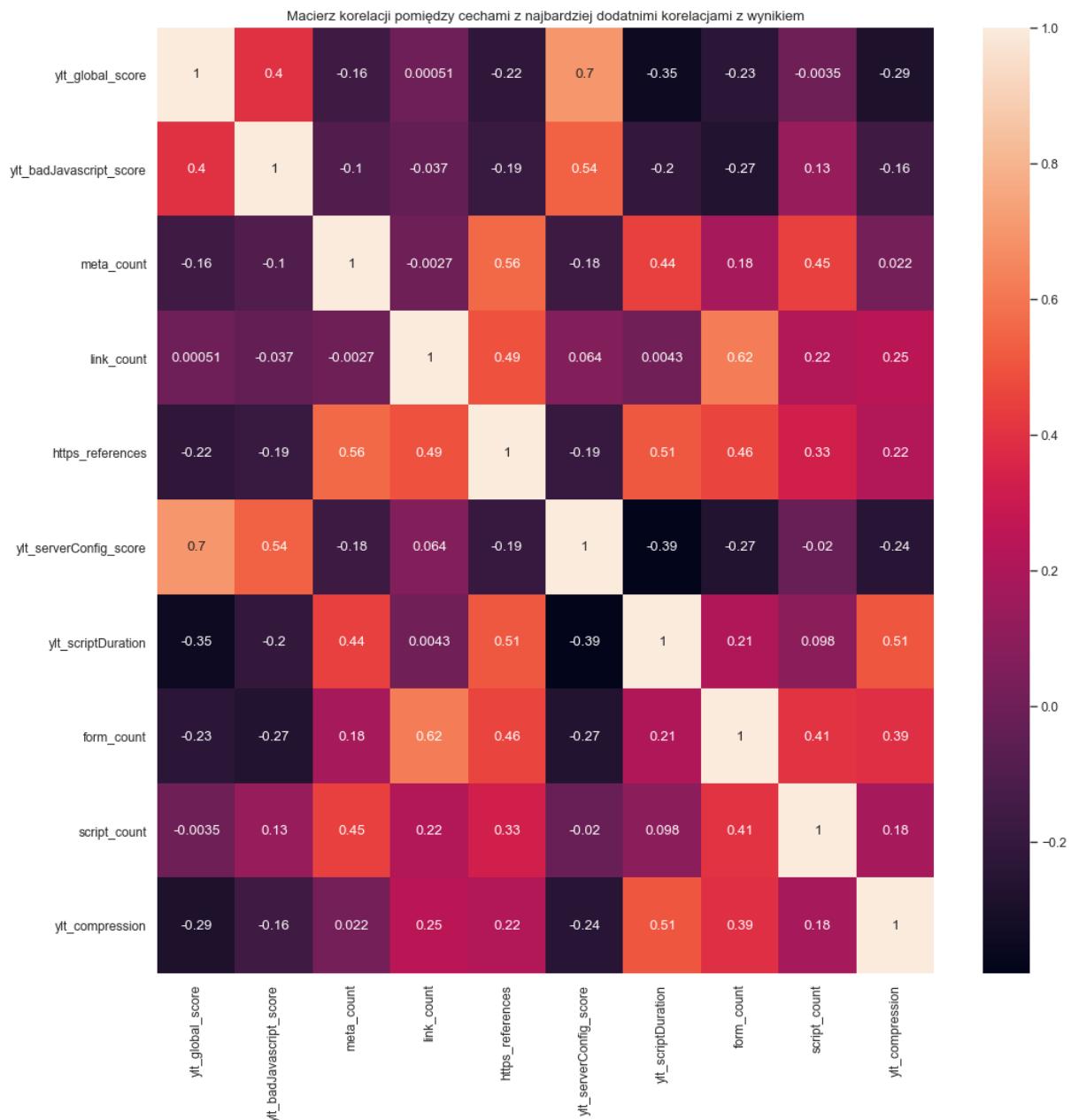
<sup>3</sup>Poynter Institute – About the International Fact-Checking Network: <https://www.poynter.org/ifcn/about-ifcn/>, dostęp na dzień 2022.01.02

### 6.3. Analiza eksploracyjna danych



**Rysunek 18.** Wykres wartości współczynnika korelacji zmiennych z danymi wynikowymi  
Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

W związku z tym, że wśród zmiennych wyróżniają się skrajne dwie grupy o wysokich wartościach bezwzględnych, postanowiliśmy przeprowadzić analizę korelacji między współczynnikami dla 10 współczynników z najwyższą wartością, 10 z najniższą oraz zbiorem 5 z najwyższą i 5 z najniższą korelacją z danymi wynikowymi.



**Rysunek 19.** Macierz korelacji 10 współczynników z najwyższą wartością korelacji z danymi wynikowymi

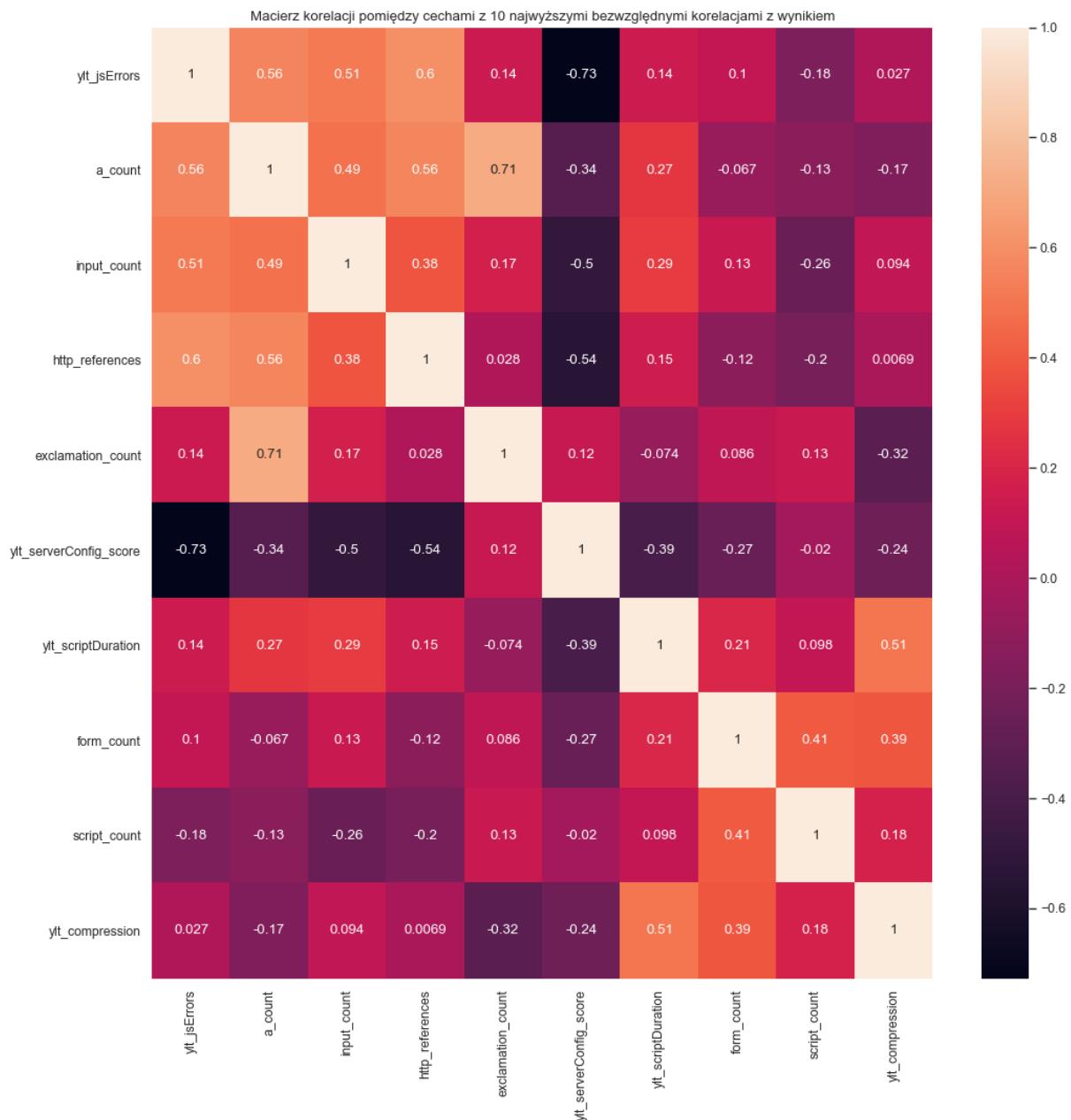
Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

### 6.3. Analiza eksploracyjna danych



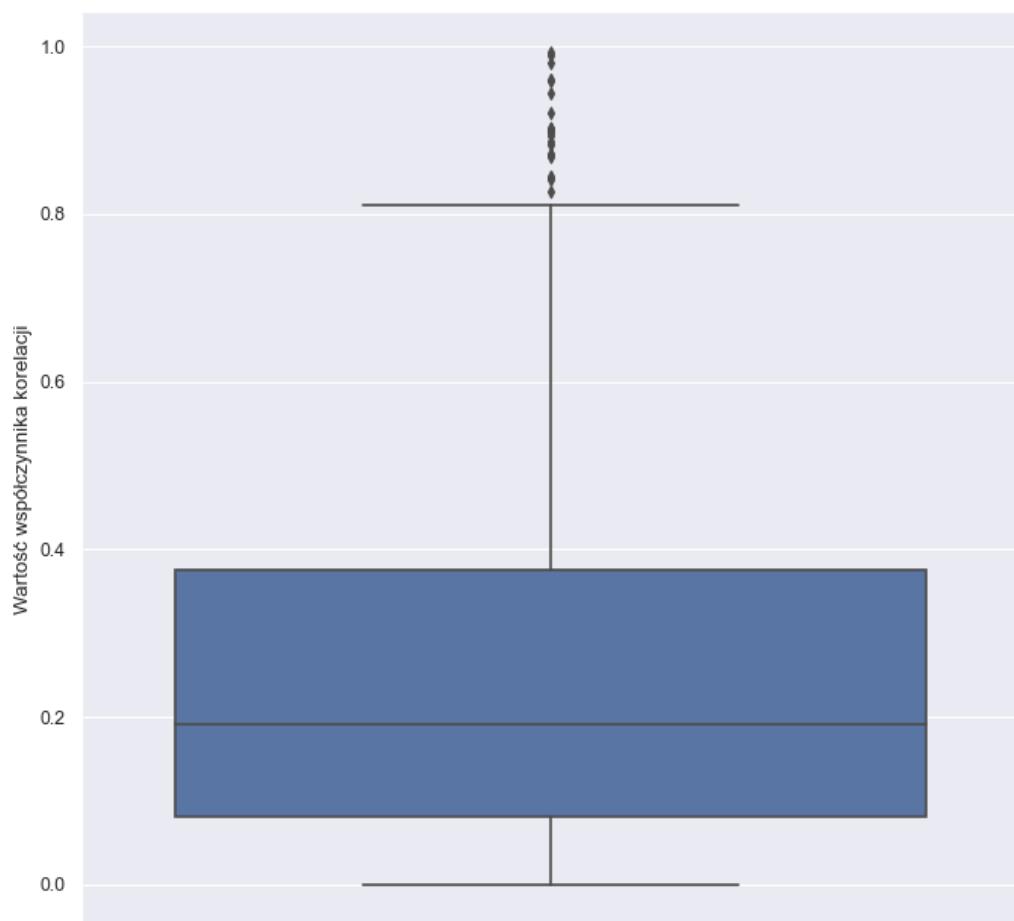
**Rysunek 20.** Macierz korelacji 10 współczynników z najniższą wartością korelacji z danymi wynikowymi

Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)



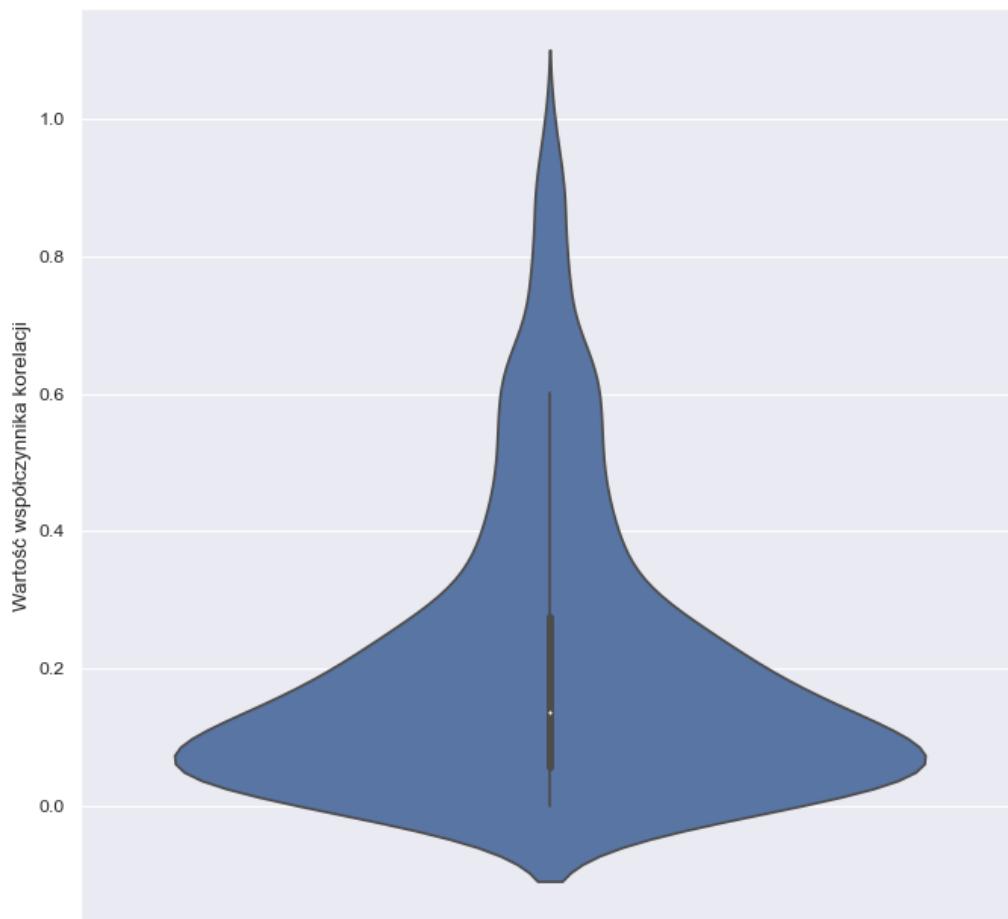
**Rysunek 21.** Macierz korelacji 5 współczynników z najwyższą wartością korelacji z danymi wynikowymi z 5 współczynnikami z najniższą wartością korelacji z danymi wynikowymi  
 Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

Jako, że macierz korelacji dla wszystkich 40 zmiennych jest nieczytelna, korelację między wszystkimi parami zmiennych oceniliśmy jedynie poprzez wyznaczenie mediany i stworzenie wykresu pudełkowego. Można na nim zauważyć, że zdecydowana większość danych nie jest ze sobą zbytnio skorelowana, 75% par wykazuje korelację na poziomie poniżej 0.4.



**Rysunek 22.** Wykres pudełkowy wartości współczynników korelacji dla wszystkich par zmiennych  
 Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

Dla pewności czy nie występują anomalie, sprawdziliśmy również rozkład na podstawie wykresu skrzypcowego, jednak nie przyniosło to dodatkowych informacji o korelacji par parametrów.



**Rysunek 23.** Wykres skrzypcowy wartości współczynników korelacji dla wszystkich par zmiennych  
 Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

Następnie stworzyliśmy tabelę ze średnią, medianą, wartościami minimalnymi, wartościami maksymalnymi oraz skośnością każdej ze zmiennych:

Nazwa zmiennej	Średnia	Medianą	Min	Max	Skośność
exclamation_count	3.481169e+01	17.5	2	298	3.565609
words_to_avoid_count	2.441558e+01	16.0	0	148	2.150967
smart_words_count	9.194805e+00	5.0	0	46	1.530041
h1_count	1.331169e+00	1.0	1	18	6.866937
h2_count	9.000000e+00	6.0	0	224	9.487610
div_count	3.914481e+02	346.0	118	4003	7.938863
iframe_count	5.681818e+00	5.0	0	25	2.027409
a_count	1.979481e+02	155.0	53	1009	2.324963
img_count	6.809091e+01	32.0	1	735	3.967205
p_count	2.027532e+02	114.5	11	3023	6.109725
span_count	2.822727e+01	14.0	0	356	5.440654
button_count	1.070130e+01	8.0	0	95	2.832565
input_count	9.545455e-01	1.0	0	2	0.088296
form_count	6.806494e+01	67.5	23	174	1.369276
script_count	4.822727e+01	36.0	22	146	1.518623
meta_count	3.072727e+01	29.0	4	63	0.060742
link_count	1.026623e+01	6.0	2	61	2.975496
style_count	1.091742e+04	8068.0	1476	69374	3.118460
text_longer_than_50_characters_count	8.376429e+03	5538.0	110	66205	3.384325
text_longer_than_100_characters_count	5.955929e+03	3966.5	0	43093	3.162590
text_longer_than_150_characters_count	3.597032e+03	3207.0	0	30211	3.582711
text_longer_than_200_characters_count	2.694805e+00	1.0	0	57	5.487002
http_references	1.391169e+02	122.0	22	876	3.305771
https_references	5.334364e+05	425252.0	189634	2337248	2.635996
total_code_length	4.095455e+01	29.0	5	262	2.850767
ylt_DOMelementsCount	1.652026e+03	1251.5	362	13635	5.416550
ylt_DOMelementMaxDepth	7.642857e+00	1.0	0	27	0.572655
ylt_scriptDuration	1.679390e+03	1418.0	98	10995	3.777168
ylt_jsErrors	4.870130e-01	0.0	0	3	1.267309
ylt_cssColors	1.623312e+02	145.0	19	455	1.795851
ylt_cssDuplicatedProperties	2.223442e+02	50.5	14	2200	3.364228
ylt_totalWeight	5.949352e+06	2539434.0	982917	91771521	4.693694
ylt_domains	3.753896e+01	28.5	8	114	1.128730
ylt_compression	2.505526e+05	238156.0	43057	836563	0.878303
ylt_totalRequests	1.347013e+02	133.0	33	336	0.792607
ylt_heavyFonts	5.578460e+04	0.0	0	369616	1.969147
ylt_global_score	4.001948e+01	52.0	-378	92	-3.754005
ylt_badJavascript_score	8.094156e+01	96.0	-116	100	-3.245585
ylt_jQuery_score	9.636364e+01	100.0	60	100	-2.730757
ylt_serverConfig_score	5.527273e+01	60.0	-67	95	-1.488574

**Tabela 1.** Wskaźniki dla zmiennych przed przetworzeniem  
Źródło: autorskie (wygenerowane za pomocą biblioteki Pandas)

W tabeli 1 widać dobrze pierwszy problem wykorzystywanego zbioru danych, większość ze zmiennych ma wysokie wartości skośności. W związku z tym dane prawoskośne przetworzyliśmy używając pierwiastka trzeciego stopnia, zaś dane lewoskośne podnieśliśmy do trzeciej potęgi. Zdecydowaliśmy się na użycie nieparzystych wartości, ze względu na obecność w danych wartości ujemnych. Po przetworzeniu tabela wyglądała następująco:

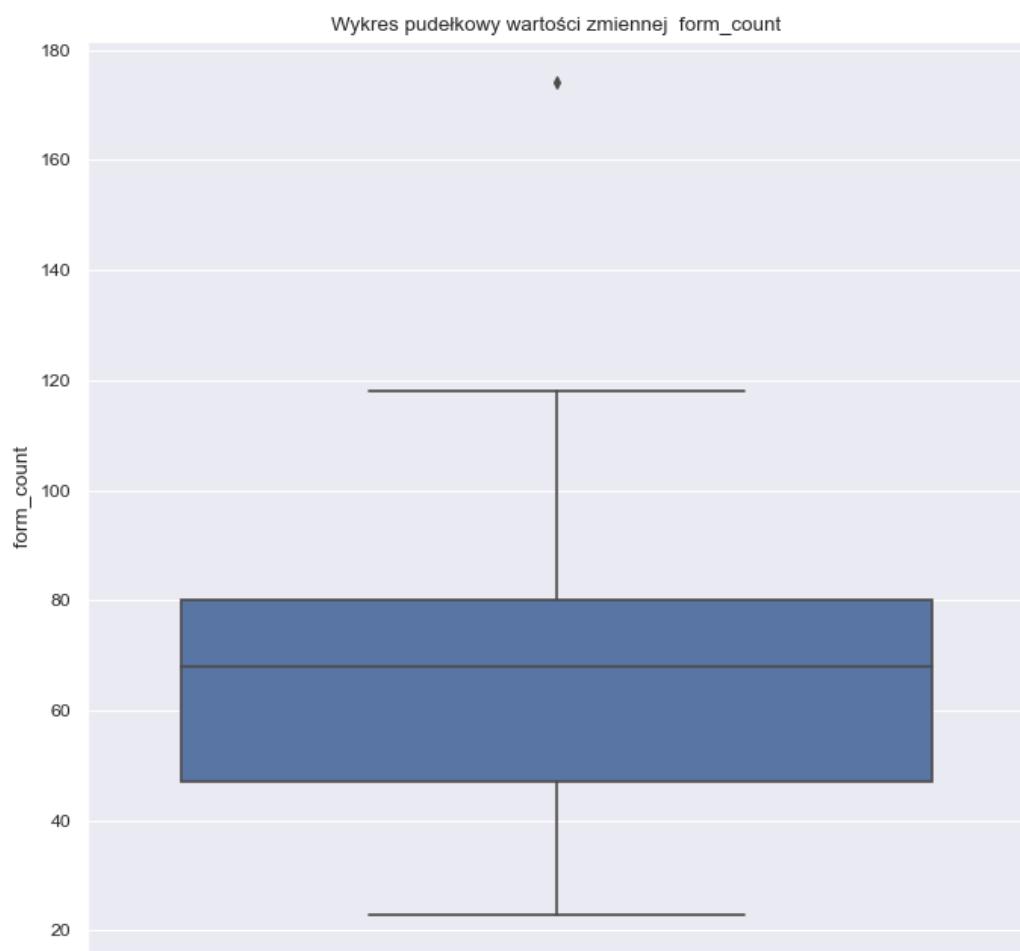
Nazwa zmiennej	Srednia	Medianą	Min	Max	Skośność
exclamation_count	2.817074	2.596011	1.259921e+00	6.679420	1.777260
words_to_avoid_count	2.580951	2.519842	0.000000e+00	5.289572	0.424439
smart_words_count	1.782432	1.709976	0.000000e+00	3.583048	-0.119172
h1_count	1.034873	1.000000	1.000000e+00	2.620741	6.530233
h2_count	1.660128	1.817121	0.000000e+00	6.073178	0.150904
div_count	7.055730	7.020349	4.904868e+00	15.877978	2.198994
iframe_count	1.689160	1.709976	0.000000e+00	2.924018	-1.520238
a_count	5.599355	5.371586	3.756286e+00	10.029910	0.803000
img_count	3.252432	3.174457	1.000000e+00	9.024624	1.794657
p_count	5.223925	4.855629	2.223980e+00	14.459259	1.319166
span_count	2.649483	2.410142	0.000000e+00	7.087341	0.097345
button_count	1.797921	2.000000	0.000000e+00	4.562903	-0.442766
input_count	0.954545	1.000000	0.000000e+00	2.000000	0.088296
form_count	4.056232	4.071602	2.843867e+00	5.582770	0.026365
script_count	3.565609	3.301927	2.802039e+00	5.265637	1.174938
meta_count	30.727273	29.000000	4.000000e+00	63.000000	0.060742
link_count	2.015432	1.817121	1.259921e+00	3.936497	1.140768
style_count	20.274350	20.056507	1.138576e+01	41.089631	1.503741
text_longer_than_50_characters_count	17.835039	17.692243	4.791420e+00	40.454198	1.468474
text_longer_than_100_characters_count	15.992109	15.829571	0.000000e+00	35.059219	1.185237
text_longer_than_150_characters_count	13.818210	14.746668	0.000000e+00	31.145002	-0.231584
text_longer_than_200_characters_count	0.777579	1.000000	0.000000e+00	3.848501	0.878528
http_references	4.976140	4.959639	2.802039e+00	9.568298	0.724619
https_references	78.195336	75.199583	5.745203e+01	132.709373	1.419878
total_code_length	3.177691	3.072317	1.709976e+00	6.398828	0.737472
ylt_DOMelementsCount	11.343928	10.775944	7.126936e+00	23.890123	1.217362
ylt_DOMelementMaxDepth	7.642857	1.000000	0.000000e+00	27.000000	0.572655
ylt_scriptDuration	11.233570	11.234607	4.610436e+00	22.236431	1.088166
ylt_jsErrors	0.366366	0.000000	0.000000e+00	1.442250	0.824729
ylt_cssColors	5.295022	5.253588	2.668402e+00	7.691372	-0.597169
ylt_cssDuplicatedProperties	4.578438	3.696231	2.410142e+00	13.005914	2.449389
ylt_totalWeight	153.374809	136.430708	9.942729e+01	451.061725	2.469119
ylt_domains	3.190554	3.054453	2.000000e+00	4.848808	0.352613
ylt_compression	250552.642857	238156.000000	4.305700e+04	836563.000000	0.878303
ylt_totalRequests	134.701299	133.000000	3.300000e+01	336.000000	0.792607
ylt_heavyFonts	18.742563	0.000000	0.000000e+00	71.765699	0.800590
ylt_global_score	-388773.733766	140608.000000	-5.401015e+07	778688.000000	-10.287242
ylt_badJavascript_score	684668.240260	884736.000000	-1.560896e+06	1000000.000000	-2.118957
ylt_jQuery_score	925142.857143	1000000.000000	2.160000e+05	1000000.000000	-2.664346
ylt_serverConfig_score	316675.272727	216000.000000	-3.007630e+05	857375.000000	0.449220

**Tabela 2.** Wskaźniki dla zmiennych po przetworzeniu

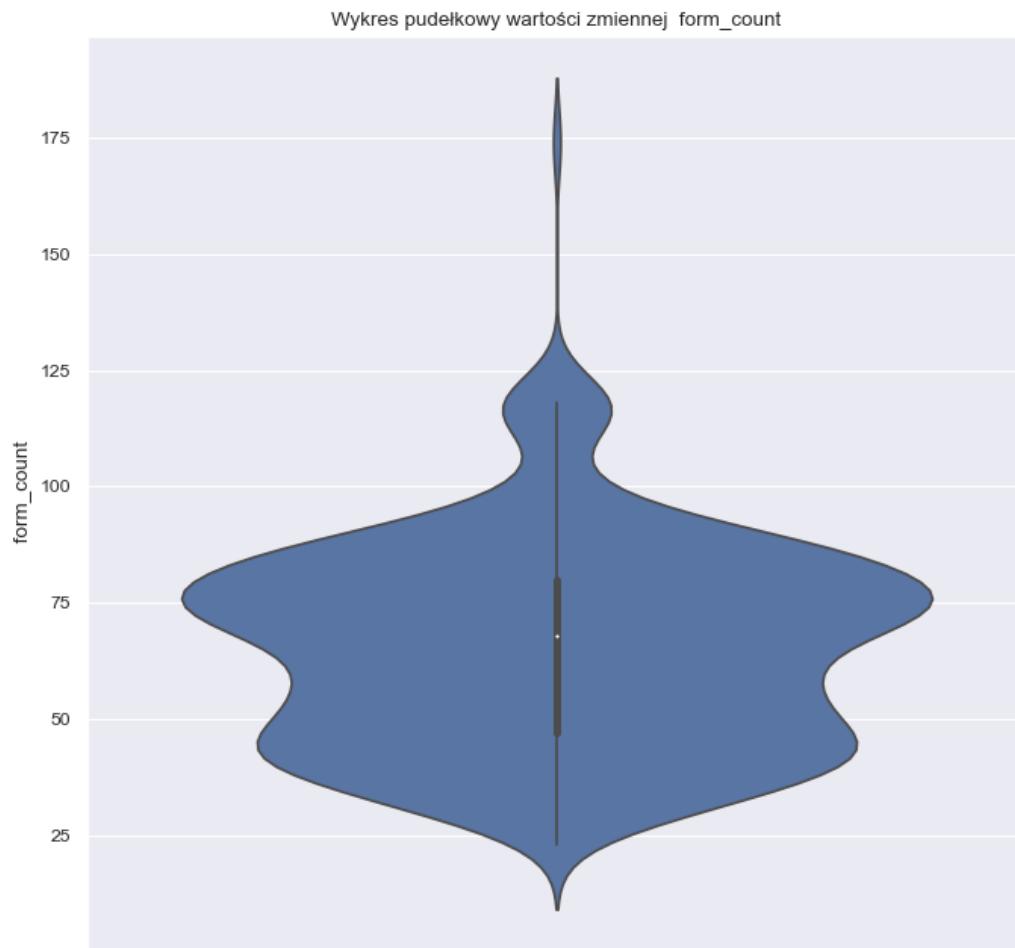
Źródło: autorskie (wygenerowane za pomocą biblioteki Pandas)

### Rozkład danych

Rozkład danych przedstawiliśmy w postaci wykresów skrzypcowych zamiast pudełkowych, ponieważ można z nich od razu zauważać więcej szczegółów, w tym, m.in. wielomodalność danych. Przykładowo dla *form\_count*, czyli liczby interaktywnych formularzy na wykresie skrzypcowym można zauważać, że w wartości mediany jest porównywalnie mniej stron, a więcej wartości gromadzi się w okolicy pierwszego i trzeciego centyla.



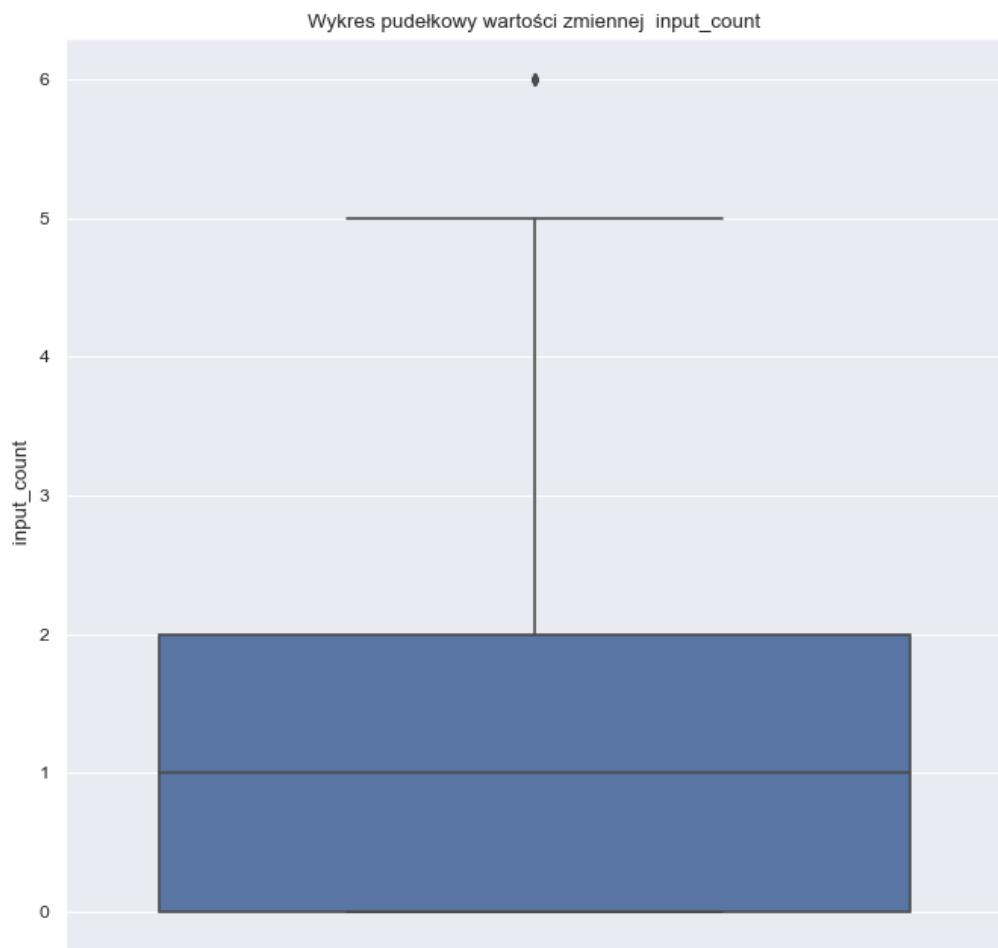
**Rysunek 24.** Wykres pudełkowy wartości parametru *form\_count*  
 Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)



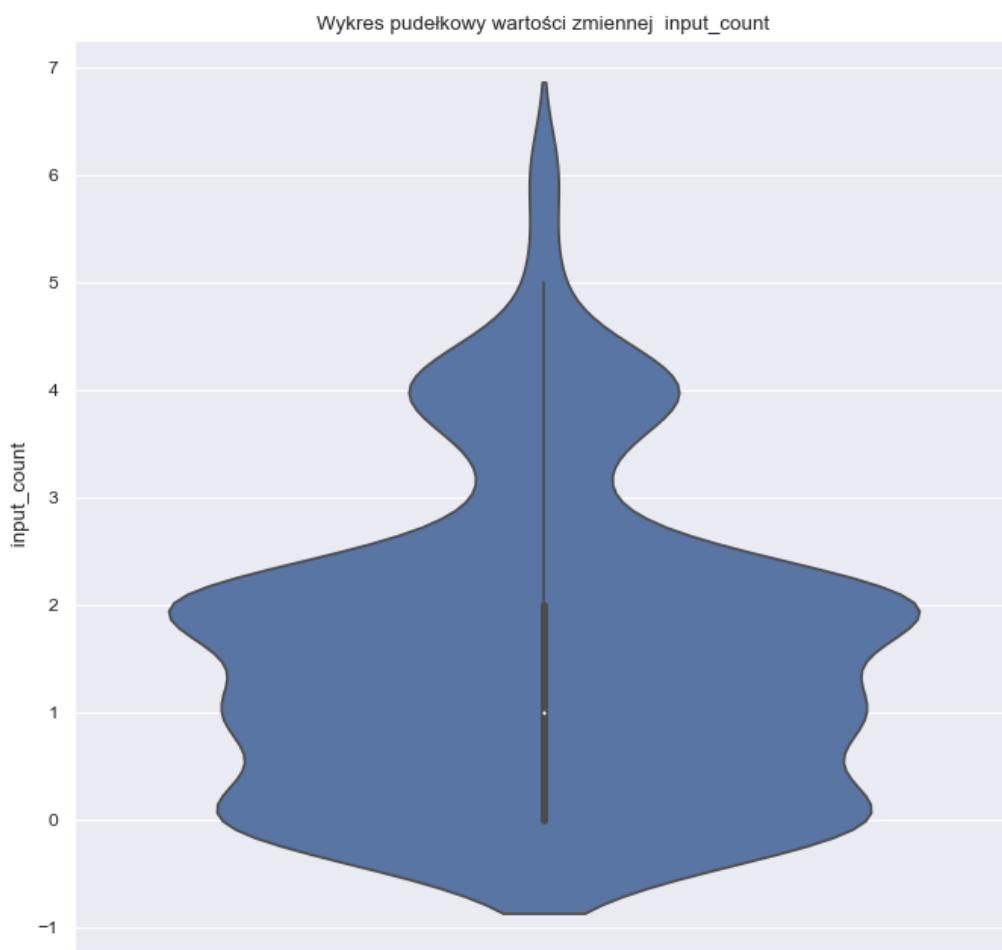
**Rysunek 25.** Wykres skrzypcowy wartości parametru form\_count

Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

Kolejnym ciekawym przypadkiem jest rozkład wartości *input\_count*, gdzie wykorzystanie wykresu pudełkowego zupełnie zaniechałoby większe nagromadzenie wartości w okolicy czwórki.



**Rysunek 26.** Wykres pudełkowy wartości parametru *input\_count*  
 Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)



**Rysunek 27.** Wykres skrzypcowy wartości parametru `input_count`

Źródło: autorskie (wygenerowane za pomocą biblioteki Seaborn)

## 6.4 Proces uczenia sieci

aa

### 6.4.1 Podział danych

Aby mieć pewność, że wyniki nie są jedynie zasługą przeuczenia modelu i wyjątkowo szczęśliwego dopasowania do zbioru testowego, uczenie sieci przeprowadzamy z wykorzystaniem zbioru walidacyjnego. W ten sposób podczas każdej generacji naszej sieci model dąży do optymalizacji zarówno na zbiorze z którego się uczy, ale także na zbiorze walidacyjnym, który przy uczeniu nie jest wykorzystywany.

Podczas uczenia zdecydowaliśmy się na podział zbiorów w stosunku 20:53:27, odpowiednio na zbiór testowy, treningowy i walidacyjny. W ten sposób wiadomo, że dopiero po uzyskaniu wysokiej dokładności predykcji na wszystkich trzech zbiorach można mówić o dobrze wytrenowanym modelu.

### 6.4.2 Metody oceny modeli

W przypadku klasyfikacji binarnej naszym celem była jak największa dokładność znajdowania wiadomości fałszywych, nawet za cenę zwiększonego błędu w postaci źle oflagowanych treści autentycznych. W związku z tym w ocenie modelu dążyliśmy do maksymalizacji wartości predykcyjnej ujemnej – zależności między liczbą prawdziwie ujemnych (poprawnie ocenionych prawdziwych wiadomości), a ogólną liczbą wyników ujemnych. W ten sposób minimalizujemy szanse na oflagowanie szkodliwych treści jako autentyczne.

Dla skali liczbowej ważniejszą kwestią jest dużo większy rozrzut wyników, przez co metody związane z macierzą pomyłek nie sprawdzają się. Dlatego skorzystaliśmy z funkcji sumy kwadratów błędów. Dążąc do minimalizacji wartości tej funkcji wiemy, że nasz model coraz lepiej radził sobie z oceną wiadomości na naszej skali.

### 6.4.3 Modyfikacje w celu poprawy wyników

### 6.4.4 Wybór najefektywniejszych rozwiązań

## 6.5 Przetwarzanie wyników

## 6.6 Wizualizacja wyników



## **Rozdział 7**

### **Uzyskane wyniki**



## **Rozdział 8**

### **Wnioski oraz podsumowanie**



# Bibliografia

- [1] AdFontesMedia, *Interactive Media Bias Chart*, Dostęp na dzień 2022.01.12, sty. 2022. adr.: <https://adfontesmedia.com/interactive-media-bias-chart/>.
- [2] Callahan, M., *How does fake news spread on Facebook?*, [Online], Dostępny w Internecie: [https://news.northeastern.edu/2019/05/14/northeastern-university-researchers-among-the-first-to-gain-access-to-facebook-data-that-could-help-us-understand-how-fake-news-spreads-on-social-media/#\\_ga=2.65547562.520465841.1640168633-2086738170.1640168633](https://news.northeastern.edu/2019/05/14/northeastern-university-researchers-among-the-first-to-gain-access-to-facebook-data-that-could-help-us-understand-how-fake-news-spreads-on-social-media/#_ga=2.65547562.520465841.1640168633-2086738170.1640168633), maj 2019.
- [3] Capitalist, G. A. E. V., *Distribution of traffic sources for fake news in the United States in 2017*, [Online], Dostępny w Internecie: <https://www.statista.com/statistics/672275/fake-news-traffic-source/>, 2017.
- [4] Chan, K. i Fang, W., „Use of the internet and traditional media among young people”, *Young Consumers: Insight and Ideas for Responsible Marketers*, t. 8, s. 244–256, list. 2007. DOI: 10.1108/17473610710838608.
- [5] Chen, B.-C., Wu, Z., Davis, L. S. i Lim, S.-N., *Efficient Object Embedding for Spliced Image Retrieval*, 2021. arXiv: 1905.11903 [cs.CV].
- [6] *Chihuahua or muffin?*, Dostęp na dzień 2021.09.10. adr.: <https://www.freecodecamp.org/news/chihuahua-or-muffin-my-search-for-the-best-computer-vision-api-cbda4d6b425d/>.
- [7] *Cross Frame Scripting Cheat Sheet*, <https://source.checkmarx.com/t/cross-frame-scripting-xfs-cheat-sheet-attack-examples-protection/303>, lut. 2020.
- [8] Daniel Funke, A. M., *Here's what to expect from fact-checking in 2019*, grud. 2018. adr.: <https://www.poynter.org/fact-checking/2018/heres-what-to-expect-from-fact-checking-in-2019/>.
- [9] *Detecting Fakenews in Social Media networks*, <https://www.sciencedirect.com/science/article/pii/S1877050918318210>, paź. 2018.
- [10] *Errors on the world's top 100 websites*, <https://rollbar.com/blog/errors-on-the-worlds-top-100-websites-and-how-to-avoid-them/>, czer. 2018.

## Bibliografia

---

- [11] *Fakenews ironic meme*, Dostęp na dzień 2022.01.10. adr.: <https://memegenerator.net/img/instances/21392935/oh-so-you-read-it-on-the-internet-well-then-i-guess-it-must-be-true.jpg>.
- [12] Géron, A., *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.
- [13] Graves, L. i Amazeen, M. A., *Fact-Checking as Idea and Practice in Journalism*, lut. 2019. DOI: 10.1093/acrefore/9780190228613.013.808. adr.: <https://oxfordre.com/communication/view/10.1093/acrefore/9780190228613.001.0001/acrefore-9780190228613-e-808>.
- [14] Hunt Allcott, M. G., *Social media and fake news in the 2016 election*, [Online], Dostępny w Internecie: [https://www.nber.org/system/files/working\\_papers/w23089/w23089.pdf](https://www.nber.org/system/files/working_papers/w23089/w23089.pdf), dostęp uzyskano 2021-09-11, 2017.
- [15] Lazer, D. i in., *THE STATE OF THE NATION: A 50-STATE COVID-19 SURVEY REPORT #18: COVID-19 FAKE NEWS ON TWITTER*, [Online], Dostępny w Internecie: <https://osf.io/vzb9t/>, paź. 2020.
- [16] Lazer, D. M. J., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S. A., Sunstein, C. R., Thorson, E. A., Watts, D. J. i Zittrain, J. L., „The science of fake news”, *Science*, t. 359, nr. 6380, s. 1094–1096, 2018. DOI: 10.1126/science.aao2998. eprint: <https://www.science.org/doi/pdf/10.1126/science.aao2998>. adr.: <https://www.science.org/doi/abs/10.1126/science.aao2998>.
- [17] Licklider, J. C. R., „Man-Computer Symbiosis”, *IRE Transactions on Human Factors in Electronics*, t. HFE-1, nr. 1, s. 4–11, 1960. DOI: 10.1109/THFE2.1960.4503259.
- [18] Stavroulakis, P. i Stamp, M., *Handbook of Information and Communication Security*, 1st. Springer Publishing Company, Incorporated, 2010, ISBN: 3642041167.
- [19] Wyatt, W. N., „American Carnival: Journalism under Siege in an Age of New Media”, 2007.

# **Wykaz skrótów i symboli**



# Spis rysunków

1	Ironiczne odniesienie do informacji umieszczanych w internecie . . . . .	2
2	Rozprzestrzenianie się fałszywych informacji w sieci . . . . .	4
3	Udział źródeł uznanych sklasyfikowanych jako fake news w tweetach poszczególnych frakcji politycznych związanych z pandemią COVID-19 . . . . .	5
4	Tablica Kanban w usłudze Youtrack . . . . .	7
5	Sytuacja problematyczna dla rozwiązań z zakresu sztucznej inteligencji . . . . .	10
6	Diagram przypadków użycia aplikacji . . . . .	16
7	Widok startowy, pozwalający na analizę wybranej strony . . . . .	18
8	Widok aplikacji pozwalający na przeglądanie danych wykorzystanych do nauki modelu sieci neuronowej . . . . .	20
9	Modal ze szczegółowymi danymi dotyczącymi wybranej strony . . . . .	21
10	Widok statystyk dla poszczególnych cech . . . . .	21
11	Efekt wykorzystania tagów typu heading w artykule zaprezentowany w wynikach wyszukiwania wyszukiwarki Google . . . . .	27
12	Przykład strony realizującej atak z wykorzystaniem XSF . . . . .	28
13	Przykładowe błędy JavaScript w konsoli przeglądarki . . . . .	30
14	Przykład blokady płatnej treści, uniemożliwiający prawidłową analizę treści strony	35
15	Architektura LSTM . . . . .	43
16	Diagram stronniczości popularnych w Stanach Zjednoczonych mediów . . . . .	45
17	Ocena wybranego serwisu uzyskana przy pomocy narzędzia Media Bias/Fact Check . . . . .	46
18	Wykres wartości współczynnika korelacji zmiennych z danymi wynikowymi . . . . .	47
19	Macierz korelacji 10 współczynników z najwyższą wartością korelacji z danymi wynikowymi . . . . .	48
20	Macierz korelacji 10 współczynników z najniższą wartością korelacji z danymi wynikowymi . . . . .	49

21	Macierz korelacji 5 współczynników z najwyższą wartością korelacji z danymi wynikowymi z 5 współczynnikami z najniższą wartością korelacji z danymi wynikowymi . . . . .	50
22	Wykres pudełkowy wartości współczynników korelacji dla wszystkich par zmiennych . . . . .	51
23	Wykres skrzypcowy wartości współczynników korelacji dla wszystkich par zmiennych . . . . .	52
24	Wykres pudełkowy wartości parametru form_count . . . . .	55
25	Wykres skrzypcowy wartości parametru form_count . . . . .	56
26	Wykres pudełkowy wartości parametru input_count . . . . .	57
27	Wykres skrzypcowy wartości parametru input_count . . . . .	58

# **Spis tabel**

1	Wskaźniki dla zmiennych przed przetworzeniem . . . . .	53
2	Wskaźniki dla zmiennych po przetworzeniu . . . . .	54



## **Spis załączników**