AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt dyplomowy

Przewidywanie opinii o produktach z wykorzystaniem technik NLP.
Predicting product reviews using NLP techniques.

Autor: *Dominik Tomalczyk*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr Katarzyna Grobler-Dębska*

Kraków, 2023/2024

Dziękuję kolegom z ds. 14 Kapitol za wiele wartościowych i ambitnych rozmów.

Spis treści

1. Wprowadzenie

Codziennie na całym świecie ludzie dokonują zakupów, tworząc ogromną ilość opinii na temat produktów i usług. Ta praktyka jest nieodłącznym elementem współczesnej konsumpcji. W dzisiejszych czasach, kiedy dostęp do Internetu jest powszechny, większość z nas wyraża swoje zdanie w Internecie, dzieląc się doświadczeniami i refleksjami na temat zakupionych przedmiotów. To właśnie te opinie, w postaci zarówno pozytywnych, jak i negatywnych recenzji, mają ogromny wpływ na decyzje potencjalnych klientów.

Opinie konsumentów pełnią kluczową rolę w procesie podejmowania decyzji zakupowych. Często są one pierwszym źródłem informacji o produkcie lub usłudze, do którego lub której jesteśmy zainteresowani. Opinie pomagają nam zrozumieć, jakie korzyści lub potencjalne ryzyko niesie ze sobą dany produkt. Oczywiście, nie zawsze wszystkie opinie są zgodne, ponieważ są one subiektywne i zależą od indywidualnych oczekiwań i doświadczeń. Niemniej jednak większość z nas skłania się ku opiniom użytkowników, którzy już przetestowali dany produkt lub skorzystali z danej usługi. Oceny i opinie są zazwyczaj wyrażane w formie tekstu połączonego z oceną skalarną w pewnej skali, na przykład w skali od 1 do 5 gwiazdek. Ten system ocen jest wyjątkowo popularny i ułatwia porównywanie różnych produktów lub usług. Dzięki niemu możemy łatwo porównywać, który produkt uzyskał lepsze recenzje.

Popularność tego rozwiązania sprawia, że niektóre aukcje lub oferty, które zdobyły wiele pozytywnych opinii, są sprzedawane znacznie częściej niż te, które posiadają niewielką liczbę lub niekorzystne recenzje. To pokazuje, jak ważna jest reputacja produktu w dzisiejszym rynku.

Mimo że opinie są subiektywne, badacze i firmy wykorzystują narzędzia NLP (Natural Language Processing) [1] w celu analizy ich sentymentu. Przewidywanie, czy opinia jest pozytywna, negatywna czy neutralna, jest jednym z najpopularniejszych zastosowań w dziedzinie uczenia maszynowego. Analiza sentymentu jest ważna nie tylko dla producentów i sprzedawców, ale również dla konsumentów, ponieważ pomaga w dokładniejszym zrozumieniu, jakie opinie są bardziej wiarygodne i rzetelne.

Warto pamiętać, że choć opinie są cenne, nie zawsze są jednoznaczne, i zawsze warto przyjrzeć się im krytycznie. Badania i analiza opinii stanowią ciekawe pole badań, które wciąż rozwija się, aby lepiej zrozumieć, jak wpływają one na nasze wybory zakupowe i jakie czynniki wpływają na to, czy dana opinia jest nacechowana pozytywnie czy negatywnie.

1.1. Motywacja pracy

Główną motywacją pracy jest wzrost popularności przetwarzania języka naturalnego i wszechstronność jego zastosowania. Techniki NLP są zdecydowanie jedną z najszybciej rozwijającą się dziedziną analizy danych. Powstaje bardzo dużo nowych modeli, bibliotek, czy technik w tej dziedzinie. Dodatkowo projekt opiera się o opinie o produktach, jak wiadomo, w internecie kupujemy coraz więcej, zostawiającym tym samym coraz więcej opinii o produktach. Podsumowując zarówno NLP jak i zakupy on-line są ogromną częścią otaczającego nas świata, zatem połączenie tych dwóch dziedzin zdaję się być ciekawym obiektem badań.

1.2. Cel pracy

Celem pracy jest analiza opinii tekstowych i klasyfikacja ich ocen skalarnych za pomocą uczenia maszynowego. Wszystkie wykorzystane opinie pochodzą z jednego z najpopularniejszych polskich portali sprzedażowych – Allegro. Zebrane opinie o produktach pochodzą z dwóch kategorii – sprzęt RTV oraz suplementy.

Projekt zakłada stworzenie zbioru danych od zera następnie oczyszczenie, przetworzenie danych. W późniejszej fazie wyliczane są statystyki i zastosowane modele klasyfikacji uczenia maszynowego.

1.3. Struktura pracy

Zakres niniejszej pracy obejmuje:

– Objaśnienie co jest motywacją pracy i jakie są poszczególne kroki projektu. Wprowadzenie do tematyki NLP. (Rozdział 1.)

– Przegląd gotowych rozwiązań, najpopularniejszych bibliotek do NLP, zwrócenie uwagi na artykuły, które poruszają tematykę NLP, analizy sentymentu lub też klasyfikacji tekstu. Objaśnienie czym w kontekście projektu jest opinia, sentyment i klasyfikacja tekstu. (Rozdział 2.)

– Opis procesu pobierania danych z serwisu Allegro.pl. Konstruowanie bazy danych jako pliki .csv. Opis rozkładu długości opinii dla poszczególnych ocen. (Rozdział 3.)

– Proces całej analizy danych, ekstrakcja cech, wyliczanie różnych statystyk ze zbioru danych. Rozdział przedstawia wszystkie kolejne kroki, jakie zostały podjęte: usuwanie duplikatów, tokenizację, usuwanie niepotrzebnych znaków i słów, lematyzację tokenów, wektoryzowanie opinii. (Rozdział 4.)

– Wybór modeli uczenia maszynowego dla klasyfikacji tekstu, proces dobierania najlepszych parametrów modeli, oraz ich trenowanie. (Rozdział 5.)

– Wybór metryk ocen modeli, objaśnienie ich i definicje. Rezultaty wyrażone w tych metrykach dla każdego poszczególnego modelu. Analiza błędów modeli i porównanie ich z podobnymi pracami. (Rozdział 6.)

– Wnioski z pracy, co z niej wynika, co jest ograniczeniem, na czym warto się skupić w celu usprawnienia jakie płyną korzyści z wykonanej pracy i perspektywa rozwoju. Co warto mieć na uwadze chcąc usprawnić projekt oraz gdzie badanie może okazać się wartościowe. (Rozdział 7.)

2. Przegląd literatury

Zważając na popularność sztucznej inteligencji[2] i technik NLP na temat klasyfikacji tekstu powstało wiele publikacji i artykułów naukowych. W pierwszej części pracy podjęta została próba zebrania i uporządkowania informacji o istniejących bibliotekach, algorytmach i artykułach mających związek z przetwarzaniem opinii.

2.1. Uczenie maszynowe w klasyfikacji tekstu

Uczenie maszynowe stanowi potężne narzędzie w dziedzinie klasyfikacji tekstu [3]. Pozwala ono na automatyczne przyporządkowanie tekstów do określonych kategorii lub etykiet na podstawie analizy treści. Metody uczenia maszynowego w klasyfikacji tekstu opierają się na wykorzystaniu algorytmów i modeli, które uczą się rozpoznawać wzorce i zależności w tekście. To umożliwia skuteczne automatyzowanie procesów, takie jak analiza sentymentu, identyfikacja tematów czy detekcja spamu.

Uczenie maszynowe w klasyfikacji tekstu znajduje zastosowanie w wielu dziedzinach, takich jak przetwarzanie języka naturalnego, analiza danych tekstowych, a także w tworzeniu systemów rekomendacyjnych czy automatycznych systemów zarządzania treścią. Dzięki coraz bardziej zaawansowanym modelom uczenia maszynowego, takim jak sieci neuronowe, klasyfikacja tekstu staje się coraz bardziej precyzyjna i efektywna, co przyczynia się do usprawnienia wielu aspektów działalności opartej na analizie tekstu.

2.2. Definicja opinii i sentymentu

W kontekście przetwarzania języka naturalnego (NLP), opinia i sentyment odnoszą się do analizy subiektywnych aspektów treści tekstowej, zwłaszcza w kontekście wyrażanych emocji, nastawienia i ocen.

Opinia [3] jest to subiektywna ocena lub wyrażenie swojego zdania na jakiś temat, usługę, produkt lub zjawisko przez użytkownika. Opinie mogą obejmować pozytywne lub negatywne

aspekty oraz niuanse oceny w zależności od kontekstu. Analiza opinii ma na celu zrozumienie tego co autor miał na myśli i jego stanowiska wobec danego tematu.

Sentyment odnosi się do emocjonalnego nacechowania wyrażonego w tekście, który może być pozytywny, negatywny lub neutralny. Analiza sentymentu w NLP polega na klasyfikowaniu tekstu pod względem jego emocjonalnego wydźwięku.

2.3. Wykorzystanie NLP w przewidywaniu opinii

W pracy Bo Pang i Lillian Lee [4] skupili się na klasyfikacji opinii z wykorzystaniem skal ocen (np. gwiazdek) w recenzjach. Zaproponowali podejście, które wykorzystuje informacje o relacjach między klasami. Badania obejmowały analizę zbioru danych recenzji filmów, a wyniki wskazywały, że uwzględnienie relacji między klasami może istotnie poprawić dokładność klasyfikacji sentymentu.

W artykule Liu [5] skupia się na różnych aspektach analizy sentymentu, obejmując definicje sentymentu, techniki analizy sentymentu, wyzwania w tej dziedzinie i zastosowania. Przedstawia również koncepcję podmiotowości w tekście, co jest istotne dla zrozumienia kontekstu analizy sentymentu.

W pracy Jayashri Khairnar i Mayura Kinikar [6] opisują kilka podejść do klasyfikacji sentymentu różnymi modelami uczenia maszynowego. Objaśniają ich wady i zalety oraz wskazują jak można je zastosować w NLP.

W artykule Horacio Saggion i Adam Funk [7] opisują zbiór narzędzi, zasobów i eksperymentów przeznaczonych dla klasyfikacji opinii. Prezentują też podejście z wykorzystaniem streszczenia opinii przed główną klasyfikacją.

2.4. Przegląd narzędzi i technik NLP

NLP cieszy się obecnie dużym zainteresowaniem i coraz więcej firm inwestuje środki w rozwój tej dziedziny. Skutkiem tego jest to, że powstało wiele gotowych otwarto-źródłowych bibliotek zapewniających gotowe rozwiązania dla podstawowych zadań przetwarzania języka naturalnego.

2.4.1. NLTK (Natural Language Toolkit)

NLTK[8] to popularna biblioteka w języku Python [9], oferująca szeroki zakres funkcji do analizy tekstu. Zawiera narzędzia do tokenizacji, stemmingu, tagowania częstości występowania słów i wiele innych.

2.4.2. Spacy

Spacy[10] to nowoczesna biblioteka NLP, znana z wydajności i dokładności analizy języka naturalnego. Zapewnia zaawansowane funkcje takie jak rozpoznawanie jednostek nazewniczych i analiza składniowa.

2.4.3. Gensim

Gensim[11] to narzędzie dedykowane głównie do przetwarzania tekstu w kontekście analizy semantycznej. Jest często wykorzystywane do modelowania tematycznego i przetwarzania tekstu w dużych korpusach.

2.4.4. Transformers

Transformers[12] to biblioteka zawierająca gotowe modele tranformatorów które aktualnie cieszą się największym zainteresowaniem. Można znaleźć tam wstępnie wytrenowane duże modele które mogą zostać wykorzystane do rozwiązywania wielu skomplikowanych problemów.

Różnego rodzaju przykłady wykorzystania NLP pojawiają się w wielu różnych źródłach, między innymi na najpopularniejszych platformach do nauki takich jak Youtube, Udemy czy też w wielu popularno naukowych publikacjach np na stronie **towardsdatascience.com** [13]. Bardzo popularną w środowisku Data Science jest też książka **Text Analytics** [3] która w całości poświęcona jest właśnie tematyce NLP. Wiele książek takich jak chociażby np. **Zaawansowane uczenie maszynowe z językiem python** [14] lub **Deep Learning. Praca z językiem Python i biblioteką Keras** [2] poświęcają również rozdziały poświęcone przetwarzaniu tekstu.

3. Metodologia

Całość projektu zarówno pobieranie danych, wizualizacje czy też modelowanie została zaimplementowania w języku Python. Język ten jest aktualnie najpopularniejszym językiem programowania w dziedzinie sztucznej inteligencji ze względu na jego ogromną wszechstronność i stosunkowo prostą składnie. Python oferuje największą ilość dostępnych bibliotek i narzędzi, które znacznie przyspieszają procesy analizy danych i tworzenia modeli predykcyjnych. Ilość dostępnych bibliotek i gotowych rozwiązań pozwala ominąć implementowanie na własną rękę najpopularniejszych rozwiązań.

3.1. Pobieranie opinii

Każdy projekt analizy danych wymaga oczywiście danych. Zazwyczaj im więcej danych tym dla nas lepiej, ponieważ umożliwia to lepsze modelowanie oraz daje więcej możliwości. Prawdą i popularnym powiedzeniem w całej branży jest to, że im lepsze mamy dane tym lepszych wyników możemy się spodziewać. Serwis Allegro z którego pochodzą opinie przechowuje je w formie zawierającej tekst, ocenę (1-5), datę i nazwę użytkownika. Proces zbierania opinii trwał blisko trzy tygodnie i odbywał się na przełomie kwietnia i maja 2023. Z perspektywy problemu klasyfikacji interesować nas będzie wyłącznie tekst i ocena, stąd informacje o dacie i nazwie użytkownika będą pomijane. W momencie tworzenia pracy serwis nie udostępnia żadnego API do pobierania takich informacji. Jednak są one ogólnodostępne stąd zostały zebrane techniką Web scrapingu.

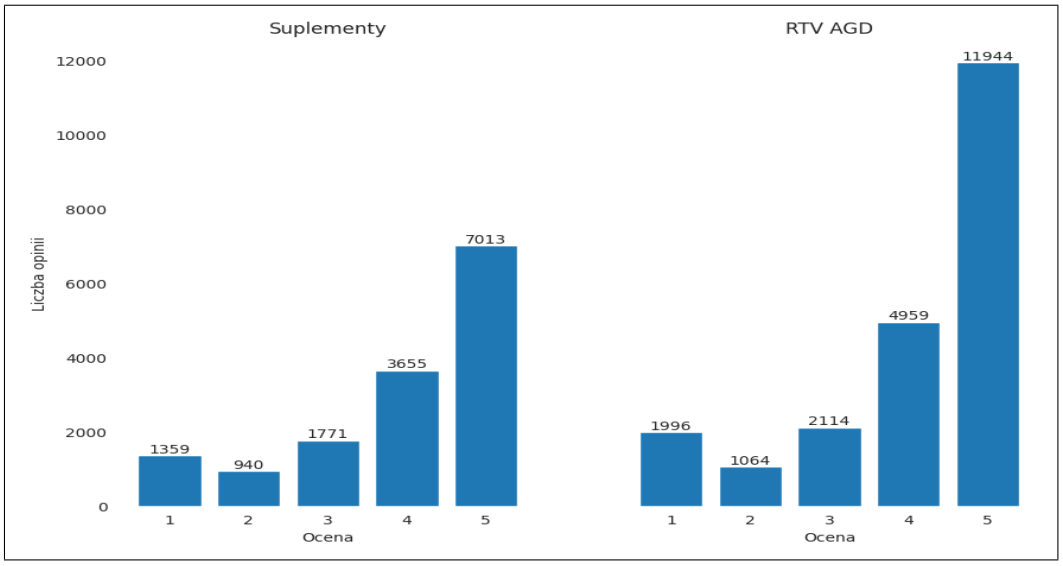
Biblioteka jaka została użyta w tym celu to Selenium [15]. Selenium to otwarte oprogramowanie wykorzystywane do automatyzacji przeglądarek internetowych. Jest popularnie używane do testowania aplikacji internetowych, a także do wykonywania zautomatyzowanych działań w przeglądarkach internetowych.

3.2. Opis zbioru danych

Zarówno pobrane opinie tekstowe i ich oceny jak i linki do aukcji, z których dane zostały pobrane, były przechowywane w plikach CSV. Pliki CSV są popularne w Data Science ze względu na ich prostotę, uniwersalność i niskie zużycie miejsca na dysku, ułatwiając zarówno przechowywanie, jak i udostępnianie danych.

Problemem był fakt, że w każdej aukcji, lwią część wszystkich opinii stanowią opinie bardzo dobre. Może wynikać to z faktu, że chociażby te produkty, które jako pierwsze opinie dostaną opinie negatywne są pozycjonowane nisko. Nisko pozycjonowane oferty mogą być ukrywane lub usuwane bo może to świadczyć o słabej jakości produktu. Jest tutaj pewna logika - produkty które wyświetlają się jako pierwsze na liście wyszukiwania to, te które sumarycznie mają przewagę opinii pozytywnych. Stąd ciężko o oferty, które mają dużo negatywnych ocen, bo takie produkty są omijane przez klientów przez co mają znacznie mniej opinii.

Rozwiązaniem problemu niezbalansowania zbioru danych było zrezygnowanie z zapisu po pewnym czasie opinii z oceną 5, a później też z oceną 4. Po pewnym czasie działania programu do bazy danych były zapisywane tylko oceny 1-3 aby oceny 5 lub 4 zbyt silnie nie przyciągały modelu w tą stronę w klasyfikacji. Finalny stan plików z opiniami wraz z ich ocenami zawiera 22077 rekordów dla opinii z kategorii RTV oraz 14738 dla kategorii suplementów.



Rys. 1. Rozkład ilości opinii dla poszczególnych ocen. Jak widać, główną część obu zbiorów stanowią oceny bardzo dobre. Mimo późniejszego zrezygnowania z zapisu opinii z oceną ’5’ i ’4’ ich przewaga w całym serwisie jest tak znaczna, że mamy tutaj do czynienia z niezbalansowanym zbiorem danych. Przewaga opinii dobrych czy też bardzo dobrych może skutkować przyciąganiem rezultatów modeli w kierunku ocen z wyższą oceną.

4. Preprocessing tekstu i Ekstrakcja cech

Rozpoczynając projekt analizy danych, pierwszym krokiem jest zazwyczaj przegląd dostępnych danych. Analiza danych jest kluczowym etapem, ponieważ pozwala zrozumieć naturę informacji, które będziemy analizować i dostosować odpowiednie narzędzia oraz podejście do problemu. Popularnymi bibliotekami które posiadają wiele wbudowanych już metod do tego typu zadań są biblioteki pandas [16] oraz Numpy [17].

Biblioteka pandas jest niezwykle przydatnym narzędziem do manipulacji danymi w formie tabelarycznej. Pozwala ona na łatwe wczytywanie danych z różnych źródeł, takich jak pliki CSV, Excel, bazy danych itp. Pandas umożliwia również obsługę brakujących danych, co jest częstym problemem w rzeczywistych zbiorach danych. Dodatkowo, biblioteka ta pozwala na wyliczenie podstawowych statystyk, grupowanie danych, a także dostosowywanie formatu danych do potrzeb projektu czy też formatu akceptowalnego przez modele uczenia maszynowego.

NumPy, z kolei, to biblioteka dedykowana do obliczeń numerycznych. Jest niezastąpiona, jeśli mamy do czynienia z macierzami i wektorami. NumPy dostarcza skuteczne i wydajne struktury danych do przechowywania i przetwarzania danych numerycznych, co jest istotne, jeśli pracujemy nad zadaniami związanymi z uczeniem maszynowym. W połączeniu z bibliotekami takimi jak pandas, NumPy pozwala na efektywną obróbkę danych i przygotowanie ich do modelowania.

4.1. Duplikaty i najpopularniejsze opinie

W fazie wstępnej obróbki z danych wszystkie opinie zostały zrzutowane na małe litery. Zasadniczo wielkość liter nie powinna wpływać na sens opinii stąd taki krok. Jako kolejny krok zostały usunięte duplikaty ze zbioru danych. Po wstępnej analizie i zliczeniu tekstowych opinii względem tekstu okazało się, że zdecydowanie najpopularniejszą opinią jest - 'ok'. Zaraz za nią drugim najpopularniejszym wpisem jest - 'polecam'.

Ciekawą statystyką wynikająca z powyższych danych jest to, że w przypadku RTV opinia 'ok' to około 11% wszystkich rekordów natomiast dla kategorii suplementy jest to około 8%.

Tabela 1. Liczba wystąpień najpopularniejszych opinii

Opinia	RTV	Suplementy
ok	2339	1179
polecam	1284	647
super	467	195
wszystko ok	203	110

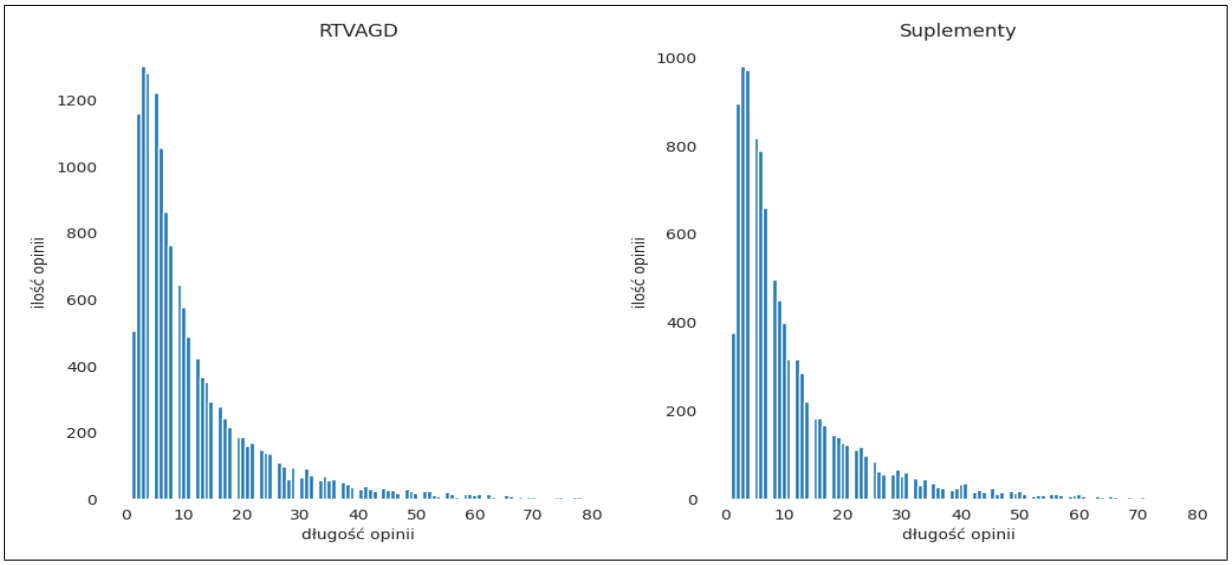
4.2. Rozkład długości opinii

Wartościowymi statystykami są te dotyczące długości opinii, wykorzystując podstawowe funkcje obiektu DataFrame z pandas wyliczamy takie wartości jak: długość maksymalna i mninimalna, mediana, średnia arytmetyczna oraz odchylenie standardowe. Tym co rzuca się w oczy

Tabela 2. Statystyki długości opinii

Statystyka	RTV	Suplementy
długość min.	1	1
długość max.	555	283
mediana	8	7
średnia	12.54	11.66
odchylenie standardowe	16.19	13.69

jest fakt że opinie z kategorii RTV są zasadniczo dłuższe niż te o suplementach. Wynika to zapewne z faktu, że sprzęty użytkowe są bardziej złożone i można o nich powiedzieć więcej w odróżnieniu do suplementów. Są to też bardziej złożone opinie o czym świadczy ich większy rozrzut (wyższa wartość odchylenia standardowego). Jest to spodziewana analogia, artykuły których używamy dłużej i mają np. kilka funkcji będą miały dłuższe i bardziej zróżnicowane opinie. Mimo to jak widać mediany wynoszą 7-8 słów, co jest tak na prawdę długością jednego zdania. Prawdziwy rozkład lepiej oddają mediany, które nie są tak podatne na wartości odstające jak średnia arytmetyczna.



Rys. 2. Rozkład długości opinii dla poszczególnych kategorii. Histogramy dla obu kategorii mają bardzo zbliżony kształt. W obu przypadkach największa część zbioru danych to opinie nie dłuższe niż 20 słów. Najliczniejsze długości wśród opinii to wartości 7 i 8 co było zaprezentowane w poprzednich tabelach. Wskazuje to na to, że najczęściej to po prostu jedno zdanie. Jest to ciekawe zjawisko, można zauważyć, że użytkownicy znacznie częściej decydują się na opinie, które mają jedynie jedno lub dwa zdania. Wpisy dużo dłuższe to dosłownie pojedyncze przypadki w całym zbiorze.

4.3. Tokenizacja

Tokenizacja tekstu, jak opisano w **Text Analytics with Python** [3], to fundamentalny etap w przetwarzaniu języka naturalnego, który pełni kluczową rolę w konwersji surowego ciągu znaków na bardziej zorganizowaną i zrozumiałą strukturę. Stanowi ona nieodłączny krok w budowie każdego modelu sztucznej inteligencji, który obejmuje szereg istotnych etapów.

Proces tokenizacji jest niezbędny, aby uczynić tekst dostępnym dla analizy i zrozumienia przez maszyny. Polega na podziale tekstu na jednostki nazywane tokenami, z których każdy jest podstawowym elementem analizy języka naturalnego. Forma tych tokenów może być różnorodna, obejmująca zarówno pełne słowa, jak i ich fragmenty, a nawet znaki interpunkcyjne, w zależności od kontekstu i celu analizy.

W kontekście naszego badania, tokenizację stosujemy na poziomie całych słów. To oznacza, że każde słowo w tekście staje się oddzielnym tokenem, co umożliwia bardziej precyzyjną analizę i zrozumienie struktury językowej.

4.4. Usuwanie niepotrzebnych znaków

Przed samym tokenizowaniem z danych zostały usunięte niepotrzebne znaki, które mają znikomy wkład do sentymentu opinii. Mowa tu o znakach interpunkcyjnych jak i o znakach które nie są literami np. emotikony. Znaki interpunkcyjne jak i emotikony występują w każdych opiniach, stąd pominięcie ich nie powinno skutkować zatraceniem informacji. Dzięki usunięciu tych znaków modele językowe i algorytmy analizy tekstu mogą skoncentrować się na samych słowach, co ułatwia zrozumienie kontekstu. Usunięcie tych znaków sprzyja również redukcji szumu. Pomijając niektóre znaki ułatwiamy również tokenizacje oraz tym samym zaoszczędzamy zasoby pamięci.

4.5. Usuwanie stop words

Poza znakami, które nie wnoszą żadnych wartościowych (z punktu naszego problemu) informacji mamy również całe słowa, które występują często w każdym zdaniu języka naturalnego. Termin ten z angielskiego nosi nazwę 'stop words' [18]. Są to słowa, które nie są nacechowane ani pozytywnie ani negatywnie, występują bardzo często w różnych kontekstach co za tym idzie nie będą wartościowe dla naszego modelu klasyfikacji.

Popularne biblioteki NLP w Pythonie zawierają gotowe listy takich słów dla niektórych języków. Podczas tworzenia pracy biblioteka NLTK wykorzystywana do tokenizacji nie zawiera jeszcze takiej listy dla języka polskiego. Na Githubie można znaleźć kilka gotowych repozytoriów z listami polskich 'stop words'. Lista wykorzystana w projekcie [19] nie spełniła jednak swojego zadania. Modele po zastosowaniu usuwania 'stop words' dawały gorsze rezultaty niż w przypadku bez usuwania. Może to wynikać złym dopasowaniem słów do problemu. Gotowa lista może zawierać słowa, które akurat w przypadku opinii w internecie zawierają wartościowe informacje i opinie bez nich tracą sens. Tutaj warto rozważyć utworzenie własnej lub zmodyfikowanie listy słów dla tego specyficznego problemu.

4.6. Stemming i lematyzacja

Operacje stemmingu i lematyzacji [3] to kolejne kroki, które mają za zadanie ujednolicenie różnych form danego słowa. W praktyce sprowadza się to do zamiany różnych odmian (przez czas lub osobę) jednego bazowego słowa. Dwa narzędzia dają podobne rezultaty jednak mają pewne różnice. Popularniejszy stemming jest techniką, która obcina ze słowa część, która ma wskazywać czas i osobę, zostaje nam jedynie pierwsza część bazowego słowa. Natomiast lematyzacja działa na zasadzie mapy, co oznacza w praktyce tyle, że rzutuje różne formy słowa na

jego podstawową formę. Nie mamy w tym przypadku obcinania części słowa a jedynie zwrócenie jego bezosobowej formy.

W momencie tworzenia pracy nie był dostępny żadnych polski stemmer. Jest jednak dostępne narzędzie do lematyzacji w języku polskim - Morfeusz2 [20]. Narzędzie to jest dostępne jako gotowa biblioteka Pythona, stąd można z niej korzystać zamiast bezpośrednio wysyłać requesty na dostępne API.

Poniżej znajduje się fragment kodu Python, który ma za zadanie przeprowadzić tokenizacje, a następnie lematyzacje każdej opinii:

Listing 4.1. Kod wykorzystujący Morfeusza2

```
from nltk.tokenize import word_tokenize
from morfeusz2 import Morfeusz

# Inicjalizacja Morfeusza
morfeusz = Morfeusz()


def morfeusz2_lemmatization(phrase, morfeusz):
    stemmed_phrase = []
    tokens = word_tokenize(phrase) # tokenizacja opinii
    for token in tokens: # każdy token jest lematyzowany
        analysis = morfeusz.analyse(token)
        # wybieranie bezokoliczników
        stemmed_phrase.append(analysis[0][2][1].split(':')[0])
    return "".join(stemmed_phrase)
```

4.7. Bag of Words

Technika ta odnosi się do zamiany słów (a w zasadzie już tokenów) na wektory. W przypadku Bag of Words [3] mamy na myśli 'wrzucenie słów do worka'. W tym podejściu całkowicie pomijamy kolejność słów w zdaniu, a interesuje nas jedynie ich liczba wystąpień. Można użyć jedynie samego zliczania słów np. poprzez wbudowaną klasę CountVectorizer w sklearn [21] lub też jego rozszerzonej wersji, która uwzględnia częstość występowania słów - TfidfVectorizer. Obie te klasy zamienią nam listy tokenów na wektory gotowe do przekazania na wejście modeli uczenia maszynowego jednak w obu przypadkach tracimy informacje o pozycji słów w zdaniach.

4.8. TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF (Term Frequency-Inverse Document Frequency) [22] to metoda używana w przetwarzaniu języka naturalnego oraz analizie tekstu, mająca na celu ocenę istotności danego słowa w kontekście danego dokumentu względem całego zbioru dokumentów. Ta technika jest szeroko stosowana w informatyce, wyszukiwaniu informacji, systemach rekomendacyjnych, analizie tekstu i wielu innych dziedzinach.

– Term Frequency (TF) - Odnosi się do liczby wystąpień danego słowa w danym dokumencie. Bardzo intuicyjnie można powiedzieć, że im częściej dane słowo występuje w dokumencie, tym ważniejsze jest dla tego konkretnego dokumentu.

– Inverse Document Frequency (IDF) - To miara rzadkości danego słowa w zbiorze dokumentów. IDF przypisuje większą wagę słowom, które występują rzadko w całym zbiorze dokumentów. Słowa, które są powszechne we wszystkich dokumentach, otrzymują niższą wagę.

Wzór na TF-IDF można przedstawić jako iloczyn TF i IDF:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

gdzie:

– t to słowo,

– d to dokument,

– D to zbiór wszystkich dokumentów.

Terminy TF i IDF mają swoje własne wzory:

$$\text{TF}(t, d) = \frac{\text{Liczba wystąpień słowa } t \text{ w dokumencie } d}{\text{Liczba wszystkich słów w dokumencie } d}$$

$$\text{IDF}(t, D) = \log\left(\frac{\text{Liczba wszystkich dokumentów w zbiorze } D}{1 + \text{Liczba dokumentów zawierających słowo } t}\right)$$

Wartości TF i IDF są używane do obliczenia ostatecznej wartości TF-IDF dla danego słowa w danym dokumencie. W praktyce, dla danego słowa w danym dokumencie, im częściej występuje (TF), a jednocześnie rzadziej występuje we wszystkich dokumentach (IDF), tym wyższa wartość TF-IDF. Wartości te pozwalają przekształcić tekst na wektory liczbowe, które mogą być używane do analizy tekstu, wyszukiwania informacji czy klasyfikacji dokumentów.

TF-IDF jest wykorzystywane w różnych dziedzinach, od systemów wyszukiwawczych, przez rekomendacje treści, po analizę sentymentu. Dzięki tej technice możliwe jest efektywne wyodrębnianie i reprezentowanie ważnych informacji z tekstów.

W projekcie jako narzędzie realizujące technikę TF-IDF a tym samym wektoryzację opinii została wykorzystana gotowa klasa biblioteki sklearn - TfidfVectorizer.

5. Modele przewidywania opinii

Klasyfikacja to jeden z rodzajów zadań w uczeniu maszynowym, w którym celem jest przypisanie danego obiektu do jednej z predefiniowanych kategorii lub klas na podstawie analizy cech tego obiektu. W skrócie, klasyfikacja polega na naukowym klasyfikowaniu obiektów na podstawie ich właściwości. Przewidywanie oceny dla opinii w tym przypadku to właśnie zadanie klasyfikacji. Jako dane wejściowe model powinien otrzymywać wektor, i zwracać wartość dyskretną. Na wejście trafia zwektoryzowana część tekstowa opinii a na wyjściu oczekujemy oceny ze skali 1-5.

5.1. Modele

Wszystkie modele wykorzystane w projekcie to popularne ogólnodostępne modele klasyfikacji. Zostały wykorzystane gotowe klasy biblioteki sklearn. Modele w tym pakiecie zawierają ujednolicone API, które pozwala na szybkie inicjalizowanie, trenowanie, predykcje i ocenę wydajności za pomocą kilku linijek kodu.

5.1.1. SVM Klasyfikator

Support Vector Machine [23], czyli Maszyna Wektorów Nośnych, to algorytm uczenia maszynowego, który znajduje zastosowanie w zadaniach klasyfikacji i regresji. Jego główną ideą jest znalezienie hiperpłaszczyzny, która najlepiej dzieli przestrzeń cech, przy jednoczesnym maksymalizowaniu odległości między punktami należącymi do różnych klas.

SVM jest oparty na koncepcji znalezienia optymalnej hiperpłaszczyzny decyzyjnej, która najlepiej separuje różne klasy. Kluczowym pojęciem jest pojęcie wektorów nośnych (support vectors) - to punkty, które leżą najbliżej hiperpłaszczyzny i mają wpływ na jej umiejscowienie. W przypadku klasyfikacji binarnej, decyzja dla nowej obserwacji x może być wyrażona jako:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b\right)$$

gdzie:

N to liczba wektorów nośnych,

α_i to współczynniki Lagrange'a,

y_i to etykieta obserwacji,

$K(x, x_i)$ to funkcja jądra (kernel function).

Funkcja jądra odpowiada za przekształcanie danych do przestrzeni o wyższej wymiarowości, co umożliwia znalezienie hiperpłaszczyzny decyzyjnej w bardziej skomplikowanych strukturach danych. Popularne funkcje jądra to np. jądro liniowe, wielomianowe i radialne (RBF).

SVM to nie tylko algorytm klasyfikacyjny, ale również teoretycznie i praktycznie zaawansowane narzędzie, które świetnie sprawdza się w przypadku zarówno liniowych, jak i nieliniowych problemów klasyfikacyjnych.

5.1.2. Random Forest

Random Forest [3], znany również jako Las Losowy, stanowi potężne narzędzie w dziedzinie uczenia maszynowego, zyskujące popularność dzięki swojej zdolności do skutecznej klasyfikacji i regresji. Ten algorytm bazuje na koncepcji zespołowego uczenia, gdzie kilka modeli jest łączonych w celu uzyskania bardziej stabilnej i precyzyjnej predykcji.

Random Forest składa się z wielu drzew decyzyjnych, a każde z tych drzew jest trenowane na innym podzbiorze danych. Proces trenowania obejmuje wybór losowego podzbioru cech podczas każdego podziału węzła, co sprawia, że drzewa są różnorodne. Różnorodność ta jest kluczowa, ponieważ pomaga unikać nadmiernego dopasowania (overfitting [24]) i przyczynia się do poprawy ogólnej zdolności do generalizacji modelu.

W matematyce, można przedstawić głosowanie dla klasyfikacji binarnej jako:

$$H(x) = \text{sign}\left(\sum_{i=1}^{N} h_i(x)\right)$$

gdzie $h_i(x)$ to wynik i-tego drzewa. W praktyce, finalna decyzja jest wynikiem głosowania drzew, a klasa przewidziana jest przez większość głosów.

Random Forest wykazuje wysoką odporność na overfitting dzięki różnorodności drzew decyzyjnych. Każde drzewo jest trenowane na innym podzbiorze danych, co oznacza, że błędy popełnione przez jedno drzewo są kompensowane przez inne. To pozwala na skuteczną redukcję wariancji modelu.

Random Forest umożliwia również ocenę ważności cech. Poprzez analizę, jak bardzo każda cecha przyczynia się do poprawy dokładności modelu, możemy identyfikować kluczowe czynniki decydujące o predykcji.

Random Forest to kompleksowe narzędzie, które oferuje zbalansowane podejście między dokładnością, stabilnością a zdolnością do wykrywania ważnych cech w danych.

5.1.3. Naiwny Klasyfikator Bayesa

Naiwny klasyfikator Bayesa (ang. Naive Bayes Classifier) [3] to probabilistyczny algorytm klasyfikacyjny, bazujący na twierdzeniu Bayesa. Jego działanie opiera się na estymacji prawdopodobieństw warunkowych dla każdej cechy w każdej klasie. Kluczową ideą jest wykorzystanie twierdzenia Bayesa do określania prawdopodobieństwa przynależności do danej klasy, mając dostęp do zbioru cech.

Model ten zakłada naiwnie, że cechy są wzajemnie niezależne, co jest uproszczeniem, ale umożliwia łatwe obliczenia. W praktyce, chociaż założenie to może być mocne, naiwność algorytmu często nie wpływa znacząco na jego skuteczność, szczególnie w przypadku dużych zbiorów danych.

Działanie naiwnego klasyfikatora Bayesa można opisać matematycznie przy użyciu twierdzenia Bayesa. Niech C oznacza klasę, a cechy obiektu to: $F_1, F_2, \ldots, F_n$

Zgodnie z twierdzeniem Bayesa, prawdopodobieństwo przynależności obiektu do danej klasy, mając cechy, można zapisać jako:

$$P(C|F_1, F_2, \ldots, F_n) \propto P(C) \cdot P(F_1|C) \cdot P(F_2|C) \cdot \ldots \cdot P(F_n|C)$$

gdzie:

- $P(C)$ to prawdopodobieństwo a priori danej klasy,

- $P(F_i|C)$ to prawdopodobieństwo warunkowe wystąpienia cechy F_i w danej klasie.

W fazie klasyfikacji, dla danego obiektu, algorytm oblicza te prawdopodobieństwa dla każdej klasy i przypisuje obiekt do klasy o najwyższym prawdopodobieństwie.

Podsumowując, naiwny klasyfikator Bayesa jest oparty na probabilistycznym podejściu, wykorzystując twierdzenie Bayesa do estymacji prawdopodobieństw przynależności do klas na podstawie cech. Mimo założenia o niezależności cech, jest to efektywny i prosty algorytm, często stosowany w zadaniach klasyfikacji tekstu, filtracji spamu i wielu innych obszarach.

5.1.4. AdaBoost Klasyfikator

AdaBoost, znany także jako Adaptive Boosting [25], to zaawansowany algorytm uczenia maszynowego, który znacząco zwiększa skuteczność klasyfikacji poprzez wykorzystanie zespołu słabych klasyfikatorów. Koncept ten wprowadzony został przez Yoava Freunda'a i Roberta Schapire'a w 1996 roku [26], a od tego czasu stał się kluczowym elementem w dziedzinie machine learning. Główną ideą AdaBoost jest skupianie się na trudnych do nauczenia przypadkach poprzez adaptacyjne dostosowywanie wag obserwacji.

AdaBoost to jeden z algorytmów boostingu, co oznacza, że buduje on zestaw słabych klasyfikatorów w sposób sekwencyjny. Klasyfikatory te są następnie ważone, przy czym większa

waga przydzielana jest błędnie sklasyfikowanym obserwacjom. To skupia uwagę na trudnych przypadkach, co poprawia zdolność modelu do generalizacji.

Matematycznie, wagi dla obserwacji oraz proces adaptacji można przedstawić za pomocą wzorów. Dla przykładu, wagi w_t dla klasyfikatora w iteracji t mogą być zaktualizowane zgodnie z:

$$w_t(i) = w_{t-1}(i) \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))$$

gdzie y_i to etykieta obserwacji, $h_t(x_i)$ to wynik klasyfikatora w iteracji t dla obserwacji i, a α_t to współczynnik wagowy dla klasyfikatora w iteracji t.

W fazie klasyfikacji, każdy klasyfikator bazowy zwraca etykietę, a ostateczna decyzja jest podejmowana na podstawie ich kombinacji, z uwzględnieniem przypisanych wag. Ostateczna etykieta $H(x)$ dla obserwacji x może być zdefiniowana jako:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\right)$$

AdaBoost znajduje zastosowanie w wielu dziedzinach, takich jak klasyfikacja obrazów, detekcja obiektów czy rozpoznawanie tekstu. Jego zdolność do adaptacji do błędów i skupianie się na trudnych przypadkach czyni go silnym narzędziem w praktyce.

AdaBoost, choć skomplikowany w swojej teorii, cieszy się popularnością ze względu na swoją efektywność i zdolność do poprawy skuteczności słabych klasyfikatorów. Algorytm ten jest jednym z pionierów w dziedzinie zespołowego uczenia maszynowego.

5.2. Dobór parametrów modeli

Podczas inicjalizowania obiektu klasy klasyfikatora praktycznie każdy model ma dostępne pewne parametry. Jest to zależne ściśle od konkretnego modelu, jednak zazwyczaj w każdym przypadku możemy zmodyfikować je na potrzeby naszego konkretnego zbioru danych. Różne kombinacje parametrów dają różne rezultaty. Tutaj nasuwa się pomysł przetestowania wielu takich kombinacji. Taka dość intuicyjna technika nosi nazwę grid search [27]. Jest to dokładnie przetestowanie wielu różnych kombinacji i zwrócenie parametrów dla których model miał najlepsza wydajność. Przed klasyfikacją każdym modelem została wykorzystana technika grid search w celu ustalenia najlepszych wartości parametrów.

6. Ewaluacja modeli

Metryki ewaluacyjne to miary które pozwalają nam ocenić wydajność wytrenowanego modelu. Dzięki uzyskaniu metryki skalarnej możemy porównać ze sobą kilka modeli i wybrać ten o największej lub najmniejszej. Dobrym przykładem jest technika grid search omawiana w rozdziale 5 która również korzysta z porównywania metryk oceny modelu aby wybierać parametry zwracające najlepsze wartości. Ponadto metryki oceny modelu umożliwiają nam niejako interpretowalność wydajności modelu.

6.1. Wybór metryk ewaluacyjnych

Każdy problem uczenia maszynowego wymaga specyficznej ewaluacji. W przypadku problemu regresji możemy porównywać różnice między zwracanym wynikiem a wartością prawdziwą. Inaczej jednak wygląda sytuacja w problemie klasyfikacji. W tym przypadku możemy jedynie rozróżnić dwa scenariusze: wynik zgodny z prawdziwą etykietą lub wynik różny od etykiety. Do oceny każdego modelu zostały zastosowane 3 metryki.

6.1.1. Dokładność

W kontekście uczenia maszynowego, metryka dokładności [28] to miara oceniająca skuteczność modelu poprzez iloraz poprawnie sklasyfikowanych przypadków do ogólnej liczby przypadków. Dokładność jest wyrażana jako procentowy stosunek prawidłowo sklasyfikowanych obserwacji do całkowitej liczby obserwacji w zbiorze danych. Wartości dokładności zawierają się w przedziale od 0 do 1.

Wzór na Acc (dokładność) można przedstawić jako:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

gdzie:

- TP to liczba prawidłowo sklasyfikowanych przypadków (True Positives),
- TN to liczba prawidłowo odrzuconych przypadków (True Negatives),

– FP to liczba przypadków błędnie sklasyfikowanych jako pozytywne (False Positives),

– FN to liczba przypadków błędnie sklasyfikowanych jako negatywne (False Negatives).

6.1.2. F1

Metryka F1 [29] jest miarą, która łączy precyzję (precision) i czułość (recall) w celu oceny wydajności klasyfikatora. Jest szczególnie użyteczna w sytuacjach, gdzie zbalansowanie między prawdziwie pozytywnymi (True Positives) a prawdziwie negatywnymi (True Negatives) jest kluczowe. Metryka F1 jest definiowana jako średnia harmoniczna precyzji i czułości, a jej wartość mieści się w przedziale od 0 do 1, gdzie wartość 1 oznacza doskonałą wydajność klasyfikatora. Użycie metryki F1 jest szczególnie korzystne w przypadkach, gdzie zarówno błędy typu I (False Positives) jak i błędy typu II (False Negatives) mają istotne konsekwencje, a jednocześnie istnieje potrzeba uwzględnienia równowagi między tymi dwoma rodzajami błędów.

Metryka F1 jest często stosowana w przypadku niezbalansowanych zbiorów danych w zadaniach klasyfikacji. Niezbalansowany zbiór danych oznacza, że liczba przypadków różnych klas znacząco się różni, co może prowadzić do wyzwań związanych z oceną wydajności modelu. Właśnie z takim zbiorem danych mamy do czynienia, ponieważ ilość opinii z oceną 5 jest większa niż pozostałych opinii.

Wzór na $F1$ można przedstawić jako:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

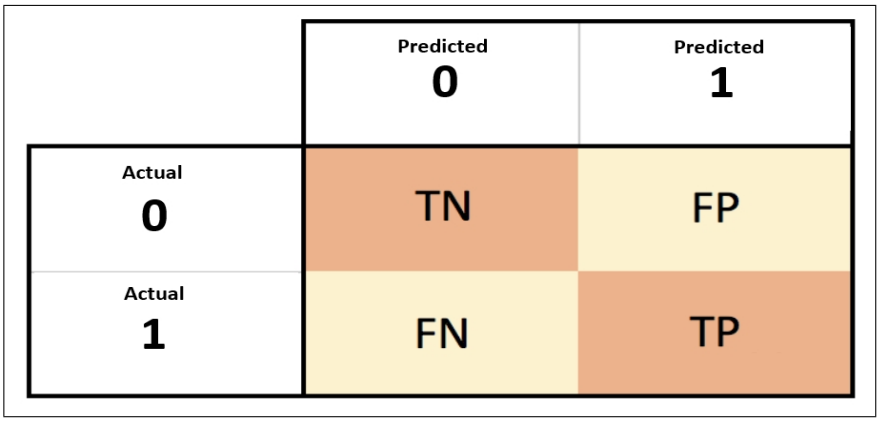
gdzie:

$$\text{P} = \frac{TP}{TP + FP} \quad \text{(Precyzja)}$$

$$\text{R} = \frac{TP}{TP + FN} \quad \text{(Czułość)}$$

6.1.3. Macierz pomyłek

Macierz pomyłek (ang. confusion matrix) [30] to narzędzie używane w uczeniu maszynowym do oceny wydajności klasyfikatora. Jest to tabela przedstawiająca liczbę poprawnych i błędnych klasyfikacji dokonanych przez model, porównując je z rzeczywistymi klasami. Tabela ta pozwala nam zobaczyć ilość źle przewidzianych przypadków dla każdej pojedynczej klasy. Z wykorzystaniem tej tabeli bez problemu możemy zaobserwować czy nasz model ma skłonność do 'przyciągania' wyników do jednej klasy lub które klasy są klasyfikowane jako które.



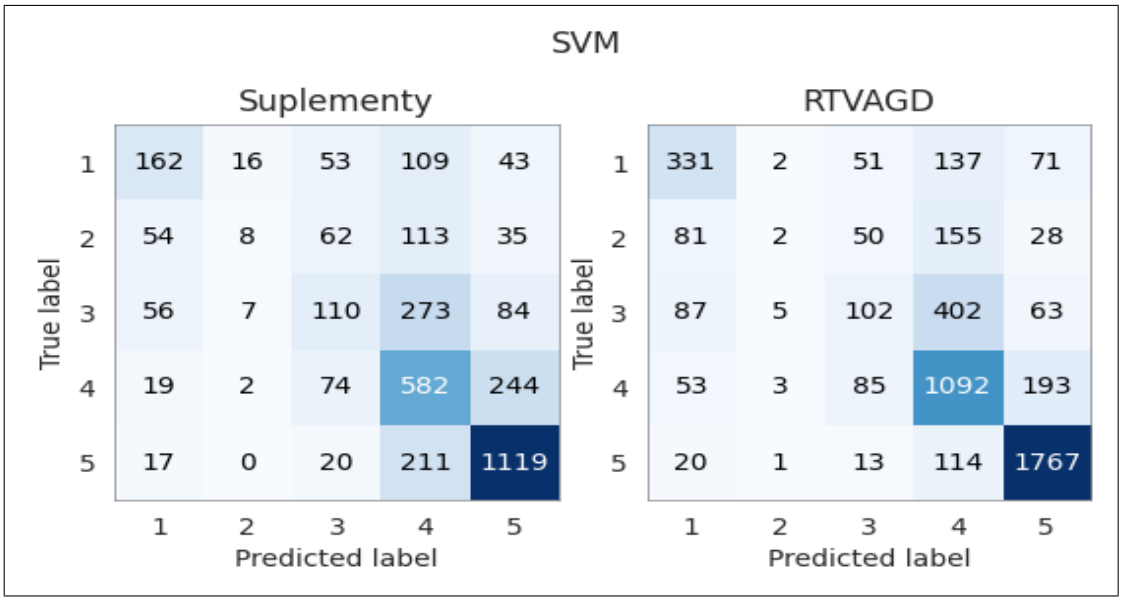
Rys. 3. Przykładowa macierz pomyłek

6.2. Wyniki eksperymentów

W celu zobrazowania wyników eksperymentów zostały wykorzystane wymienione w poprzednim podrozdziale 3 miary: dokładność, miara F1, macierz pomyłek. Wszystkie te metryki zostały wyliczone dla każdego modelu. Zostały wykorzystane gotowe rozwiązania do obliczeń z biblioteki sklearn oraz narzędzia do wizualizacji z pakietu Matplotlib [31].

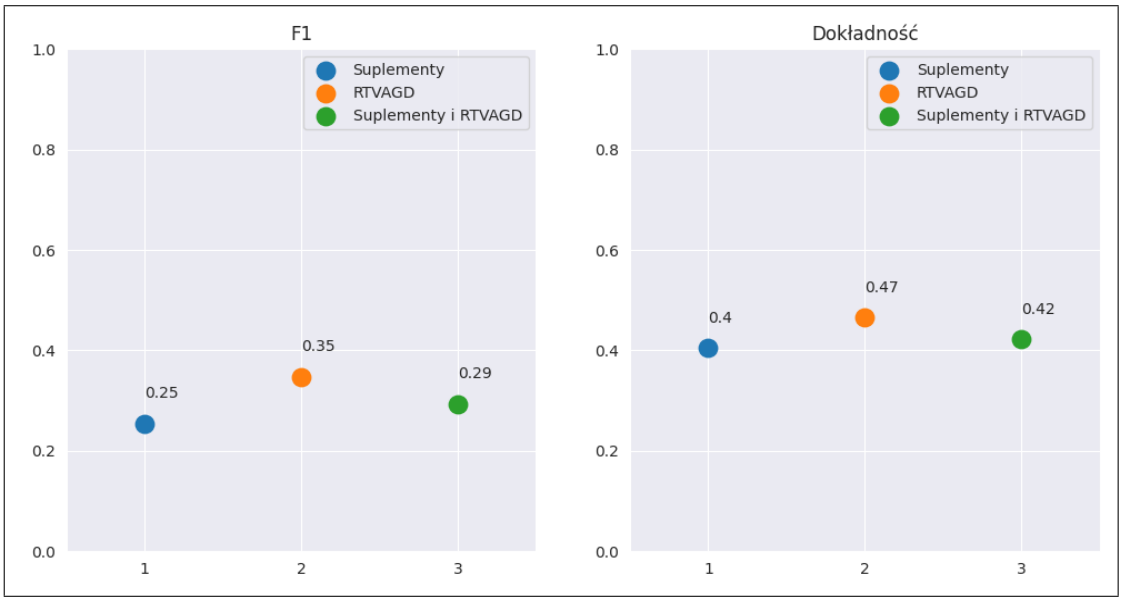
6.2.1. Rezultaty - SVM Klasyfikator

Rys. 4. Dokładność i F1 SVM. Najwyższe F1 jak i dokładność zostały osiągnięte dla zbioru danych ze sprzętami RTVAGD. Rezultaty są znacznie lepsze niż klasyfikacja losowa i można stwierdzić, że model nauczył się pewnych wzorców.

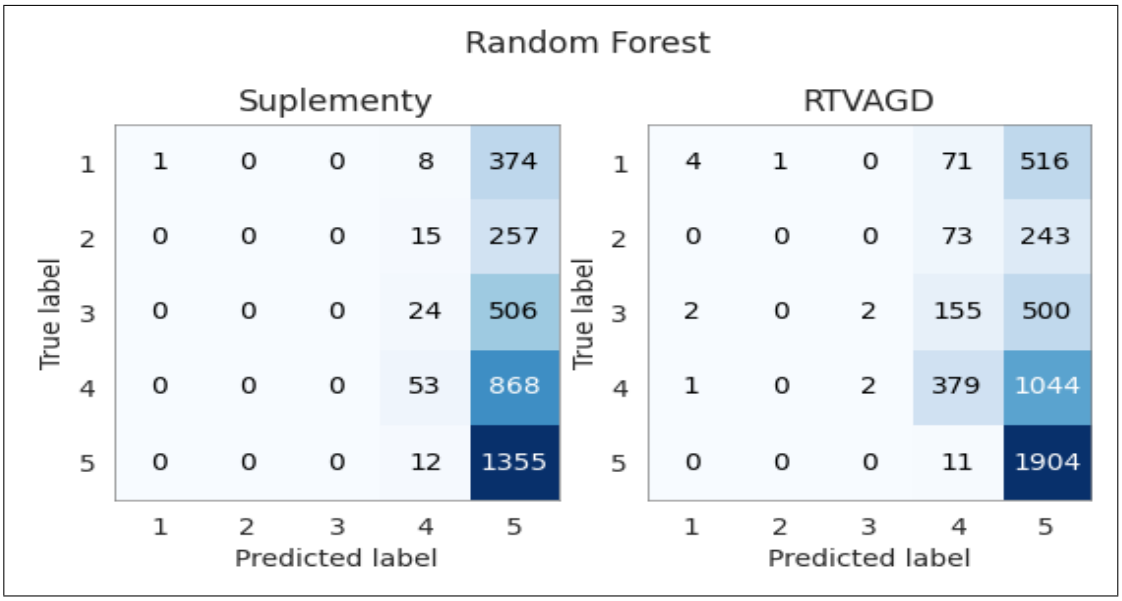


Rys. 5. Macierze pomyłek SVM. Z macierzy pomyłek wynika, że model ma problemy z opiniami o ocenie '2', z pewnością jest to spowodowane tym, że jest to najmniej liczna klasa w zbiorze danych. Opinie z ocenami '4' i '5' uzyskały najwyższą dokładność.

6.2.2. Rezultaty - Random Forest

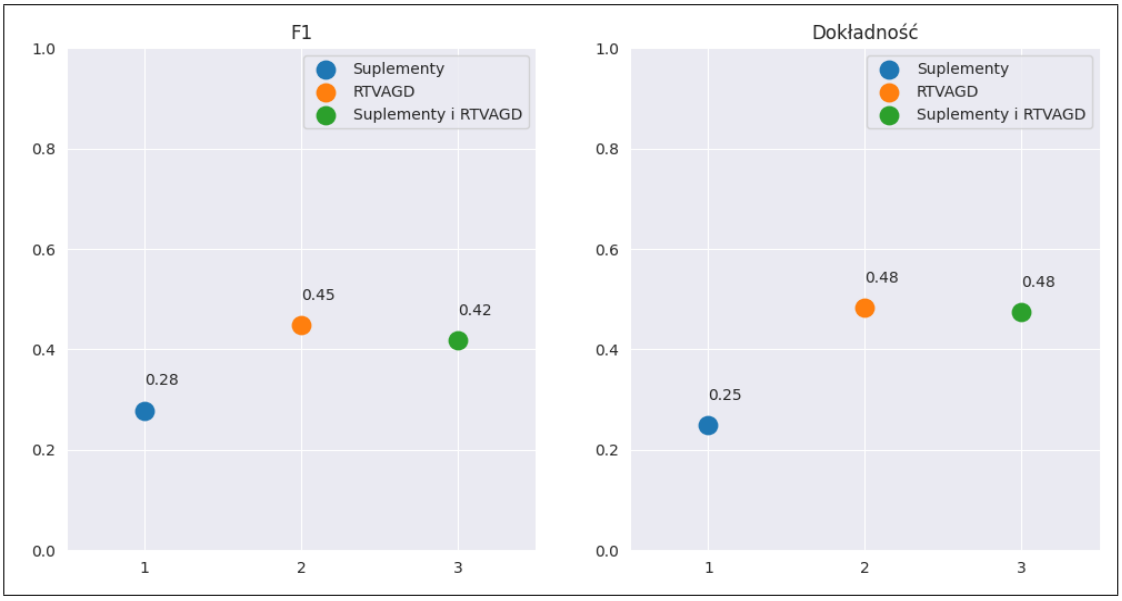


Rys. 6. Dokładność i F1 Random Forest. Wyniki są znacznie gorsze niż w przypadku SVM. Może być to spowodowane zbyt małym rozmiarem zbioru danych jak na ten model. Znów jednak to dla kategorii RTVAGD wyniki są nieco wyższe niż w pozostałych przypadkach.

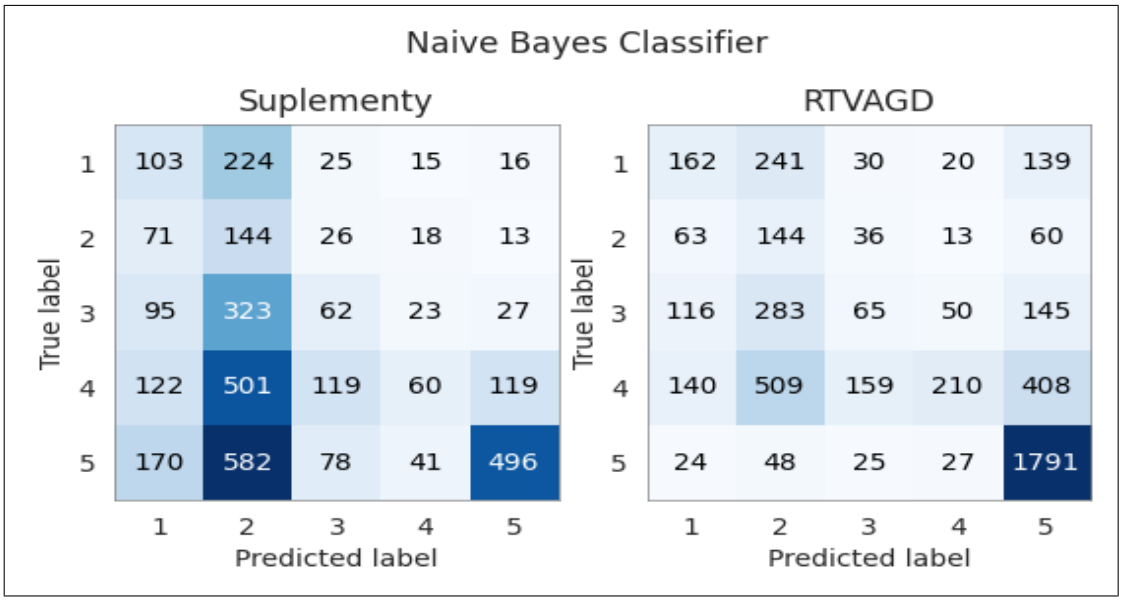


Rys. 7. Macierze pomyłek Random Forest. W przypadku suplementów, model praktycznie wszystkie oceny przewidział jako ’5’. Z kolei w RTVAGD praktycznie wszystkie predykcje to ’4’ i ’5’. Mamy tutaj silne przyciąganie w stronę dobrych opinii.

6.2.3. Rezultaty - Naiwny Klasyfikator Bayesa

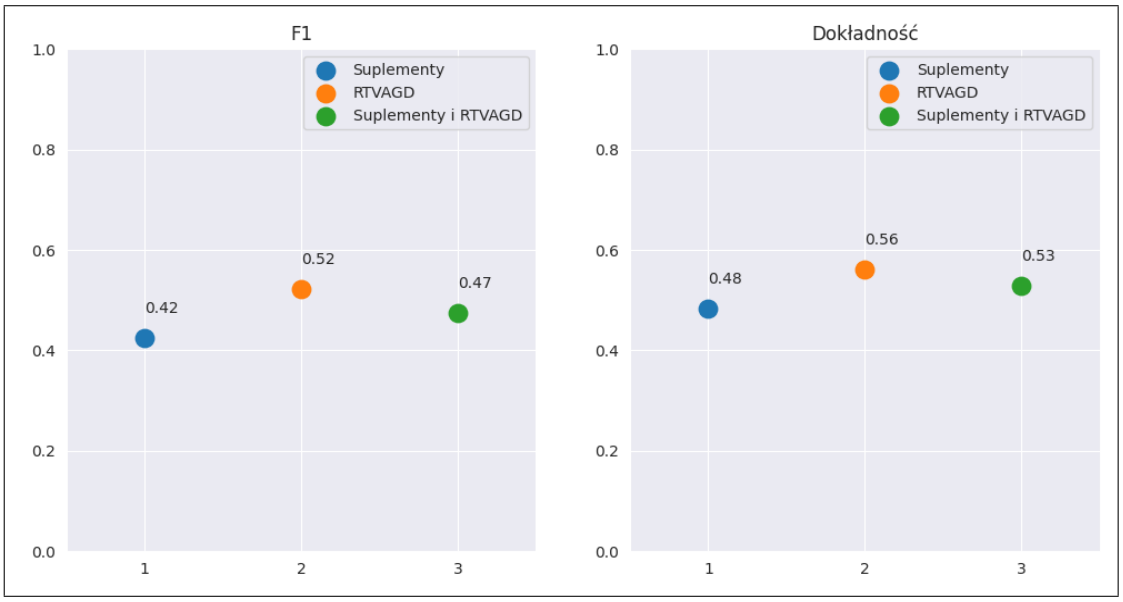


Rys. 8. Dokładność i F1 Naiwny Klasyfikator Bayesa. Wyniki znacznie lepsze niż Random Forest i gorsze niż SVM. Ponownie zbiór opinii o suplementach ma najmniejsze wartości metryk ewaluacyjnych. Jednak wciąż są to znacznie wyższe wartości niż klasyfikacja losowa.

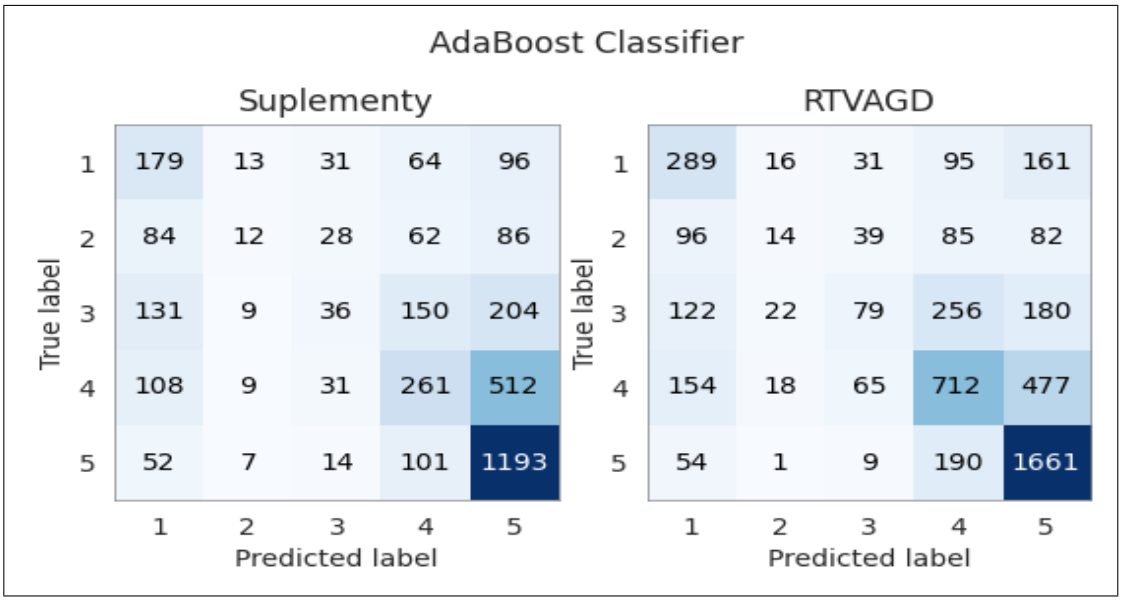


Rys. 9. Macierze pomyłek Naiwny Klasyfikator Bayesa. Zupełnie inne zachowanie niż w przypadku poprzednich modeli, tym razem to klasa '2' oraz '5', stały się najczęstszym wynikiem modelu. Wyniki jednak są mocno zaszumione i odbiegają od prawdziwych wartości.

6.2.4. Rezultaty - AdaBoost Klasyfikator



Rys. 10. Dokładność i F1 AdaBoost Klasyfikator. AdaBoost Klasyfikatory to popularne i wydajne modele. Jednak tym razem nie udało się osiągnąć wyższych wartości niż SVM. Są to drugie najlepsze rezultaty. Ponownie w kategorii RTVGAD mamy lepsze rezultaty.



Rys. 11. Macierze pomyłek AdaBoost Klasyfikator. W przypadku tej macierzy pomyłek mamy tutaj bardzo mało wyników z klasy '2'. Znaczą część stanowią oceny '4' i '5', jest też trochę tych z oceną '1'. Jeśli zadanie miałoby na celu przykładowo przewidywanie opinii ze środka zakresu, model ten miałby najlepszą z użytych w projekcie wydajność.

6.3. Analiza błędów

Różne modele dały różne rezultaty. Dobrą techniką jest podstawienie tych samych danych do różnych modeli uczenia maszynowego. Mimo użycia tego samego zbioru danych wyniki otrzymane z różnych modeli mogą się znacznie od siebie różnić co widać w tym przypadku. Biblioteka sklearn daje nam łatwość w testowaniu wielu gotowych rozwiązań zmieniając dosłownie kilka linijek kodu. Co za tym idzie wybierając jedynie inny model możemy znacznie poprawić efekty naszej pracy.

Najlepsze wartości F1 i dokładności osiągnął jednak SVM klasyfikator natomiast najgorsze Random Forest. Dokładność modeli waha się na poziomie od około 0.2 do 0.65. Jest to duża rozbieżność i należy przeanalizować przyczyny tak dużego zróżnicowania. Przyczyn może być wiele, zależą one głównie od każdego konkretnego modelu.

W przypadku SVM mamy tutaj do czynienia z najlepszymi rezultatami. Podobnie jak w pracy Bo Pang and Lillian Lee z 2005 roku [4] to właśnie SVM Klasyfikator osiąga największa dokładność w tego typu zadaniach. W artykule autorzy dokonywali klasyfikacji 4 klasowej, mimo to pomimo, że w projekcie była klasyfikacja 5 klasowa poziomy dokładności są do siebie mocno zbliżone. W macierzy pomyłek również nie widać silnego przyciągania dla jednej konkretnej klasy.

Random Forest z kolei dał najgorsze rezultaty. Pierwszym co się nasuwa to fakt, że zbiór danych jest stosunkowo niewielki. Może być to nie wystarczająca ilość danych dla tego modelu. W macierzy pomyłek widać, że wszystkie rekordy są przyciągane do klasy z oceną '5' oraz '4'. W innych klasach znajduje się dosłownie 5 rekordów. Model ten w przypadku tego konkretnego zadania zdaję się być zupełnie niepraktyczny.

Naiwny Klasyfikator Bayesa dał wyniki znacznie lepsze niż Random Forest jednak gorsze niż SVM. Jest to prosty model, który nieraz może dać zaskakująco dobre rezultaty mimo swoich ograniczeń i uproszczeń. W tym przypadku jednak nie uzyskał najlepszych wyników. Mimo to jego popularność i prostota były warte przetestowania jego wydajności.

AdaBoost Klasyfikator to jedyny klasyfikator na liście który korzysta z opisanej wcześniej techniki boostingu. Jak widać na wykresach wyniki są nieco gorsze od najlepszego modelu. Jest to popularny model który obecnie często wykorzystywany jest do zadań klasyfikacji. W tym przypadku jednak nie zdołał osiągnąć lepszych wyników niż SVM.

6.4. Podsumowanie użytych metod

Podsumowując najlepszym modelem dla tego konkretnego zadania okazał się SVM. Osiągnął on dokładnośc na poziomie 57% dla zbioru danych o suplementach, 67% dla zbioru danych o artykułach RTVAGD oraz dokładność na poziomie 62% dla zbioru danych zawierającego połączone wszystkie zebrane rekordy. Wartości F1 jakie osiągnął model to 0.54 dla suplementów, 0.63 dla RTVAGD oraz 0.58 w przypadku połączonego zbioru danych. Widać tutaj, że klasyfikacja opinii o sprzętach RTVAGD osiąga lepsze rezultaty, zarówno pod względem F1 jak i dokładności. Może wynikać to między innymi z faktu, że ten zbiór danych jest większy, mediana długości opinii jest większa, lub po prostu w przypadku opinii o tych produktach różnice między klasami są bardziej widoczne. Zgodnie z dobrą praktyką zostały przetestowane różne modele. Mimo tego że faworytami zdawały się być częściej używane modele zespołowe takie jak Random Forest czy AdaBoost nie zdołały one uzyskać lepszych wyników niż SVM. W przypadku innych zadań czy chociażby innego (lub po prostu większego/mniejszego) zbioru danych z cała pewnością warto powtórzyć eksperyment i przetestować od nowa wszystkie modele.

7. Podsumowanie

7.1. Wnioski z pracy

Celem pracy było pobranie opinii od użytkowników z aukcji o produktach, zapis wpisów do plików csv, oczyszczenie danych, wektoryzacja tekstu i klasyfikacja opinii względem ich ocen. Cel został osiągnięty poprzez pozyskanie opinii techniką web scrappingu, wykorzystując bibliotekę Selenium. Inżynieria cech została wykonana przy użyciu najpopularniejszych bibliotek NLP w Pythonie, takich jak pandas czy nltk. Część obejmująca uczenie maszynowe została zaimplementowana wykorzystując bibliotekę sklearn.

W ramach pracy zostały pobrane opinie o produktach z dwóch kategorii z serwisu Allegro.pl. Wpisy były pobierane z wykorzystaniem biblioteki Selenium i przechowywane w plikach .csv. Opinie były pobierane przez kilka tygodni na przełomie kwietnia i maja 2023. Wszystkie dane zostały oczyszczone, przeanalizowane pod kątem częstości wystąpień i długości wpisów. Następnie zastosowano techniki NLP, takie jak usuwanie stop words, lematyzacja oraz wektoryzacja za pomocą Tf-idf. Kolejnym krokiem było wytrenowanie 4 różnych modeli uczenia maszynowego w celu klasyfikacji opinii. Aby dobrać odpowiednie parametry tych modeli, wykorzystana została technika grid search. Zostały wybrane metryki ewaluacji modeli, takie jak F1, dokładność i macierze pomyłek. Najlepsze rezultaty osiągnął SVM, uzyskując wyniki na poziomie cytowanego wcześniej artykułu naukowego. Natomiast najgorsze wyniki uzyskał Random Forest, który przyciągał praktycznie wszystkie dane do oceny 4 lub 5.

Z badania zostały wyciągnięte wnioski, przeanalizowane co można usprawnić i zidentyfikowane potencjalne problemy. Cały proces obejmował kompleksowe podejście, począwszy od pozyskania danych aż do klasyfikacji opinii, co pozwoliło na dogłębną analizę i zrozumienie opinii użytkowników na platformie Allegro.pl.

Klasyfikacja opinii, czyli wpisów, które są szczególnie subiektywne nie jest łatwym zadaniem. Dla każdego użytkownika każda ocena ma nieco inną wartość, podobnie też ze słowami, których używają. Co widać w zbiorze danych niektóre opinie, które pod względem tekstu są identyczne mają nieraz zupełnie inne oceny. Warto mieć na uwadze, że w rozważanym przypadku mieliśmy do czynienia z 5 klasami. Jest to więcej niż w bardziej typowym przypadku, gdzie rozróżniamy jedynie pozytywna/negatywna lub pozytywna/neutralna/negatywna. Stąd też

wyniki jakie otrzymamy będą stosunkowo gorsze. Mimo tak dużego problemu jakim jest klasyfikacja tekstów silnie subiektywnych, wyniki jakie zostały uzyskane są znacznie lepsze niż klasyfikacja losowa.

7.2. Ograniczenia i kierunki dalszych badań

Pomimo znaczącego postępu w dziedzinie NLP, istnieją istotne wyzwania związane z subiektywnością tekstu w opiniach, które mogą znacząco ograniczać skuteczność systemów sztucznej inteligencji.

Jednym z głównych problemów jest złożoność natury ludzkiego języka, który często jest zależny od kontekstu i tematu. Różnice kulturowe i indywidualne czy chociażby ironie i sarkazmy wpływają na interpretację słów, co sprawia, że budowanie uniwersalnych modeli NLP staje się wyzwaniem.

Dodatkowo, subiektywność tekstu w opiniach oznacza, że dane wyrażone w zdaniach mogą być trudne do uchwycenia i analizy przez algorytmy. Różnice w stylu pisania, sformułowaniach i używanych słowach mogą prowadzić do błędnej interpretacji treści. Co więcej, jedno zdanie może być rozumiane inaczej w zależności od kontekstu, co utrudnia stworzenie modeli NLP zdolnych do dokładnego zrozumienia intencji autora. Rozwiązanie tego problemu wymaga dalszego rozwoju zaawansowanych technologii, które uwzględniają kontekst, ton, styl i emocje zawarte w tekście. Integracja sztucznej inteligencji z bardziej zaawansowanymi narzędziami analizy tekstu może pomóc w lepszym zrozumieniu subiektywności i kontekstu wypowiedzi.

7.3. Potencjalne korzyści i wyzwania

Wyniki otrzymane z modeli są mimo wszystko znacznie lepsze niż klasyfikacja losowa. Jak widać modele były w stanie wyłapać pewne zależności i są w stanie (z pewną dokładnością) klasyfikować wpisy. Praktycznym zastosowaniem do jakiego można wykorzystać modele klasyfikujące może być próba wprowadzenia jedynie tekstowych opinii o produktach i usunięciu ocen. Oceny możemy przewidywać jako statystykę widoczną jedynie dla dostawcy produktu. Takie rozwiązanie jest aktualnie stosowane, nie tyle co w przypadku opinii o produktach a w przypadku 'łapek w górę/dół' w najpopularniejszych social mediach.

Kolejną kwestią jest to, że klasyfikacja może być wartościowa z perspektywy analizy danych, gdyż daje to możliwość przeprowadzenia kolejnych eksperymentów. Jednym z takich potencjalnych eksperymentów może być np. próba określenia jak nowo zapisana opinia dobrze pasuje do pozostałych z tej klasy oraz sugestia użytkownikowi jaką ocenę posiadają podobne opinie.

7.4. Perspektywy rozwoju

Jest kilka usprawnień które można rozważyć myśląc o rozwoju projektu. Jak w każdym projekcie analizy danych z całą pewnością warto powiększyć zbiór danych. Zazwyczaj sprawdza się tutaj powiedzenie, że im więcej danych tym lepiej.

Poza ilością danych warto poprawić też ich jakość. Można to zrobić np. dodając ograniczenie co do długości opinii. Dłuższe wpisy będą miały więcej słów przez co ich wektory będą się bardziej różnić a próba ich klasyfikacji może okazać się łatwiejsza. Kolejnym pomysłem będzie dodanie własnej unikalnej na potrzeby tego przypadku listy stop words które należy usunąć z tekstów. Przez co skupiamy się jedynie na bardziej wartościowych słowach w tekście.

W kontekście samych wektorów można również eksperymentować ze sposobem zamiany tekstu na wektory. Popularną obecnie techniką w przypadku dużych modeli jest tzw. Word Embedding [32]. Czyli techniki która mapuje każde słowo na wektor w przestrzeni n-wymiarowej. Podobnie jak w przypadku wielu takich technologii mamy gotowe wektory dla słów angielskich. W momencie tworzenia pracy nie udało mi się znaleźć polskich odpowiedników. Można pokusić się o wytrenowanie konkretnej przestrzeni embeddings dla tego szczególnego przypadku klasyfikacji.

Bibliografia

[1] Czym jest NLP. URL: https://www.ibm.com/topics/natural-language-processing. Data dostepu do strony: 2023-10-21.

[2] Francois Chollet. „*Deep Learning. Praca z językiem Python i biblioteką Keras*". Wydawnictwo Helion, 2019.

[3] Dipanjan Sarkar. „*Text analytics with Python*". apress, 2019.

[4] Bo Pang i Lillian Lee. „*Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*". W: *Association for Computational Linguistics* (2005).

[5] Bing Liu. „*Sentiment Analysis and Subjectivity*". W: *Journal of Data Analysis and Information Processing* (2010).

[6] Mayura Kinikar Jayashri Khairnar. „*Machine Learning Algorithms for Opinion Mining and Sentiment Classification*". W: *International Journal of Scientific and Research Publications* (2013).

[7] Adam Funk Horacio Saggion. „*Interpreting SentiWordNet for Opinion Classification*". W: *European Language Resources Association* (2010).

[8] Biblioteka NLTK dla języka Python. URL: https://www.nltk.org/. Data dostepu do strony: 2023-10-19.

[9] Język programowania Python. https://www.python.org/. Data dostepu do strony: 2023-10-26.

[10] Biblioteka spacy dla języka Python. URL: https://spacy.io/. Data dostepu do strony: 2023-10-22.

[11] Biblioteka Gensim dla języka Python. URL: https://radimrehurek.com/gensim/. Data dostepu do strony: 2023-10-21.

[12] Biblioteka transformers dla języka Python. URL: https://huggingface.co/docs/transformers/. Data dostepu do strony: 2023-10-13.

[13] Towards Data Science URL: https://towardsdatascience.com/. Data dostepu do strony: 2023-10-23.

[14] John Hearty. „*Zaawansowane uczenie maszynowe z językiem Python*". Wydawnictwo Helion, 2016.

[15] Biblioteka Selenium dla języka Python. URL: https://selenium-python.readthedocs.io/. Data dostepu do strony: 2023-10-19.

[16] Biblioteka Pandas dla języka Python. URL: https://pandas.pydata.org/. Data dostepu do strony: 2023-10-15.

[17] Biblioteka Numpy dla języka Python. URL: https://numpy.org/. Data dostepu do strony: 2023-10-23.

[18] Definicja stop words. https://www.opinosis-analytics.com/knowledge-base/stop-words-explained/. Data dostepu do strony: 2023-10-15.

[19] Polskie stop words. https://github.com/bieli/stopwords/blob/master/polish.stopwords.txt. Data dostepu do strony: 2023-10-15.

[20] Narzędzie do lematyzacji w języku polski - Morfeusz2. http://morfeusz.sgjp.pl/. Data dostepu do strony: 2023-10-14.

[21] Biblioteka sklearn dla języka Python. URL: https://scikit-learn.org. Data dostepu do strony: 2023-10-23.

[22] Definicja Tf-idf. https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3. Data dostepu do strony: 2023-10-18.

[23] SVM Klasyfikator. https://scikit-learn.org/stable/modules/svm.html. Data dostepu do strony: 2023-10-21.

[24] Nadmierne dopasowanie w uczeniu maszynowym. https://aws.amazon.com/what-is/overfitting/. Data dostepu do strony: 2023-10-26.

[25] AdaBoost Klasyfikator: https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/. Data dostepu do strony: 2023-10-14.

[26] E. Schapire Yoav Freund. „*A decision-theoretic generalization of on-line learning and an application to boosting*". W: *ATT Labs* (1996).

[27] Technika Grid Search. https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee. Data dostepu do strony: 2023-10-15.

[28] Definicja dokładności. https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=pl. Data dostepu do strony: 2023-10-15.

[29] Definicja F1. https://www.v7labs.com/blog/f1-score-guide. Data dostepu do strony: 2023-10-15.

[30] Definicja Macierzy pomyłek. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. Data dostepu do strony: 2023-10-15.

[31] Biblioteka Matplotlib dla języka Python. URL: https://matplotlib.org/. Data dostepu do strony: 2023-10-15.

[32] Word embedding. https://machinelearningmastery.com/what-are-word-embeddings/. Data dostepu do strony: 2023-10-15.