Zestawienie metod uczenia maszynowego do przeprowadzania analizy sentymentu

Paweł Michnowicz

Polsko-Japońska Akademia Technik Komputerowych

University of Oxford

Promotor:

Dr. hab. Grzegorz Marcin Wójcik

Warszawa 2022

Spis treści

Spis rysunków

Rozdział 1

Wprowadzenie do analizy sentymentu

Poniższy rozdział poświęcony jest omówieniu zagadnień teoretycznych oraz przedstawieniu istniejącego stanu wiedzy na temat pojęcia analizy sentymentu. Dodatkowo opisane zostały sposoby implementacji oraz zastosowania danej metody.

1.1 Ogólna definicja

Analiza sentymentu/wydźwięku jest pewnego rodzaju analizą tekstu dokonywaną automatycznie lub półautomatycznie. Jej zadaniem jest identyfikacja nastawienia emocjonalnego autora w odniesieniu do badanego obiektu. Analizowany może być cały tekst lub poszczególne zdania, najczęściej wyniki rozróżnia się trzema etykietami: pozytywny, negatywny i neutralny.

1.2 Historia rozwoju

Prekursorem dzisiejszego internetu były sieci LAN, gdzie komputery w firmach łączyły się lokalnie co dawało możliwość szybkiej wymiany danych cyfrowych. Produkcja danych zaliczyła swój przełom wraz z powstaniem sieci ogólnoświatowej (ang. *World Wide Web*) w latach dziewięćdziesiątych. Internet stawał się bardziej powszechny gdzie coraz częściej różne organizacje ale i ludzie byli w stanie wytwarzać gigabajty danych. Kolejnym etapem, który miał wpływ na wzrost ogólnodostępnych danych było powstanie mediów społecznościowych. Użytkownik poruszając się w wirtualnym świecie pozostawia ślad cyfrowy w postaci danych. Są one cennym źródłem informacji dla korporacji, które mogą je wykorzystać do określenie przewidywań zachowań

Rysunek 1.1: Liczba wyszukiwań hasła “Sentiment Analysis”

konsumenckich, preferencji klientów oraz dostosowania do nich oferty marketingowej. W dzisiejszym świecie często to my i nasze aktywności stają się produktem.

Przez lata, przed rozwojem internetu i mediów społecznościowych, obszar badań na temat analizy sentymentu był w dużym stopniu ignorowany. Dopiero po około roku 2004 stała się ona jedną z najszybciej rozwijających się dziedzin naukowych. [15] Przyrost popularności można zauważyć na rys. 1.1 ukazującym ilość wyszukiwań hasła Śentiment Analysis”w wyszukiwarce Google.

Przed okresem wzrostu dostępności danych tekstowych zawierających recenzje oraz sondaże, badania opinii publicznej opierały się głównie na przeprowadzaniu ankiet na pewnej grupie badawczej. Przykładem takiego projektu może być praca z 1940 roku zatytułowana ”The Cross out Technique as a Method in Public Opinion Analysis”[19]. Dokonano w niej analizy opinii publicznej, głównie w obszarze politycznym, gdzie zbiorem danych były przeprowadzone ankiety na studentach oraz ich rodzicach.

W momencie gdy nastąpił rozwój internetu oraz mediów społecznościowych dostęp do ogromnej ilości danych stał się powszechny i można było go wykorzystywać. Aktualnie ludzie zamieszczają swoje recenzje oraz opinie na temat danych produktów na ogólnodostępnych stronach, forach, serwisach społecznościowych lub bezpośrednio na stronach producentów. Dzięki temu firmy nie musiały już przeprowadzać badań opinii publicznej za pomocą ankiet gdyż wszystkie potrzebne informacje były powszechnie dostępne. Jednak w przeciwieństwie do badań ankietowych opinie nie były wyrażane bezpośrednio za pomocą odpowiedzi na konkretne pytania lecz swobodnie przez autora. W związku z tym, że pozyskiwane dane są mało ustrukturyzowane to wymagają dużego zaangażowania w ich przygotowanie. Firmy miały dostęp do ogromnej ilości danych tekstowych, tak więc aby usprawnić proces analizy powstawały różnego rodzaju systemy automatycznego wykrywania emocji

Jedną z pierwszych publikacji wokół tematu analizy sentymentu wykorzystującą uczenie maszynowe była praca z 2002 roku zatytułowana "Thumbs up? Sentiment Classification using Machine Learning Techniques" [**?**]. Autorzy wykorzystali zbiór recenzji filmowych oraz stworzyli model klasyfikujący wydźwięk pojedynczych zdań. Posłużyli się takimi algorytmami jak Naiwny Bayes (ang. *Naive Bayes*), maszyną wektorów nośnych (ang. *Support-vector machine*) oraz klasyfikatora maksymalnej entropii (ang. *Max Entropy Classifier*). W tym samym roku przez innych autorów została opublikowana praca o podobnej nazwie "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews", gdzie został zaproponowany model uczenia nienadzorowanego do klasyfikacji opinii jako etykiety przyjmował wartości "rekomendujące" oraz "odradzające". Korzystano ze zbioru recenzji na remat aut, banków, filmów oraz celów podróży.

1.3 Zastosowania

Metoda analizy wydźwięku stała się popularna w wielu dziedzinach takich jak:[18] [1] [16]

- W biznesie, poprzez wsparcie firm biznesowych w monitorowaniu reputacji oraz opinii na temat firmy lub marki wykorzystując wszelkie opinie konsumentów na temat produktów i usług.
- W finansach, mogą dostarczać analitycznych informacji dla inwestorów finansowych, na podstawie opinii rynkowych.
- W polityce, gdzie kampanie polityczne są często zależne od opinii wyrażanych przez wyborców.
- W ekonomii gdzie może zostać użyta jako wsparcie do wskazania korelacji pomiędzy wiadomościami ekonomicznymi, a zachowaniami wskaźników finansowych na giełdzie.
- W socjologi, na przykład może pomóc określić nastroje społeczne w stosunku do nowych zjawisk pojawiających się w przestrzeni publicznej.

1.4 Poziomy analizy

Analizy sentymentu dokonuje się na różnym poziomie ziarnistości, można ją rozdzielić na analizę trzech różnych poziomów: poziomu dokumentu, zdania lub aspektu / cechy.

1.4.1 Stopień dokumentu

Poddany analizie zostaje cały dokument i na bazie całego tekstu zostaje określony konkretny wydźwięk emocjonalny (pozytywny, negatywny lub neutralny). Zakłada się, że cały dokument wyraża sprecyzowaną opinie na temat pojedynczego obiektu, dlatego nie stosuję się tej analizy na dokumentach, które oceniają wiele podmiotów na raz.

1.4.2 Stopień zdania

Klasyfikacja wydźwięku jest determinowana na podstawie analizy pojedynczych zdań i do każdego zdania przypisana jest konkretna emocja. Poprzez ten typ analizy dokonuje się również klasyfikacji subiektywności na podstawie, której oceniamy czy zdanie jest obiektywne, zawierające faktyczne informacje lub jest zdaniem subiektywnym, z którego jesteśmy w stanie wyciągnąć opinie lub wydźwięk emocjonalny. Następnie poddaje się analizie tylko zdania subiektywne na podstawie, których wyciąga się interesujący nas wydźwięk. [10]

1.4.3 Stopień aspektu / cechy

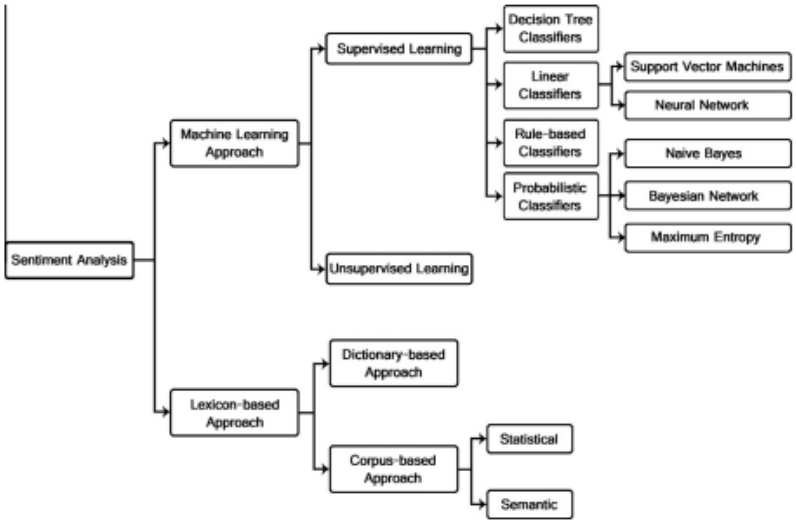
Zarówno analizy na poziomie dokumentu jak i poziomie zdania ukazują ogólne nastawienie do ocenianego obiektu, nie pokazuje jakie cechy oceniane są pozytywnie a jakie negatywnie. W przeciwieństwie do zdań badających wydźwięk na poziomie aspektu gdzie mamy możliwość określenia emocji do konkretnej cechy badanego podmiotu. Można to zauważyć na przykładzie zdania

"Robert Lewandowski ma niezwykłą technikę, jednak jego szybkość biegu pozostawia wiele do życzenia."

gdzie można wyodrębnić dwie cechy: technika oraz szybkość biegu, pierwsza z cech została oceniona pozytywnie, zaś druga negatywnie. Analizując teksty w ten sposób można określić różne oceny na temat wielu cech analizowanego obiektu.

1.5 Podejścia

Podejścia do analizy wydźwięku można podzielić na dwie różne techniki. Pierwsza z nich opiera się na wykorzystaniu słowników, gdzie do każdego słowa przypisany jest konkretny wydźwięk. Druga technika bazuje na statystycznych metodach uczenia maszynowego. Podział ten został zaproponowany w publikacji [13] i zaprezentowany na rys. 1.2.



Rysunek 1.2: Podział podejść do analizy sentymentu

1.5.1 Podejścia leksykalne

Podejścia leksykalne są podzielone na dwie kategorie, pierwsza z nich bazuje na wykorzystywaniu słowników, druga wykorzystuje korpusy tekstów.

1.5.1.1 Metoda słownikowa

Pierwsza metoda bazuje na wykorzystaniu przygotowanych słowników zawierających słowa, którym przypisany jest wydźwięk emocjonalny. Bierze ona pod uwagę liczbę występowań kluczowych wyrażeń, którym dopisany jest konkretny sentyment (pozytywny lub negatywny), np. w poniższym zdaniu

"Pies jest spokojny, ale na widok kota staje się niebezpieczny i agresywny."

Kluczowymi wyrażeniami nadające wydźwięk emocjonalny są słowa "spokojny" posiadające wydźwięk pozytywny oraz wyrażenia "agresywny" i "niebezpieczny" posiadające wydźwięk negatywny. Ze względu na przewyższającą ilość słów o negatywny sentymencie całe zdanie również zostanie tak sklasyfikowane.

1.5.1.2 Metoda korpusowa

W odróżnieniu od metody słownikowej, technika ta wykorzystuje korpusy językowe, czyli zbiór tekstów, który: [17]

- Zawiera metadane dostarczające informacje na temat m.in autora, daty i miejsca wydania.

- Określa warstwy językowe tekstu, czyli to jaki język został użyty z uwzględnieniem wszelakich gwar, żargonów oraz stylów funkcjonalnych (urzędowy, naukowy, potoczny itd.).
- Zawiera również informacje na temat tego w jakich formach funkcjonuje dane słowo np. rozpoznanie poszczególnych części mowy oraz ustalenie ich form i odmian.

Metoda korpusowa jest już bardziej złożona od poprzedniej. Wykorzystuje ona reguły składniowe i gramatyczne danego języka, więc analizowany jest cały tekst, a nie pojedyncze wyrażenia. Wyróżnia się dwie metody korpusowe statystyczną i semantyczną. Dzięki analizie reguł możliwe jest rozpoznanie części tekstu, które mają wpływ na sentyment, wyróżnia się negacje (negation), neutralizacje (nullification) oraz ograniczenia (limitation). Dodatkowo, wykorzystuje się reguły odnoszące się do znaczenia słów (synonimy, wyrazy przeciwstawne, wyrazy bliskoznaczne, wieloznaczność słów) oraz do zasad gramatyki i fleksji. [6] [22]

W określaniu wydźwięku całej frazy ważną rolę pełnią również spójniki w zdaniach złożonych. Dla przykładu spójniki łączne takie jak "i", "oraz" oznacza, że wyrażenia kluczowe w zdaniach składowych mają taką samą orientacje, a spójniki przeciwstawne "ale", "jednakże" wskazują na zmianę wydźwięku w dalszej części zdania. [13].

1.5.2 Podejścia wykorzystujące uczenie maszynowe

Uczenie maszynowe to dziedzina nauki, która jest podzbiorem sztucznej inteligencji, wykorzystująca odpowiednio wielkie zbiory danych w celu podejmowania decyzji lub precyzowania przewidywań. Ogromny wzrost dostępności danych oraz zwiększone możliwości ich przechowywania spowodowało szybki rozwój dziedziny uczenia maszynowego oraz ogromne zainteresowanie wokół tej sfery. Są rozwijane lub tworzone nowe algorytmy, a sama dziedzina znajduję zastosowanie w kolejnych obszarach.

Dziedzina skupia się na stworzeniu modelu matematycznego, który na podstawie przekazanych danych ma za zadanie przypisanie odpowiedniej etykiety w przypadku klasyfikacji, wartości numerycznej w przypadku regresji lub odpowiedniej grupy w przypadku grupowania do każdej badanej jednostki. Model matematyczny udoskonalany jest przy pomocy dostarczanych do niego informacji, poprzez przyswajanie coraz większej ilości zbioru danych, co umożliwia mu nauczenie się pewnych wzorców i korelacji dzięki czemu jest on w stanie wykonać konkretne zadanie z większą efektywnością. Dane są w postaci wartości pewnych atrybutów danego obiektu np. gdy chcemy przekazać dane na temat pracowników firmy to do modelu przekazujemy

informacje na temat kolejnych obiektów, przykładowymi cechami mogą być: “płeć”, “stanowisko”, “zarobki”.

Parametry modelu aktualizowane są w kolejnych iteracjach, regulacja dokonywana jest na podstawie tego jak bardzo model się myli. Błąd modelu określany jest poprzez wybraną metrykę, która wyznacza rozbieżność pomiędzy prawdziwymi wartościami, a tymi przewidzianymi przez model.

Nie przekazujemy do modelu wszystkich danych jakie mamy, dokonujemy podziału zbioru danych na dwa niezależne podzbiory: treningowy i testowy. Proces uczenia, czyli regulacji parametrów modelu odbywa się na zbiorze treningowym. Następnie do modelu przekazywane są dane ze zbioru testowego dla których dokonywana jest predykcja, dzięki porównaniu otrzymanych wyników z prawdziwymi możemy określić skuteczność modelu. Podział danych na dwa podzbiory powoduje, że możemy sprawdzić jak model radzi sobie z danymi, z którymi wcześniej nie miał styczności.

Uczenie maszynowe zgodnie z rysunkiem 1.2 jest podzielone na dwie różne kategorie: uczenie nadzorowane (ang. *supervised learning)* oraz uczenie nienadzorowane *(ang. unsupervised learning).*

W podejściu uczenia z nadzorem tworzymy model matematyczny, który za zadanie ma przewidzieć odpowiednią klasę na podstawie danych wejściowych, tworzony jest tzw. model predykcyjny. W procesie uczenia dostarczane są dane wyposażone w zbiór atrybutów oraz etykiety, czyli wartość wyjściową, którą model będzie musiał przewidzieć. Możemy wyznaczyć dwa główne typy problemów predykcyjnych, w zależności od rodzaju etykiety:

- regresja - etykieta ma wartość ciągłą,
- klasyfikacja - etykieta ma wartość dyskretną.

W przypadku regresji popularną metryką wykorzystywaną w procesie uczenia oraz do ewaluacji skuteczności modelu jest błąd średniokwadratowy, który jest sumą kwadratów różnic pomiędzy wartością przewidywaną, a faktyczną.

$$MSE = \sum_{i=1}^{D} (x_i - y_i)^2 \tag{1.1}$$

W przypadku klasyfikacji korzystamy z jednej z metryk, które są wyliczane na podstawie wartości z macierzy pomyłek ukazanej na rysunku **??**, który ukazany jest w dalszej części pracy. Posługujemy się wartościami takimi jak :

- TP (ang. *True Positive*) - ilość poprawnie zaobserwowanych klas pozytywnych
- TN (ang. *True Negative*) - ilość poprawnie zaobserwowanych klas negatywnych
- FP (ang. *False Positive*) - ilość błędnie zaobserwowanych klas pozytywnych
- FN (ang. *False Negative*) - ilość błędnie zaobserwowanych klas negatywnych

Najczęściej stosowaną metryką jest dokładność (ang.*accuarcy*), jest wyrażona poprzez stosunek poprawnie sklasyfikowanych wartości do wszystkich wartości, wobec czego otrzymujemy prawdopodobieństwo prawidłowej klasyfikacji.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.2}$$

Kolejną dostępną metryką jest precyzja (ang. *precision*), która jest wyrażona poprzez stosunek prawdziwie pozytywnych do sumy wartości wszystkich wartości, które zostały sklasyfikowane pozytywnie. Dzięki czemu otrzymujemy informacje o tym ile z pozytywnie sklasyfikowanych przypadków zostało przyporządkowane poprawnie.

$$Precision = \frac{TP}{TP + FP} \tag{1.3}$$

Następną metryką jest czułość (ang. *recall*), jest wyrażona poprzez stosunek prawdziwie pozytywnie wartości do sumy wszystkich pozytywnych wartości, włącznie z tymi błędnie sklasyfikowanymi. Otrzymujemy prawdopodobieństwo poprawnej klasyfikacji pod warunkiem, że przypadek jest pozytywny.

$$Recall = \frac{TP}{TP + FN} \tag{1.4}$$

W celu połączenia dwóch wyżej wymienionych metryk, wprowadzona nową o nazwie F-miara(ang. *F-score* łącząca w sobie precyzje oraz czułość poprzez średnią harmoniczną.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{1.5}$$

Tablica 1.1: Zestawienie algorytmów wykorzystujących różne podejścia

Dataset	*pSenti*	*LexiconOnly*	*LearningOnly*
Miscellaneous(Editor)	89.65%	79.40%	90.78%
Browser(Editor)	89.94%	76.94%	91.39%
Browser(Customer)	79.60%	75.50%	80.54%
Antivirus(Customer)	78.55%	70.60%	82.91%
Video(Customer)	83.55%	75.95%	85.83%
Action Games(Customer)	78.77%	71.55%	82.93%
Utilities(Customer)	78.80%	73.70%	82.03%
Movie Reviews	82.30%	66.00%	86.85%

1.5.3 Podejście hybrydowe

Wymienione podejścia nie muszą być wykorzystywane osobno, mogą ze sobą współpracować i budowane algorytmy mają możliwość stosowania zarówno metody leksykalnej jak i uczenia maszynowego jednocześnie, takie podejście nazywane jest hybrydowym. Zazwyczaj tak budowane algorytmy osiągają lepsze wyniki, niż w przypadku gdy stosujemy tylko jedną z wymienionych metod. Przykład porównania ze sobą wszystkich metod został ukazany w publikacji [14]. Jako zbiór danych posłużyły recenzje filmów z serwisu IMDB oraz recenzje oprogramowań z CNET. W przetaczanej publikacji stworzony model *pSenti* osiągnął znacznie większą wydajność niż model oparty na leksykalnych metodach oraz porównywalną do uczenia maszynowego, wyniki zostały zaprezentowane w tabelce 1.1. Przewagą stworzonego algorytmu nad modelami opartymi o uczenie maszynowe jest fakt, że rezultaty dzięki zastosowanym leksykalnym metodom są ustrukturyzowane oraz są łatwiejsze w analizie i wizualizacji.

1.6 Etapy postępowania

1.6.1 Zdobycie i przygotowanie danych

Pierwszym krokiem jest zdobycie oraz oczyszczenie danych, zważając na to, że stworzony model będzie zależny od jakości danych, które mu dostarczymy jest to najważniejszy krok.

W celu dokonania analizy sentymentu poszukujemy danych zawierających opinie na przykład na temat danego produktu. W tym celu możemy przeszukać komentarze na stronie producenta, forum dyskusyjnych lub mediów społecznościowych. Dane

możemy pozyskać ze stron internetowych za pomocą interfejsu API, umożliwiającym komunikacje z serwerami. W celu wykonania jak najskuteczniejszej analizy dane powinny zostać zebrane z maksymalnie dostępnej ilości źródeł, aby zapewnić kompleksowość.

W przypadku gdy stosujemy podejście wykorzystujące uczenie maszynowe pobrane dane wymagają dokonania wstępnego przetworzenia, które można podzielić na kilka kroków:

- Usunięcie interpunkcji, adresów URL oraz niechcianych znaków takich jak pozostałości języka HTML.
- Usunięcie z tekstów wielkich liter.
- Usunięcie stopword'ów, czyli najczęściej występujących słów w danym języku nie niosących ze sobą żadnych istotnych treści.
- Dokonanie tokenizacji, czyli podziału całego tekstu na pojedyncze tokeny (np. podział na pojedyncze słowa).
- Ujednolicenie form wyrazów, czyli skrócenie danego słowa do jego formy podstawowej, za pomocą jednej z dwóch metod:
  - Lematyzacja, gdzie uzyskana podstawowa wersja słowa będzie bezokolicznikiem dla czasowników, w przypadku rzeczownika otrzymamy mianownik w liczbie pojedynczej.
  - Stemming, stosuje się go poprzez usunięcie z danego wyrazów wszelkich przedrostków oraz przyrostków poprzez co otrzymujemy tzw. rdzeń. W przeciwieństwie do lematyzacji, otrzymany wynik nie musi być poprawnie skonstruowanym słowem.

Gdy stosujemy jedną z metod leksykalnych konieczne jest aby tekst w większości pozostał w swojej oryginalnej postaci, w tym przypadku przy obróbce tekstu skupiamy się głównie na usuwaniu niechcianych znaków, które pojawiły się podczas pobierania tekstu, takich jak pozostałości po języku HTML lub adresów URL.

1.6.2 Klasyfikacja tekstu

Stosując jedno z podejść leksykalnych konieczne jest rozróżnienie informacji zawartych w tekście, są one klasyfikowane jako fakty oraz opinie. Część zdania zawierającą opis emocji lub stosunek do pewnych podmiotów, osób lub wydarzeń wyrażaną przez podmiot jest uznawana za opinie, a sama część zdania uznawana za subiektywną. Obiektywne wyrażenia na temat podmiotu zawierające konkretny opis bez wydźwięku emocjonalnego są nazywane faktami. W momencie gdy otrzymamy opracowane dane pierwszy krokiem jest dokonanie podziału zdania na część zawierającą subiektywną wartość oraz faktyczne informacje. Dzięki czemu poprzez analizę części subiektywnej jesteśmy w stanie wywnioskować nacechowanie emocjonalne zawarte w danym tekście. [11] [7]

Juz w 1990 roku Janyce Marbury Wiebe w swojej pracy zaproponował model pozwalający wykrywać subiektywność w analizowanych zdaniach. [21]

1.6.3 Ekstrakcja informacji

Kolejnym etapem jest ekstrakcja cech z analizowanych tekstów, czyli wydobycie kluczowych wyrażeń, które nadają wydźwięk emocjonalny. W przypadku analizy na poziomie aspektu konieczna jest ekstrakcja obiektów oraz ich cech. Traktowane są one często jako dwa oddzielne zadanie ze względu na ich odmienne charakterystyki. W przypadku obiektów chodzi o wydobycie podmiotów, które będą poddawane analizie, są to najczęściej nazwy produktów, usług, wydarzeń, osób fizycznych lub nazw organizacji. Wydobywając cechy produktu wyciągamy własności danego podmiotu, którym dopisywany jest wydźwięk emocjonalny. [12]
Analizując poniższe przykłady:

"Jakość dźwięku z mojego nowego JBL'a jest niesamowita."
"Spotkałem nieznajomą dziewczynę na dworcu, miała naprawdę piękne włosy"

W przypadku tych zdań rolę obiektów pełnią "JBL" oraz "nieznajoma dziewczyna", podczas gdy ich cechami, którym nadawany jest wydźwięk emocjonalny są "jakość dźwięku" oraz "włosy".

W przypadku analiz bazujących na uczeniu maszynowym ekstrakcja cech polega na przekształceniu tekstu do odpowiedniej formy, która będzie mogła być zrozumiana

przez klasyfikator czyli do wektora cech. Najczęściej jest on reprezentowany przez model Bag-of-Words lub TFIDF, które zostały szerzej opisane w następnym rozdziale. Dane dostarczane do tych modeli są najczęściej w formie jednej lub kilku wyrazowych fraz czyi za pomocą modeli n-gram. W momencie gdy przekazywane są pojedyncze wyrazy stosujemy unigramy, jeśli wykorzystywane są pary kolejnych słów stosujemy bigramy, w przypadku gdy model chce wykorzystać trzy kolejne wyrazy jako pojedynczą strukturę nazywamy to trigramami. Jest możliwość stosowania cztero- lub pięciowyrazowych struktur, jednak są one rzadko stosowane.
Analizując poniższe zdanie:

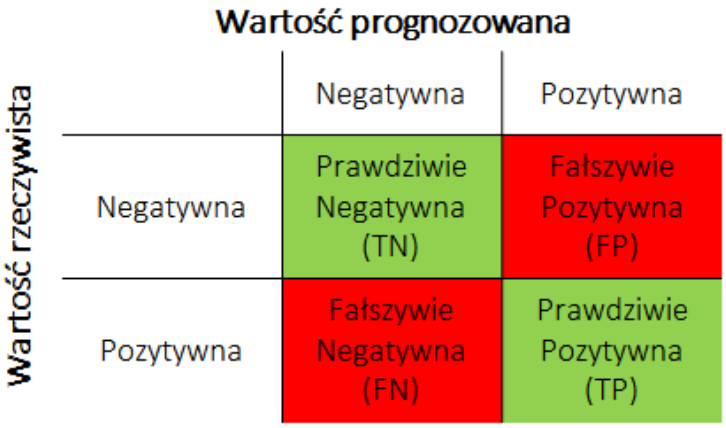
"Koncert był zorganizowany na dobrym poziomie"

Stosując unigramy otrzymamy: ['Koncert', 'był', 'zoorganizowany', 'na', 'dobrym', 'poziomie']
bigramy: ['Koncert był', 'był zoorganizowany', 'zoorganizowany na', 'na dobrym', 'dobrym poziomie']
trigramy: ['Koncert był zorganizowany', 'był zorganizowany na', 'zorganizowany na dobrym', 'na dobrym poziomie']

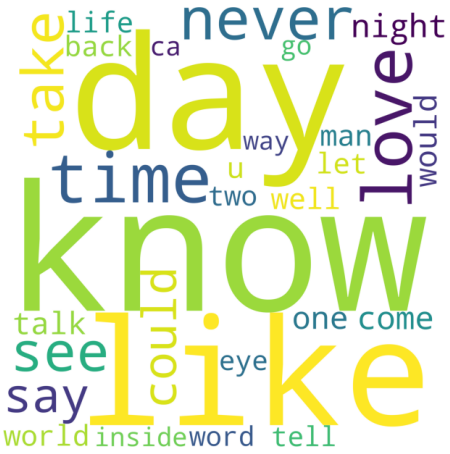
1.6.4 Klasyfikacja i wizualizacja wyników

Następnie stosując jedno z dostępnych podejść do klasyfikacji sentymentu, które zostały opisane w poprzednim podrozdziale, dokonujemy analizy sentymentu przez co otrzymujemy rezultat wydźwięku dla badanego tekstu.

Ostatnim krokiem jest odpowiednie wizualizowanie otrzymanych wyników, jednym z najbardziej powszechnym sposobem na ukazanie rezultatów jest zastosowanie macierzy pomyłek, gdzie zestawiamy ze sobą otrzymane wartości pozytywne oraz negatywne do rzeczywistych wartości. W ten sposób otrzymujemy liczbę poprawnie oraz błędnie zdefiniowanych wartości dla pozytywnego oraz negatywnego wydźwięku. Macierz pomyłek można zastosować też w przypadku gdy analizujemy więcej klas np. gdy oprócz pozytywnego i negatywnego wydźwięku jest możliwe zdefiniowanie tekstu jako neutralny. Dzięki otrzymanym wartością w macierzy jesteśmy w stanie policzyć takie miary jak dokładność, precyzja, czułość oraz f-miarę. Przykładowa macierz pomyłek została ukazana na rysunku 1.3



Rysunek 1.3: Macierz pomyłek



Rysunek 1.4: Przykładowa chmura słów

Jedną z form do wizualizacji danych tekstowych jest zastosowanie chmury słów (ang. *Word Cloud*). Metoda działa w ten sposób, że tworzona jest grafika, na której widnieją konkretne słowa, a ich wielkość jest zależna od przyjętej miary, którą najczęściej jest ilość wystąpień. W ten sposób można ukazać w sposób graficzny jakie słowa mogą mieć wpływ na wydźwięk pozytywny lub negatywny. Przykładowa chmura słów została ukazana na rysunku 1.4.

Rozdział 2

Uczenie Maszynowe

Poniższy rozdział poświęcony jest bliższemu przyjrzeniu się algorytmom uczenia maszynowego. Zostaną opisane modelu, które zostały wykorzystane w dalszej części pracy.

2.1 Wektorowa reprezentacja tekstu

Modele uczenia maszynowego przyjmują dane w postaci numerycznej, więc aby wykorzystać je w przypadku przetwarzania języka naturalnego konieczne jest przekonwertowania tekstu do postaci wektorowej. W przypadku gdy do modelu przekazujemy dane tekstowe to w takiej sytuacji traktujemy tekst jako wektor cech. Istnieje kilka sposobów na przedstawienie danych w formie wektorowej, najczęściej stosowany jest model worka słów *(ang.bag-of-words)* oraz technika TFIDF. W pierwszym z nich w utworzonym wektorze przedstawiamy słowa ze względu na ich częstotliwość występowania. Natomiast w drugim skupiamy się na określeniu trafności, która jest wysoka w przypadku gdy słowo występuje często w konkretnym dokumencie, a rzadko w pozostałych.

2.1.1 Bag-of-words

Poprzez model worka słów możemy przedstawić dane tekstowe za pomocą wektorów, zawierających numeryczne informacje, które reprezentują częstość występowania danego słowa. Posiadając zbiór dokumentów, pierwszym krokiem jest dokonanie wstępnego przetworzenia tekstu tj. zastosowanie takich kroków jak ujednolicenie tekstu, tokenizacja, usunięcie stop-wordów itd. Następnie tworzony jest z nich słownik, który zawiera wszystkie unikalne tokeny użyte w zbiorze. Finalnie każdy dokument jest odwzorowany za pomocą wektora cech o długości stworzonego słownika, gdzie każdy

Tablica 2.1: Reprezentacja dokumentów za pomocą modelu Bag-of-words

	Ala	mieć	kot	Robert	chcieć	psa	bawić	się
Tekst 1	1	1	1	0	0	0	0	0
Tekst 2	0	0	1	1	1	1	0	0
Tekst 3	1	0	2	1	0	0	1	1

element wektora odpowiada danemu tokenowi, a wartość jest reprezentowana poprzez ilość wystąpień w przekształcanym dokumencie, gdzie liczba 0 oznacza, że dane słowo nie pojawiło się ani razu, każda liczba naturalna odpowiada ilości wystąpienia danego słowa

Załóżmy, że do modelu Bag-of-words chcemy przekazać 3 dokumenty, których treść prezentuje się następująco:

Tekst 1: *"Ala ma kota"*
Tekst 2: *"Robert chcę psa i kota"*
Tekst 3: *"Kot Ali i kot Roberta bawią się"*

Po wcześniejszej obróbce tekstu z uzyskanych wyżej termów tworzymy słownik, który ostatecznie składa się z danych tokenów:

Słownik: *"Ala; mieć; kot; Robert; chcieć; psa; bawić; się"*

Za pomocą uzyskanego słownika, możemy wyrazić każdy z tekstów przy użyciu wektora, gdzie kolejne elementy odpowiadają liczbie wystąpień danego tokenu w tekście. Reprezentacja tekstu za pomocą wektorów została przedstawiona w tabeli 2.1.

2.1.2 TF-IDF

Kolejną omawianą metodą jest częstość termów - odwrotna częstość dokumentów *(ang. Term Frequency - Inverse Document Frequency)*, najczęściej określana za pomocą skrótu TF-IDF. W przeciwieństwie do modelu worka słów technika TF-IDF nie skupia się wyłącznie na określeniu częstotliwości występowania słów w dokumencie, określa on statystycznie jak ważne jest dany token w dokumencie. Im częściej dane słowo występuje w tekście tym jego znaczenie jest większe, jednak gdy to samo słowo pojawia się w wielu dokumentach niesie ono mało znaczące informacje.

Algorytm składa się z dwóch składowych. W celu określenia wartości ważności konkretnego tokenu za pomocą algorytmu obliczamy częstość termu *(ang. Term Frequency)* oraz odwrotną częstość termu *(ang. Inverse Document Frequency)*. Pierwszy

Tablica 2.2: Reprezentacja dokumentów za pomocą modelu TF-IDF

	Ala	mieć	kot	Robert	chcieć	psa	bawić	się
Tekst 1	0.54783	0.72033	0.42544	0	0	0	0	0
Tekst 2	0	0	0.34520	0.44451	0.58448	0.58448	0	0
Tekst 3	0.35645	0	0.55364	0.35645	0	0	0.46869	0.46869

z nich opisany w równaniu 2.1 jest zliczeniem wystąpień danego tokena w dokumencie podzielonym przez ich sumaryczną ilość całym tekście, co określa częstotliwość występowania danego słowa.

$$\mathrm{TF}_{i,j} = \frac{n_{i,j}}{|d_j|} \tag{2.1}$$

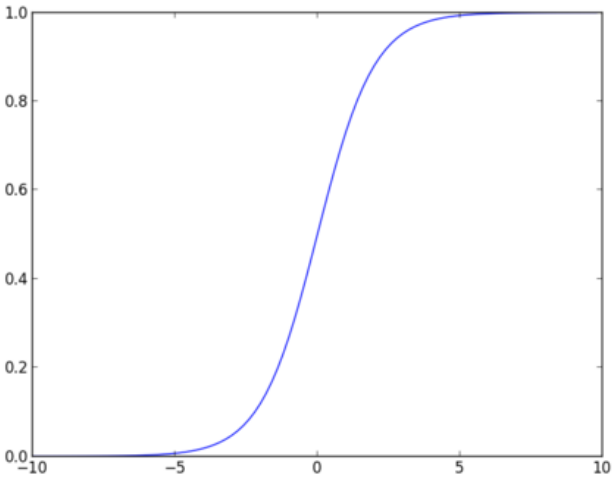
gdzie:
$n_{i,j}$ - reprezentuję ilość wystąpień słowa "i" w dokumencie "j"
$|d_j|$ - reprezentuje ilość wszystkich tokenów w dokumencie "d"

Drugi składnik opisany w równaniu 2.2 obliczany jest za pomocą logarytmu z liczby wszystkich dokumentów podzielonej przez ich ilość, w którym analizowane słowo się znalazło. Dzięki czemu nadaje się mniejszą ważność słowu, które występuje w wielu dokumentach.

$$\mathrm{IDF}_i = log\frac{|D|}{d : x_i \in d} \tag{2.2}$$

gdzie:
$|D|$ - reprezentuję ilość wszystkich dokumentów
$d : x_i \in d$ - reprezentuje ilość dokumentów zawierających rozpatrywany token

Finalną wartość TF-IDF otrzymujemy poprzez przemnożenie ze sobą dwóch wyżej opisanych składowych. Stosując tę technikę otrzymujemy wartości z przedziału od 0 do 1, gdzie im większa wartość tym większą wagę przypisuje się danemu tokenowi. Dokonano obliczeń na przykładowych zdaniach z poprzedniego podrozdziału i umieszczono je w tabeli 2.2.



Rysunek 2.1: Wykres funkcji sigmoidalnej

2.2 Regresja logistyczna

Regresja logistyczna jest modelem klasyfikacji, w którym możemy łatwo ocenić wpływ pojedynczych cech na ostateczną decyzje przypisania obiektu do konkretnej klasy. Jest to możliwe dzięki temu, że podwaliny regresji logistycznej stanowi funkcja liniowa, zaprezentowana we wzorze 2.3.

$$\mathrm{Y} = W_0 + W_1X_1 + W_2X_2...W_iX_i \tag{2.3}$$

gdzie:

X_i - dane wejściowe, w przypadku zadań analizy języka naturalnego są to kolejne wyrazy wektorowej reprezentacji tekstu

W_i - parametry, nazywane również wagami, określają ważność kolejnych danych wejściowych

Regresja logistyczna nosi swoją nazwę od funkcji logistycznej, którą wykorzystuje do przekształcenia wartości rzeczywistych otrzymywanych z funkcji liniowej. Funkcja logistyczna inaczej nazywana funkcją sigmoidalną ma charakterystyczny kształt przedstawiony na rysunku 2.1, który niezależnie od przekazanych wartości przyjmuje wartości od 0 do 1, co często może być interpretowane jako prawdopodobieństwo zajścia jakiegoś zdarzenia. W przypadku klasyfikacji jest to wartość, która determinuje

przynależność do danej klasy, dodatkowo musimy ująć funkcje decyzyjną, która mówi nam jaki ustalić próg powyżej, którego otrzymujemy konkretną klasę. Wzór funkcji sigmoidalnej wygląda następująco: [4]

$$\mathrm{Y} = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-(W_0 + W_1 X_1 + W_2 X_2)}} \tag{2.4}$$

Jak można zauważyć w powyższym wzorze, wykorzystujemy funkcje liniową traktując ją jako argument funkcji sigmoidalnej. Uczenie modelu regresji logistycznej polega na dobraniu prawidłowych wartości parametrów W_i za pomocą metody MLE *(ang. maximum likelihood estimation)*, która przypisuje odpowiednie wartości poprzez uzyskiwanie jak największej wartości dla funkcji prawdopodobieństwa. W przypadku analizy sentymentu gdzie wartość 1 odpowiada klasie pozytywnej, dobiera się takie współczynniki aby wartość wyjściowa była maksymalnie zbliżona do 1. [2]

2.3 Naiwny klasyfikator bayesowski

Jest to model klasyfikacji, który wywodzi się, ze znanego w świecie statystyki, twierdzenia Bayesa, nazwa twierdzenia pochodzi od nazwiska autora Thomasa Bayes'a. Wykorzystuje popularnie nazywane naiwnym, założenie o wzajemnej niezależności paradyktorów, które zakłada, że poszczególne cechy w naszym zbiorze danych są niezależne od siebie. Jest ono nazywane naiwnym, gdyż praktycznie zawsze zostaje ono naruszone, ponieważ zazwyczaj poszczególne cechy są od siebie w jakiś sposób zależne. Jednak to założenie pozwala znacząco uprościć obliczenia, a jego naruszenie nie wpływa aż tak mocno na samo działanie algorytmu. Algorytm bayesowski przeprowadza analizę języka naturalnego nie biorąc pod uwagę kontekstu wypowiedzi w jakim te słowa występują. Skupia się on na wykorzystaniu informacji o liczbie wystąpień słów z pominięciem kolejności, poprzez co stosowany jest do niego model bag-of-words. Pomimo tego algorytm bayesowski okazuje się często dobrym wyborem ze względu na swoją prostotę oraz szybkość działania. [20]

$$\mathrm{P}(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{2.5}$$

gdzie:

$P(A|B)$ - Prawdopodobieństwo warunkowe, tj. prawdopodobieństwo zajścia zdarzenia A pod warunkiem zajścia zdarzenia B

$P(B|A)$ - Prawdopodobieństwo zajścia zdarzenia B pod warunkiem zajścia zdarzenia A

$P(A)\ oraz\ P(B)$ - Prawdopodobieństwo zajścia zdarzenia A oraz zdarzenia B

Klasyfikator wykorzystuje powyższe twierdzenie poprzez podstawienie w miejsce zdarzenia B zmiennej określającej daną klasę, a zdarzenie A zastępowane jest wektorem zmiennych objaśniających co w przypadku analizy języka naturalnego stanowi wektorową reprezentacje tekstu. Stosując powyższe podstawienia otrzymujemy następujący wzór:

$$\mathrm{P}(y|x_1, x_2, x_3...x_n) = \frac{P(x_1, x_2, x_3...x_n|y) * P(y)}{P(x_1, x_2, x_3...x_n)} \tag{2.6}$$

Korzystając z założenia o niezależności paradyktorów możemy przekształcić powyższy wzór do poniższej wersji:

$$\mathrm{P}(y|x_1, x_2, x_3...x_n) = \frac{P(x_1|y) * P(x_2|y) * P(x_3|y)...P(x_n|y)) * P(y)}{P(x_1, x_2, x_3...x_n)} \tag{2.7}$$

Wzór ten można zapisać w bardziej optymalny sposób:

$$\mathrm{P}(y|x_1, x_2, x_3...x_n) = \frac{P(y) * \prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, x_3...x_n)} \tag{2.8}$$

Wykorzystując powyższy wzór algorytm oblicza prawdopodobieństwo przynależności do każdej z możliwych klas. W przypadku analizy sentymentu policzy prawdopodobieństwo osobno dla wydźwięku negatywnego oraz pozytywnego, następnie porównując otrzymane wyniki wybierze klasę z większym prawdopodobieństwem.

Ze względu na to, że będziemy porównywać ze sobą dwa otrzymane wyniki prawdopodobieństwa, możemy zignorować obliczanie wartość $P(x_1, x_2, x_3...x_n)$ ponieważ nie wpływa ona w żadnym stopniu na końcowy rezultat wybrania odpowiedniej klasy.

Obliczenie prawdopodobieństwa wystąpienia danej klasy $P(y)$ jest możliwe na kilka sposobów. Wystarczające przybliżenie tej wartości może być określane jako stosunek ilości dokumentów zawierających daną klasę do sumy wszystkich dokumentów:3

$$\mathrm{P}(y_i) = \frac{Ilo\ dokumentw\ klasy\ y_i}{Ilo\ wszystkich\ dokumentw} \tag{2.9}$$

Prawdopodobieństwo warunkowe $P(x_i|y_j)$ jest wyrażane poprzez stosunek ilości wystąpień tokenu x_i w wektorach, którym przypisana jest klasa y_j do ilości wszystkich tokenów w dokumentach klasy y_j

$$\mathrm{P}(x_i|y_j) = \frac{Liczba\ tokenw\ x_i\ w\ dokumentach\ klasy\ y_j}{Liczba\ wszystkich\ tokenw\ w\ dokumentach\ klasy\ y_j} \tag{2.10}$$

2.4 K-najbliższych sąsiadów

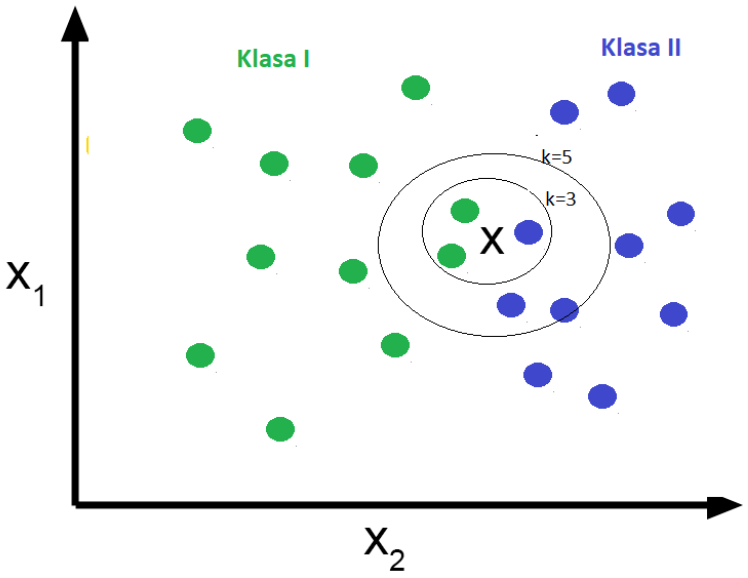
Algorytm k-najbliższych sąsiadów*(ang. k-nearest neighbors)* różni się od pozostałych metod tym, że pominięty jest tutaj etap trenowania. Klasyfikacja nowego przykładu oparta jest na podstawie porównania go do zestawu danych treningowych.

Zasada działania algorytmu polega na ty, że dla nowej instancji przeszukiwany jest zbiór w celu znalezienia najbliższych przykładów(sąsiadów) i na ich podstawie określana jest klasa. Mamy pewien parametr „K", który informuje nas o tym ile sąsiadów bierzemy pod uwagę przy klasyfikacji. Zmienną wyjściową determinuje się na podstawie najczęściej występującej klasy wśród „K" instancji, które bierzemy pod uwagę. Co zostało zaprezentowane na rysunku 2.2. Dla k=3 nowa instancja zostałaby przypisana do klasy I ze względu na przewyższającą ilość reprezentantów tej klasy wśród 3 najbliższych sąsiadów. Dla k=5 zostałaby przypisana klasa II ponieważ tym razem trzech z pięciu sąsiadów należy do tej klasy.

Wyznaczenie najbliższych sąsiadów odbywa się za pomocą miary odległości określaną przez wybraną metrykę. Najbardziej powszechną metodą jest stosowanie odległości Euklidesowej, która równa się pierwiastkowi sumy kwadratów różnic pomiędzy wszystkimi atrybutami dwóch punktów. Podczas obliczeń przy określaniu nowej klasy można pomijać pierwiastek gdyż nie ma on ostatecznie wpływu przy porównywaniu odległości.

$$||x - x^k|| = \sqrt{\sum_{i=1}(x_i - x_i^k)^2} \tag{2.11}$$

W przypadku analizy języka naturalnego do klasyfikacji nowej instancji posługujemy się odległością pomiędzy wektorami, które zostały utworzone za pomocą wektorowej reprezentacji tekstu.

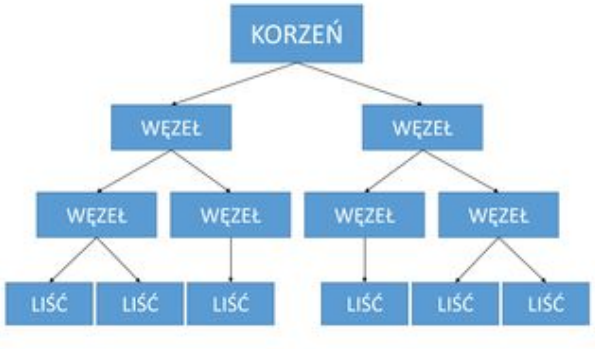


Rysunek 2.2: Klasyfikacja za pomocą algorytmu knn

2.5 Drzewo decyzyjne

Kolejnym modelem uczenia maszynowego, który omówimy są drzewa decyzyjne. Może on służyć zarówno do regresji jak i klasyfikacji. Algorytm można traktować jako przeprowadzanie procesu decyzyjnego, gdzie w kolejnych krokach następuje podział zbioru przykładów na osobne podzbiory według pewnego kryterium dotyczącego cech instancji. Możemy to rozumieć jako ciągłe powtarzanie podziału przestrzeni obiektów na podprzestrzenie, aż do uzyskania podzbiorów spełniających kryterium stopu.

Korzystając z teorii grafów, drzewo decyzyjne możemy opisać jako graf spójny nie zawierający cykli, który jest również grafem skierowanym. Pierwszy z wierzchołków został opisany jako korzeń, zawierający wszystkie próbki treningowe. Wychodząc z korzenia dokonujemy podziału naszych danych na kolejne podzbiory według ustalonego kryterium. Utworzone nowe wierzchołki są nazywane węzłami, gdzie każdy z nich posiada własny lokalny zestaw przykładów stanowiący pewną część wszystkich danych użytych do stworzenia drzewa. Proces ten jest powtarzany, gdzie od nowo powstałych podzbiorów tworzy się kolejne, lokalne zestawy danych. W dalszych węzłach zawierają się tylko instancje spełniające wszystkie kryteria znajdujące się na ścieżce idącej od korzenia do badanego węzła. Proces ten jest powtarzany, aż węzeł nie spełni kryterium stopu. Kiedy zostaje ono spełnione nie powstają już kolejne podzbiory, a wierzchołek nazywany jest liściem, czyli końcowym węzłem w którym dokonywana jest klasyfikacji. W przypadku dokonywana predykcji wszystkie elementy



Rysunek 2.3: Klasyfikacja za pomocą algorytmu knn

tego podzbioru są przypisywane do pewnej klasy. Kryterium stopu są spełnione w następujących przypadkach:

- Gdy wszystkie elementu podzbioru są tej samej klasy.
- Gdy podzbiór zawiera mniejszą ilość przykładów niż minimalna ilość nadana jako hiperparametr.
- Gdy głębokość utworzonego wierzchołka jest równa maksymalnej głebokości drzewa ustalonej jako hiperparametr.
- Gdy nie istnieje kryterium, które dzieli aktualny węzeł na więcej niż jeden niepusty zbiór.

Przykładowa reprezentacja graficzna modelu została zaprezentowana na rysunku 2.3, przedstawia drzewo o głębokości równej cztery, można zauważyć jak w kolejnych krokach wychodząc od korzenia tworzą się nowe wierzchołki nazywane węzłami, a końcowe z nich liśćmi.

Kryterium, którym posługujemy się w momentach stosowania kolejnych podziałów zbioru przykładów jest ustalane na podstawie funkcji, która określa jakie pytanie będzie zadawane. Każdego podziału dokonujemy zadając pytanie tylko na temat jednej z cech instancji. Z uwagi na dużą ilość możliwości dokonywania podziałów konieczna jest ocena każdego z nich i wybór najbardziej optymalnego rozwiązania. Dokonując wyboru najefektywniejszego podziału chcemy uzyskać jak największość czystość otrzymanych podzbiorów, która jest określana na podstawie rozkładu klas w węźle. Czystość zbioru jest największa w momencie gdy w danym węźle znajdują się instancje tylko jednej klasy więc dążymy do tego aby w otrzymywanych podzbiorach uzyskiwać podziały zdominowane przez jedną z klas. [5]

Jednym ze sposobów określenia czystości zbioru jest wyliczenie wskaźnika Gini'ego, który zaprezetowany jest w poniższym wzorze:

$$\mathrm{G} = 1 - \sum_{k=1}^{n} p_k^2 \tag{2.12}$$

gdzie: p_k jest udziałem konkretnej klasy w otrzymanym podzbiorze.
Udział określa się poprzez stosunek ilości danej klasy do sumy wszystkich elementów w tym podziale, więc w przypadku gdy mamy 3 klasy, a w otrzymanym podzbiorze mamy taką samą ilość każdej z tych klas to wskaźnik będzie równy $\frac{1}{3}$. Podział dobieramy w ten sposób aby wskaźnik Gini'ego był jak najmniejszy. [5]

Inną miarą, która może posłużyć do określenia czystości zbioru jest Entropia, która jest określona poniższym wzorem:

$$\mathrm{E}ntropia = -\sum_{k=1}^{n} p_k * log_2\ p_k \tag{2.13}$$

Tak samo jak w przypadku parametru Gini zależy nam na takim dobraniu podziału aby minimalizować wartość entropii, algorytm drzew decyzyjnych dążąc do minimalizacji wartości entropii dąży do tego aby w podziałach były jak najbardziej jednorodne klasy.

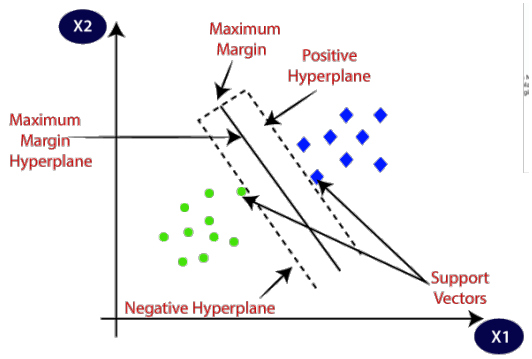
Główną zaletą drzew decyzyjnych jest fakt, że są one bardzo łatwe do interpretowania ze względu na to, że widać cały proces decyzyjny, który został zastosowany, który jest również optymalny do przedstawienia w formie graficznej. Są łatwe do zbudowania, dobrze radzą sobie z dużymi zbiorami danych, a same dane nie wymagają skomplikowanego przygotowania danych.
Jako wady drzew decyzyjnych można wymienić to, że są one skłonne do szybkiego przeuczenia co często stanowi główny problem przy budowania modelu, dlatego bardzo ważne w budowaniu drzew jest dobieranie odpowiednich hiperparametrów, aby tego uniknąć. Dodatkowo drzewa decyzyjne są często bardzo złożonym procesem, który wymaga dużej pamięci obliczeniowej, ze względu na rosnąco wykładniczo ilość możliwych sposobów utworzenia drzew razem z wzrostem ilości atrybutów. Innym problemem z algorytmami budowanymi na bazie drzew jest niestabilność, gdzie nawet niewielka zmiana danych wejściowych może mieć duży wpływ na budowę całego drzewa, dobrym sposobem aby uniknąć tego problemu jest budowania lasów losowych, które poprzez uśrednianie wyników potrafią ograniczyć tą niestabilność. [9]

2.5.1 Lasy losowe

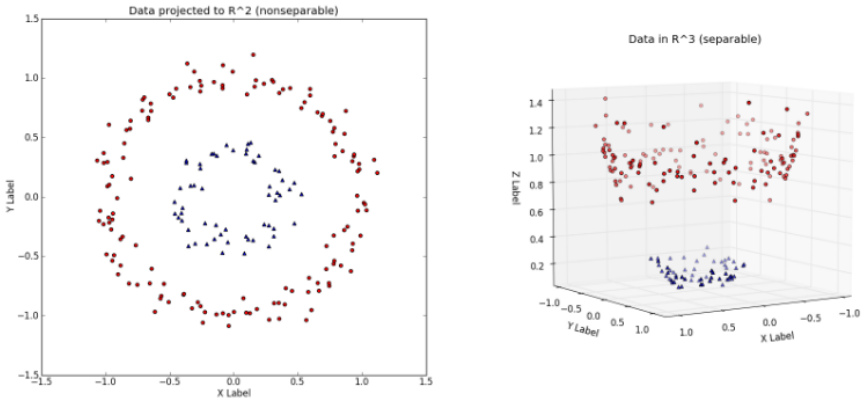
W celu ograniczenia niestabilności, czyli zbytniego dopasowania do danych uczących zamiast drzewa losowego można stosować podejście wielomodelowe, które w tym przypadku nazywane jest algorytmem lasów losowych. Sugerując się nazwą modelu, może nam ona dawać pewien obraz tego jak algorytm działa, mianowicie jest to połączenie wielu drzew decyzyjnych, które łącznie podejmują decyzje co do klasyfikacji. Każde drzewo decyzyjne daje nam swoją propozycje doboru klasy, a co za tym idzie prognoza z największą ilość głosów staje się ostatecznym wynikiem zaproponowanym przez model lasu losowego. Drzewa stosowane w tym algorytmie nie wykorzystują całego zbioru treningowego. Każde z drzew używa losowej próbki z tych danych, dzięki czemu moc obliczeniową konieczną do przeprowadzenia operacji zostaje ograniczona oraz otrzymujemy różne wyniki dla pojedynczych drzew, klasa z największą ilość głosów jest końcową predykcją modelu.

2.6 Maszyna wektorów nośnych

Algorytm maszyny wektorów nośnych *(ang. Support vector machines)* wśród pozostałych algorytmów wyróżnia się mocnym podłożem matematyczny, stara się definiować problem klasyfikacji za pomocą rozwiązywania problemu liniowej separowalności zbioru danych w przestrzeni. Ogólnie algorytm próbuje znaleźć hiperpłaszczyznę, czyli podprzestrzeń $(n-1)$wymiarową dzielącą przestrzeń (n)wymiarową, którą separuje na dwa rozłączne obszary, gdzie w każdym z nich znajdują się tylko i wyłącznie instancje jednej z klas. Tworzy się granica decyzyjna, dzięki której łatwo zdefiniować przynależność nowych przykładów do prawidłowej klasy bazując na lokalizacji w przestrzeni. Do problemu znalezienia współczynników hiperpłaszczyzny można stosować algorytm sekwencyjnej minimalnej optymalizacji *(ang. Sequential Minimal Optimization)*. Odległość między linią wyznaczającą granice, a punktami położnymi najbliżej niej nazywana jest marginesem *(ang. margin)*, który naturalnie wyznacza przestrzeń w kształcie pasa między dwoma klasami, na krańcach tej przestrzeni znajdują się próbki, które wyznaczają tzw. wektory nośne *(ang. support vectors)*, są to punkty kluczowe do konstruowania klasyfikatora. Algorytm stara się wybrać taką hiperpłaszczyznę z maksymalnie dużymi marginesami *(ang. Maximal-Margin hyperplane)*. Przykładowe zobrazowanie działania maszyny wektorów nośnych zostało zaprezentowane na rysunku 2.4, gdzie została wyznaczona hiperpłaszczyzna z maksymalnym marginesem, która rozdziela obiekty na pozytywne oraz negatywne, zostały również zaznaczone wektory nośne. [3]



Rysunek 2.4: Działa algorytmu maszyny wektorów nośnych



Rysunek 2.5: Rzutowanie danych na przestrzeń o większym wymiarze w algorytmie SVM

Teoretycznie aby dobrać odpowiednią hiperpłaszczyznę dane powinny być separowalne liniowo jednak często zbiór, który taki nie jest można poprzez odpowiednią transformacje uczynić zdatnymi dla modelu. Możemy tego dokonać poprzez zwiększeniu rozmiaru danych poprzez zastosowanie funkcji, która dodawałaby kolejny wymiar w przestrzeni poprzez co zwiększają się możliwości rozdzielania zbioru. Zobrazowane to zostało na rysunku 2.5 gdzie widać, że początkowo na rysunku po lewej separacja liniowa jest niemożliwe, a po zrzutowaniu zbioru z przestrzeni dwuwymiarowej na trójwymiarową z łatwością można dokonać podziału. [8]

Często mimo rzutowania danych na przestrzeń o większym wymiarze dane mogą wciąż być nieseparowalne, co powoduje niemożliwość utworzenia podziału poprzez hiperpłaszczyzne, która jednoznaczane rozdzielała by przypadki na dwie klasy. W takim przypadku dobrym sposobem może być dokonanie podziału w sposób niedoskonały, czyli dopuszczając aby kilka obserwacji treningowych zostało celowo umieszczonych po nieprawidłowej stronie granicy, co może rzutować na lepsze dopasowanie się hiperpłaszczyzny do pozostałych danych, co jest nazywane jako łagodny margines *(ang.*

soft margin). Ilość przykładów znajdujących się po nieprawidłowej stronie podziału powinna pozostawać nieduża w porównaniu do pozostałych punktów znajdujących się po prawidłowej stronie granicy. Odpowiedni podział danych jest kontrolowany poprzez zmienną dopełniającą, która przyjmuje wartości od 0 do 1 w przypadku gdy instancja jest zdefiniowana po odpowiedniej stronie, w momencie gdy obiekt znajduje po nieprawidłowej części podziału przypisywana mu jest wartość większa od 1. Wartości zmiennej dopełniające ma wpływ na kontrolowanie rozmiaru marginesów.

2.7 Hiperparametry i parametry

Sposóby analizowania danych wejściowych przez algorytmy uczenia maszynowego działają w zależności od tego jakie ma on dobrane hiperparametry oraz parametry. Dla każdego typu modelu występują różne, unikalne rodzaje parametrów, podyktowane sposobem działania algorytmu. Hiperparametry, różnią się od parametrów tym, że są nadawane przez użytkownika jeszcze przed procesem uczenia, podczas gdy parametry są dostrajane przez model w jego trakcie. Występują również modele nieparametryczne, które działają tylko i wyłącznie na podstawie nadanych hiperparametrów, jednym z przykładów jest model k-najbliższych sąsiadów, do którego przekazujemy tylko informacje o wybranej metryce oraz ilości sąsiadów branych pod uwagę przy klasyfikacji.

Jako, że wydobywanie optymalnych hiperparametrów do modelu jest zadaniem użytkownika w tworzonym projekcie na potrzeby pracy użyłem metody przeszukiwania siatki *(ang. Grid-search)*. Działa ona w ten sposób, że do siatki przekazujemy hiperparametry jako argumenty do konstruktora modelu. Model przeszukiwania siatki daje możliwość, która pozwala przeszukiwać różne, wybrane przez nas, wartości hiperparametrów. W celu maksymalizacji wybranej metryki oceny modelu tworzy się tzw. przesztrzeń hiperparametrów, z której kolejno są testowane wszystkie kombinacje podanych wartości.

Ocena najlepszego zestawu hiperparametrów odbywa się za pomocą walidacji krzyżowej. Jako, że dobór odpowiednich hipeparametrów chcemy dokonać jeszcze na etapie uczenia tj. nie używając danych ze zbioru testowego. Jednym z podejść może być podział naszych danych na trzy zestawu danych: treningowy, walidacyjny, na którym dokonalibyśmy wyboru hiperparametrów oraz testowy. Takie podejście jednak redukuje nam ilość próbek danych, na których chcielibyśmy dokonać procesu treningowego. Lepszym rozwiązaniem jest zastosowanie wyżej wspomnianej walidacji krzyżowej, która polega na podziale naszego zbioru treningowego na wybraną ilość

```
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()
param_grid = {'criterion':['gini', 'entropy']
              "max_depth":np.arange(3,30,5) ,
              'min_samples_leaf': np.arange(1,20,5)}
grid_search = GridSearchCV(classifier, param_grid, cv=10)
```

Rysunek 2.6: Przykładowy wygląd modelu siatki przeszukiwania

mniejszych części, następnie model jest oceniany na jednej z tych części podczas gdy trenowany był na pozostałych próbkach. Proces powtarzany jest w ten sposób aby dokonać oceny na każdej z części zestawu treningowego, a końcowa wartość naszego modelu jest ustalona jako średnia wartość wszystkich uzyskanych wyników.

Przykładowy sposób implementacji siatki przeszukiwania, używając biblioteki sckit-learn, został ukazany na rysunku 2.6, gdzie stworzyliśmy siatkę przeszukiwania dla drzewa decyzyjnego. Jak widać został ustalone różne hiperaparametry do sprawdzenia. Algorytm zostanie sprawdzony dla dwóch różnych kryteriów oceny czystości modelu: Gini oraz Entropia i pięciu różnych wartości zarówno dla maksymalnej głębokości jak i minimalnej ilości próbek na liściu. Dodatkowo został nadany parametr “cv”, który określa na ile mniejszych częsci zostanie podzielony zbiór treningowy podczas walidacji krzyżowej. Jako, że chcemy dokonać oceny modelu na wszystkich możliwych kombinacjach tych hiperparametrów to dla zaprezentowanej na rysunku 2.6 siatki daje nam to sto różnych możliwości zestawiania tych zmiennych, co już stanowi zadanie mocno wymagające obliczeniowo dla komputera, dlatego w celu zmniejszenia ilości czasu poświęconego na uczenie staramy się nie przekazywać zbyt dużej ilości parametrów do siatki.

Rozdział 3

Wykorzystane narzędzia

3.1 Pandas

Rysunek 3.1: Logo oprogramowania Pandas

Pandas to open-source'owa biblioteka pythona, najbardziej popularne narzędzie wykorzystywane do analizy oraz manipulacji danych. Za jego pomocą czyszczenie oraz modyfikacja całych zbiorów stają się dużo łatwiejsze ,daje ona możliwość przeprowadzania tych procesów na całym zbiorze danych za pomocą pojedynczych komend. Uzupełnianie brakujących wartości jest często zautomatyzowane, gdzie łatwo można wypełnić brakujące wartości poprzez zastosowanie jednej z proponowanych reguł np. średniej wartości z pozostałych instancji. Umożliwia wczytywanie danych z plików o różnym formacie, obsługuje najpopularniejsze z nich takie jak: *.csv* i *.json*. W dużej mierze wykorzystuje ona inną biblioteke *NumPy*, która ułatwia szybkie działania na macierzach liczb, przedstawiane za pomocą wielowymiarowych tablic *ndarray*, na podstawie których budowane są najbardziej powszechne struktury zapisu danych w tej bibliotece czyli *DataFrame*. Jak sama nazwa wskazuje dane są przedstawiane w postaci ramki danych, czyli dwuwymiarowej tabelarycznej struktury danych gdzie kolejne kolumny oznaczają cechy, którym przypisywana jest konkretna wartość, mogą one różnić się od siebie typem danych. Wiersze symbolizują kolejne instancje, gdzie każdy kolejny wiersz to osobny przykład, któremu przypisywane są odpowiedni wartości cech wymienionych w kolumnach.

3.2 Scikit-learn

Rysunek 3.2: Logo oprogramowania Scikit-learn

Scikit-learn jest kolejną open-source'owa biblioteka pythona, wykorzystywaną najczęściej do tworzenia modeli uczenia maszynowego, zawiera one wszystkie najpopularniejsze algorytmy oraz różne narzędzie służące do przygotowania danych do modelu. Scikit-learn jest niezwykle łatwą w obsłudze biblioteką, gdzie właściwie cały proces tworzenia, trenowania oraz dokonania predykcji za pomocą modelu uczenia maszynowego można dokonać w kilku linijkach, co zostało zaprezentowane na rysunku 3.3, gdzie w pierwszej i drugiej linijce dokonano wczytania modelu regresji logistycznej, za pomocą metody *.fit()* wytrenowano model na podstawie danych treningowych, a na końcu wykorzystując metodę *.predict()* uzyskano wyniki dla zbioru testowego. Biblioteka została wykorzystana również w procesie opracowywania danych, za jej pomocą dokonano podziału zbioru na treningowy oraz testowy oraz dokonano transformacji tekstu do jego wektorowej postaci metodą TFIDF. Oprogramowanie posiada również funkcje wspomagające proces ewaluacji modelu, w projekcie magisterskim dzięki tej bibliotece stworzono macierz pomyłek dla każdego modelu oraz wyliczono wartość dokładności na podstawie, której wybrane zostały najlepsze modele.

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()
classifier.fit(train_x,train_y)
pred_y = classifier.predict(test_x)
```

Rysunek 3.3: Implementacja modelu uczenia maszynowego w Scikit-learn

3.3 NLTK

NLTK jest następną biblioteka wykorzystywaną w projekcie, która również należy do open-source'owych bibliotek pythona. Rozwijając skrót nazwy otrzymujemy *Natural Language Toolkit* co można przetłumaczyć jako zestaw narzędzi do naturalnego języka

Rysunek 3.4: Logo oprogramowania NLTK

i tym właśnie jest to oprogramowanie, służy do pracy z danymi w postaci języka naturalnego, jest to jedna z największych bibliotek wspomagających NLP. W projekcie magisterskim została wykorzystana do procesu przygotowania danych. Za jej pomocą dokonano tokenizacji tekstu do pojedynczych wyrazów, usunięto słowa najbardziej powszechne nie wpływające na ogólne nacechowanie wypowiedzi (tzw. *stopwords)*, przeprowadzono proces lematyzacji, czyli sprowadzenie słowa do jego podstawowej wersji (bezokolicznik dla czasowników, mianownik liczby pojedynczej dla rzeczowników). Za pomocą biblioteki można również dokonać klasyfikacji tekstu za pomocą niektórych modeli uczenia maszynowego.

3.4 Matplotlib i WordCloud

Rysunek 3.5: Logo oprogramowania Matplotlib

Następne dwie dwie biblioteki użyte w projekcie wspomagają procesy tworzenia graficznej reprezentacji danych. Pierwsza z nich, czyli Matplotlib jest najczęściej stosowana do tworzenia wszelkiego rodzaju wykresów wykorzystując dane w postaci DataFrame, które otrzymywano dzięki bibliotece pandas. W projekcie magisterskim wykorzystano ją do graficznego przedstawienia macierzy pomyłek, wspomagając się jednocześnie biblioteką Scikit-learn. WordCloud jest niedużą biblioteką pythona służącą głównie do przedstawienia danych tekstowych za pomocą chmury słów, przykładowa grafika została ukazana w początkowych rozdziała na rysunku 1.4, gdzie wielkość słowa jest uzależniona od częstości jego występowania w tekście przekazywanym do funkcji tworzącej chmurę.

Rysunek 3.6: Logo frameworka Django

3.5 Django

Django jest to darmowy i open-source'owy framework napisany w języku P python, służący do tworzenia stron internetowych, sprawia że dzięki swoim zbiorem narzędzi tworzenie aplikacji staje się szybkie i proste. Tworzenie różnych stron internetowych często wymaga powtarzania wykonywania tych samych kroków dla każdego projektu tj. stworzenie systemy uwierzytelniania, formularzy do tworzenia obiektów, systemu przypominania i zmiany hasła itd. Z tego powodu został stworzony framework Django, który dostarcza odpowiednich narzędzi, które sprawiają, że większość tych rzeczy jest szybka i łatwa do wykonania. W projekcie magisterskim używając tego frameworka utworzono stronę internatową, gdzie można dokonać klasyfikacji wprowadzonego tekstu, która zostaje dokonana na podstawie najlepiej działających algorytmów uczenia maszynowego.

Rozdział 4

Projekt magisterski

4.1 Cel projektu

Celem projektu było stworzenie kilku modeli opartych na różnych algorytmach uczenia maszynowego dokonujących analizy sentymentu w języku angielskim. Modele były uczone na podstawie etykietowanego zbioru recenzji filmowych. Następnie z pośród stworzonych modeli wybrano najlepsze z nich, które zostały zastosowane w stworzonej aplikacji internetowej, której za zadanie ma klasyfikować wprowadzony tekst przez użytkownika jako negatywny lub pozytywny.

4.2 Zbiór danych

Dane zostały uzyskane z platformy *www.kaggle.com*, która daje mnóstwo możliwości ludziom zajmujących się tematyką Data Science i uczenia maszynowego, jedną z nich jest szansa na zdobycie wysokiej jakości danych, które są prawie gotowe do wykorzystania w projekcie. Pozyskano z tej platformy duży zbiór danych zawierających etykietowane recenzje filmowy pochodzącej z IMDB. Zbiór zawiera 25000 recenzji, gdzie dokładnie połowa z nich jest sklasyfikowana jako pozytywna, a pozostała część jako negatywna, co daje nam po 12500 recenzji dla każdej z klas. Średnia liczba symboli dla każdej recenzji wynosi 1305 znaków, a średnia liczba słów wynosi 234 dokładnie wyrazów. Sumując ze sobą te liczby wychodzi nam, że tworzone przez nas algorytmy przepuszczą przez siebie prawie sześć miliona słów, zestawiając to z faktem że np. Pismo Święte posiada około miliona słów, daje nam to naprawdę ogromną wartość.

4.3 Proces przygotowania danych

Dane zostały wczytane dzięki bibliotece pandas do obiektu DataFrame, dzięki któremu ułatwione jest przeprowadzanie procesów modyfikacji danych na całej strukturze danych.

4.3.1 Usunięcie znaków specjalnych

Chcąc dokonać analizy na podstawie danych tekstowych konieczne jest usunięcie znaków specjalnych oraz liczb, które nie mają wpływu na wydźwięk emocjonalny wypowiedzi i często stanowią pewną pozostałość po procesie wydobycia danych. Dokonano tego posługując się wbudowanym modułem *re* aby dopasować odpowiednie wyrażenie regularne, dzięki którym pozbyliśmy niechcianych symboli.

```
# removing special characters and numbers
df['text'] = df['text'].replace(r'[^A-Za-z ]+', '', regex=True)
```

Rysunek 4.1: Usunięcia znaków specjalnych

4.3.2 Tokenizacja

W związku z tym, że początkowo dysponujemy monolitycznym tekstem w pierwszym kroku przygotowania danych konieczne jest podzielenie go na pojedyncze obiekty zwane tokenami, dzięki czemu model będzie w stanie przyjąć je w takiej formie. Uzyskane fragmenty tekstu są dzielone poprzez narzucone kryterium, czasami tokeny mogą stanowić fragmenty oddzielane znakami interpunkcyjnymi, jednak najbardziej powszechnym sposobem, zastosowanym również w tym projekcie, jest podział tekstu ze względu na białe znaki dzięki czemu jako tokeny otrzymujemy pojedyncze wyrazy. W projekcie tokenizacji dokonana za pomocą wbudowanego pakietu biblioteki NLTK.

```
# tokenize texts
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')

df['tokens'] = df['text'].apply(word_tokenize)
df.head()
```

Rysunek 4.2: Tokenizacja

4.3.3 Usunięcie stopwords'ów

W każdym języku występują słowa, które są bardzo powszechne i pojawiają się w każdym rodzaju tekstu, same w sobie nie niosą ze sobą żadnej wartości informacyjnej ani ładunku emocjonalnego. Chcąc ograniczyć wielkość wektorów przekazywanych do modelu oraz zmniejszyć wpływ zaszumienia spowodowany występowaniem stopwords'ów chcemy je usunąć z naszego zbioru danych. Dokonujemy tego pobierając z korpusu biblioteki *NLTK* listę niechcianych słów, a potem przechodząc iteracyjne przez wszystkie tokeny usuwamy te, które zdefiniowane są jako stopwrds'y.

```
#stop words
from nltk.corpus import stopwords
nltk.download('stopwords')

stop = stopwords.words('english')
df['stop'] = df['tokens'].apply(lambda x: [word for word in x if (word not in stop)])

df.head()
```

Rysunek 4.3: Usunięcia stopswords'ów

4.3.4 Lematyzacja

Do modelu dane będziemy przekazywać w formie wektora, który uwzględnia częstotliwość występowania danego słowa w dokumencie. Stosując lematyzacje sprowadzamy wszystkie wyrazy do formy podstawowej tego słowa, dzięki czemu słowa o takim samym znaczeniu ale różnej odmiane będą sprowadzane do tej samej formy i traktowane jako to samo wyrażenie. Dokonujemy dzięki bibliotece *NLTK*, która dostarcza nam metodę *lemmatize* stosowaną iteracyjnie na każdym z wyrazów z wszystkich dokumentów.

```
# lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

lemon = WordNetLemmatizer()
df['lemon'] = ''

for ind,text in enumerate(df['stop'].values):
  df['lemon'][ind]  = [lemon.lemmatize(t) for t in text]
```

Rysunek 4.4: Lematyzacja

4.3.5 Wektorowa reprezentacja tekstu

Kolejnym etapem jest sprowadzenie otrzymanego zbioru danych do postaci wektorowej, której idea szerzej została opisana w rozdziale drugim. W projekcie zdecydowano się na zastosowanie techniki TFIDF, używając bibliotekę scikit-learn oraz wbudowaną metodę *TfidfVectorizer*.

```
# TFIDF
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features = 1000,  max_df = 0.4,
                             lowercase=False, preprocessor=' '.join)

X = vectorizer.fit_transform(df['lemon']).toarray()
```

Rysunek 4.5: Reprezentacja tekstu za pomocą TFIDF

4.3.6 Podział danych

Po dokonanym procesie uczenia modelu konieczne jest dokonanie sprawdzenie skuteczności otrzymanego modelu, żeby tego dokonać konieczne jest podzielenie danych na dwa zbiory: treningowy oraz testowy. Całego procesu uczenia i dopasowywania parametrów modelu dokonuje się na zbiorze treningowym, żeby potem moć dokonać sprawdzenia skuteczności modelu na danych, których algorytm nie widział wcześniej. W ten sposób chronimy nasz model przed przetrenowaniem oraz zbytnim dopasowaniu się do danych uczących. W projekcie podziału dokonano za pomocą biblioteki scikit-learn oraz wbudowanej metodzie *model selection*, za pomocą której podzielono dane

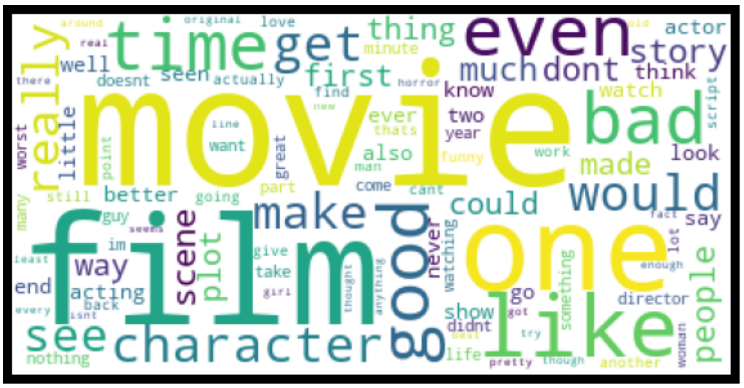
```
# Diving dataset into train and test sets
from sklearn import model_selection, preprocessing

train_x, test_x, train_y, test_y = model_selection.train_test_split(X, df['sentiment'])
```

Rysunek 4.6: Podział danych na dwa zbiór treningowy i testowy

4.3.7 Wizualizacja danych

Przed przekazaniem przygotowanych danych dalej do procesu trenowania modelu warto aktualny stan danych zwizualizować. Forma chmury słów zapewnia intuicyjny dla każdego człowieka sposób na zademonstrowanie danych i wyróżnienie najistotniejszych elementów, które bardziej przykuwają wzrok ze względu na ich większy

rozmiar. W projekcie wykonano dwie chmury słów, osobną dla recenzji pozytywnych, a drugą dla negatywnych. Dzięki rozdzieleniu dwóch typów recenzji można zauważyć porównując ze sobą dwie chmury, które słowa miały wpływ na nadanie pozytywnego lub negatywnego ładunku emocjonalnego. Chmury słów utworzono uwzględniając sto najczęściej pojawiających się wyrazów, gdzie ich wielkość na grafice była zależna od częstości wystąpień danego słowa we wszystkich dokumentach jednej z klas.



Rysunek 4.7: Chmura słów dla recenzji pozytywnych

Rysunek 4.8: Chmura słów dla recenzji negatywnych

4.4 Dobieranie hiperparametrów

W celu uzyskania jak najlepszych algorytmów konieczne jest nadanie im możliwie najbardziej optymalnych i dostosowanych hiperparametrów, które ukierunkują proces trenowania modelu. W celu uzyskania najbardziej dopasowanych hipeparametrów konieczne jest przetestowanie algorytmu na conajmniej kilku różnych zestawach ich wartości. W projekcie posłużono się metodą przeszukiwania siatki *(ang. Grid-search)*, która jest dostarczona przez biblioteke *sckit-learn*. Dzięki wykorzystaniu

Bibliografia

[1] Khurshid Ahmad. *Affective computing and sentiment analysis: Emotion, metaphor and terminology*, volume 45. Springer Science & Business Media, 2011.

[2] Jason Brownlee. *Machine learning algorithms from scratch with Python*. Machine Learning Mastery, 2016.

[3] Jason Brownlee. *Machine learning algorithms from scratch with Python*. Machine Learning Mastery, 2016.

[4] Jason Brownlee. *Machine Learning Mastery with Weka: Analyze Data, Develop Models, and Work Through Projects*. Machine Learning Mastery, 2016.

[5] Jason Brownlee. *Machine Learning Mastery with Weka: Analyze Data, Develop Models, and Work Through Projects*. Machine Learning Mastery, 2016.

[6] Aleksander Buczyński and Aleksander Wawer. Automated classification of product review sentiments in polish. *Intelligent Information Systems*, pages 213–217, 2008.

[7] Mukund Deshpande and Avik Sarkar. Bi and sentiment analysis. *Business Intelligence Journal*, 15(2):41–49, 2010.

[8] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[9] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[10] Ashish Katrekar and Big Data Analytics AVP. An introduction to sentiment analysis. *GlobalLogic Inc*, 2005.

[11] Bing Liu. Sentiment analysis: A multi-faceted problem. *IEEE Intelligent Systems*, 25(3):76–80, 2010.

[12] Bing Liu. *Aspect and Entity Extraction*, page 137–188. Cambridge University Press, 2015.

[13] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

[14] Andrius Mudinas, Dell Zhang, and Mark Levene. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the first international workshop on issues of sentiment discovery and opinion mining*, pages 1–8, 2012.

[15] Mika V. Mäntylä, Daniel Graziotin, and Miikka Kuutila. The evolution of sentiment analysis—a review of research topics, venues, and top cited papers. *Computer Science Review*, 27:16–32, 2018.

[16] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis (foundations and trends (r) in information retrieval), 2008.

[17] Łukasz Piątkowski. Zastosowanie analiz korpusowych w badaniach leksykogramatycznych: perspektywa porównawcza niemiecko-polska. *Zeszyty Naukowe Towarzystwa Doktorantów Uniwersytetu Jagiellońskiego. Nauki Humanistyczne*, (21 (2)), 2018.

[18] Filipe N Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5(1):1–29, 2016.

[19] R Stagner. The cross-out technique as a method in public opinion analysis. 11, 1940.

[20] Krzysztof Tomanek et al. Analiza sentymentu–metoda analizy danych jakościowych. przykład zastosowania oraz ewaluacja słownika rid i metody klasyfikacji bayesa w analizie dan. *Przegląd Socjologii Jakościowej*, 10(2):118–136, 2014.

[21] Janyce Marbury. Wiebe. Recognizing subjective sentences: A computational investigation of narrative text. 1990.

[22] Katarzyna Wójcik and Janusz Tuchowski. Wykorzystanie metody opartej na wzorcach w automatycznej analizie opinii konsumenckich. *Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu*, (385):314–324, 2015.