Uniwersytet im. Adama Mickiewicza w Poznaniu

Wydział Matematyki i Informatyki

Informatyka

Analiza wydźwięku komentarzy studentów do wykładu

Sentiment Analysis of Students' Comments on Lectures

Praca dyplomowa magisterska

Kinga Agnieszka Kramer

Numer albumu: 429412

Promotor:

prof. UAM dr hab. Krzysztof Jassem

Zakład Przetwarzania Języka Naturalnego

Poznań, 2018

Poznań, dnia ……..…

(data)

OŚWIADCZENIE

Ja, niżej podpisana *Kinga Agnieszka Kramer*, studentka Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt.: *Analiza wydźwięku komentarzy studentów do wykładu* napisałam samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałam z pomocy innych osób, a w szczególności nie zlecałam opracowania rozprawy lub jej części innym osobom, ani nie odpisywałam tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w wersji drukowanej jest całkowicie zgodny z egzemplarzem pracy dyplomowej w wersji elektronicznej. Jednocześnie przyjmuję do wiadomości, że przypisanie sobie, w pracy dyplomowej, autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego stanowi podstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego.

[ ]* - wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum UAM

[ ]* - wyrażam zgodę na udostępnianie mojej pracy w zakresie koniecznym do ochrony mojego prawa do autorstwa lub praw osób trzecich

*Należy wpisać TAK w przypadku wyrażenia zgody na udostępnianie pracy w czytelni Archiwum UAM, NIE w przypadku braku zgody. Niewypełnienie pola oznacza brak zgody na udostępnianie pracy.

…….....

czytelny podpis studenta

Streszczenie

Celem niniejszej pracy magisterskiej jest opis autorskiego narzędzia do automatycznej klasyfikacji komentarzy studentów do wykładu przy wykorzystaniu wybranych metod uczenia maszynowego. Początkowe rozdziały prezentują wiedzę teoretyczną z zakresu przetwarzania języka naturalnego oraz uczenia maszynowego. Wprowadzają również w temat analizy wydźwięku. Kolejny rozdział omawia wybrane prace i eksperymenty dotyczące analizy wydźwięku. Następnie opisane są narzędzia, które zostały wykorzystane w projekcie. Ostatni rozdział przedstawia rezultaty osiągnięte w ramach projektu – zaimplementowane metody klasyfikacji wraz z ich ewaluacją.

Słowa kluczowe: analiza wydźwięku, klasyfikacja, uczenie maszynowe, przetwarzanie języka naturalnego.

Abstract

The aim of this master thesis is to describe a tool, designed by this author, for automatic classification of students' comments to lectures using selected machine learning methods. The initial chapters present theoretical knowledge in the field of natural language processing and machine learning. They also introduce the topic of the sentiment analysis. The next chapter discusses selected works and experiments on the sentiment analysis. Then, the tools used in the project are described in detail. The last chapter presents the results achieved in the project – the implemented classification methods and their evaluation.

Key words: sentiment analysis, classification, machine learning, natural language processing.

Spis treści

1. Wstęp

Analiza wydźwięku wykorzystywana jest w wielu aspektach codziennego życia. Zajmuje się problemem analizy tekstów, takich jak posty, recenzje i komentarze, które są umieszczane przez użytkowników na forach dyskusyjnych, blogach i w mediach społecznościowych. Każdego dnia w Internecie pojawiają się ogromne zbiory wypowiedzi użytkowników, których ręczna analiza wymaga bardzo dużych nakładów pracy człowieka. W takich sytuacja bardzo przydatna staje się automatyczna analiza wydźwięku, która w łatwy sposób pomaga firmom zrozumieć nastawienie konsumentów w stosunku do ich marki, produktu lub usługi.

Komentarze stanowią szczególny rodzaj dokumentów ze względu na swoją zawartość – ich głównym celem nie jest przekazanie obiektywnej informacji, ale subiektywnego nastawienia autora tekstu do produktu, usługi lub wydarzenia. Każdy człowiek ocenia zjawiska, produkty, czy usługi w bardzo indywidualny sposób, ponieważ każdy inaczej postrzega otaczającą go rzeczywistość. Ze względu na subiektywność komentarzy, analiza wydźwięku staje się trudnym zagadnieniem przetwarzania języka naturalnego.

Najbardziej popularnym zastosowaniem analizy wydźwięku jest klasyfikowanie tekstu do jednej z trzech klas – pozytywny, negatywny lub neutralny. Coraz częściej stosuje się również klasyfikację wieloklasową, przyznając tekstom wcześniej określone etykiety liczbowe lub słowne. Dzięki narzędziom przetwarzania języka naturalnego oraz uczenia maszynowego możliwe jest stworzenie systemu, który będzie automatycznie klasyfikować wypowiedzi do ustalonych klas.

W niniejszej pracy została przeprowadzona dyskusja na temat analizy wydźwięku oraz wybranych metod uczenia maszynowego. Opisany został proces automatycznej klasyfikacji komentarzy wykorzystujący następujące metody uczenia maszynowego: naiwny klasyfikator Bayesa, k-najbliższych sąsiadów, regresja logistyczna, maszyna wektorów nośnych oraz sieci neuronowe.

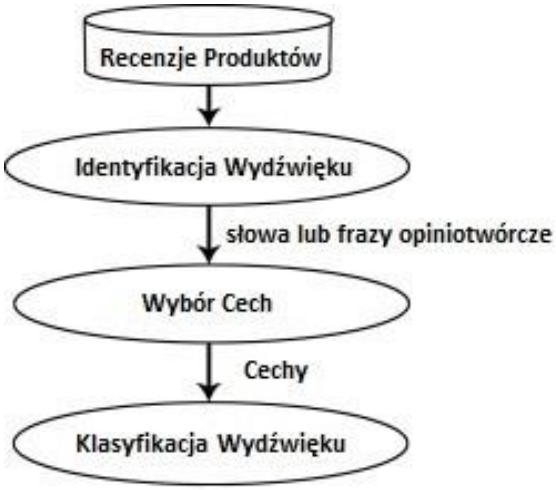
W ramach projektu magisterskiego zgromadzono korpus komentarzy wykorzystany do trenowania i testowania klasyfikatorów. Przeprowadzono serię eksperymentów mających na celu porównanie różnych metod uczenia maszynowego. Uzyskany efekt końcowy pracy stanowi narzędzie automatycznie klasyfikujące komentarze.

2. Wprowadzenie do analizy wydźwięku

2.1. Analiza wydźwięku

Analiza wydźwięku (ang. *sentiment analysis*), nazywana również eksploracją opinii (ang. *opinion mining*), jest dziedziną badań, która analizuje opinie, odczucia, oceny, postawy i emocje ludzi w stosunku do takich podmiotów, jak produkty, usługi, organizacje, osoby, problemy, zdarzenia, tematy i ich atrybuty [LIU, 2012]. Ma na celu określenie postawy mówiącego lub autora tekstu w odniesieniu do jakiegoś tematu lub emocjonalnej reakcji na dane wydarzenie. Celem analizy wydźwięku jest identyfikowanie uczuć zawartych w opiniach, a następnie klasyfikowanie tych opinii ze względu na pojawiające się w nich słowa nacechowane emocjonalnie [MEDH, 2014].

Przykładowy proces analizy wydźwięku został pokazany na rysunku 2.1. W poniższym przykładzie zbiór danych stanowią recenzje produktów. Pierwszym etapem jest identyfikacja wydźwięku, czyli wybór słów lub fraz opiniotwórczych mogących zawierać wydźwięk. Wybrane słowa i frazy stanowią cechy, będące danymi wejściowymi. Ostatni etap stanowi klasyfikacja wydźwięku, która prowadzi do określenia wypowiedzi jako pozytywnej, negatywnej lub neutralnej.



Rys. 2. 1. Proces analizy wydźwięku na przykładzie recenzji produktów. Na podstawie [1].

Analiza wydźwięku może opierać się na dwóch różnych założeniach. Po pierwsze, niektóre słowa wyrażają emocje. Po drugie, istnieją słowa, których wypowiedzenie może wywołać emocje. Zatem analiza wydźwięku z jednej strony wskazuje na stan emocjonalny autora wypowiedzi, natomiast z drugiej, może służyć określeniu

emocjonalnego efektu, jaki dana wypowiedź wywołuje [TOMA, 2014]. W niniejszej pracy skupiono się na pierwszym z tych założeń.

Badając analizę wydźwięku sprawdza się nastawienie autora tekstu w całej jego wypowiedzi lub w jej poszczególnych zdaniach. Można również dokonać analizy fragmentu wypowiedzi odnoszącego się do konkretnego obiektu lub pojęcia. Automatyczna analiza wydźwięku jest przydatna, gdy posiadamy bardzo dużą liczbę tekstów, których wydźwięk chcemy poznać, a dostępne teksty nie są otagowane lub istniejące tagi nie są wystarczające [JURA, 2012].

Mnogość aplikacji i narzędzi do analizy tekstu pozwala na wykorzystanie statystycznego podejścia do analizy wydźwięku. Dostępne narzędzia otwartego oprogramowania wykorzystują uczenie maszynowe, metody statystyczne oraz techniki przetwarzania języka naturalnego w celu zautomatyzowania analizy wydźwięku na dużych kolekcjach tekstów, takich jak strony internetowe, grupy dyskusyjne, recenzje online, blogi oraz media społecznościowe [XIAN, 2017].

W procesie analizy wydźwięku można skupić się na kilku aspektach wypowiedzi, które dotyczą nastawienia, czyli stosunku autora tekstu w odniesieniu do jakiegoś tematu. Do aspektów tych należą: właściciel nastawienia, cel nastawienia, typ nastawienia – zestaw typów (np. lubić, kochać, nienawidzić itd.) lub określenie, czy dana wypowiedź jest pozytywna, negatywna, lub neutralna. Kolejnym aspektem jest również forma tekstu zawierającego nastawienie – zdanie lub cały dokument [JURA, 2012].

Analizę wydźwięku bada się głównie na trzech poziomach. Pierwszym z nich jest poziom dokumentu (ang. *document level*). Zadanie na tym poziomie polega na sklasyfikowaniu, czy cały dokument zawiera pozytywne lub negatywne nastawienie. Drugim poziomem jest poziom zdania (ang. *sentence level*), na którym określa się czy dane zdanie zawiera opinię pozytywną, negatywną, czy neutralną. Neutralny wydźwięk może oznaczać brak opinii. Ostatni poziom stanowi poziom jednostki i aspektu (ang. *entity and aspect level*). Dana osoba może wyrazić różne opinie na temat różnych aspektów tego samego podmiotu np. *Wykład był prowadzony w nudny sposób, ale slajdy bardzo mi się podobały* [LIU, 2012].

Wyróżniamy kilka stopni trudności analizy. Do najprostszych zadań należy sprawdzenie, czy nastawienie osoby piszącej dany tekst jest negatywne, czy pozytywne.

Trudniejsze zadanie stanowi ocena nastawienia tekstu w ustalonej skali, np. od 1 do 10. Bardzo zaawansowane analizy polegają na wykryciu źródła, celu lub złożonego typu postawy [JURA, 2012].

2.2. Podejścia

W analizie wydźwięku wyróżnia się dwa odmienne podejścia do zagadnienia klasyfikacji tekstów. Pierwszym z nich jest podejście wykorzystujące metody słownikowe. Metody te mogą być oparte na zbiorach słów opiniujących lub na całych korpusach tekstów. Biorą pod uwagę występowanie słów i wyrażeń kluczowych. Analizie poddaje się zawartość tekstów i wypowiedzi, wybierając istotne słowa i frazy. Zakłada się, że istnieją pewne słowa, które ludzie często używają do wyrażania silnych uczuć. Wystarczy zatem stworzyć listę takich słów i polegać wyłącznie na nich, aby sklasyfikować dany tekst [LEE, 2002]. Podejście wykorzystujące metody słownikowe wykorzystuje znaczenia analizowanych wyrazów i bierze pod uwagę reguły leksykalne danego języka. W takim podejściu niezbędna jest znajomość gramatyki danego języka oraz specyfiki wypowiedzi związanej ze stosowanym słownictwem – podobieństw i różnic znaczeń słów oraz zasób słów danego języka [TOMA, 2014].

Drugie podejście natomiast wykorzystuje metody statystyczne. W tym przypadku korzysta się z technik traktujących tekst jako obiekt. Charakteryzuje się go za pomocą danych ilościowych, które odzwierciedlają zawartość dokumentu np. liczbę słów lub fraz. Traktowany w ten sposób tekst–obiekt reprezentowany jest w postaci wektora w wielowymiarowej przestrzeni wyznaczonej przez zbiór cech (ang. *features*) opisujących dokumenty. Wybór różnych cech może prowadzić do odmiennych wyników [TOMA, 2014].

Podejście do analizy wydźwięku z wykorzystaniem metod statystycznych składa się z trzech etapów: tokenizacji, czyli odpowiedniego podziału tekstu, ekstrakcji cech, czyli dokonania wyboru cech, na podstawie których zostanie wytrenowany klasyfikator oraz etapu klasyfikacji.

Podczas procesu tokenizacji, jeśli teksty pochodzą ze stron internetowych, należy znaleźć sposób przetwarzania znaczników HTML i XML. Należy zwracać uwagę na znaczniki występujące na portalach społecznościowych (nazwy, hasztagi) oraz na

zdania pisane w całości wielkimi literami, które mogą oznaczać "krzyczenie". Bardzo ważną właściwością tekstu są również występujące emotikony.

Kolejnym krokiem podczas klasyfikacji wydźwięku jest wyodrębnienie i wybranie odpowiednich cech tekstu. Metody wyboru cech można podzielić na metody oparte o słowniki, które wymagają pracy człowieka oraz metody statystyczne, które są w pełni automatyczne. Częściej używanym podejściem jest wykorzystanie metod statystycznych.

W trakcie analizy wydźwięku należy zwrócić uwagę na:

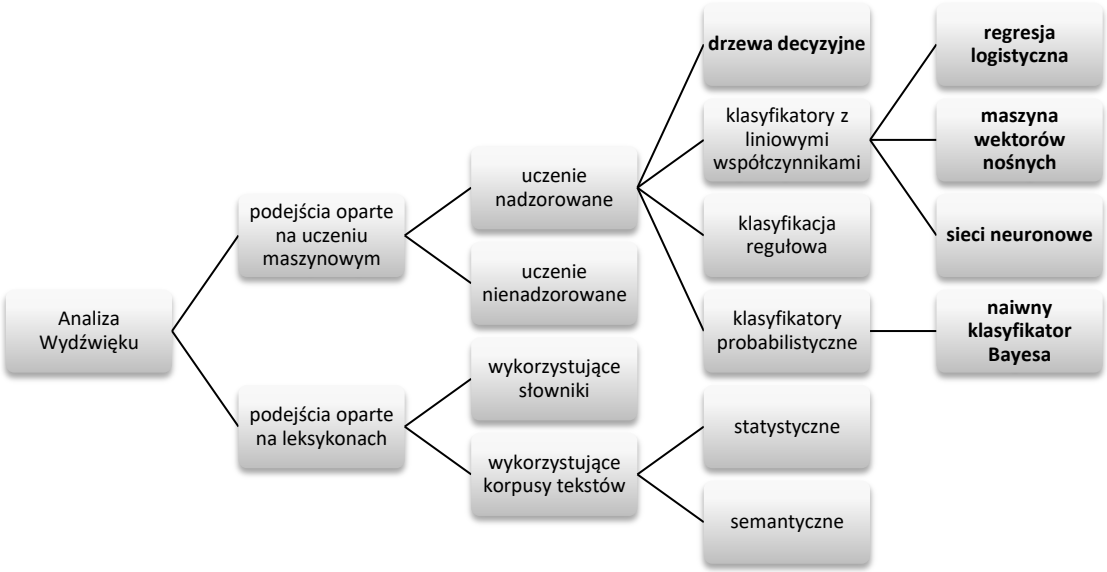
- częstość występowania słów lub sekwencji słów,
- współwystępowanie słów,
- części mowy – np. przymiotniki stanowią ważną część wyrażanej opinii,
- słowa i frazy opiniujące,
- negacje,
- wykrzykniki,
- słowa pisane wielkimi literami,
- akronimy,
- emotikony [MEDH, 2014].

Podczas procesu klasyfikacji używa się różnych metod uczenia maszynowego np.:

- naiwny klasyfikator Bayesowski (ang. *naive Bayes classifier*),
- regresja logistyczna (ang. *logistic regression*),
- maszyna wektorów nośnych (ang. *support vector machine*),
- sieci neuronowe (ang. *neural networks*),
- drzewa decyzyjne (ang. *decision tree classifiers*) [MEDH, 2014].

Powyższe klasyfikatory zostały omówione szczegółowo w rozdziale 3.1.

Przykładowe podejścia do analizy wydźwięku przedstawiono na rysunku 2.2.



Rys. 2. 2. Podejścia do analizy wydźwięku. Na podstawie [2].

2.3. Problemy

Do problemów występujących podczas analizy wydźwięku zaliczamy:

- subtelność – brak negatywnych słów wypowiedzianych wprost np. *Wykład był bardzo ciekawy, ale drugi raz bym nie na niego nie poszedł.*
- wystąpienie wielu słów pozytywnych w tekście o wydźwięku negatywnym np. *Spodziewałem się świetnego wykładu, ciekawej tematyki i interesujących przykładów, ale było to najnudniejsze 1,5 godziny mojego życia.*
- kontekst wypowiedzi – dane słowo w zależności od kontekstu może mieć wydźwięk pozytywny lub negatywny np.
    - *Bardzo podobał mi się dzisiejszy wykład*.
    - *Bardzo podobało mi się to, że prowadzący skończył szybciej wykład*.

  Pierwsza wypowiedź wydaje się być wypowiedzią pozytywną. Druga natomiast sugeruje, że wykład mógł być nudny, dlatego jego autor cieszy się ze skrócenia wykładu, co wydaje się być opinią negatywną.
- zdania, w których występują słowa zawierające wydźwięk, mogą nie wyrażać opinii np. *Czy ktoś mógłby polecić mi dobrą i łatwą książkę, żeby poszerzyć wiedzę z wykładu?*
- sarkazm np. *Wspaniały wykład! W końcu mogłem się wyspać.*

- posiadanie wydźwięku pomimo braku słów zawierających opinię np. *Ten wykład trwał stanowczo zbyt długo* [LIU, 2012].

2.4. Przykłady zastosowań

Analiza wydźwięku wykorzystywana jest w wielu aspektach codziennego życia. W ostatnich latach nastąpił dynamiczny wzrost znaczenia opinii, szczególnie w mediach społecznościowych. Zbieranie opinii z Internetu stało się jednym z ważniejszych elementów badań nad opinią publiczną. Socjolodzy badają opinie i reakcje na różne tematy, przeglądając blogi, fora dyskusyjne i grupy w mediach społecznościowych.

Opinie konsumenckie stanowią bardzo ważny element strategii marketingowych przedsiębiorstw. Koncerny sprawdzają, co klienci myślą na temat ich produktów lub usług oraz co sądzą o konkurencji. Badają również opinie na temat nowych produktów, które planują wprowadzić na rynek. Z pomocą automatycznej analizy wydźwięku można sprawdzać np. czy treści wiadomości e-mail otrzymane od klientów świadczą o ich zadowoleniu, czy niezadowoleniu z danego produktu.

Kolejnym zastosowaniem analizy wydźwięku jest badanie nastawienia wyborców w stosunku do konkretnego kandydata, partii politycznej lub w stosunku do danej reformy. Za pomocą analizy wydźwięku można przewidywać wyniki wyborów lub tendencje rynkowe w oparciu o nastroje wyborców lub konsumentów. Badacze są również w stanie określić stronniczość ideologiczną danej osoby lub grupy [MEDH, 2014].

3. Uczenie maszynowe

3.1. Metody uczenia maszynowego

3.1.1. Naiwny klasyfikator Bayesa

Klasyfikatory bayesowskie są prostymi klasyfikatorami statystycznymi, opartymi na twierdzeniu Bayesa, które można wyrazić wzorem:

$$P(h|d) = \frac{P(d|h) \cdot P(h)}{P(d)}$$

W powyższym równaniu:

- $P(h|d)$ jest prawdopodobieństwem hipotezy h przy danych d. Nazywa się je prawdopodobieństwem a posteriori.
- $P(d|h)$ jest prawdopodobieństwem danych d zakładając, że hipoteza h była prawdziwa.
- $P(h)$ jest prawdopodobieństwem prawdziwości hipotezy h (niezależnie od danych). Nazywa się je prawdopodobieństwem a priori.
- $P(d)$ jest prawdopodobieństwem danych (niezależnie od hipotezy) [BROW, 2016].

Klasyfikatory bayesowskie przewidują prawdopodobieństwo przynależności danej próbki do konkretnej klasy. Naiwny klasyfikator Bayesa (ang. *Naive Bayes Classifier*, NBC) zakłada wzajemną niezależność predykatów, czyli niezależne prawdopodobieństwa wystąpienia cech. Drugim założeniem jest branie pod uwagę jedynie liczby wystąpień słów w klasie, z pominięciem ich kolejności, czyli stosowanie modelu „worka ze słowami" (ang. *bag-of-words*). Oba założenia są w rzeczywistości niepoprawne, ale wystarczające do otrzymania akceptowalnych wyników. Pomimo uproszczonych założeń, w wielu rzeczywistych sytuacjach naiwne klasyfikatory Bayesa dają znacznie lepsze wyniki, niż można by oczekiwać [JURA, 2012].

Naiwne klasyfikatory Bayesa różnią się między sobą zastosowanym rozkładem prawdopodobieństwa. Wyróżnia się trzy podejścia oparte na następujących rozkładach prawdopodobieństwa:

- Rozkład normalny (Gaussa)

  Wektor cech zawiera wartości z przestrzeni ciągłej. Dla danych treningowych należy obliczyć jedynie średnią oraz odchylenie standardowe. Jest to proste podejście, ale najrzadziej stosowane w problemie klasyfikacji dokumentów.
- Rozkład wielomianowy

  Wektor cech zawiera informację o liczebności poszczególnych termów w multizbiorze. Rozkład ten jest najczęściej używany w rozwiązywaniu problemu klasyfikacji dokumentów. Zlicza się wystąpienia poszczególnych słów w dokumentach.
- Rozkład zero-jedynkowy (Bernoulliego)

  Wektor cech zawiera informację, czy dana cecha występuje lub nie występuje w próbce. W odróżnieniu do rozkładu wielomianowego, nie jest istotna częstotliwość występowania danego termu, ale informacja, czy dana cecha wystąpiła w próbce [MCCA, 1998].

Stosując wielomianowy rozkład prawdopodobieństwa, klasyfikację rozpoczyna się od obliczenia prawdopodobieństwa dla każdej klasy, czyli liczby wystąpień dokumentów danej klasy podzieloną przez liczbę wystąpień wszystkich dokumentów.

$$P(c_i) = \frac{liczba\ wystąpień\ klasy\ c_i}{suma\ wystąpień\ wszystkich\ klas}$$

Następnie oblicza się prawdopodobieństwa warunkowe, czyli częstotliwość wystąpienia słowa x_j w klasie c_i:

$$P(x_j|c_i) = \frac{\#\ termu\ x_j\ w\ dokumentach\ klasy\ c_i}{\#\ wszystkich\ termów\ w\ dokumentach\ klasy\ c_i}$$

Kolejnym etapem jest obliczenie prawdopodobieństwa przynależności do danej klasy pod warunkiem wystąpienia danego wektora cech, wykorzystując wzór Bayesa:

$$P(c_i|d') = P(c_i|x_1, \dots, x_n) = \frac{P(c_i) \cdot P(x_1, \dots, x_j|c_i)}{P(x_1, \dots, x_j)}$$

Ważnym założeniem naiwnego klasyfikatora Bayesa jest założenie niezależności występujących cech:

$$P(x_1, \dots, x_n|c_i) = P(x_1|c_i) \cdot \dots \cdot P(x_n|c_i)$$

W ostatnim etapie zachodzi wybór klasy, dla której obliczone prawdopodobieństwo jest największe:

$$\bar{c} = argmax_{c_i} P(c_i|x_1, \dots, x_j)$$

zbiór	dokument	słowa	klasa
uczący	d_1	hamulec, bagażnik, koło	samochód
	d_2	bagażnik, dzwonek, hamulec, kierownica, koło, koło, koło, kosz	rower
	d_3	bagażnik, bagażnik, dywanik, fotel,	samochód
	d_4	kierownica bagażnik, dywanik, kierownica, kierownica	samochód
testowy	d’	hamulec, kierownica, koło	?

Tab. 3.1. Przykładowa klasyfikacja dokumentu z wykorzystaniem naiwnego klasyfikatora Bayesa

Na przykładzie zbioru dokumentów zaprezentowanych w tabeli 3.1. obliczenie przynależności do klas z wykorzystaniem naiwnego klasyfikatora Bayesa przebiega następująco:

$$P(samochód) = \frac{3}{4} \qquad P(rower) = \frac{1}{4}$$

$$P(samochód|hamulec, kierownica, koło)$$

$$= \frac{P(samochód) \cdot P(hamulec|samochód) \cdot P(kierownica|samochód) \cdot P(koło|samochód)}{P(hamulec) \cdot P(kierownica) \cdot P(koło)}$$

$$= \frac{\frac{3}{4} \cdot \frac{1}{12} \cdot \frac{1}{4} \cdot \frac{1}{12}}{\frac{2}{20} \cdot \frac{4}{20} \cdot \frac{4}{20}} \approx \frac{\frac{3}{2304}}{\frac{32}{8000}} = \mathbf{0,326}$$

$$P(rower|hamulec, kierownica, koło)$$

$$= \frac{P(rower) \cdot P(hamulec|rower) \cdot P(kierownica|rower) \cdot P(koło|rower)}{P(hamulec) \cdot P(kierownica) \cdot P(koło)}$$

$$= \frac{\frac{1}{4} \cdot \frac{1}{8} \cdot \frac{1}{8} \cdot \frac{3}{8}}{\frac{2}{20} \cdot \frac{4}{20} \cdot \frac{4}{20}} \approx \frac{\frac{3}{2048}}{\frac{32}{8000}} = \mathbf{0,366}$$

Zatem dokument d' zaliczamy do klasy $rower$.

3.1.2. K-najbliższych sąsiadów

Metoda k-najbliższych sąsiadów (ang. *k-nearest neighbors*, KNN), w odróżnieniu od pozostałych metod opisanych w niniejszej pracy, nie zawiera etapu uczenia. Reprezentację modelu stanowi cały zbiór danych treningowych. Dane mogą być przechowywane przy użyciu złożonych struktur, aby jak najskuteczniej wyszukiwać i dopasowywać nowe wzorce do istniejącego modelu [BROW, 2016].

Przewidywanie następuje bezpośrednio przy użyciu danych treningowych. Dla każdego nowego punktu danych algorytm przeszukuje cały zestaw danych treningowych w poszukiwaniu K najbardziej podobnych instancji (sąsiadów). Zmienną wyjściową dla klasyfikacji stanowi najczęściej występująca klasa wśród K sąsiadów [BROW, 2016].

Aby określić, które z K instancji w zbiorze treningowym są najbardziej podobne do nowych danych, wykorzystuje się miary odległości. W przypadku zmiennych wejściowych o wartościach rzeczywistych najpopularniejszą miarą odległości jest odległość euklidesowa, obliczana jako pierwiastek kwadratowy z sumy kwadratów różnic między dwoma punktami we wszystkich atrybutach wejściowych:

$$d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Innymi popularnymi miarami odległości są:

- Odległość Hamminga – oblicza odległość miedzy dwoma wektorami binarnymi:

$$d(a, b) = \sum_{i=1}^{n} |a_i - b_i|$$

$$a = b \Longrightarrow 0$$

$$a \neq b \Longrightarrow 1$$

- Odległość Taksówkowa (ang. *Manhattan Distance*) – oblicza odległość między wektorami rzeczywistymi, wykorzystując sumę ich różnic bezwzględnych:

$$d(a, b) = \sum_{i=1}^{n} |a_i - b_i|$$

- Odległość Minkowskiego – stanowi uogólnienie odległości Euklidesowej i Taksówkowej:

$$d(a,b) = \left(\sum_{i=1}^{n} (|a_i - b_i|)^q \right)^{1/q}$$

[BROW, 2016]

Dla $k = 1$ algorytm sprowadza się do równania:

$$h(x) = f(argmin_{x_i \in x} d(x_i, \bar{x}))$$

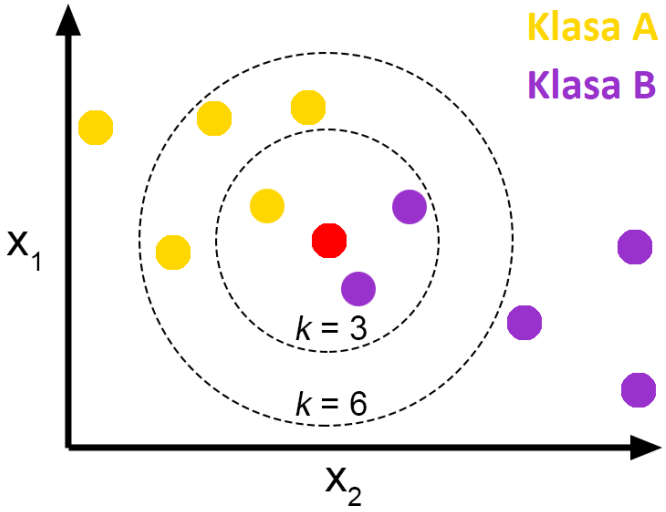
W powyższym równaniu $d(x_i, \bar{x})$ jest odległością Euklidesową między wektorami x_i oraz $\bar{x}$, czyli jako klasę przewidywaną danej instancji przyjmuje się klasę najbliższego elementu ze zbioru trenującego.

Optymalną wartość K można znaleźć przez dostrajanie algorytmu. Dobrą metodą jest wybór kilkunastu wartości i sprawdzanie, która z nich daje najlepsze wyniki. Złożoność obliczeniowa KNN wzrasta wraz z rozmiarem zbioru danych treningowych. W przypadku bardzo dużych zestawów treningowych, można wylosować próbkę z zestawu danych treningowych, z której następnie oblicza się K-najbardziej podobnych instancji [BROW, 2016].

Przy użyciu KNN do klasyfikacji, wynikiem jest klasa z najwyższą frekwencją z K najbardziej podobnych instancji. Prawdopodobieństwo klasy można obliczyć jako znormalizowaną częstotliwość próbek należących do każdej klasy w zestawie K najbardziej podobnych instancji. Zakładając problem klasyfikacji binarnej, prawdopodobieństwo może zostać obliczone ze wzoru:

$$p(c = 0) = \frac{zliczenia(c = 0)}{zliczenia(c = 0) + zliczenia(c = 1)}$$

Dobrą praktykę stanowi wybór wartości K parzystej dla nieparzystej liczby klas oraz wartości K nieparzystej dla parzystej liczby klas w celu uniknięcia remisu. W sytuacji, gdy wystąpi remis należy rozszerzyć wartość K o 1 i poszukać następnej najbardziej podobnej instancji w zbiorze [BROW, 2016].



Rys. 3.1. Działanie algorytmu k-najbliższych sąsiadów dla $k = 3$ oraz $k = 6$. Na podstawie [3].

Na rysunku 3.1. przedstawiono zbiór treningowy złożony z pięciu punktów należących do klasy A (oznaczonych żółtym kolorem) oraz pięciu punktów należących do klasy B (oznaczonych fioletowym kolorem). Do wykresu dodano nowy punkt oznaczony kolorem czerwonym. Jeśli za k podstawiona zostanie wartość 3, to najbliżej nowego punku leżą dwa punkty należące do klasy B i jeden punkt należący do klasy A. W takiej sytuacji punkt ten zostanie przypisany do klasy B. Jeśli natomiast do k przypisze się wartość 6, to w sąsiedztwie nowego punktu leżą cztery punkty należące do klasy A i dwa punkty należące do klasy B. W tym przypadku nowy punkt zostanie przypisany do klasy A.

3.1.3. Maszyna wektorów nośnych

Maszyna wektorów nośnych (ang. *Support Vector Machines*, SVM) jest algorytmem, który znajduje liniową funkcję separującą, wyznaczającą podział przestrzeni na obszary odpowiadające klasom decyzyjnym. Algorytm SVM rzutuje daną przestrzeń cech na przestrzeń o większej liczbie wymiarów, co pozwala osiągnąć własność zwaną liniową separowalnością [CHIH, 2002].

Liczbowe zmienne wejściowe danych tworzą n-wymiarową przestrzeń. Jeśli występują dwie zmienne wejściowe, to algorytm SVM utworzy przestrzeń dwuwymiarową. Hiperpłaszczyzna jest linią dzielącą przestrzeń zmiennej wejściowej. Algorytm dokonuje takiego wyboru hiperpłaszczyzny, aby jak najlepiej oddzielić w przestrzeni

punkty należące do różnych klas. W dwóch wymiarach można zwizualizować to jako linię, przy założeniu, że wszystkie punkty wejściowe mogą być nią całkowicie oddzielone. Przykładowe równanie opisujące tę linię może wyglądać w następujący sposób:

$$B0 + (B1 \times X1) + (B2 \times X2) = 0$$

Współczynniki $B1$ oraz $B2$ określają nachylenie linii, a współczynnik $B0$ określa punkt przecięcia. Wartości $X1$ i $X2$ stanowią dwie zmienne wejściowe. Nowy punt może zostać sklasyfikowany przy pomocy takiej linii. Podstawiając wartości wejściowe do równania liniowego, można wyliczyć, czy punkt leży nad lub pod linią [BROW, 2016].

Odległość między linią a najbliższymi punktami danych określana jest jako margines (ang. *margin*). Najlepszą lub optymalną linią, która może oddzielić dwie klasy, jest linia o największym marginesie. Algorytm SVM poszukuje hiperpłaszczyzny z największym marginesem (ang. *Maximal-Margin hyperplane*). Margines obliczany jest jako prostopadła odległość od linii do najbliższych punktów. Tylko te punkty są istotne dla określenia linii i konstruowania klasyfikatora. Punkty te nazywane są wektorami nośnymi (ang. *support vectors*) [BROW, 2016].

Model SVM jest reprezentacją przykładów jako punktów w przestrzeni, mapowanych w taki sposób, że przykłady należące do poszczególnych klas są oddzielone od siebie za pomocą wyraźnej przerwy. Nowe przykłady są mapowane do tej samej przestrzeni. Przynależność do danej klasy zostaje przypisana w oparciu o to, po której stronie przerwy się znajdują. Równanie służące do przewidywania nowego przykładu wygląda w następujący sposób:

$$f(x) = B0 + \sum_{i=1}^{n} (a_i \times (x \times x_i))$$

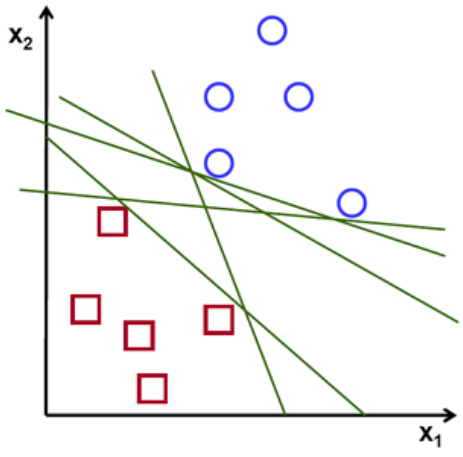
W powyższym równaniu x stanowi wektor wejściowy, a współczynniki $B0$ i a_i (dla każdego przykładu) muszą być oszacowane przez algorytm na podstawie danych treningowych [BROW, 2016].

Model SVM jest tworzony przy użyciu procedury optymalizacji. Można użyć optymalizacji numerycznej w celu wyszukania współczynników hiperpłaszczyzny lub

metody SMO (ang. *Sequential Minimal Optimization*), rozdzielającej problem na pod problemy, które można rozwiązać obliczeniowo [BROW, 2016].

Algorytm SVM został stworzony do klasyfikacji binarnej, ale może zostać wykorzystany również do klasyfikacji wieloklasowej. Obecnie istnieją dwa rodzaje podejść do klasyfikacji wieloklasowej. Jednym z nich jest budowanie i łączenie kilku klasyfikatorów binarnych, podczas gdy drugie polega na bezpośrednim rozpatrywaniu wszystkich danych w jednym sformułowaniu optymalizacyjnym. Stosuje się następujące podejścia:

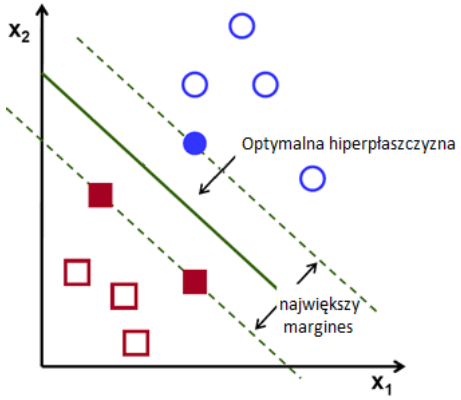
- *One-againts-all* – konstruuje się k modeli (gdzie k jest liczbą klas), otrzymując k funkcji decyzyjnych. Nowy przykład zostaje przypisany do klasy, dla której wartość funkcji decyzyjnej jest największa.
- *One-againts-one* – konstruuje się $\frac{k\cdot(k-1)}{2}$ modeli, otrzymując $\frac{k\cdot(k-1)}{2}$ klasyfikatorów, z których każdy jest trenowany na danych z dwóch klas. Nowy przykład zostanie sklasyfikowany do najczęściej występującej klasy.
- *Directed Acyclic Graph Support Vector Machines* (DAGSVM) – etap trenowania jest identyczny jak dla podejścia *one-againts-all*. Podejście to wykorzystuje grafy acykliczne posiadające $\frac{k\cdot(k-1)}{2}$ węzłów oraz k liści, stanowiących klasy [CHIH, 2002].



Rys. 3.2. Przykładowy podział hiperpłaszczyzny dla klasyfikacji binarnej [4].

Na rysunku 3.2. przedstawiono przykładowe podziały hiperpłaszczyzny dla problemu klasyfikacji binarnej. Istnieje wiele linii mogących stanowić rozwiązanie. Algorytm SVM poszukuje hiperpłaszczyzny, która zapewnia największą minimalną odległość od

przykładów treningowych. Przykładowe rozwiązanie tego problemu przedstawiono na rysunku 3.3.



Rys 3.3. Optymalne rozwiązanie problemu klasyfikacji algorytmem SVM. Na podstawie [5].

3.1.4. Regresja logistyczna

Regresja logistyczna zawdzięcza swoją nazwę funkcji stanowiącej podstawę tej metody, czyli funkcji logistycznej. Funkcja logistyczna, nazywana również funkcją sigmoidalną, została opracowana przez statystyków w celu opisania właściwości wzrostu populacji, szybkiego wzrostu i maksymalnego wykorzystania zdolności środowiska. Jest to krzywa w kształcie litery S, która może przyjmować dowolną wartość rzeczywistą i odwzorowywać ją na wartość pomiędzy 0 a 1, ale nigdy dokładnie na tych granicach:

$$y = \frac{1}{1 + e^{-v}}$$

W powyższym równaniu e stanowi podstawę logarytmów naturalnych, a wartość v jest rzeczywistą wartością liczbową, którą należy przekształcić [BROW, 2016].

W modelu regresji logistycznej wartości wejściowe (x) są łączone liniowo za pomocą wag lub wartości współczynników, w celu przewidywania wartości wyjściowej (y). Modelowana wartość wyjściowa jest zawsze wartością liczbową. Przykład równania regresji logistycznej może wyglądać w następujący sposób:

$$y = \frac{e^{B0+B1\times x}}{1 + e^{B0+B1\times x}}$$

W powyższym wzorze y stanowi przewidywaną wartością wyjściową, $B0$ jest wartością nazywaną progiem (ang. *bias*) – uproszczonym założeniem modelu, które ułatwia uczenie się funkcji celu, a $B1$ jest współczynnikiem dla pojedynczej wartości wejściowej (x) [BROW, 2016].

Współczynniki algorytmu regresji logistycznej szacuje się na podstawie danych treningowych. Odbywa się to za pomocą estymacji metodą największej wiarygodności (ang. *maximum likelihood estimation*, MLE), która próbuje znaleźć wartości parametrów maksymalizujących funkcję prawdopodobieństwa, biorąc pod uwagę dane obserwacje. Zakładając klasyfikację binarną, dobranie najlepszych współczynników pozwala uzyskać model przewidujący wartość bardzo zbliżoną do 1 dla domyślnej klasy i wartość bardzo zbliżoną do 0 dla drugiej klasy [BROW, 2016].

Przewidywanie z wykorzystaniem modelu regresji logistycznej polega na podstawieniu liczb do równania regresji logistycznej, z wcześniej wyliczonymi współczynnikami i obliczeniu wyniku. Przykładem może być model przewidujący, czy dana osobą jest kobietą czy mężczyzną, w zależności od jej wzrostu. Zakładając, że wartości bliskie 0 wskazują przynależność do klasy kobiet, a wartości bliskie 1 skazują na klasę mężczyzn, poszukując klasyfikacji dla wzrostu 150 cm, można wykorzystać poniższe równanie:

$$y = \frac{e^{B0+B1\times x}}{1 + e^{B0+B1\times x}}$$

Przy założeniu współczynników $B0 = -100$ i $B1 = 0.6$, przyjmie ono postać:

$$y = \frac{e^{-100+0.6\times 150}}{1 + e^{-100+0.6\times 150}}$$

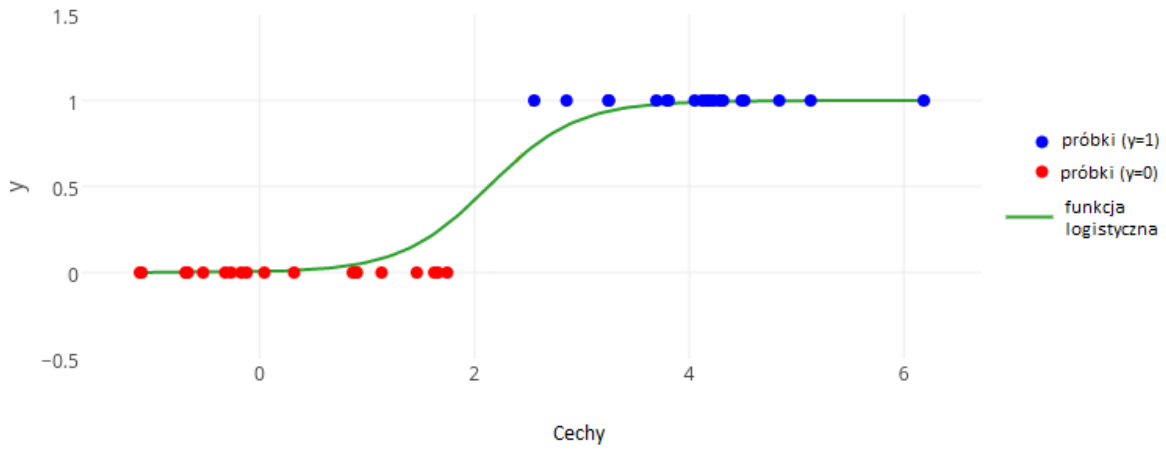
$$y = 0.0000453978687$$

W problemie klasyfikacji poszukuje się wyraźnej odpowiedzi w kwestii przynależności do danej klasy, dlatego można przyjąć binarne prawdopodobieństwa:

$$przewidywanie = 0 \quad JEŻELI \quad p(mężczyzna) < 0.5$$

$$przewidywanie = 1 \quad JEŻELI \quad p(mężczyzna) \geq 0.5$$

Otrzymany wynik jest bardzo bliski 0, czyli przyjmujemy $p(mężczyzna) < 0.5$. Zatem powyższy przykład powinien zostać przypisany do klasy kobiet.



Rys. 3.4. Przykładowy wykres funkcji regresji logistycznej dla problemu klasyfikacji binarnej. Na podstawie [6].

Na rysunku 3.4. przedstawiono przykładowy wykres funkcji regresji logistycznej rozwiązujący problem klasyfikacji binarnej. W powyższym przykładzie model przewiduje wartość bardzo zbliżoną do 1 dla klasy niebieskiej oraz wartość bardzo zbliżoną do 0 dla klasy czerwonej.

3.1.5. Sieci neuronowe

Sieci neuronowe (ang. *neural networks*, NN) są zbiorami prostych jednostek obliczeniowych, które przetwarzają dane, komunikują się ze sobą i pracują równolegle. Inspirację do stworzenia sztucznych sieci neuronowych stanowiła budowa naturalnych neuronów występujących w układach nerwowych istot żywych [STEF, 2016].

Sieć neuronowa składa się z prostych jednostek przetwarzania – neuronów oraz ukierunkowanych, ważonych połączeń pomiędzy tymi neuronami. Waga połączenia między dwoma neuronami i i j jest określana jako $w_{i,j}$ [ZHAN, 2000]. Budowa sztucznego neuronu została pokazana na rysunku 3.5. Do jego podstawowych elementów należą [STEF, 2016]:

- n wejść neuronu z wagami w_i,
- sygnał wyjściowy y,

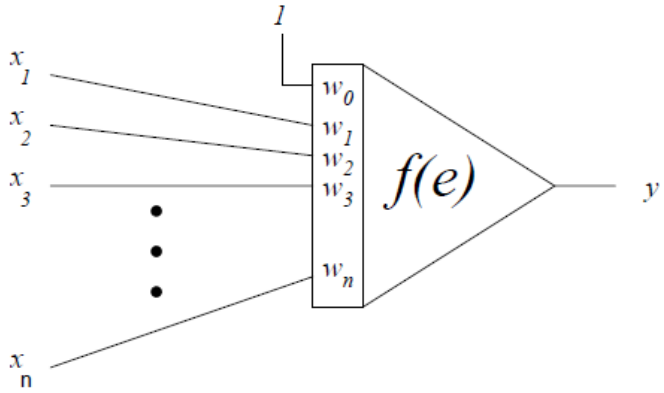
- pobudzenie e neuronu

$$e = \sum_{i=0}^{n} w_i \cdot x_i = w^T x$$

W powyższym równaniu w jest wektorem wag, a x wektorem sygnałów wejściowych.

- funkcja aktywacji

$$y = f(e)$$



Rys. 3.5. Budowa sztucznego neuronu. Na podstawie [7].

Funkcja aktywacji może przyjąć następujące formy:

- funkcja liniowa

$$f(x) = ax + b$$

- funkcja logistyczna

$$f(x) = \frac{1}{1 + e^{-x}}$$

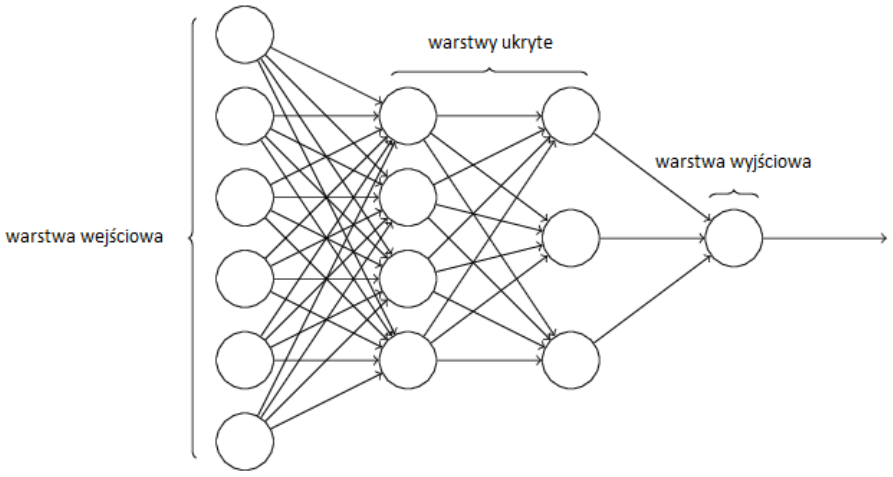
- tangens hiperboliczny

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Wyróżnia się dwa typy sieci neuronowych. Pierwszym z nich są sieci jednokierunkowe (ang. *feedforwarded*), które nie zawierają sprzężeń zwrotnych. Drugim natomiast są sieci rekurencyjne (ang. *feedback*), w których neurony tworzą sprzężenia zwrotne, liczne i skomplikowane zamknięte pętle, w których impulsy mogą długo krążyć i zmieniać się [TADE, 2007].

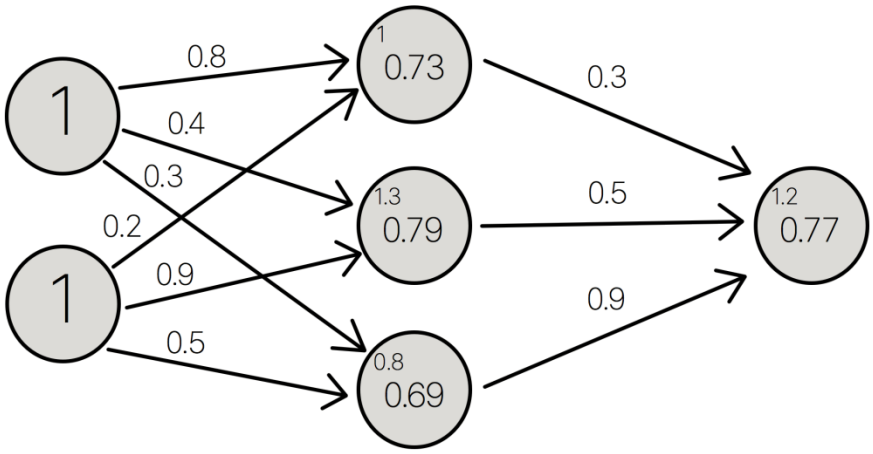
Najprostsza jednokierunkowa sieć neuronowa przedstawiona na rysunku 3.6. składa się z trzech warstw:

- Warstwa wejściowa – służy do wprowadzania danych do sieci. Zawiera tyle neuronów, ile wymiarów mają wektory wejściowe.
- Warstwy ukryte – w tych warstwach zachodzą wszystkie obliczenia. W najprostszej wersji może występować tylko jedna warstwa ukryta. W wersji, w której występuje kilka warstwa ukrytych, ostatnia musi mieć tyle neuronów, ile występuje możliwych klas.
- Warstwa wyjściowa – wyznacza odpowiedź sieci [TADE, 2007].



Rys. 3.6. Jednokierunkowa sieć neuronowa. Na podstawie [8].

Jedną z głównych zalet sieci neuronowych jest to, że są to metody bardzo uniwersalne, mogące dostosować się do danych bez wyraźnej specyfikacji funkcjonalnej. Potrafią przybliżać dowolną funkcję z dowolną dokładnością. Sieci neuronowe elastycznie modelują złożone relacje, które występują w świecie rzeczywistym [ZHAN, 2000].



Rys. 3.7. Przykładowa sieć neuronowa z dwoma neuronami wejściowymi. Na podstawie [9].

Przykładowa sieć neuronowa została pokazana na rysunku 3.7. W przykładzie użyto jednej warstwy ukrytej zawierającej trzy neurony. Synapsom przypisano losowe wagi pomiędzy 0 a 1. Następnie zsumowano iloczyny wejść z odpowiadającym im zestawem wag, aby uzyskać pierwsze wartości dla ukrytej warstwy:

$$1 \cdot 0.8 + 1 \cdot 0.2 = 1$$

$$1 \cdot 0.4 + 1 \cdot 0.9 = 1.3$$

$$1 \cdot 0.3 + 1 \cdot 0.5 = 0.8$$

W kolejny etapie zastosowano sigmoidalną funkcję aktywacji na uzyskanych sumach, w celu wyliczenia ostatecznych wartości dla warstwy ukrytej:

$$S(1.0) = 0.73105857863$$

$$S(1.3) = 0.78583498304$$

$$S(0.8) = 0.68997448112$$

Następnie zsumowano iloczyn wyników ukrytej warstwy z drugim zestawem wag, aby określić sumę wyjściową:

$$0.73 \cdot 0.3 + 0.79 \cdot 0.5 + 0.69 \cdot 0.9 = 1.235$$

W ostatnim etapie zastosowano funkcję aktywacji, do otrzymania ostatecznego wyniku:

$$S(1.235) = 0.7746924929149283$$

3.2. Ewaluacja modelu

3.2.1. Walidacja na odłożonym zbiorze testowym

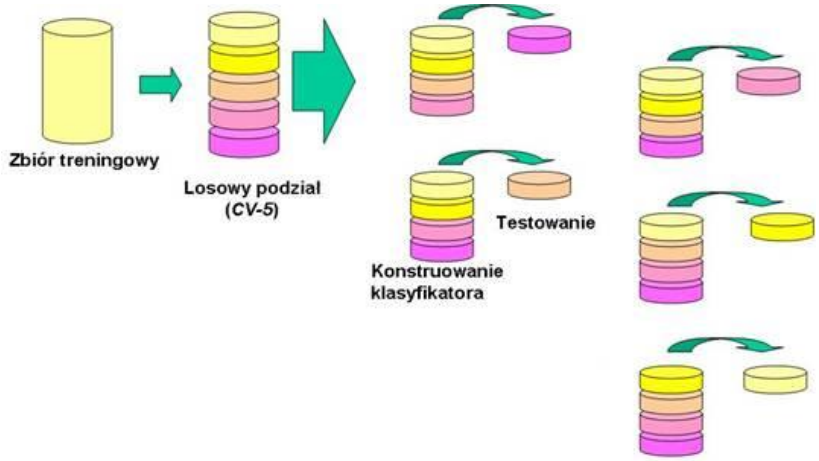
Walidacja na odłożonym zbiorze testowym (ang. *hold-out validation*) polega na podziale zbioru danych na zbiór uczący oraz zbiór testowy. Proces uczenia zachodzi przy wykorzystaniu zbioru treningowego, a testowanie – z wykorzystaniem zbioru testowego. Przeważnie stosuje się jeden z dwóch poniższych podziałów danych:

- 10% stanowi zbiór testowy, a 90% zbiór uczący,
- 20% stanowi zbiór testowy, a 80% zbiór uczący.

Wykorzystując tę metodę należy pamiętać o konieczności równomiernego rozkładu przykładów należących do danej klasy w obu zbiorach.

3.2.2. K-krotna walidacja krzyżowa

K-krotna walidacja krzyżowa (ang. *k-fold cross validation*) pozwala trenować i testować model k razy na różnych podzbiorach danych treningowych, w celu oszacowania jego poprawności. Proces k-krotnej walidacji krzyżowej został przedstawiony na rysunku 3.8. Metoda ta polega na losowym podziale zbioru treningowego na k równolicznych podzbiorów (najczęściej przyjmuje się $k = 10$). Na każdym etapie zbiorem testowym jest jeden z k podzbiorów, a pozostałe $k - 1$ podzbiorów stanowią zbiór uczący. Każdy z podzbiorów zostaje wykorzystany tylko jeden raz w procesie testowania oraz $k - 1$ razy podczas trenowania. Na końcu uśrednia się otrzymane wyniki walidacji.



Rys. 3.8. Schemat k-krotnej walidacji krzyżowej. Na podstawie [10].

3.2.3. Macierz konfuzji

W problemie klasyfikacji istnieją dwie kategorie błędu: uznanie przykładu pozytywnego za negatywny oraz uznanie negatywnego za pozytywny. Istnieją także dwa przypadki prawidłowej diagnozy. Prawidłową klasyfikację określa się jako prawda, a niepoprawną jako fałsz. W ten sposób powstają cztery warianty które, razem tworzą macierz konfuzji (ang. *Confusion Matrix*), która stanowi podstawę oceny jakości klasyfikacji.

Terminy *pozytywny* i *negatywny* odnoszą się do przewidywania klasyfikatora (określanego jako oczekiwanie), a określenia *prawda* i *fałsz* odnoszą się do tego, czy klasyfikacja odpowiada rzeczywistości (obserwacji).

	Zaklasyfikowany do klasy pozytywnej	**Zaklasyfikowany do klasy negatywnej**
Należy do klasy pozytywnej	TP (*True positive*)	FN (*False negative*)
Należy do klasy negatywnej	FP (*False positive*)	TN (*True negative*)

Tab. 3.1. Macierz konfuzji dla klasyfikacji binarnej.

Skróty zastosowane w tabeli 3.2. oznaczają:

- TP – przykłady poprawnie sklasyfikowane do klasy pozytywnej,
- FN – przykłady błędnie sklasyfikowane do klasy negatywnej,
- FP – przykłady błędnie sklasyfikowane do klasy pozytywnej,
- TN – przykłady poprawnie sklasyfikowane do klasy negatywnej.

3.2.4. Miary jakości

3.2.4.1. Dokładność

Dokładność (ang. *accuracy*) określa procent poprawnie sklasyfikowanych przykładów. Miara ta ocenia ogólną efektywność algorytmu [SOKO, 2006].

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \times 100\%$$

3.2.4.2. Precyzja

Precyzja (ang. *precision*) określa skuteczność w wykrywaniu danego zjawiska, czyli stosunek przykładów poprawnie sklasyfikowanych jako pozytywne do wszystkich przykładów sklasyfikowanych jako pozytywne [SOKO, 2006].

$$precision = \frac{TP}{TP + FP}$$

3.2.4.3. Pokrycie

Pokrycie (ang. *recall*) określa stosunek przykładów poprawnie sklasyfikowanych jako pozytywne do wszystkich rzeczywiście pozytywnych przykładów [SOKO, 2006].

$$recall = \frac{TP}{TP + FN}$$

Rys. 3.9. Porównanie wartości pokrycia i precyzji. Na podstawie [11].

Na rysunku 3.9. przedstawiono wartości miar służące do ewaluacji modelu. Wysoka wartość precyzji z niską wartością pokrycia (1) występuje, gdy algorytm poprawnie sklasyfikował daną grupę przykładów, ale do klasy tej należy znacznie więcej dokumentów. Wysoką wartość pokrycia z niską wartością precyzji (2) osiąga się, gdy algorytm przyporządkował do danej klasy znacznie więcej przykładów niż powinien, ale pokrywa większość poprawnie sklasyfikowanych dokumentów. Wynik optymalny (3), z wysoką wartością precyzji i pokrycia jest trudny do osiągnięcia.

3.2.4.4. Specyficzność

Specyficzność (ang. *specificity*) ocenia skuteczność algorytmu dla pojedynczej klasy. Określa stosunek przykładów poprawie sklasyfikowanych jako negatywne do

wszystkich negatywnych przykładów. Inaczej nazywana jest też wskaźnikiem TP (ang. *TP rate*) [SOKO, 2006].

$$specificity = \frac{TN}{TN + FP}$$

4. Nawiązania do innych projektów

4.1. Różne podejścia do ekstrakcji cech

W ostatnich latach metody oparte na uczeniu maszynowym służące do klasyfikacji wydźwięku znalazły szerokie zastosowanie ze względu na ich rosnącą jakość. Sposób ekstrakcji złożonych cech, a nie wyłącznie prostych cech, a także odkrywanie, które typy cech są bardziej wartościowe, stanowią dwa kluczowe zagadnienia w metodach wykorzystujących uczenie maszynowe. Do tej pory zaproponowano różne metody ekstrakcji cech, takich jak: pojedyncze wyrazy, n-gramów słów, wzory leksykalno-syntaktyczne i wiele innych nowatorskich modeli [ZHAN, 2015].

Pang i in. (2002) po raz pierwszy zastosowali metodę opartą na uczeniu maszynowym w celu klasyfikacji wydźwięku. Wykorzystali model n-gramowy i porównali działanie następujących klasyfikatorów: naiwny klasyfikator Bayesa, klasyfikator Maximum Entropy oraz maszynę wektorów nośnych. Najlepsze wyniki klasyfikacji przy użyciu unigamów jako zestawu cech uzyskali dla modelu SVM. W ciągu ostatnich kilku lat zaproponowali różne metody selekcji cech i modele klasyfikacji.

Yessenalina, Yue i Cardie (2010) zaproponowali wspólne dwupoziomowe podejście do klasyfikacji wydźwięku na poziomie dokumentu, które jednocześnie wydobywa subiektywne zdania i przewiduje wartość wydźwięku na podstawie tych wyodrębnionych zdań. Zhai i in. (2011) wyekstrahowali słowa, zadania, ich podciągi i grupy zawierające wydźwięk jako wybrane cechy.

Wang, Li, Song, Wei i Li (2011) zaproponowali skuteczną metodę selekcji cech opartą na wskaźniku dyskryminacyjnym Fishera dla klasyfikacji wydźwięku tekstu.

Yao, Wang i Yin (2011) wykorzystali metody uczenia maszynowego oparte na danych statystycznych do wyboru cech i redukcji wymiarów dla klasyfikacji wydźwięku recenzji chińskich hotelów. Xia, Zong i Li (2011) skorzystali z platformy programistycznej do integracji różnych zestawów cech i algorytmów klasyfikacji w celu zwiększenia ogólnej wydajności.

Abbasi, France, Zhang i Hsinchun (2011) zaproponowali metodę selekcji cech tekstu opartą na regułach o nazwie *Feature Relation Network* (FRN), która uwzględnia

informacje semantyczne, a także wykorzystuje składniowe relacje między cechami n-gramów.

Wang, Yin, Yao i Liu (2013) oraz Wang, Yin, Zheng i Liu (2014) przyjęli częstotliwość dokumentów (ang. *document frequency*), entropię względną (ang. *information gain*), statystyki chi-kwadrat (anf. *chi-squared statistic*) i wzajemne informacje (ang. *mutual information*) do wyboru cech, a następnie zastosowali metodę ważenia *Boolean weighting* do ustawienia wag cech i skonstruowania modelu przestrzeni wektorowej.

Moraes, Valiati i Neto (2013) przyjęli standardowy kontekst ewaluacji z popularnymi nadzorowanymi metodami doboru i ważenia cech w tradycyjnym modelu „worka ze słowami" (ang. *bag-of-words*).

4.2. Analiza wydźwięku recenzji i komentarzy użytkowników

Chakankar, Mathur i Venuturimilli (2012) zastosowali uczenie maszynowe do analizy wydźwięku tekstów generowanych przez użytkowników w Internecie. Zbiór danych stanowiły recenzje filmów oraz komentarze zamieszczone na portalach społecznościowych i forach internetowych. Zastosowali metody uczenia nadzorowanego: naiwny klasyfikator Bayesa oraz maszynę wektorów nośnych, w celu wykrycia pozytywnego lub negatywnego wydźwięku w tekście [CHAK, 2012].

Przeanalizowali trzy oddzielne zestawy danych, z których dwa zawierały recenzje filmów, a trzeci składał się z obelg występujących w komentarzach użytkowników. Pierwszy zbiór danych składał się z 25 tysięcy recenzji filmów stanowiących zbiór treningowy oraz z zestawu 25 tysięcy recenzji wykorzystanych do testowania. W każdym zestawie wystąpił równomierny podział między pozytywnymi a negatywnymi recenzjami. Drugi zbiór składał się z 2 tysięcy przetworzonych recenzji filmów, z równomiernym podziałem na recenzje pozytywne i negatywne. Dla tego zbioru zastosowano 10-krotną walidację krzyżową.

Trzeci zbiór danych składał się z 3947 przykładów stanowiących zbiór treningowy, z czego 1049 (26,6%) zawierało obelgi. Zestaw danych testowych zawierał 2467 przykładów, z czego 693 (26,2%) stanowiło obelgi.

W każdym ze zbiorów usunięto tagi html oraz powtórzenia liter. Użyto podejścia „worka ze słowami", stosując unigramy, bigramy i trygramy. Zastosowano trzy różne

podejścia: zliczenia częstości termów, podejście zerojedynkowe – sprawdzenie czy dana cecha występuje oraz wektory ważone metodą *tfidf*. Jako cechy wybrano N najczęściej występujących n-gramów.

Dla pierwszego zbioru recenzji naiwny klasyfikator Bayesa uzyskał dokładność wynoszącą 0.86936, natomiast maszyna wektorów nośnych uzyskała dokładność równą 0.89632. Dla drugiego zbioru recenzji, po zastosowaniu walidacji krzyżowej uzyskano dokładność wynoszącą 0.85 dla naiwnego klasyfikatora Bayesa oraz 0.89 dla maszyny wektorów nośnych. W zbiorze wykrywającym obelgi naiwny klasyfikator Bayesa uzyskał dokładność 0.851, a maszyna wektorów nośnych – 0.8415. Dla tego zbioru dodatkowo użyto regresji logistycznej, która uzyskała dokładność wynoszącą 0.845.

4.3. Wieloklasowa analiza wydźwięku

Bouazizi i Ohtsuki (2017) starali się zaklasyfikować wypowiedzi użytkowników serwisu *Tweeter* [1] do jednej z następujących klas: *miłość, szczęście, zabawa, neutralny, nienawiść, smutek, gniew*. Dla każdej wypowiedzi ekstrahowali różne zbiory cech, odwoływali się do zbioru uczącego i wykorzystywali metody uczenia nadzorowanego [BOUA, 2017].

Zbiór danych treningowych składał się z 21 tysięcy wypowiedzi, które zostały ręcznie sklasyfikowane do 7 klas. Każda z klas zawierała 7 tysięcy wypowiedzi. Zbiór danych testowych zawierał 19740 wypowiedzi. Każdy z nich został ręcznie sklasyfikowany do jednej z 7 klas.

Zastosowali następujące podejścia do wyboru cech:

- cechy oparte na wydźwięku,
- cechy oparte na interpunkcji,
- cechy syntaktyczne i stylistyczne,
- cechy semantyczne,
- unigramy i najczęściej występujące słowa,
- cechy oparte na wzorcach.

[1] https://twitter.com data dostępu: 03.05.2018

Pomimo dużej liczby klas uzyskana dokładność wynosi 60,2%. Poszczególne wartości dokładności dla każdej z klas przedstawiono w tabeli 4.1. Najlepsze wyniki otrzymano dla klas *miłość* oraz *nienawiść*, co świadczy o tym, że wypowiedzi należące do tych klas można łatwo odróżnić od pozostałych klas.

klasa	**dokładność**
miłość	0.752
szczęście	0.543
zabawa	0.407
neutralny	0.678
nienawiść	0.909
smutek	0.521
gniew	0.622
średnia	0.602

Tab. 4.1. Wartości dokładności dla klasyfikacji wieloklasowej.

5. Narzędzia użyte w projekcie

5.1. BeautifulSoup

BeautifulSoup [2] jest biblioteką języka Python służącą do analizowania dokumentów HTML oraz XML. Umożliwia ekstrahowanie określonych treści z pobranej strony internetowej. Zapewnia idiomatyczne sposoby nawigacji, przeszukiwania i modyfikowania drzewa parsowania (ang. *parse tree*). Udostępnia zestaw narzędzi do analizy dokumentów i wydobywania potrzebnych informacji. Umożliwia automatyczną konwersję dokumentów wejściowych do zestawu znaków Unicode i dokumentów wyjściowych do systemu kodowania UTF-8. Jest popularnym, szybkim i elastycznym paserem dokumentów.

W projekcie autorskim bibliotekę zastosowano w celu ekstrakcji treści komentarzy z plików html. Przykładowy fragment pliku wejściowego został pokazany na rysunku 5.1. Sposób użycia biblioteki pokazano na rysunku 5.2. a fragment formatu pliku wyjściowego na rysunku 5.3.

```
<div class=col-md-10 data label-aligned name=answer style=overflow:auto><p>Ocena wykładu: 8.</p>
    <p>Komentarz:</p><p>Wykład był poprowadzony dobrze, na plus fakt, że było przedstawionych dużo
       przykładów różnych wyrażeń regularnych i ich wywołania w Pythonie.</p>
</div>
<div class=col-md-10 data label-aligned name=answer style=overflow:auto><p>Ciekawy wykład fajnie
       przeprowadzony.</p>
    <p><br data-mce-bogus=1></p><p>7/10</p></div>
<div class=col-md-10 data label-aligned name=answer style=overflow:auto><p>Ciekawy wykład. Praktyczne
       zastosowania NLP. 9/10<br></p>
</div>
```

Rys. 5.1. Przykładowy fragment pliku html zawierający komentarze.

```
from bs4 import BeautifulSoup
in_f = open('comments.html', 'r')
out_f = open('comments.txt', 'w')

for line in in_f:
    comment = BeautifulSoup(line, "lxml").get_text()
    out_f.write(comment.encode('utf-8'))

in_f.close()
out_f.close()
```

Rys. 5.2. Ekstrakcja treści komentarzy za pomocą biblioteki *BeautifulSoup*.

[2] https://www.crummy.com/software/BeautifulSoup/ data dostępu: 15.04.2018

```
Ocena wykładu: 8.Wykład był poprowadzony dobrze, na plus fakt, że było
przedstawionych dużo przykładów różnych wyrażeń regularnych i ich
wywołania w Pythonie.
Ciekawy wykład fajnie przeprowadzony.7/10
Ciekawy wykład. Praktyczne zastosowania NLP. 9/10
```

Rys. 5.3. Ostateczny format komentarzy.

5.2. Scikit-learn

Scikit-learn [3] jest biblioteką otwartego oprogramowania służącą do uczenia maszynowego napisaną w języku Python. Zapewnia szereg nadzorowanych i nienadzorowanych algorytmów uczenia maszynowego za pośrednictwem spójnego interfejsu. Zawiera różne algorytmy klasyfikacji, regresji oraz klastrowania. Udostępnia również narzędzia do przetwarzania tekstu.

Biblioteka *Scikit-learn* koncentruje się wyłącznie na modelowaniu danych. Nie zapewnia funkcjonalności umożliwiających wczytywanie danych oraz manipulowanie nimi, dlatego została zaprojektowana do współpracy z numerycznymi i naukowymi bibliotekami języka Python: *NumPy* oraz *SciPy*.

Do najpopularniejszych funkcjonalności dostarczanych przez *scikit-learn* należą:

- klastrowanie (ang. *Clustering*) – służy do grupowania nieoznaczonych danych,
- walidacja krzyżowa (ang. Cross Validation) – służy do oceny wydajności nadzorowanych modeli,
- tworzenie zestawów danych (ang. *Datasets*) – służy do generowania zestawów danych o określonych właściwościach w celu badania zachowania modelu,
- redukcja wymiarów (ang. *Dimensionality Reduction*) – służy do zmniejszenia liczby cech w danych,
- metody Ensemble (ang. *Ensemble Methods*) – służą do łączenia przewidywań różnych nadzorowanych modeli,
- ekstrakcja cech (ang. *Feature Extraction*) – służy do definiowania atrybutów danych wejściowych,

[3] http://scikit-learn.org/ data dostępu: 15.04.2018

- wybór cech (ang. *Feature Selection*) – służy do identyfikacji znaczących cech,
- dostrajanie parametrów (ang. *Parameter Tuning*) – służy do uzyskania jak najlepszych modeli nadzorowanych,
- tworzenie nadzorowanych modeli (ang. *Supervised Models*) – służy do tworzenia modeli liniowych, probabilistycznych i neuronowych.

W projekcie autorskim zostały zawarte następujące klasyfikatory:

- **Naiwny klasyfikator Bayesa**
  sklearn.naive_bayes.MultinomialNB
- **Regresja logistyczna**
  sklearn.linear_model.LogisticRegression
- **Maszyna wektorów nośnych**
  sklearn.svm.SVC
- **K-najbliższych sąsiadów**
  sklearn.neighbors.KNeighborsClassifier
- **Sieci neuronowe**
  sklearn.neural_network.MLPClassifier

Biblioteka udostępnia również funkcje, spośród których w projekcie użyto:

- ***Stratified K-Folds cross-validator***
  sklearn.model_selection.StratifiedKFold
  StratifiedKFold jest funkcją, która dzieli zbiór danych k razy na zbiór uczący i zbiór testowy, w celu przeprowadzenia walidacji krzyżowej. Podziały są tworzone w taki sposób, aby zachować taki sam procent próbek, jaki wystąpił w całym zbiorze. Każdy z k podzbiorów zostaje użyty $k-1$ razy w procesie uczenia oraz raz na etapie testowania. Przykład zastosowania funkcji pokazano na rysunku 5.4.

```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=10)

results =[]

for train, test in skf:
    model.fit(X[train], Y[train])
    results.append(accuracy_score(y[test], model.predict(X[test])))

print "Walidacja krzyżowa (10-krotna): ", np.mean(results)
```

Rys. 5.4. Przykładowe zastosowanie modułu *KFold.*

- ***TfidfVectorizer***

  sklearn.feature_extraction.text.TfidfVectorizer

  TfidfVectorizer jest funkcją przekształcającą zbiór wektorów, które reprezentują częstość wystąpień termów w dokumentach na wektory ważone metodą *tfidf*. Przykład zastosowania funkcji pokazano na rysunku 5.5.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features = 60000, ngram_range = (1, 3),
    sublinear_tf = True)

train_comments = []
for comment in train["komentarz"]:
    train_comments.append( " ".join(tokenize(comment)))

train_x = vectorizer.fit_transform(train_comments)
```

Rys. 5.5. Przykładowe zastosowanie modułu *TfidfVectorizer*

5.3. Moduł re

Moduł *re* [4] jest biblioteką napisaną w języku Python, która zapewnia operacje dopasowywania wyrażeń regularnych. Funkcje dostępne w tym module pozwalają sprawdzić, czy dany ciąg pasuje do danego wyrażenia regularnego. Moduł ten został użyty w celu ekstrakcji oceny z komentarzy. Jego zastosowanie zostało zaprezentowane na rysunku 5.5. Funkcja *findall* zwraca listę wszystkich dopasowań stanowiących potencjalne oceny zawarte w komentarzach.

[4] https://docs.python.org/2/library/re.html data dostępu: 15.04.2018

```
match = re.findall('\d+', comment)
match = re.findall('\d+/\d+', comment)
match = re.findall('\d+ pkt', comment)
match = re.findall('Ocena: \d+', comment)
match = re.findall('Ocena \d+', comment)
```

Rys. 5.6. Przykłady zastosowań wyrażeń regularnych do ekstrakcji ocen z komentarzy.

6. Projekt magisterski

6.1. Cel projektu

Głównym celem projektu jest stworzenie narzędzia obsługiwanego z linii poleceń, które służy do analizy wydźwięku tekstów. Program dokonuje klasyfikacji komentarzy, przyznając im ocenę w skali od 1 do 10. W wyniku klasyfikacji program zwraca przewidzianą kategorię punktową. W ramach projektu dokonano również porównania jakości zaimplementowanych metod klasyfikacji.

6.2. Gromadzenie korpusu

W projekcie użyto zgromadzonego od podstaw zbioru komentarzy. Każdy komentarz składał się z dwóch elementów: treści oraz oceny w skali od 1 do 10. Korpus został pozyskany z serwisu *Cyber Academy* [5]. Zawarte w nim komentarze dotyczyły zajęć przeprowadzonych w latach 2014 – 2017 z następujących przedmiotów:

- Przetwarzanie Języka Naturalnego,
- Tworzenie Systemów Informatycznych,
- Uczenie Maszynowe.

Komentarze zostały wyekstrahowane z plików *html* za pomocą biblioteki *BeautifulSoup*, a następnie zapisane do pliku w formacie *tsv*. Przykładowy fragment pliku zawierający korpus komentarzy został przedstawiony na rysunku 6.1.

```
7   7/10 Ciekawy wykład, sporo przykładów i nowej wiedzy
10  Bardzo przyjemny wykład. Ocena: 10/10
9   9/10 Ciekawe wykonanie prezentacji.
8   8/10 Było troche ciekawych rzeczy.
8   8. Ciekawa forma wykładu.
5   5/10 wykład jak co dzień
9   9/10. Ciekawy wykład i jego tematyka.
4   Ten wykład zbytnio mnie nie porwał.
7   7/10 dość ciężki wykład
8   Bardzo dobry wykład 8/10
9   Ocena: 9/10 Dobra treść merytoryczna, sporo się nauczyłem.
4   4/10-nie wszystkie treści były zrozumiałe. Niepotrzebne wstawki kodu.
```

Rys. 6.1. Przykładowy fragment korpusu komentarzy.

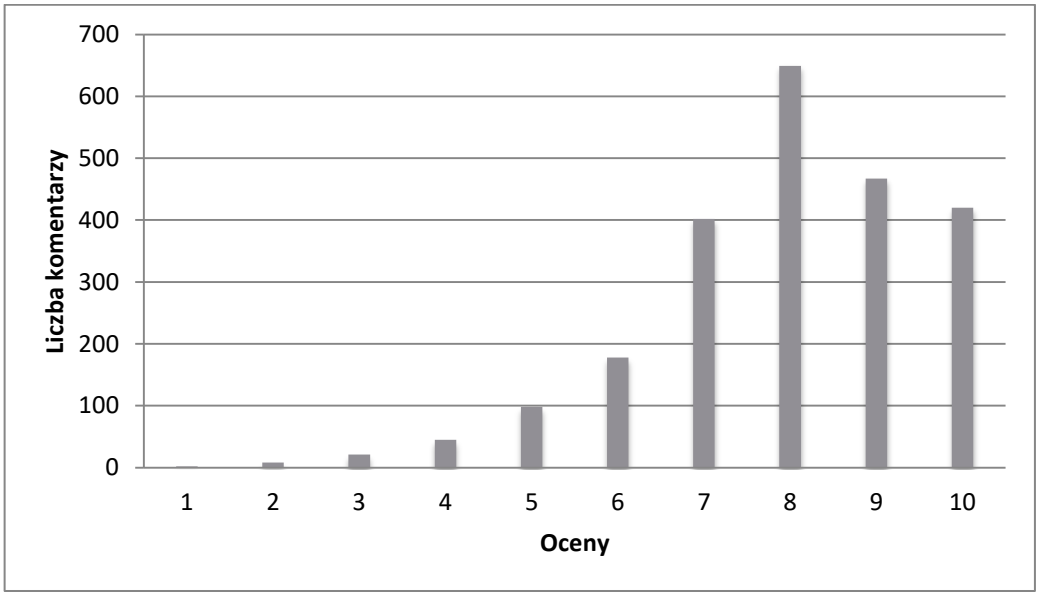
[5] https://cyberacademy.wmi.amu.edu.pl/ data dostępu: 17.11.2017

Łącznie pozyskano 2290 kompletnych komentarzy. Usunięto komentarze niekompletne – zawierające jedynie ocenę lub nie zawierające oceny oraz komentarze napisane w innym języku niż język Polski. Liczbowe podsumowanie korpusu przedstawiono w tabeli 6.1.

Ocena	Liczba
1	2
2	8
3	21
4	45
5	98
6	178
7	402
8	649
9	467
10	420
Łącznie	**2290**

Tab. 6.1. Liczbowe podsumowanie korpusu komentarzy.

Analizując dane zawarte w tabeli 6.1. zauważyć można bardzo nierównomierny rozkład komentarzy należących do poszczególnych klas, który został pokazany na rysunku 6.2.



Rys. 6.2. Rozkład komentarzy dla każdej z klas.

6.3. Przygotowanie danych do uczenia i testowania

Przygotowanie danych rozpoczęto od usunięcia ocen zawartych w treści komentarzy. Wykorzystano moduł *re*, w celu dopasowania wyrażenia regularnego, a następnie usunięto wszystkie liczby występujące w komentarzach. Proces czyszczenia komentarzy został przedstawiony na rysunku 6.3.

```
def clean(comment):
    comment_text = re.sub("\d+"," ", comment)

    return comment_text
```

Rys. 6.3. Usuwanie ocen z komentarzy.

W kolejnym etapie przeprowadzono tokenizację, czyli podział zdań na pojedyncze słowa oraz usunięcie znaków interpunkcyjnych. Wszystkie słowa zostały również sprowadzone do małych liter. Proces tokenizacji został przedstawiony na rysunku 6.4.

```
def tokenize(comment):
    comment_text = clean(comment)
    comment_clean = comment_text.translate(None, string.punctuation)
    words = comment_clean.lower().replace("\n", " ").split()

    return(words)
```

Rys. 6.4. Tokenizacja komentarzy.

Ostatnim etapem przygotowania danych był proces wektoryzacji zbioru danych przy użyciu funkcji *TfidfVectorizer*. Parametr *ngram_range* ustala dolną i górną granicę zakresu wartości *n* dla różnych n-gramów, które zostaną wyekstrahowane. W projekcie wykorzystano unigramy, bigramy oraz trigramy. Funkcja *TfidfVectorizer* buduje słownik, w którym cechy zostają uporządkowane według częstotliwości występowania słów i uwzględnia maksymalnie tyle cech, ile podano jako parametr *max_features*. W opisywanym projekcie parametrowi temu przypisano wartość 60 tys. cech. Proces wektoryzacji został przedstawiony na rysunku 6.5.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer( max_features = 60000, ngram_range = ( 1, 3 ),
    sublinear_tf = True )

clean_train_comments = []
for comment in train["komentarz"]:
    clean_train_comments.append( " ".join(tokenize(comment)))

train_x = vectorizer.fit_transform(clean_train_comments)


clean_test_comments = []
for comment in test["komentarz"]:
    clean_test_comments.append( " ".join(tokenize(comment)))

test_x = vectorizer.transform(clean_test_comments)
```

Rys. 6.5. Wektoryzacja cech.

6.4. Przeprowadzone eksperymenty

6.4.1. Klasyfikacja dziesięcioklasowa

Pierwszy eksperyment został przeprowadzony na danych należących do 10 klas, stanowiących oceny od 1 do 10. Rozkład komentarzy w poszczególnych klasach przedstawiono w tabeli 6.1. zawartej w rozdziale 6.2. Należy zwrócić uwagę na duże różnice w liczności komentarzy należących do poszczególnych klas.

Podczas eksperymentu przeprowadzono proces klasyfikacji za pomocą następujących metod uczenia maszynowego:

- naiwny klasyfikator Bayesa (NB),
- regresja logistyczna (LR),
- maszyna wektorów nośnych (SVM),
- sieci neuronowe (NN),
- k-najbliższych sąsiadów (kNN).

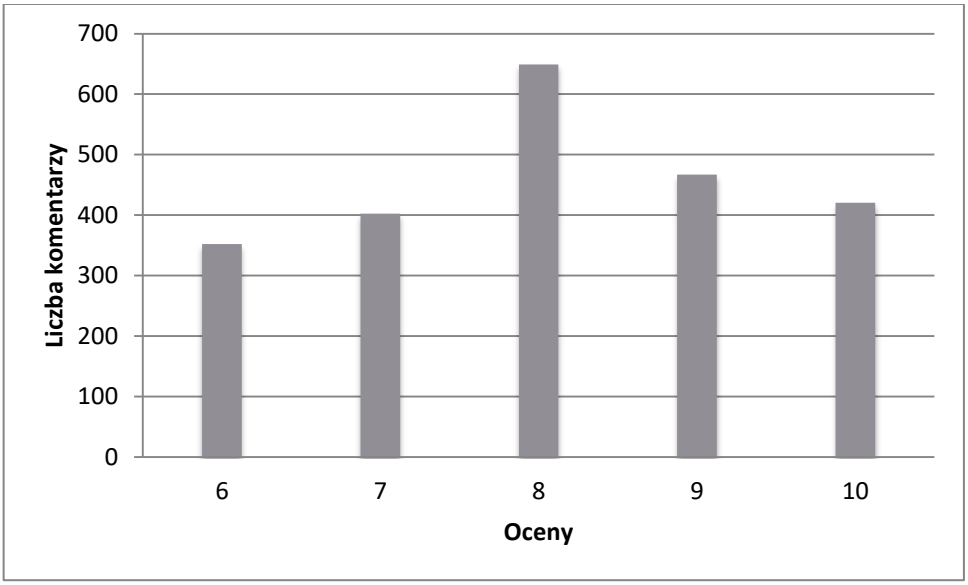
Dodatkowo w celu porównania zaimplementowanych metod, utworzono dodatkowy klasyfikator, nazwany klasyfikatorem naiwnym. Nie zawierał on etapu trenowania. Bez względu na dane wejściowe, przyznawał on każdemu komentarzowi najczęściej występującą klasę w całym zbiorze danych.

6.4.2. Klasyfikacja pięcioklasowa

Ze względu na bardzo nierównomierny rozkład komentarzy należących do 10 klas, przeprowadzono dodatkowy eksperyment polegający na zredukowaniu klas do 5 kategorii ocen oraz dokonania klasyfikacji przy użyciu tych samych metod uczenia maszynowego. Komentarze, które posiadały oceny od 1 do 6 zostały połączone w jedną klasę, której przypisano etykietę *6*. Podejście to pozwoliło uzyskać bardziej zbilansowane klasy. Liczbowe podsumowanie korpusu komentarzy zostało przedstawione w tabeli 6.2. Rozkład komentarzy dla każdej z pięciu klas zaprezentowano na rysunku 6.6.

Ocena	Liczba
6	352
7	402
8	649
9	467
10	420
Łącznie	**2290**

Tab. 6.2. Liczbowe podsumowanie korpusu komentarzy dla klasyfikacji pięcioklasowej.



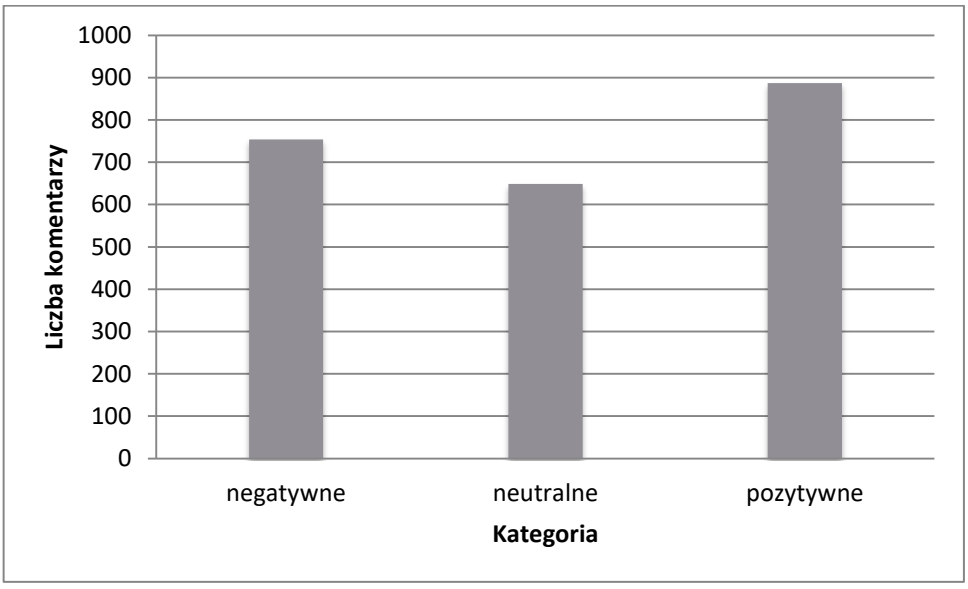
Rys. 6.6. Rozkład komentarzy dla każdej z klas.

6.4.3. Klasyfikacja trójklasowa

W życiu codziennym przeważnie nie wykorzystuje się skali ocen, aby sklasyfikować wypowiedzi, lecz dzieli się je na wypowiedzi zawierające wydźwięk pozytywny, negatywny lub neutralny. Ze względu na to popularne podejście, w projekcie autorskim wykonano dodatkowy eksperyment. Komentarze, które posiadały oceny od 1 do 7 zostały połączone w jedną klasę, której przypisano etykietę *negatywny*. Komentarze ocenione na 8 otrzymały etykietę *neutralny*, a komentarze posiadające ocenę 9 lub 10 otrzymały etykietę *pozytywny*. Takie podejście pozwoliło uzyskać znacznie lepiej zbilansowane klasy, niż przy pierwszych dwóch eksperymentach. Liczbowe podsumowanie korpusu komentarzy zostało przedstawione w tabeli 6.3. Rozkład komentarzy dla każdej z pięciu klas zaprezentowana na rysunku 6.7.

Kategoria	Liczba
negatywne	754
neutralne	649
pozytywne	887
Łącznie	**2290**

Tab. 6.3. Liczbowe podsumowanie korpusu komentarzy dla klasyfikacji trójklasowej.



Rys. 6.7. Rozkład komentarzy dla każdej z klas.

6.5. Porównanie metod uczenia maszynowego

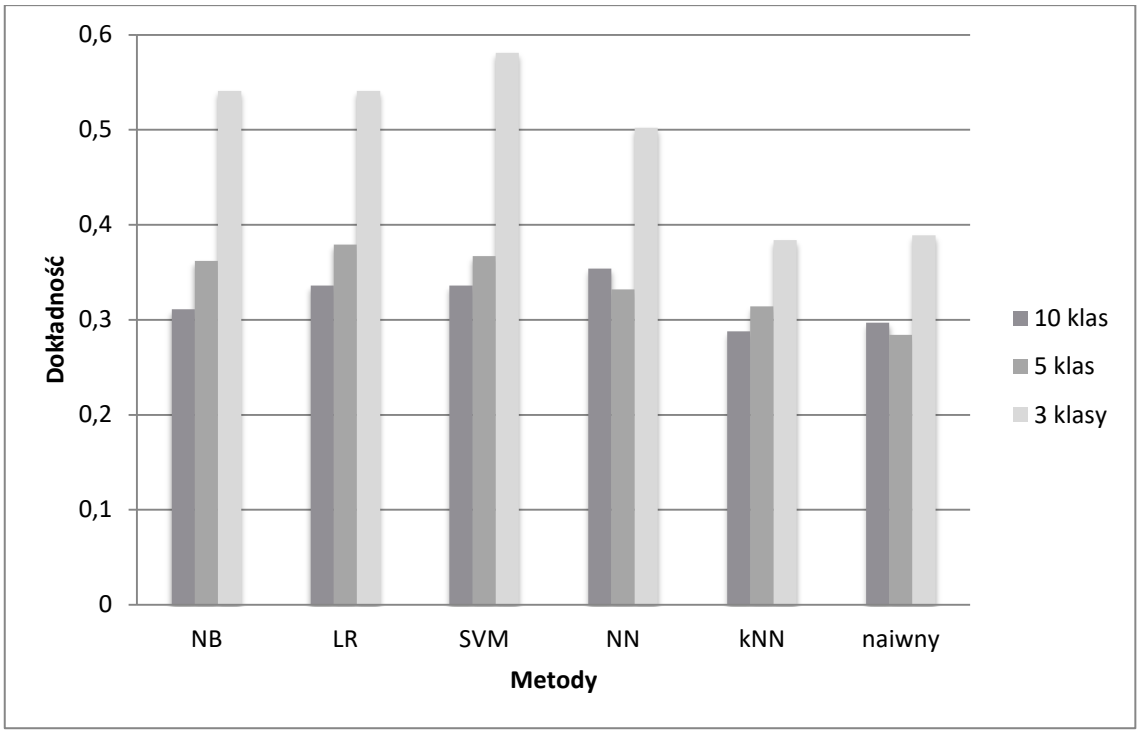
6.5.1. Dokładność

Wartości dokładności zostały zaprezentowane w tabeli 6.4. Dla klasyfikacji dziesięcioklasowej najlepszy wynik, wynoszący 0.354, uzyskały sieci neuronowe. W klasyfikacji pięcioklasowej najlepsza okazała się regresja logistyczna, która uzyskała dokładność równą 0.379. Dla klasyfikacji trójklasowej najlepszą metodą była maszyna wektorów nośnych z dokładnością wynoszącą 0.581.

Metoda	10 klas	5 klas	3 klasy
Naiwny klasyfikator Bayesa	0.311	0.362	0.541
regresja logistyczna	0.336	**0.379**	0.541
maszyna wektorów nośnych	0.336	0.367	**0.581**
sieci neuronowe	**0.354**	0.332	0.502
k-najbliższych sąsiadów	0.288	0.314	0.384
klasyfikator naiwny	0.297	0.284	0.389

Tab. 6.4. Porównanie wartości dokładności dla trzech eksperymentów.

Taki sposób oceny nie umożliwił wskazania najlepszej metody, ale pokazał, że metody te są lepsze niż klasyfikator naiwny, który dowolnemu komentarzowi przypisuje najczęściej występującą klasę w całym zbiorze. Porównanie wartości dokładności dla trzech eksperymentów przedstawiono na rysunku 6.8.



Rys. 6.8. Porównanie wartości dokładności dla trzech eksperymentów.

6.5.2. Pierwiastek błędu średniokwadratowego

Pierwiastek błędu średniokwadratowego (RMSE, ang. *root mean squared error*) można zapisać za pomocą poniższego wzoru:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

W powyższym wzorze n oznacza liczbę próbek, y_i oznacza kategorię punktową przyznaną przez klasyfikator, a $\hat{y}_i$ stanowi poprawną ocenę, przyznaną przez autora komentarza.

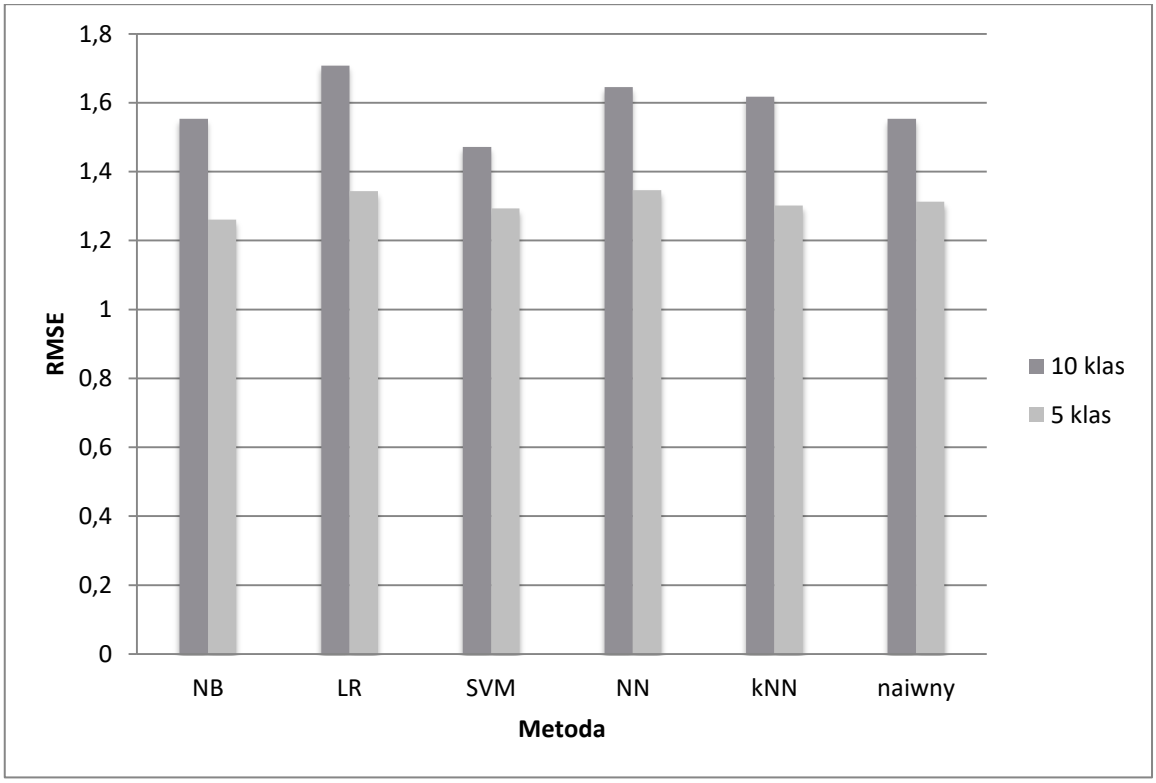
Miara RMSE pokazuje, o ile średnio myśli się dany klasyfikator w przyznawaniu kategorii punktowej. Wartości miary RMSE dla klasyfikatora dziesięcioklasowego oraz pięcioklasowego zostały przedstawione w tabeli 6.5. Dla klasyfikacji dziesięcioklasowej najniższą wartość miary RMSE, wynoszącą 1.472, uzyskała maszyna wektorów nośnych. W problemie klasyfikacji pięcioklasowej najlepszy

okazał się naiwny klasyfikator Bayesa, który uzyskał wartość miary RMSE wynoszącą 1.261.

Metoda	10 klas	5 klas
naiwny klasyfikator Bayesa	1.553	**1.261**
regresja logistyczna	1.708	1.343
maszyna wektorów nośnych	**1.472**	1.293
sieci neuronowe	1.645	1.346
k-najbliższych sąsiadów	1.617	1.302
klasyfikator naiwny	1.553	1.313

Tab. 6.5. Porównanie wartości miary RMSE dla dwóch eksperymentów.

Metoda ta również nie wskazuje, który z zaimplementowanych klasyfikatorów jest najlepszy. Zauważyć można, że każda z metod myli się o ok. 1.5 w przypadku dziesięciu klas oraz o ok. 1.3 w klasyfikacji pięcioklasowej. Porównanie wartości miary RMSE przedstawiono na rysunku 6.9.



Rys. 6.9. Porównanie wartości miary RMSE dla dwóch eksperymentów.

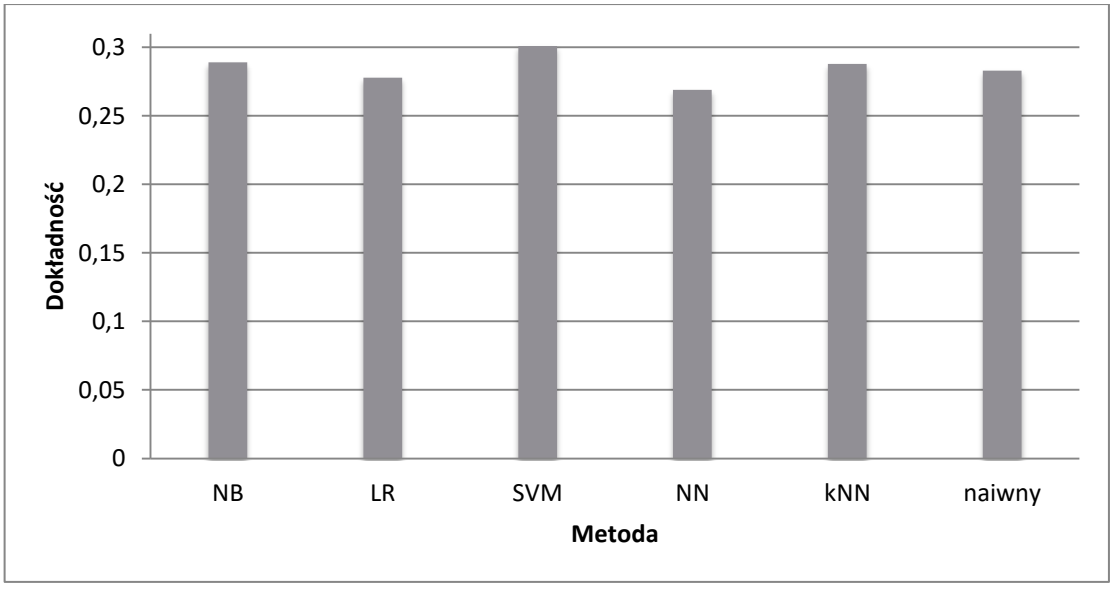
6.5.3. Walidacja krzyżowa

W projekcie autorskim dokonano walidacji krzyżowej przy użyciu funkcji *StratifiedKFold*, która podzieliła zbiór komentarzy 10 razy na zbiór uczący i zbiór testowy, przy zachowaniu procentowego rozłożenia próbek w zbiorach. 90% zbioru stanowił zbiór trenujący, a 10% stanowił zbiór testowy. Wartości dokładności oraz miary RMSE dla klasyfikacji dziesięcioklasowej zostały przedstawione w tabeli 6.6.

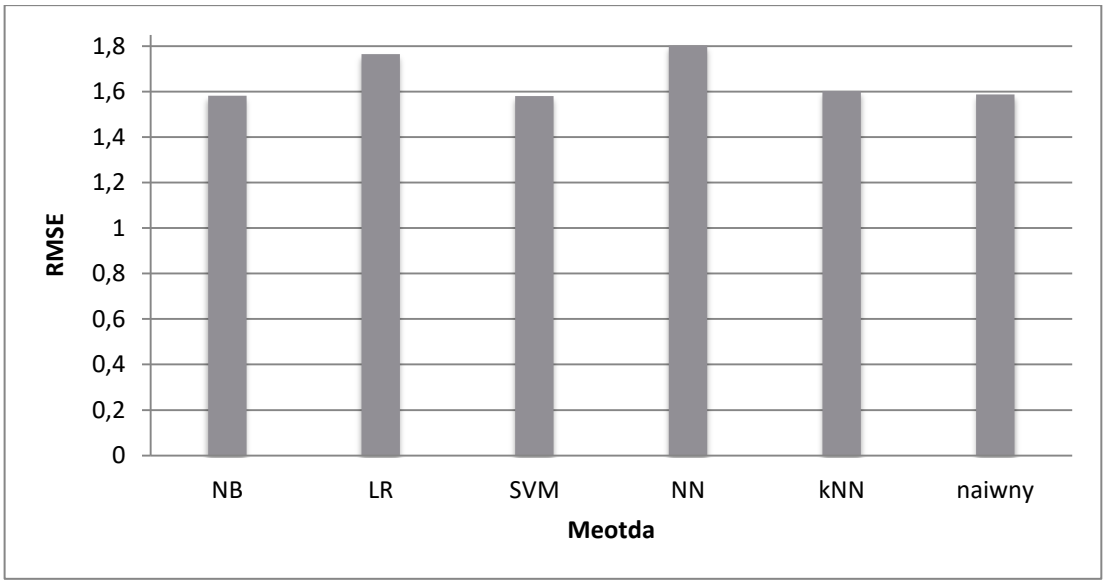
Metoda	Dokładność	RMSE
naiwny klasyfikator Bayesa	0.289	1.582
regresja logistyczna	0.278	1.765
maszyna wektorów nośnych	**0.301**	**1.581**
sieci neuronowe	0.269	1.803
k-najbliższych sąsiadów	0.288	1.603
klasyfikator naiwny	0.283	1.587

Tab. 6.6. Porównanie wartości dokładności oraz miary RMSE dla klasyfikacji dziesięcioklasowej.

W tym przypadku można stwierdzić, że najlepszą metodą jest maszyna wektorów nośnych, która uzyskała najwyższą wartość dokładności, wynoszącą 0.301 oraz najniższą wartość miary RMSE, wynoszącą 1.581. Porównanie wartości dokładności przedstawiono na rysunku 5.7. a wartości miary RMSE na rysunku 6.8



Rys. 6.7. Porównanie wartości dokładności dla klasyfikacji dziesięcioklasowej.



Rys. 6.8. Porównanie wartości miary RMSE dla klasyfikacji dziesięcioklasowej.

6.5.4. Porównanie z innymi eksperymentami

W rozdziale 4.3. przedstawiono opis i wyniki eksperymentu polegającego na klasyfikacji wypowiedzi użytkowników serwisu *Tweeter*. Eksperyment ten uzyskał prawie dwukrotnie wyższą średnią dokładność klasyfikacji wynoszącą 0.602. Średnia wartość dokładności dla eksperymentu autorskiego wynosiła 0.323.

Głównym podobieństwem obu eksperymentów była klasyfikacja wieloklasowa. Natomiast podstawową różnicą pomiędzy nimi jest to, że klas eksperymentu klasyfikującego *Tweety* nie można uporządkować rosnąco. Kategorie nie zależą od siebie, a w niektórych przypadkach wręcz wzajemnie się wykluczają – np. wypowiedź nie może jednocześnie być neutralna i wyrażać gniewu.

Kolejną różnicą miedzy eksperymentami jest to, że wypowiedzi użytkowników serwisu *Tweeter* przydzielono ręcznie do wybranych kategorii, a nie zostały one przyporządkowane bezpośrednio przez ich autorów, jak w eksperymencie autorskim.

Ważnym aspektem, odróżniającym eksperymenty autorskie od wszystkich pozostałych projektów jest rozmiar korpusu wykorzystanego do trenowania i testowania. W eksperymentach opisanych w rozdziałach 4.2. oraz 4.3. wykorzystano zbiory uczące i testowe zawierające po ok. 20 tys. wypowiedzi. W projekcie autorskim cały zbiór komentarzy był prawie 10 razy mniejszy i wynosił 2290 komentarzy.

6.6. Przykładowe wyniki

10/10 Wykład jako przedstawienie prezentacji był bardzo ciekawy.	
naiwny klasyfikator Bayesa	**10**
regresja logistyczna	**10**
maszyna wektorów nośnych	**10**
sieci neuronowe	**10**
k-najbliższych sąsiadów	**10**

Tab. 6.7. Przykład poprawnie sklasyfikowany wszystkimi metodami.

Ciekawe przykłady i trafne wnioski. Ocena: 7	
naiwny klasyfikator Bayesa	**8**
regresja logistyczna	**8**
maszyna wektorów nośnych	**8**
sieci neuronowe	**8**
k-najbliższych sąsiadów	**8**

Tab. 6.8. Przykład niepoprawnie sklasyfikowany wszystkimi metodami.

6/10 Wykład ciekawszy, niż dwa poprzednie, jednak wydaje się, że chwilami treści na wykładzie były zaprezentowane zbyt szybko.	
naiwny klasyfikator Bayesa	**8**
regresja logistyczna	**6**
maszyna wektorów nośnych	**8**
sieci neuronowe	**9**
k-najbliższych sąsiadów	**8**

Tab. 6.9. Przykład poprawnie sklasyfikowany tylko za pomocą jednej metody.

Dużo logiki mało informatyki. 8/10	
naiwny klasyfikator Bayesa	**8**
regresja logistyczna	**6**
maszyna wektorów nośnych	**8**
sieci neuronowe	**5**
k-najbliższych sąsiadów	**8**

Tab. 6.10. Przykład poprawnie sklasyfikowany przez trzy z pięciu metod.

Wykład w porządku ale pytania (zadania) czasem były nie do końca jasne. Moja ocena: 7	
naiwny klasyfikator Bayesa	**9**
regresja logistyczna	**9**
maszyna wektorów nośnych	**7**
sieci neuronowe	**7**
k-najbliższych sąsiadów	**8**

Tab. 6.11. Przykład poprawnie sklasyfikowany przez dwie z pięciu metod.

6.7. Wnioski

Zbiór zawierający 2290 komentarzy, w którym występuje 10 klas, jest niewystarczający do otrzymania zadowalających wyników. Przy tak małym zbiorze redukcja klas o połowę nie poprawia znacząco dokładności klasyfikacji. Aby otrzymać zadowalające wyniki, należy dążyć do jak najbardziej zbilansowanych klas.

Analiza wydźwięku jest trudnym zagadnieniem, ze względu na subiektywność komentarzy. Każdy człowiek ocenia zjawiska, produkty, czy usługi w bardzo subiektywny i indywidualny sposób, ponieważ każdy inaczej postrzega otaczającą go rzeczywistość. Przykładem tego mogą być wypowiedzi wchodzące w skład korpusu przedstawione poniżej:

- *7/10 Przeciętny wykład.*
- *7/10 Ciekawy wykład, sporo przykładów i nowej wiedzy*

- *Ocena: 7/10 Komentarz: Ok*
- *7 ciekawe informacje, ale słabe przygotowanie (niedziałające piloty)*
- *Zadanie grupowe są raczej średnie, za duży chaos i wiele kwestii spornych. Ocena 7/10*
- *Ciekawy pomysł na prowadzenie wykładu. Interesująca odskocznia od tradycyjnych zajęć. 7/10*

Każdy z powyższych komentarzy uzyskał ocenę 7, jednak każdy z nich różni się wydźwiękiem.

Otrzymane wyniki eksperymentów autorskich nie umożliwiają wyboru najlepszej metody klasyfikacji. W przeprowadzonym eksperymencie, bez względu na typ klasyfikatorów – czy są to drzewa decyzyjne, klasyfikatory probabilistyczne, czy klasyfikatory z liniowymi współczynnikami, każda z metod uzyskuje zbliżone wartości dokładności oraz miary RMSE.

7. Podsumowanie

Celem projektu zaprezentowanego w niniejszej pracy było stworzenie narzędzia służącego do automatycznej analizy wydźwięku komentarzy studentów do wykładu. Zgromadzono korpus 2290 komentarzy należących do 10 klas. Powstały korpus użyto do trenowania oraz testowania. Przeprowadzono trzy eksperymenty polegające na klasyfikacji dziesięcioklasowej, pięcioklasowej oraz trójklasowej. W każdym eksperymencie przetestowano pięć tych samych metod uczenia maszynowego oraz porównano wyniki ewaluacji.

Otrzymane wyniki ewaluacji są gorsze niż wyniki otrzymane w innych projektach, opisanych w niniejszej pracy. Głównym problemem w każdym z przeprowadzonych eksperymentów autorskich był zbyt mały korpus komentarzy. Ze względu na swoją specyfikę – komentarze studentów dotyczące wykładów, nie istniała możliwość uzyskania komentarzy pochodzących z innych źródeł niż serwis *Cyber Academy*.

System ten w postaci zaprezentowanej w niniejszej pracy umożliwia wprowadzenie rozmaitych ulepszeń. Stanowi bazę do dalszego rozwoju w kierunku aplikacji z graficznym interfejsem użytkownika. Istnieje możliwość zaimplementowania innych metod uczenia maszynowego, które nie zostały wybrane w projekcie, a mogłyby uzyskać lepsze wyniki np. drzewa decyzyjne (ang. *Decision Trees*). Aby poprawić dokładność działania narzędzia należałoby wytrenować zaimplementowane metody na znacznie większym korpusie komentarzy, co nie było możliwe w chwili tworzenia projektu.

Bibliografia

[BOUA, 2017] Mondher Bouazizi, Tomoaki Ohtsuki, A Pattern-Based Approach for Multi-Class Sentiment Analysis in Twitter, 2017.

[BROW, 2016] Jason Brownlee, Master Machine Learning Algorithms, 2016.

[CHAK, 2012] Abhijit Chakankar, Sanjukta, Pal Mathur, Krishna Venuturimilli, Sentiment analysis of users' reviews and comments, 2012.

[CHIH, 2002] Chih-Wei Hsu, Chih-Jen Lin, A Comparison of Methods for Multiclass Support Vector Machines, IEEE Transactions on Neural Networks, s.415- 425, 2002.

[JURA, 2012] Dan Jurafsky, Christopher Manning, Sentiment Analysis – Stanford NLP, https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html, 2012.

[KRIE, 2007] David Kriesel, A Brief Introduction to Neural Networks, 2007.

[LEE, 2002] Lillian Lee, Bo Pang, Thumbs up? Sentiment Classification using Machine Learning Techniques, 2002.

[LIU, 2012] Bing Liu, Sentiment Analysis and Opinion Mining, 2012.

[MCCA, 1998] Andrew McCallum, Kamal Nigam, A Comparison of Event Models for Naive Bayes Text Classification, 1998.

[MEDH, 2014] Walaa Medhat, Ahmed Hassan, Hoda Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal, t. 5, s. 1093–1113, 2014.

[SOKO, 2006] Marina Sokolova, Nathalie Japkowicz, Stan Szpakowicz - Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation, AI 2006: Advances in Artificial Intelligence, s. 1015-1021, 2006.

[STEF, 2016] Jerzy Stefanowski, Sztuczne sieci neuronowe, 2016.

[TADE, 2007] Ryszard Tadeusiewicz, Odkrywanie właściwości sieci neuronowych, 2007.

[TOMA, 2014] Tomanek Krzysztof, Analiza sentymentu – metoda analizy danych jakościowych. Przykład zastosowania oraz ewaluacja słownika RID i metody

klasyfikacji Bayesa w analizie danych jakościowych, Przegląd Socjologii Jakościowej, t. 10, nr 2, s. 118–136, 2014.

[XIAN, 2017] Xiangfeng Dai, Irena Spasic, Frederic Andres, A Framework for Automated Rating of Online Reviews Against the Underlying Topics. Proceedings of the SouthEast Conference, str.164–167, 2017.

[ZHAN, 2000] Guoqiang Peter Zhang, Neural Networks for Classification: A Survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C :Applications and Reviews, 2000.

[ZHAN, 2015] Dongwen Zhang, Hua Xu, Zengcai Su, Yunfeng Xu, Chinese comments sentiment classification based on word2vec and SVM^{perf}, Expert Systems with Applications, t. 42, s. 1857-1863, 2015.

Spis rysunków

[1] Walaa Medhat, Ahmed Hassan, Hoda Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal, t. 5, s. 1094, 2014.

[2] Walaa Medhat, Ahmed Hassan, Hoda Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal, t. 5, s. 1095, 2014.

[3] Data pobrania zasobu: 26.02.2018
http://bdewilde.github.io/assets/images/2012-10-26-knn-concept.png

[4] Data pobrania zasobu 26.02.2018
https://docs.opencv.org/2.4/_images/separating-lines.png

[5] Data pobrania zasobu 26.02.2018
https://docs.opencv.org/2.4/_images/optimal-hyperplane.png

[6] Data pobrania zasobu: 27.02.2018
https://plot.ly/~florianh/140.png

[7] Jerzy Stefanowski, Sztuczne sieci neuronowe, s. 3, 2016.

[8] Data pobrania zasobu: 03.03.2018
http://neuralnetworksanddeeplearning.com/images/tikz11.png

[9] Data pobrania zasobu: 06.05.2018
https://i.imgur.com/IDFRq5a.png

[10] Data pobrania zasobu 07.03.2018
http://edu.pjwstk.edu.pl/wyklady/adn/scb/wyklad9/w9_files/image017.jpg

[11] Data pobrania zasobu: 11.03.2018
https://www.h5.com/wp-content/uploads/2017/05/recall-precision-tradeoff.png