AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ GEOLOGII, GEOFIZYKI I OCHRONY ŚRODOWISKA

Praca inżynierska

Zastosowanie uczenia maszynowego w wykrywaniu stanów depresyjnych na podstawie analizy komentarzy i tekstów użytkowników sieci internetowej

Application of Machine Learning in Detecting Depressive States Based on Analysis of User Comments and Texts on the Internet

Autor:	*Kacper Müller*
Kierunek studiów:	*Inżynieria i analiza danych*
Opiekun pracy:	*dr inż. Mateusz Zaręba*

Kraków, 2023

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Dla Antka i Kacpra

Spis treści

1. Wprowadzenie

Praca skupia się na wykorzystaniu sztucznej inteligencji do wykrywania stanów depresyjnych u użytkowników sieci internetowej. Badane są modele które, na podstawie analizy komentarzy i tekstów użytkowników, będą w stanie automatycznie rozpoznawać, czy dana osoba jest w stanie depresyjnym. W tym celu zostaną wykorzystane techniki przetwarzania języka naturalnego, algorytmy uczenia maszynowego i uczenia głębokiego.

1.1. Opis problemu

Coraz więcej osób, szczególnie młodych, doświadcza stanów depresyjnych [1]. Coraz więcej osób, próbuje odebrać sobie życie [2]. W latach 2021 i 2022 zauważono nagły wzrost prób samobójczych w przedziałach wiekowych 7-12 oraz 13-18 [3]. Przyczyny wzrostu samobójstw mogą być różne, jedną z nich mogła być pandemia koronawirusa, która wybuchła w 2019 roku [4].

Pomoc takim osobom można nieść dopiero po wykryciu stanu depresyjnego. Sposobem, aby pomóc w wykrywaniu tego, może być sprawdzanie treści udostępnianych w internecie. Problemem jednak jest to, że ilość udostępnianych danych w internecie jest ogromna i cały czas wzrasta szybkość ich przybywania. Zespoły moderatorów forów, stron, czy grup nie są w stanie wszystkiego kontrolować — polegają oni na zgłoszeniach treści przez użytkowników, ale i takich przypadków jest za dużo. Z pomocą przychodzą algorytmy sztucznej inteligencji, które mogą bardzo szybko (nawet na bieżąco) przetwarzać treści, jakie użytkownicy przesyłają do internetu. To właśnie modele uczenia maszynowego, jeśli zostaną dobrze wyuczone, mogą z dużym stopniem wiarygodności stwierdzić, iż użytkownik jest w stanie depresyjnym i przesłać tę informację do weryfikacji np. moderatorowi, czy psychologowi. Ważne jest, aby ostateczną decyzję podejmował człowiek, gdyż zawsze należy pamiętać, że modele mogą się mylić.

1.2. Cel pracy

Celem pracy jest pomoc ludziom cierpiącym na depresję poprzez stworzenie jak najlepszego modelu sztucznej inteligencji, który może pomóc w wykrywaniu stanu depresyjnego użytkownika, na podstawie tego co opublikował w sieci.

2. Wstęp teoretyczny

2.1. Depresja

Depresja nie jest zwykłym poczuciem smutku. Jest to dużo bardziej złożone zaburzenie psychiczne. Objawami są również niska pewność siebie, uczucie pustki, niemożliwość odczuwania przyjemności czy radości z życia, brak apetytu, bezsenność lub nadmierna senność, myśli samobójcze. Czasami osoby chore nie odczuwają smutku, lecz występują u nich inne wyżej wymienione objawy. Depresja przeszkadza również w koncentracji, zaburza pamięć, utrudnia wykonywanie nawet prostych czynności fizycznych i znacząco utrudnia kontakt z innymi.

Przyczyny depresji szukane są w różnych aspektach, między innymi biologicznym (np. zaburzenia neuroprzekaźników) lub psychicznym (np. występowanie konfliktów między pragnieniami a ideałami, do których osoby dążą). Ważnym jest, aby pamiętać, że depresja to nie choroba, z której można się wyleczyć, zmieniając swoje nastawienie. Trudno jest się z niej wyleczyć, przede wszystkim ze względu na to, jak bardzo utrudnia normalne funkcjonowanie człowieka.

Istnieją metody na leczenie depresji. Często stosowane są leki przeciwdepresyjne, ale mają one skutki uboczne i mogą uzależnić. Można również stosować leczenie elektrowstrząsami (sportretowane w filmie Lot nad kukułczym gniazdem), rTMS, czy psychoterapię. [5] [6] [7] [8] [9] [10] [11] [12]

2.2. Sztuczna inteligencja

Sztuczna inteligencja to inteligencja wykazywana przez artefakty, czyli przedmioty sztuczne, nie ludzi, zwierzęta, czy przyrodę. Obecnie sztuczna inteligencja to głównie programy komputerowe i mniej lub bardziej zaawansowane algorytmy, w dużej mierze wykorzystujące statystykę. Obecnie najczęściej używanymi metodami SI, są te zaliczane do uczenia maszynowego i uczenia głębokiego (które jest podzbiorem uczenia maszynowego).

2.2.1. Uczenie maszynowe

Uczenie maszynowe to podzbiór metod sztucznej inteligencji. Algorytmy z tej grupy wykorzystują dane, aby się na nich uczyć i podejmować coraz to lepsze decyzje. Im więcej dostępnych danych, tym lepiej będą wykonywać swoje zadanie.

Algorytmy te można podzielić na przyjmowane przez nich dane — sposób uczenia się. Wyróżniane są cztery rodzaje uczenia się modeli: nadzorowane, nienadzorowane, częściowo nadzorowane oraz wzmocnione. [13]

1. W uczeniu nadzorowanym dane wejściowe muszą posiadać również oczekiwany wynik. Do uczenia tego zaliczamy algorytmy klasyfikacji (na których skupia się ta praca) oraz predykcji. Przykładowo: chcąc przewidywać cenę domu, na podstawie jego powierzchni, musimy podać algorytmowi dane uczące go, składające się zarówno z powierzchni jak i ceny, którą będziemy chcieli przewidywać. Dopiero tak nauczony algorytm będzie w stanie przewidywać ceny domów o nowych, wcześniej niepodanych powierzchniach.

2. W uczeniu nienadzorowanym algorytm nie potrzebuje w ciągu uczącym oczekiwanych wyników, gdyż sam je generuje. Algorytmy w tej kategorii zajmują się klasteryzacją. Przykładowo: chcąc, aby algorytm podzielił grupę ludzi na otyłych i nieotyłych, podajemy jedynie masy ludzi i ilość grup, na jakie chcemy ich podzielić — w tym przypadku 2. Nie podajemy natomiast początkowo, do jakiej grupy należy która osoba.

3. Algorytmy częściowo nadzorowane są połączeniem obu wcześniej wymienionych grup.

4. W uczeniu wzmocnionym model dostaje reguły i dozwolone działania. Korzystając z dozwolonych działań, obserwuje ich skutki i uczy się, aby zbliżać się coraz bardziej do zadanego wyniku. Przykładowo: samochód autonomiczny dostaje zasady ruchu drogowego i dozwolone akcje: poruszanie się do przodu, do tyłu i skręcanie, a jego zadaniem jest dotrzeć do celu — mniejszy dystans do celu (bez łamania reguł) oznacza, że podjął właściwe kroki.

Warto zwrócić tutaj uwagę na podział danych na dane uczące i testowe. Modele sztucznej inteligencji uczą się na części danych, ale ocena ich polega na sprawdzaniu ich działania na innej części — wcześniej niewidzianej przez model. Unika się przez to przeuczenia modelu (overfit'u), czyli sytuacji gdzie model idealnie ocenia dane, na których był uczony, lecz kiepsko radzi sobie z nowymi danymi.

2.2.2. Uczenie głębokie

Uczenie głębokie wykorzystuje sztuczne sieci neuronowe, na wzór najlepszej maszyny do podejmowania decyzji, jaką jest ludzki mózg. Występują tutaj grupy sztucznych neuronów — tworzące warstwy sieci, które są ze sobą połączone.

Słowo „głębokie" oznacza, w tym wypadku, że sieć posiada wiele warstw ukrytych, czyli takich pomiędzy warstwami wejściowymi i wyjściowymi. Obecnie stosuje się bardzo złożone sieci, ze skomplikowanymi, pod względem działania, neuronami. Łączą się one w architektury, które też często są przeplatane ze sobą. Dzięki swojej złożoności dają one imponujące wyniki, w postaci generowania tekstów i obrazów, zbliżonych do tych generowanych przez człowieka. Są to przykładowo modele: GPT, Llama, Dall-e, czy Stable Diffusion.

Minusem jednak tak skomplikowanych modeli jest to, że wymagają bardzo dużo mocy obliczeniowej, a przez to energii [14], aby je wyuczyć i używać. Dodatkowo, przez swoje skomplikowanie, metody uczenia głębokiego są czarnymi skrzynkami. Decyzje podejmowane przez metody standardowego uczenia maszynowego są wytłumaczalne: można zobaczyć, przykładowo, jak wyglądają stworzone drzewa decyzyjne i dlaczego model podjął taką, a nie inną decyzję. Metody uczenia głębokiego to zbiór wag na neuronach, przez co odczytanie i zrozumienie, dlaczego model podjął taką, a nie inną decyzję, jest bardzo trudne i czasochłonne.

2.3. Przetwarzanie języka

Należy pamiętać o jednej ważnej rzeczy: algorytmy sztucznej inteligencji działają na liczbach, nie na wyrazach. Pierwszym problemem, jaki trzeba rozwiązać, chcąc przetwarzać tekst, jest to, w jaki sposób przedstawić zdania za pomocą liczb. Rozwiązanie opisane będzie w rozdziale 3.3. Po rozwiązaniu tego należy zdać sobie sprawę, jak ciężko przedstawić język, jako funkcję matematyczną. Nie jest to jednak nic dziwnego, gdyż rozwijany jest on od około 70 000 lat [15]. Dziedzina, która zajmuje się takimi problemami, nazywana jest przetwarzaniem języka naturalnego (natural language processing — NLP). [16]

Większość modeli w pracy będzie ignorowała bardzo istotną część języka - kolejność słów, czy gramatykę. Mimo tego, wyniki takiej klasyfikacji, będą bardzo dobre. Zbadane zostaną również i takie modele, które będą to uwzględniały.

3. Metody

3.1. Dane

Dane zostały pobrane ze strony Reddit. Strona ta łączy różnego rodzaju fora. Do pobrania postów z nich, została użyta biblioteka Pythona: PRAW. Aby uniknąć bias'u modelu więcej danych (stosunek 2:1), jest niedepresyjnych – 'non-suicidal'. Są to dane pobrane z forów (zwanych subreddit'ami): 'learnprogramming', 'patientgamers', 'midlyinfuriating', 'antiwork', 'ThoughtsYouCanFeel', 'Advice', 'rant', 'CasualConversation', 'Needafriend', 'changemyview', 'DebateReligion', 'teenagers', 'PoliticalDiscussion', 'truegaming' oraz 'relationships', z każdego z nich zostało pobrane około 1000 postów. Fora zostały dobrane tak, aby posty z nich dotyczyły różnych tematów, np. gry komputerowe, programowanie, ogólne porady, religie, relacje międzyludzkie. Zostały także dodane fora, które mogą być bardziej problematyczne dla modelu, np. 'midlyinfuriating' lub 'rant'. Użytkownicy postują tam swoje narzekania, często używając słownictwa nacechowanego negatywnie, ale nie mogą być one zaliczane, przez model, do danych depresyjnych.

Dane depresyjne – 'suicidal' pobrane zostały z forów: 'SuicideWatch' oraz 'depression_help'. Z pierwszego zostało pobrane około 5000 postów, a z drugiego około 3000. Na subreddit'ach tych użytkownicy piszą o walce z depresją i myślami samobójczymi.

Dane zostały od razu poddane wstępnemu oczyszczeniu, co opisane jest w sekcjach 3.1.1 oraz 3.1.2. Łącznie, po tym zabiegu, zostało około 20500 wierszy, z czego około 7500 to dane suicidal.

3.1.1. Dane — detale

Dane były pobierane po około 1000 najnowszych postów z każdego subreddita. Po pobraniu od razu usuwane były z tekstu wyrażenia typu: '[m23]', '(23F)' (oznaczające płeć i wiek osoby piszącej), 'tldr', czy linki url. Nie usunięcie tych wyrazów mogłoby powodować, że model podejmuje decyzje w oparciu o informacje, które nie powinny zmieniać wyniku klasyfikacji. Posty później łączone były w tabele (osobne dla dwóch kategorii: suicidal i nonsuicidal). Tabele te składały się z kolumn:

– Indeks numeryczny

– ID — ID posta, potrzebne z przyczyn technicznych, do zapamiętania ostatnio zapisanego posta

– Title — tytuł posta

– Is text? — kolumna wskazująca czy post składa się z tekstu (wtedy wartość 1), czy np. z wideo, obrazka itp. (wartość 0)

– Text — zawartość posta (NA w przypadku, gdy kolumna ’Is text?’ posiada wartość 0)

	ID	Title	Is text?	Text
0	15oxhb8	Looking for junior dev students!	True	Last week I created a post about me wanting to...
1	15ox85b	How to run code submitted on website safely on...	True	I want to create a sort of Leetcode clone for ...
2	15ownfp	Why does copy and sudoku change, when I'm copy...	True	sudoku=\[\r \n\[0, 0, 0, 0, 0, 0, 0, 0, 0\],\...
3	15owguy	Any free/low cost resources for learning email...	True	I'm currently a Bootcamp student, and I don't ...
4	15ovzi1	[C#] Can't access methods from another C# file.	True	Here's my main that is trying to access that c...

Rys. 3.1. Początek tabeli non-suicidal

3.1.2. Czyszczenie danych

Następnie dane zostały kolejny raz oczyszczone. Zostawiono tylko te wiersze, które miały wartość 1 kolumny ’Is text?’, a wszystkie białe znaki zamienione zostały w pojedynczą spację. Następnie te posty, które w tytule miały wartości NA, zostały usunięte, a NA w tekście zostały zamienione na ” (pusty łańcuch znaków). Tytuł został połączony z tekstem: do końca tytułu zostało dodane ’. ’ i tak zmieniony został dołączony na początek kolumny z treścią posta. Ostateczna tabela składała się z kolumn:

– Indeks numeryczny

– Text — połączony tekst: tytuł i zawartości posta

– Suicidal — wskazuje czy post pochodzi z forów suicidal (wartość 1), czy nie (wartość 0)

	Text	Suicidal
0	i think im going to kill myself soon. i just h...	1
1	i do not genetically qualify for the human exp...	1
2	how long have you been uninterested in living?...	1
3	nobody loves me, i’ll never be loved. i’m so d...	1
4	i’m ready to go. this isn’t worth it anymore. ...	1
...	...	...
20584	why did my girlfriend break up with me?. my gi...	0
20585	my bf and i are going through a rough patch ...	0
20586	my friend has never been in a romantic relati...	0
20587	i slept with another man while separated from...	0
20588	girlfriend doesn’t want to spend any time with...	0

Rys. 3.2. Początek i koniec tabeli głównej

3.2. Natural Language Processing

Jak opisane to zostało w rozdziale 2.3, dziedzina NLP zajmuje się przetwarzaniem języka ludzkiego przez komputery. Najprostszą metodą na to jest zliczenie słów i używanie tych liczb w modelach — opisane w rozdziale 3.3.1. Istnieje jednak wiele innych technik i dobrych praktyk, które polepszają działanie modeli.

Przykładowo, można stosować bardziej złożone techniki zliczania słów, jak na przykład TF-IDF, opisane w rozdziale 3.3.2. Można również zliczać występowanie par, lub trójek słów — ze zdania „Ala ma kota", zamiast podzielić zdanie na pojedyncze słowa „Ala", „ma" i „kota", można użyć „Ala ma" i „ma kota". Są to tak zwane n-gramy. Zanim jednak przejdziemy do zliczania słów, można zastanowić się nad usuwaniem słów, które nie wznoszą żadnej informacji do tekstu, a jedynie upiększają go. Są to tak zwane stop words. Przykładowo, w języku angielskim, będą to słowa typu: the, a, an.

Kolejnym sposobem polepszenia wyników modeli jest normalizacja. Polega ona na między innymi zamianie wszystkich liter na małe, rozwijaniu skrótów, usuwaniu znaków interpunkcyjnych i typograficznych. Można także spróbować metod stemming'u, czyli usuwania końcówek słów: -ing i tym podobnych. [17]

Należy pamiętać, że nie w każdym przypadku wymienione sposoby będą działały. Czasem mogą one popsuć model, dlatego też należy badać wpływ poszczególnych metod na wyniki.

3.3. Wektoryzacja słów

Algorytmy posługują się liczbami, aby więc mogły klasyfikować słowa, należy najpierw zamienić tekst na ciąg liczb. Na początku została sprawdzona metoda Count Vectorizer, a później TF-IDF. Dlatego, że ta druga daje dużo lepsze wyniki, Count Vectorizer został zastosowany tylko w pierwszym modelu. Wejściem dla modelu będzie wektor o długości równej ilości różnych słów we wszystkich dokumentach (tekstach z postów). Długość ta może być jednak zmniejszona, używając tylko np. 1000 najbardziej popularnych słów. Dla jednego tekstu większość wartości będzie równa 0, gdyż odpowiadające jej słowo nie będzie występowało.

3.3.1. Count Vectorizer

Count Vectorizer, jak sama nazwa wskazuje, tworzy wektor na podstawie czystej ilości wystąpień wyrazów [18]. Kłopotem są słowa często występujące w zdaniach, szczególnie w języku angielskim, typu: a, an, the, and. Słowa te nie powinny wpływać mocno na wynik klasyfikacji, gdyż nie przekazują konkretnej informacji, są tylko upiększeniem języka. Niemniej jednak mogą mieć znaczenie. Może się okazać, że np. osoby w ciężkim stanie psychicznym mogą zwracać mniejszą uwagę na słowa mające małe znaczenie, a skupiają się na tych bardziej znaczących.

3.3.2. TF-IDF

Drugim sposobem była metoda TF-IDF [19] [20]. Zmniejsza ona wartości słów często występujących w języku. Zmniejsza to wpływ słów niewnoszących informacji do tekstu. Metoda używa następującego wzoru do wyliczania wartości przypisanej danemu słowu w wektorze:

$$w_{i,j} = tf_{i,j} \cdot log(\frac{N}{df_i}) \tag{3.1}$$

gdzie:
$tf_{i,j}$ = ilość wystąpień słowa i w dokumencie j
df_i = ilość dokumentów, w których występuje słowo i
N = ilość wszystkich dokumentów

3.4. Naiwny klasyfikator Bayesa

Model ten jest najprostszym użytym modelem do klasyfikacji tekstu. Uczenie polega na pobraniu wartości z wektora wejściowego (z metody Count Vectorizer lub TF-IDF) każdego ze słów i obliczenia dla nich prawdopodobieństwa wystąpienia, w danej grupie klasyfikacyjnej (w tym przypadku są dwie: suicidal/nonsuicidal).

$$p_s = \frac{\text{ilość wystąpień słowa}}{\text{ilość wystąpień wszystkich słów}}$$

Podczas klasyfikacji: słowa tekstu do sklasyfikowania są zamieniane na wyliczone prawdopodobieństwa i wszystkie są wymnażane przez siebie. Dodatkowo wymnażane są też przez prawdopodobieństwo a priori — dla badanych danych wynosi ono około 1/3 dla kategorii suicidal i 2/3 dla nonsuicidal. Jeśli wymnożone prawdopodobieństwo dla kategorii suicidal wyszło więcej niż dla drugiej, zdanie klasyfikowane jest jako depresyjne.

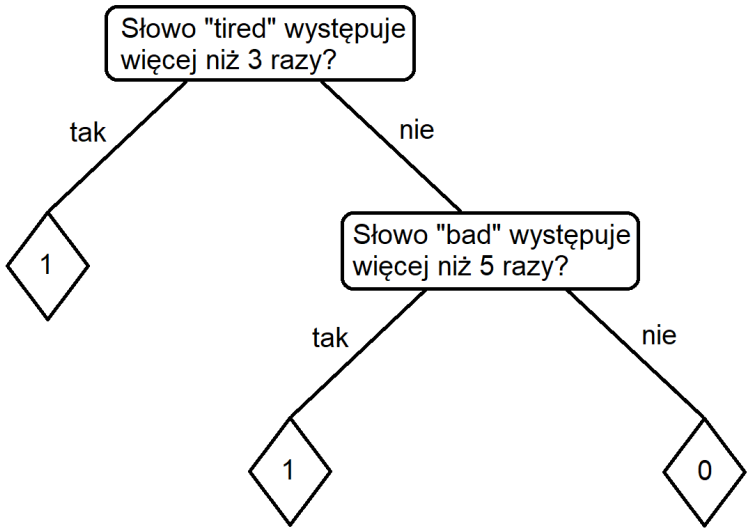
Pojawia się problem w przypadku, gdy w tekście do sklasyfikowania pojawia się słowo, które nie wystąpiło w tekstach uczących model. Prawdopodobieństwo tego słowa zostanie oszacowane na zero, co uniemożliwia określenie prawdopodobieństwa wymnożonego. Aby to rozwiązać, naiwny Bayes posługuje się parametrem α. Parametr ten dodaje się do każdej wartości w wektorze wejściowym, co pozwala uniknąć zerowych prawdopodobieństw. [21]

3.5. Las losowy

Algorytm Random Forest tworzy n drzew decyzyjnych. Każde z drzew korzysta z części danych i ustala podziały, aby jak najlepiej klasyfikować dane. W trakcie predykcji każde z drzew zwraca wynik 0 lub 1 (w tym przypadku). Aby ustalić, jaki jest ostateczny wynik klasyfikacji, wybiera się ten, który wybrało najwięcej drzew decyzyjnych. [22] [23]

Drzewa decyzyjne to algorytmy, które opierają się na strukturach drzewa binarnego. Każde rozgałęzienie to decyzja [24]. Ilość rozgałęzień zależy od danych i od przyjętego kryterium podziału. W tym

przypadku każde rozgałęzienie to uwzględnienie kolejnego zliczenia któregoś ze słów (lub wcześniej użytego, jednak z innym ograniczeniem), będzie więc ich stosunkowo dużo.



Rys. 3.3. Prosty przykład działania drzewa decyzyjnego.

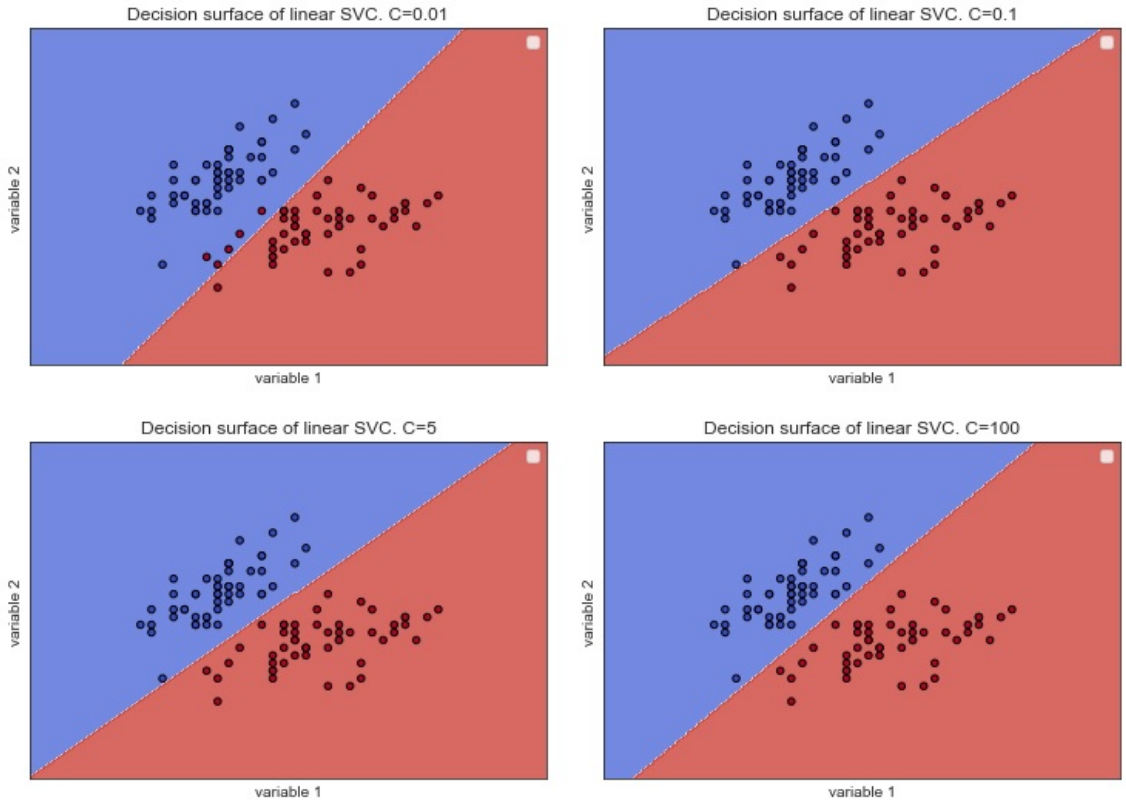
W przypadku lasu losowego, dostępnych jest ogromna ilość parametrów do zmiany. Między innymi: ilość drzew (n), kryterium podziału (na jego podstawie ustalane są kolejne podziały w drzewach), maksymalna głębokość drzewa, maksymalna ilość próbek z danych do utworzenia drzewa, czy minimalna ilość próbek w liściu. Jest to bardzo dobrze działające narzędzie, lecz łatwo jest je przeuczyć. [25]

3.6. SVM

Algorytm SVM (Support Vector Machine) stara się tak dobrać hiperpłaszczyznę w przestrzeni stworzonej przez dane (wektory wejściowe), aby jak najlepiej rozdzielić pozytywne i negatywne klasyfikowane przypadki. W swojej podstawowej, liniowej wersji nie jest on zbyt użyteczny, szczególnie dla bardziej skomplikowanych danych (jak właśnie w przypadku tekstu), ale można go polepszyć, korzystając z funkcji jądrowych. [26]

W tym algorytmie najważniejszymi parametrami są: funkcja jądrowa, parametr C i parametr gamma. Funkcja jądrowa służy do przekształcenia danych. Dopiero po tym SVM dobiera hiperpłaszczyznę. Liniowa funkcja jądrowa prawdopodobnie nie da dobrych wyników dla bardziej skomplikowanych wartości, za to wielomianowe funkcje lub funkcja RBF mogą dać dużo lepsze wyniki. [27] [28] [29]

Parametr C w uproszczeniu pozwala jednocześnie na zniwelowanie wpływu outlier'ów na model, jak i lepsze dobranie hiperpłaszczyzny, gdy punkty uczące są stosunkowo bardzo blisko siebie — granica między nimi byłaby trudna do znalezienia. Dla większych wartości C model będzie starał się, aby żaden punkt nie był źle zaklasyfikowany (nie był po złej stronie hiperpłaszczyzny). Dla mniejszych wartości C będzie na to mniej uważał, a za to będzie starał się stworzyć hiperpłaszczyznę, która będzie lepiej generalizowała model — unika wtedy overfittingu. [30]



Rys. 3.4. Wizualizacja działania metody liniowej SVM w zależności od parametru C.

Natomiast parametr gamma, stosowany tylko dla funkcji jądrowych nieliniowych, określa kształt tych funkcji. Większe wartości lepiej dopasowują się do danych, ale, jak poprzednio, należy uważać na overfit. [31]

Jako że dane dotyczące naturalnego języka są skomplikowane, można wywnioskować, że lepsze dla modelu będzie używanie wyższych wartości parametrów C i gamma, jak i nieliniowych funkcji jądrowych.

3.7. Ocena modeli uczenia maszynowego

Poprawność modeli była mierzona metodą k-fold (k równe 4), która jednocześnie uwzględnia wszystkie dane jako dane testowe i uczące, bez overfit'u. Były brane pod uwagę statystyki F1, zbalansowanej dokładności (balanced accuracy), średniej precyzji (average precision) oraz dokładności (accuracy). Ze względu na weryfikację krzyżową (k-fold) dla każdej ze statystyk wychodziły po cztery wartości, a więc z tych wartości brana była średnia.

Statystyki liczone są ze wzorów:

1. $F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$
2. $Balanced\_accuracy = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$

3. $Average\_precision = \sum_n (R_n - R_{n-1}) P_n$

4. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

gdzie:

TP = ilość tekstów poprawnie sklasyfikowanych jako suicidal

FP = ilość tekstów niepoprawnie sklasyfikowanych jako suicidal

TN = ilość tekstów poprawnie sklasyfikowanych jako nonsuicidal

FN = ilość tekstów niepoprawnie sklasyfikowanych jako nonsuicidal

$R_n = \frac{TP}{TP+FN}$, wartości liczone do n-tego tekstu

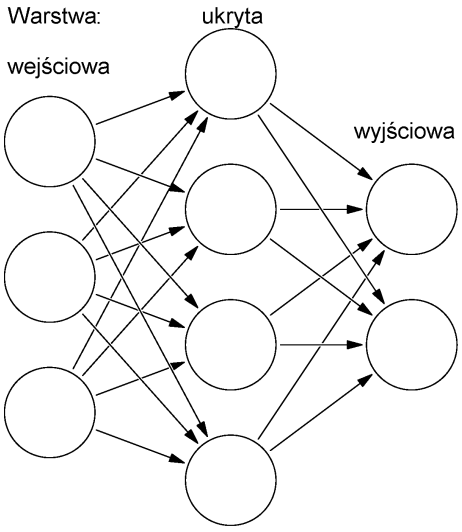
$P_n = \frac{TP}{TP+FP}$, wartości liczone do n-tego tekstu [32]

Najprostszą w interpretacji statystyką jest dokładność, mówi ona bezpośrednio jaką część przypadków model dobrze zaklasyfikował. Nie zawsze jednak jest ona najlepszą statystyką, szczególnie w przypadku niezbalansowanego zbioru danych. Przykładowo, w zbiorze o rozkładzie 95:5 przypadków pozytywnych do negatywnych, wystarczy, że model będzie zwracał cały czas 1, a otrzyma wartość dokładności 0.95. [33]

3.8. Modele uczenia głębokiego

Następnie zostały zbadane modele uczenia głębokiego.

Modele te wykorzystują sztuczne sieci neuronowe. Sieci neuronowe składają się z neuronów i połączeń między nimi. Neurony dzielą się na warstwy: warstwa wejściowa (tyle neuronów ile wartości w wektorze wejściowym), warstwy ukryte (to będzie dostosowywane, aby osiągnąć lepsze wyniki), warstwa wyjściowa (jeden neuron — jego wartość to prawdopodobieństwo zaklasyfikowania tekstu jako suicidal).



Rys. 3.5. Przykładowa jednokierunkowa sztuczna sieć neuronowa - multi layer perceptron, z tylko jedną warstwą ukrytą.

Każdy neuron wymnaża wartość z innych neuronów przez wartość wagi na połączeniu z tymi neuronami, a następnie wszystkie wyniki sumuje. Wynik sumowania przekazuje do funkcji aktywacji, np. funkcji sigmoidalnej lub ReLU (równanie 3.2). Wartość z tej funkcji przekazuje do kolejnych neuronów. Funkcja ReLU zwraca wartość wejściową, lub 0 jeśli wartość ta jest ujemna. Funkcja sigmoidalna zwraca wartości z zakresu 0-1, będzie ona użyta w warstwie wyjściowej, która będzie ustalała, czy sprawdzany tekst zostanie zaklasyfikowany pozytywnie (gdy wartość wynosi 0.5 i więcej), czy negatywnie. [34]

$$wn = f(\sum_i w_i \cdot n_i) \tag{3.2}$$

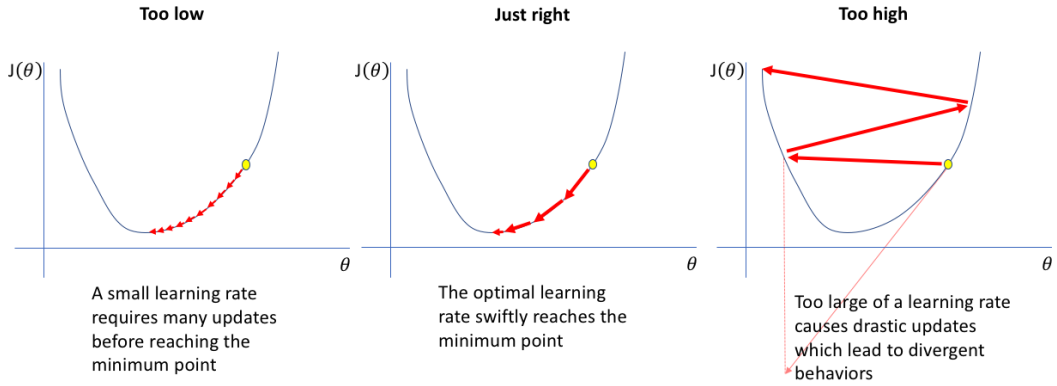
gdzie:
f = funkcja aktywacji
w_i = wartość wagi łączącej neuron z neuronem i
n_i = wartość przekazywana przez neuron i

Uczenie sieci polega na odpowiednim dobraniu wag na połączeniach między neuronami. Wykorzystywany jest algorytm wstecznej propagacji błędów [35]. Stara się on zminimalizować funkcję loss (funkcję straty). Funkcja ta może się różnić w zależności od implementacji, lecz najczęściej zwraca ona średni kwadrat różnicy pomiędzy wartością obecnie zwracaną przez sieć a oczekiwaną — prawidłową (równanie 3.3). Wizualizacja tej funkcji znajduje się w rozdziale 3.10.

$$MSE = \frac{1}{N}\sum_i^N (p_i - y_i)^2 \tag{3.3}$$

gdzie:
N = ilość przewidywanych punktów
p_i = obliczona wartość wyjściowa
y_i = oczekiwana wartość wyjściowa

Uczenie dzieli się na epoki; podczas każdej epoki wszystkie dane wejściowe przechodzą przez sieć. Im większe sieci, tym więcej epok potrzebują, aby optymalnie nauczyć się danych. Należy również uważać, aby epok nie było za dużo, gdyż nastąpi przeuczenie sieci. Parametrem, który odpowiada za to jak mocno dane dostosowują wagi w sieci, jest learning rate, współczynnik uczenia. Jest to swego rodzaju wielkość kroku wykonywanego przez sieć w kierunku znalezienia minimum funkcji straty.



Rys. 3.6. Wizualizacja doboru różnych wartości współczynnika uczenia[36].

Wyższy współczynnik przyspieszy znalezienie minimum funkcji loss. Może jednak dojść do sytuacji, że sieć będzie omijała optymalne rozwiązanie, jak na obrazku po prawej. Zbyt niski współczynnik może jednak bardzo wydłużyć proces uczenia sieci.

Tak jak w przypadku algorytmów ML, na wejście sieci neuronowych, należy podać liczby. W tym przypadku jednak, nie korzysta się z wektoryzatorów wykorzystanych w poprzednich modelach, lecz z tokenizacji [37]. Tokenizer'y, w przeciwieństwie do wektoryzatorów, nie tylko dzielą zdania na pojedyncze słowa, ale czasem i słowa na kilka tokenów. Można sprawdzić działanie przykładowego tokenizer'a na stronie: https://platform.openai.com/tokenizer.

3.9. Zaawansowane uczenie głębokie

Uczenie głębokie może być polepszone poprzez zastosowanie specjalnych warstw z bardziej złożonymi, pod względem działania, neuronami. Niektóre z nich umożliwiają rozumienie kontekstu zdania i uwzględniają kolejność słów — czego żaden z wcześniej omawianych modeli nie robił.

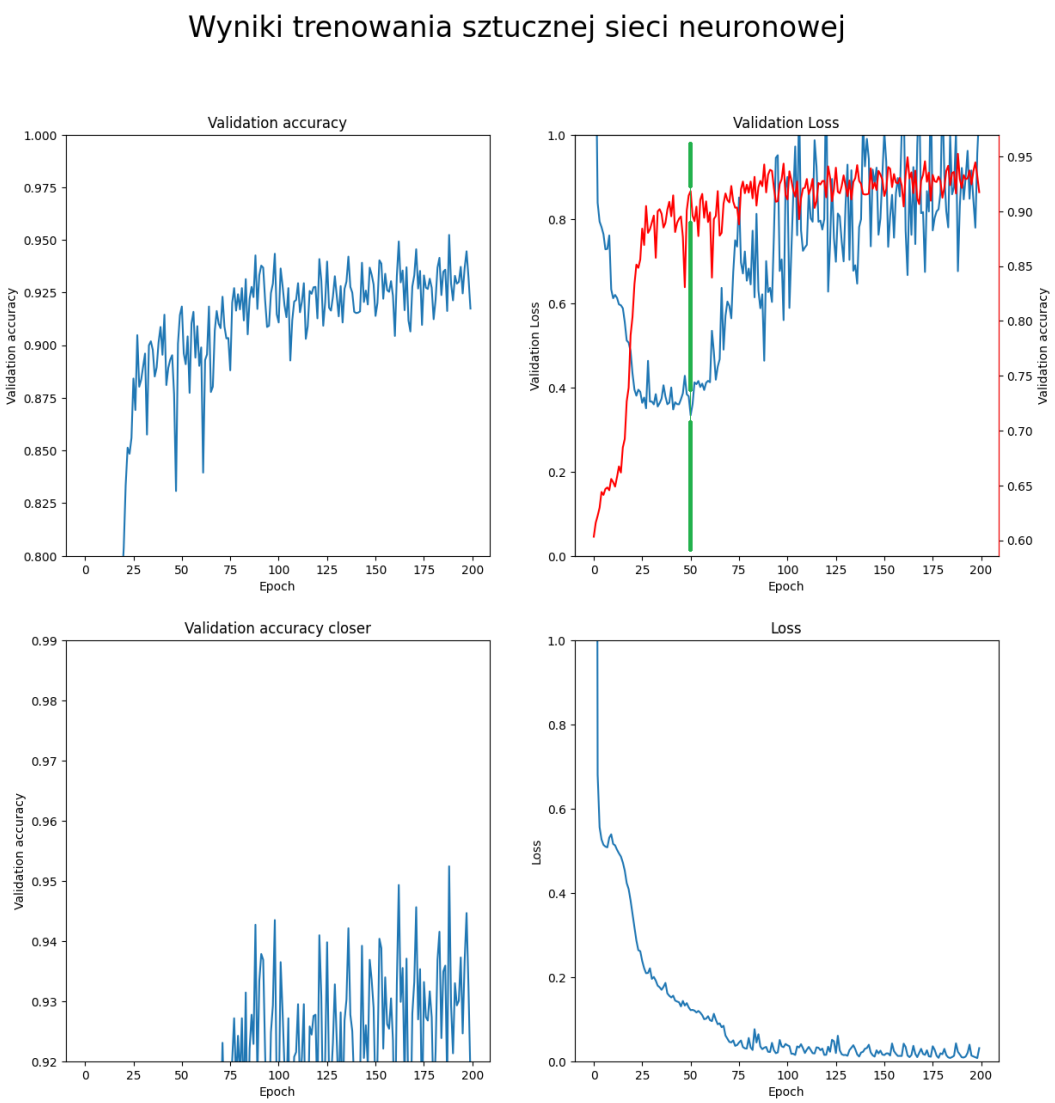
Metody wymienione w rozdziale 3.3 i modele z nich korzystające opierają się jedynie na ilości poszczególnych słów w zdaniach. Co więcej, generowane przez nie wektory wejściowe są ogromne, choć można to zmniejszyć np. biorąc pod uwagę tylko N najczęściej występujących tokenów. Obie te wady rozwiązuje embedding.

Warstwa embedding'owa zmniejsza wymiarowość naszych danych i grupuje podobne słowa. Słowa o podobnym znaczeniu, nacechowaniu, używane są w podobnym kontekście i przez to będą blisko siebie w wymiarze embedding'owym (dostaną wektory o poszczególnych wartościach zbliżonych do siebie). [38] [39]

Kolejnym usprawnieniem jest zastosowanie warstw rekurencyjnych. Sieci składające się z nich nazywane są sieciami RNN (recurrent neural networks). Neurony w takich sieciach posiadają dodatkowo sprzężenie zwrotne. Umożliwia to uwzględnienie kolejności tokenów, a nie tylko faktu ich wystąpienia i ich ilości. [40] [41]

3.10. Ocena modeli uczenia głębokiego

Ocena modeli uczenia głębokiego różni się od oceny standardowych modeli. Pod uwagę będą brane: funkcja loss oraz dokładność (accuracy) danych testowych. Ponadto, ustalenie najlepszej wartości dokładności jest trochę subiektywne. Celem tworzącego sieć jest wybranie momentu, w którym model uczenia głębokiego osiągnął najwyższą dokładność dla danych testowych (która przeważnie zwiększa się z każdą epoką), ale jednocześnie nie dopuścić do sytuacji kiedy nastąpił overfit, czyli przeuczenie. Aby to uczynić, należy patrzeć głównie na funkcję straty: będzie się ona zmniejszała, wraz ze wzrostem dokładności, ale gdy tylko zaczyna wzrastać, oznacza to przeuczenie się sieci (mimo że dokładność dla danych testowych może nadal się zwiększać). Najlepszym zatem modelem, ze wszystkich epok sieci, będzie ten, który osiągnął najmniejszą wartość funkcji loss, a odpowiadająca jej dokładność wykorzystywana będzie do porównania tej sieci z innymi. Na rysunku poniżej można prześledzić zmiany dokładności i funkcji loss danych treningowych i testowych dla kolejnych epok uczenia się sieci neuronowej. Przedrostek "Validation"w tytule oznacza wartości dla danych testowych, a jego brak wartości dla danych treningowych.



Rys. 3.7. Przykładowe wykresy z trenowania sieci neuronowej. Na zielono zaznaczony punkt idealnego wyuczenia i start przeuczenia sieci.

Pokazany powyżej przykład dość dobrze obrazuje moment przeuczenia się sieci. Nie zawsze jednak jest on tak dobrze widoczny. W tym przypadku minimum funkcji loss (dla danych testowych) wypadło około pięćdziesiątej epoki i odpowiada ono wartości dokładności na poziomie nieco poniżej 0.9 (dokładne wartości odczytywane są z programu, nie wykresów).

4. Wyniki

4.1. Naiwny klasyfikator Bayesa

4.1.1. Count Vectorizer

Na początku zbadana została opcja zamiany wszystkich liter na małe [42]. Wyłączenie jej dawało lepsze wyniki, z takim więc ustawieniem przeprowadzane są kolejne badanie (również w przypadku tf-idf [43], we wszystkich modelach).

Możliwym do ustawienia parametrem w algorytmie Bayes'a jest wspomniany wcześniej parametr α. Do sprawdzenia, jaka wartość będzie dawała najlepsze wyniki, została użyta metoda grid search. Z dostępnych wartości 0.0000001, 0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95, 0.9999, 1, 2, 5, najlepsze wyniki dawała wartość najmniejsza. Jest to zrozumiałe, gdyż słowa nowe (nieznajdujące się w słowach uczących) to prawdopodobnie słowa bardzo specyficzne i nie powinny bardzo zmieniać wyniku klasyfikacji.

Dla tak dobranego parametru wyniki były następujące:
Średni wynik $test\_f1$: 0.900
Średni wynik $balanced\_accuracy$: 0.930
Średni wynik $average\_precision$: 0.924
Średni wynik $test\_accuracy$: 0.917

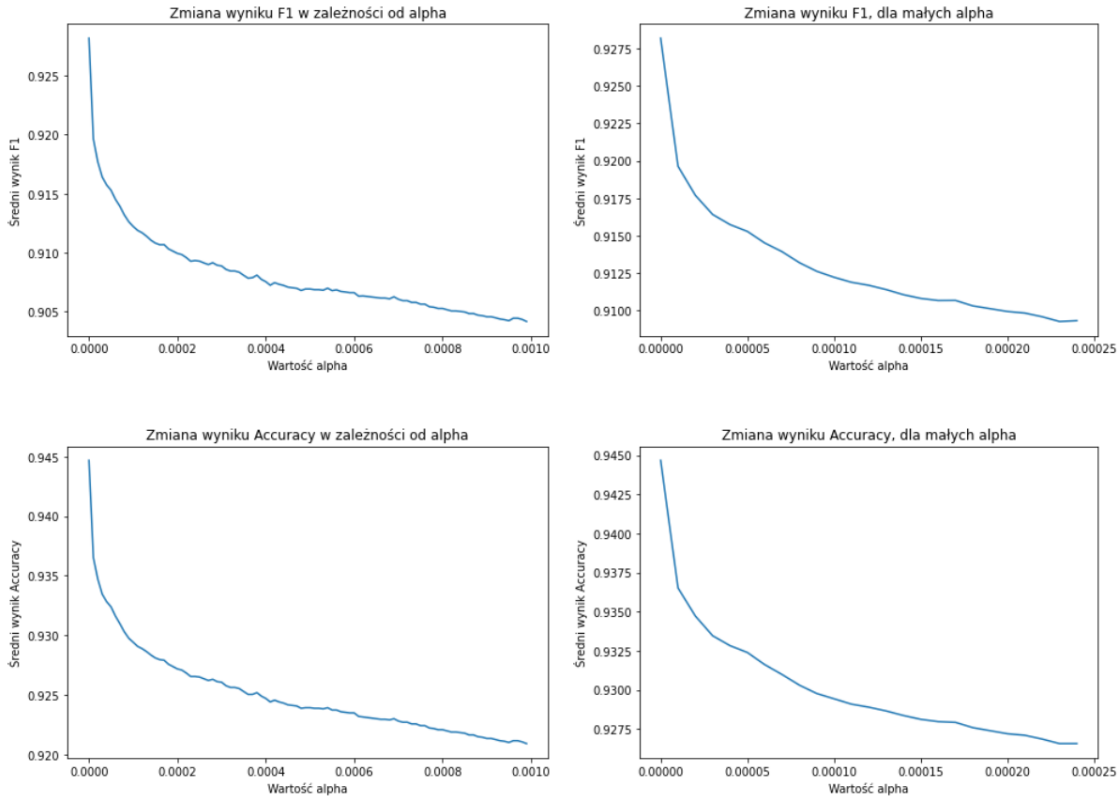
Wnioskując z accuracy, model zaklasyfikował poprawnie 91.7% przypadków.

4.1.2. TF-IDF

Aby polepszyć wyniki modelu, zastosowany został drugi rodzaj wektoryzacji: TF-IDF. Jak wspomniano w rozdziale 2, taki rodzaj wektoryzacji zmniejsza udział słów często występujących. Po wykorzystaniu Tfidf wyniki poprawiły się, niektóre statystyki wyraźnie się polepszyły:
Średni wynik $test\_f1$: 0.931
Średni wynik $balanced\_accuracy$: 0.951
Średni wynik $average\_precision$: 0.980
Średni wynik $test\_accuracy$: 0.946

Co ciekawe, kompletne usunięcie stop-words (np. 'a', 'the', 'is') pogorszyło wyniki.

4.1.3. Parametr alpha



Rys. 4.1. Wykresy ilustrujące wartość statystyk, w zależności od wartości parametru alfa

Jak widać na rysunku powyżej, im mniejsza wartość parametru alfa, tym lepiej działa model. Najmniejszą, bezpieczną, wartością jest 1e-14. Mniejsze wartości mogłyby prowadzić do błędów zaokrągleń.

Reasumując, najlepszy zbadany model, wykorzystujący naiwny algorytm Bayesa, korzystał z TF-IDF, oraz minimalnego parametru alfa: 1e-14.
Taki model daje następujące wyniki:
Średni wynik $test\_f1$: 0.935
Średni wynik $balanced\_accuracy$: 0.953
Średni wynik $average\_precision$: 0.978
Średni wynik $test\_accuracy$: 0.950

4.2. Las losowy

Pierwszy model lasu losowego został uruchomiony z następującymi parametrami: ilość drzew równa 200, kryterium podziału Giniego, brak ograniczenia głębokości drzew, dane pobierane były metodą

bootstrap i ich ilość wynosiła tyle ile wierszy wejściowych. Model dał następujące wyniki:
Średni wynik $test\_f1$: 0.967
Średni wynik $balanced\_accuracy$: 0.974
Średni wynik $average\_precision$: 0.997
Średni wynik $test\_accuracy$: 0.976

Jak widać, nawet bez dogłębnego badania parametrów, model daje bardzo dobre wyniki.

Następnie zostały sprawdzone różne wartości parametrów. Badane były: ilość drzew, kryterium wyboru, maksymalna głębokość, maksymalna ilość próbek pobieranych do tworzenia drzewa i minimalna ilość próbek w drzewie. Wszystkie parametry są sprawdzane za pomocą grid search. Okazało się jednak, że pierwsze parametry były bardzo dobrze dobrane. Jedyna potencjalna zmiana to kryterium podziału. Zastosowanie parametrów takich jak wyżej, ale z kryterium entropii, dało następujące wyniki:
Średni wynik $test\_f1$: 0.968
Średni wynik $balanced\_accuracy$: 0.974
Średni wynik $average\_precision$: 0.997
Średni wynik $test\_accuracy$: 0.977

We wszystkich przypadkach wyniki wychodziły bardzo podobne i w każdym przypadku modele używały parametrów, które mocno zwiększały złożoność modelu. Jest to jednak zrozumiałe; tłumaczy to złożoność języka naturalnego.

4.3. SVM

W tej metodzie również stosowana była metoda grid-search. Badane były parametry: C: 0.1, 0.5, 1.0, 5.0, 100, funkcja jądrowa: liniowa, rbf, sigmoidalna, gamma: 'scale' lub dobranie automatyczne. Wartość $scale$ parametru $gamma$ jest równa [44]:

$$\frac{1}{n \cdot variance(X)} \tag{4.1}$$

gdzie:
n = ilość danych wejściowych
$variance(X)$ = wariancja tych danych

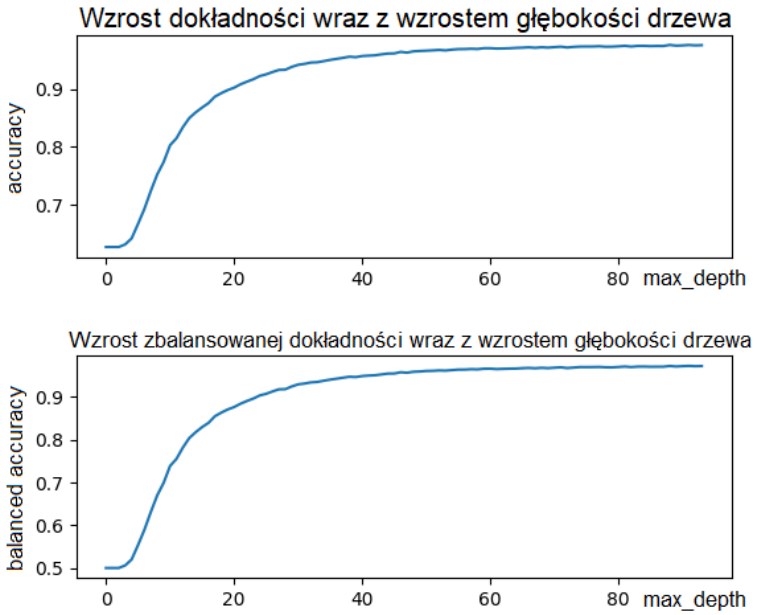
Z takich wartości najlepszym modelem okazał się ten stosujący parametry: C równy 5, $gamma$ równy $scale$ i funkcję jądrową RBF. Wyniki takiego modelu:
Średni wynik $test\_f1$: 0.965
Średni wynik $balanced\_accuracy$: 0.975
Średni wynik $average\_precision$: 0.993
Średni wynik $test\_accuracy$: 0.973

Zostały również poddane dalszym badaniom parametry $gamma$ i C. Najlepsza wartość $gamma$ się nie zmieniła. W przypadku parametru C wyższe wartości dawały bardzo podobne wyniki, lecz zwiększały czas obliczeń. Ostatecznie więc najlepszymi parametrami i wynikami modelu są te podane wyżej.

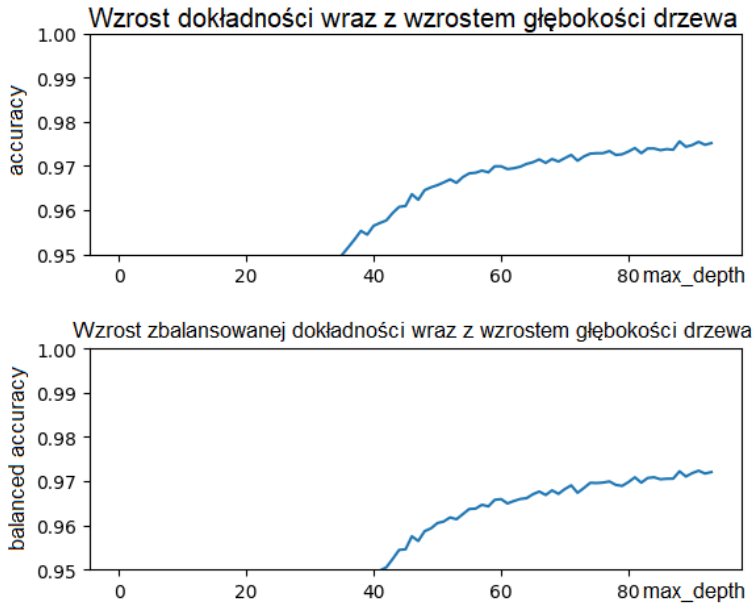
4.4. Sprawdzenie wyników

Wyniki badań okazały się zaskakująco dobre. Las losowy dał wyniki, które nawet dla bardzo skomplikowanych modeli uczenia głębokiego byłyby wysokie. Nie wynikają one jednak z nadmiernego dopasowania się modelu do danych, gdyż są one wynikiem weryfikacji k-fold. Przyczynami „problemu", może być rozkład danych 2:1, jak było to opisane w rozdziale 3.7, głębokość drzew modelu lasu losowego, lub średnia długość postów.

W przypadku głębokości drzew można zauważyć, że wraz z jej wzrostem, precyzja klasyfikacji cały czas się zwiększa.



Rys. 4.2. Wizualizacja, jak zwiększają się statystyki, wraz z zwiększeniem maksymalnej głębokości drzewa.



Rys. 4.3. Przybliżenie, jak zwiększają się statystyki, wraz z zwiększeniem maksymalnej głębokości drzewa.

Jak można zauważyć, im wyższe ograniczenie głębokości drzewa, tym lepsze wyniki klasyfikacji. Im głębsze drzewo, tym więcej słów jest uwzględnionych w modelu, a więc w przypadku NLP wartość tego parametru powinna być wysoka.

Dla danych w rozkładzie 1:1 wyniki lasu losowego, o parametrach takich samych jak najlepszy model z rozdziału 3.5, wyglądały następująco:
Średni wynik $test\_f1$: 0.953
Średni wynik $balanced\_accuracy$: 0.952
Średni wynik $average\_precision$: 0.995
Średni wynik $test\_accuracy$: 0.952
Są więc niższe, lecz nie dużo niższe. W porównaniu z modelem wykorzystującym kryterium entropii średnia precyzja wyszła nawet większa.

Kolejną rzeczą, jaka została sprawdzona, to długość poszczególnych tesktów. Średnia ilość słów w postach nonsuicidal wyniosła 279, natomiast w suicidal 189. Mogłoby się wydawać, że to jest przyczyna wysokich wyników, lecz po sprawdzeniu modelu lasu losowego na zmienionych danych (zmiana danych polegała na przycięciu wszystkich postów do długości 180 słów), wyniki były jeszcze lepsze:
Średni wynik $test\_f1$: 0.983
Średni wynik $balanced\_accuracy$: 0.988
Średni wynik $average\_precision$: 0.998
Średni wynik $test\_accuracy$: 0.988

4.5. Model uczenia głębokiego

W przypadku prostych sieci neuronowych MLP wyniki nie były tak dobre jak we wcześniejszych przypadkach. Większość z nich dawała wyniki dokładności wachającę się w przedziale od 0.9 do 0.93. Raz udało się zauważyć wynik 0.94, na epoce 62. Sieć składała się z warstw: 100, 500, 500 i 50 neuronów z funkcją aktywacji ReLU i na końcu z jednego neuronu z funkcją sigmoidalną. Learning rate: 0.00005.

Został również sprawdzony wpływ zmniejszenia wszystkich liter na małe i, co interesujące, w tym wypadku polepszyło to wyniki. Dokładność na poziomie 0.94 występowała często, a nawet udało się uzyskać 0.952. Jednak wymagało to dużo więcej uczenia. Wynik taki ukazał się na epoce 316.

4.6. Sieć RNN

Sieć RNN uzyskiwała dobre wyniki już po kilku epokach, lecz pojedyncza epoka uczyła się ponad 200 razy dłużej niż w poprzednim modelu. Dawała też bardziej stabilne i lepsze wyniki niż sieć MLP (przeważnie dokładność wynosiła około 0.97). Najwyższy wynik wyniósł 0.975, uzyskany przez sieć składającą się z 128-wymiarowej warstwy embedding'owej, warstwy 64 prostych neuronów rekurencyjnych, 250 i 200 neuronów zwykłych z funkcją aktywacji ReLU i jednego neuronu wyjściowego sigmoidalnego. Sieć wykorzystywała krok uczenia równy 0.0001 i uzyskała taki wynik w piątej epoce.

Jak poprzednio został zbadany wpływ zmniejszenia liter na małe, lecz w tym przypadku, tak jak w wypadku modeli ML, pogorszyło to wyniki. Najlepszym okazał się wynik 0.969, który został osiągnięty przez sieć zbudowaną dokładnie tak samo jak sieć opisana powyżej. Co bardziej zaskakujące, został on otrzymany z taką samą wartością learning rate i w tej samej epoce. Warte uwagi jest też to, że zmiana ilości neuronów rekurencyjnych z 64 na 32 pogorszył dokładność o zaledwie 0.002.

5. Podsumowanie

W pracy zostały przebadane różne algorytmy sztucznej inteligencji do klasyfikacji. Z podejść ML: naiwny klasyfikator Bayes'a, SVM i las losowy, najlepszym okazał się ten ostatni, choć metoda SVM była też bardzo dobra. Random forest poprawnie zaklasyfikował 97.7% przypadków. Użycie wektoryzacji TF-IDF znacząco poprawiło wyniki, a usunięcie wielkich liter je pogorszyło. Ignorowanie kapitalizacji liter jest zalecaną praktyką w NLP, w tym przypadku jednak to, czy użytkownik używał kapitalizacji, czy nie, niosło przydatne informacje dla zadania klasyfikacji.

W modelach uczenia głębokiego sieć RNN dała lepsze wyniki od zwykłych sieci MLP. Kwalifikowała dobrze 97.5% postów. Nie była to jednak bardzo rozbudowana sieć (choć dużo bardziej niż w przypadku standardowego uczenia maszynowego). Być może większe, bardziej skomplikowane sieci — wykorzystujące więcej neuronów, lub bardziej zaawansowane neurony, czy architektury, jak na przykład LSTM, dałyby lepsze wyniki.

Tak jak sprawdzono w rozdziale 4.4, wysokie statystyki raczej nie są wynikiem błędu badań. Dane zostały sprawdzone poprzez rozkład 1:1, a także przycięcie ilości słów we wszystkich postach. Pierwsza metoda lekko zmniejszyła poprawność działania modeli, druga zaś zwiększyła, obie jednak nieznacznie.

Mimo że, dla użytych tutaj danych, las losowy dał najlepsze wyniki, a uczenie głębokie (pomimo dużo większej złożoności) nie poprawiły ich, należy pamiętać, że dostępne dane nie odzwierciedlały w 100% danych, z jakimi w rzeczywistości można się spotkać. Gdyby w przyszłości stworzono specjalistyczny zbiór danych, dla takiego problemu, należałoby wypróbować wszystkie użyte w tej pracy modele (i nie tylko), aby mieć pewność, że znajdziemy najlepsze rozwiązanie. Jest też wiele metod, które nie zostały sprawdzone w tym badaniu, a które mogłyby polepszyć wyniki. Przykładowo: inne algorytmy uczenia maszynowego (np. regresja logistyczna, gradient boost), zaawansowane metody uczenia głębokiego (np. LSTM, transformery) czy zastosowanie n-gramów do wektoryzacji.

Bibliografia

[1] CDC. „YOUTH RISK BEHAVIOR SURVEY DATA SUMMARY TRENDS REPORT". W: (2021).

[2] CDC. „Increase in Suicide Mortality in the United States, 1999–2018". W: (2018).

[3] Demagog. „Samobójstwa w 2022 roku. Przedstawiamy dane policji". W: (2023).

[4] Malshani L. Pathirathna i in. „Impact of the COVID-19 pandemic on suicidal attempts and death rates: a systematic review". W: *BMC Psychiatry* (2022). DOI: *https://doi.org/10.1186/s12888-022-04158-w*.

[5] Łukasz Święcicki. „Depresja w pytaniach pacjentów i odpowiedziach eksperta". W: ().

[6] J. Wciórka J. Rybakowski S. Pużyński. *Psychiatria. T. 2.* Elsevier Urban Partner, 2010, s.305–375. ISBN: 978-83-7609-102-0.

[7] Włodzimierz Strzyżewski Adam Bilikiewicz. *Psychiatria: podręcznik dla studentów medycyny*. Państwowy Zakład Wydawnictw Lekarskich, 1992, s.323–341. ISBN: 83-200-1688-6.

[8] Jacek Wciórka Stanisław Pużyński. *Klasyfikacja zaburzeń psychicznych i zaburzeń zachowania w ICD-10. Opisy kliniczne i wskazówki diagnostyczne*. UWM „Vesalius", 2007, s.107–116. ISBN: 83-85688-25-0.

[9] Pedro Ruiz Benjamin James Sadock Virginia Alcott Sadock. *Kaplan Sadock's Synopsis of Psychiatry Behavioral Sciences/Clinical Psychiatry. Wydanie 11*. Wolters Kluwer, 2015, s.107–116. ISBN: 978-1609139711.

[10] Janusz Rybakowski Jacek Wciórka Stanisław Pużyński. *Metody leczenia, Zagadnienia etyczne, prawne, publiczne, społeczne*. Elsevier Urban Partner, 2010, s.65–123, 199–204. ISBN: 978-83-7609-110-5.

[11] J. Gerber S. Puschmann K. Petrowski P. Joraschky T. Hummel S. Negoias I. Croy. „Reduced olfactory bulb volume and olfactory sensitivity in patients with acute major depression". W: *Neuroscience 169 (1)* (2010), s. 415–421. DOI: *10.1016/j.neuroscience.2010.05.012*.

[12] Gabbard G.O. *Psychodynamic Psychiatry in Clinical Practice: Fourth Edition*. American Psychiatric Publishing, 2005, s. 195–224. ISBN: 978-1-58562-185-9.

[13] Katarzyna Kwiecień. „Cztery typy uczenia maszynowego". W: (2018).

[14] Paul Lin Wendy Torell Maria A. Torres Arango Victor Avelar Patrick Donovan. „The AI Disruption: Challenges and Guidance for Data Center Design". W: (2023).

[15] George Poulos. „When did humans first start to speak? How language evolved in Africa". W: *The conversation* (2022).

[16] Elizabeth D. Liddy. „Natural Language Processing". W: (2001).

[17] Krystyna Piątkowska. „Klasyfikacja tekstu, czyli „text classification"". W: *Blog statystyczny* (2020).

[18] Basics of CountVectorizer. „Pratyaksh Jain". W: *Medium, Towards Data Science* (2021).

[19] Dylan Kaplan. „Machine Learning 101: CountVectorizer vs TFIDFVectorizer". W: *enjoymachinelearning.com* (2022).

[20] Juan Ramos. „Using TF-IDF to Determine Word Relevance in Document Queries". W: (2003).

[21] Joshua Starmer. „StatQuest. "Naive Bayes"YouTube". W: (2020).

[22] Joshua Starmer. „StatQuest. "StatQuest: Random Forests Part 1 - Building, Using and Evaluating"YouTube". W: (2018).

[23] Leo Breiman. „Random Forests". W: 2001. DOI: *https://doi.org/10.1023/A:1010933404324*.

[24] Ted Pedersen. „A Decision Tree of Bigrams is an Accurate Predictor of Word Sense". W: *Second Meeting of the North American Chapter of the Association for Computational Linguistics*. 2001.

[25] *Dokumentacja scikit-learn, RandomForestClassifier*.

[26] PEDRO PENZUTI PACHECO SHAVIRA NARRANDES YANG WANG SHUJUN HUANG NIANGUANG CAI i WAYNE XU. „Applications of Support Vector Machine (SVM) Learning in Cancer Genomics". W: (2018). DOI: *10.21873/cgp.20063*.

[27] Joshua Starmer. „StatQuest. "Support Vector Machines Part 1 (of 3): Main Ideas!!!"YouTube". W: (2019).

[28] Joshua Starmer. „StatQuest. "Support Vector Machines Part 2: The Polynomial Kernel (Part 2 of 3)"YouTube". W: (2019).

[29] Joshua Starmer. „StatQuest. "Support Vector Machines Part 3: The Radial (RBF) Kernel (Part 3 of 3)"YouTube". W: (2019).

[30] Pushkar Mandot. „What is the Significance of C value in Support Vector Machine?" W: (2017).

[31] A Man Kumar. „C and Gamma in SVM". W: *Medium* (2018).

[32] *Dokumentacja scikit-learn, average*$_p recision$.

[33] Sumeet Kumar Agrawal. „Metrics to Evaluate your Classification Model to take the right decisions". W: *Analytics Vidhya* (2023).

[34] Joshua Starmer. „StatQuest. "Neural Networks Pt. 1: Inside the Black Box"YouTube". W: (2020).

[35] Joshua Starmer. „StatQuest. "Neural Networks Pt. 2: Backpropagation Main Ideas"YouTube”. W: (2020).

[36] Jeremy Jordan. „Setting the learning rate of your neural network.” W: (2018).

[37] Jonathan J. Webster Chunyu Kit. „TOKENIZATION AS THE INITIAL PHASE IN NLP”. W: ().

[38] AssemblyAI. „A Complete Overview of Word Embeddings”. W: *YouTube* (2022).

[39] Joshua Starmer. „StatQuest. "Word Embedding and Word2Vec"YouTube”. W: (2023).

[40] Joshua Starmer. „StatQuest. "Recurrent Neural Networks (RNNs)"YouTube”. W: (2022).

[41] Alex Sherstinsky. „Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. W: *Physica D: Nonlinear Phenomena* 404 (2020), s. 132306. ISSN: 0167-2789. DOI: *https://doi.org/10.1016/j.physd.2019.132306*.

[42] *Dokumentacja scikit-learn, CountVectorizer*.

[43] *Dokumentacja scikit-learn, TfidfVectorizer*.

[44] *Dokumentacja scikit-learn, RandomForestClassifier*.