1. Wstęp
2. Podstawy teoretyczne
3. Projekt?
4. Implementacja?
5. Badania
6. Podsumowanie

ROZDZIAŁ 2 – ZAGADNIENIA TEORETYCZNE

# Uczenie maszynowe

Dziedzina uczenia maszynowego, która powstała w latach 60-tych XX wieku, obecnie odgrywa bardzo ważną rolę w gwałtownie rozwijającym się sektorze analityki dużych zbiorów danych. Algorytmy uczenia wykorzystywane są przy wielu zadaniach przewidywania lub klasyfikowania danych.

Terminem uczenia maszynowego możemy określić gałąź informatyki oraz sztucznej inteligencji, zajmującej się utworzeniem modelu, który jest w stanie samodzielnie, pod wpływem otrzymywanych danych, uczyć się, czyli maksymalizować swoją skuteczność w wykonywaniu konkretnego zadania.

Istnieją różne rodzaje modeli uczenia maszynowego różniące się sposobem uczenia. Główną role odgrywa wpływ człowieka na dostarczane dane. Dwoma podstawowymi typami są:

* Uczenie nadzorowane – dane otrzymywane przez algorytm uczący posiadają etykiety, które są wykorzystywane w procesie uczenia, a także do określenia, przy pomocy różnych metryk, jak dobrze działa finalnie stworzony model
* Uczenie nienadzorowane – dane otrzymywane przez algorytm uczący nie posiadają etykiet. Zadaniem algorytmu jest zidentyfikowanie relacji i wzorców w przekazanych danych bez pomocy człowieka

Proces nadzorowanego uczenia, na którym skupia się ta praca, składa się z trzech głównych, powtarzających się części:

* Procesu decyzyjnego – Algorytm użyty w procesie uczenia na podstawie otrzymanych danych dokonuje predykcji.
* Funkcji błędu – Funkcja ta służy do oceny predykcji dokonanej w procesie decyzyjnym z prawdziwą wartością dostarczaną z danymi. Jest to miara dokładności modelu.
* Procesu optymalizacji – Metoda, której celem jest zaktualizowanie działania algorytmu używanego w procesie decyzyjnym, biorąca pod uwagę wartość funkcji błędu, w taki sposób aby zminimalizować błąd w następnej iteracji.

# Głębokie uczenie maszynowe

Terminy „Uczenie maszynowe” i „Głębokie uczenie” są często używane zamiennie, jednak oznaczają one inne podejścia. Samo głębokie uczenie jest poddziedziną uczenia maszynowego, a główną różnicą tych metod jest sposób uczenia się algorytmu. Głębokie uczenie maszynowe wykorzystuje wielowarstwowe sieci neuronowe, które cechują się bardzo duża elastycznością, a także umiejętnością wykorzystywania danych, które nie były wstępnie przetwarzane.

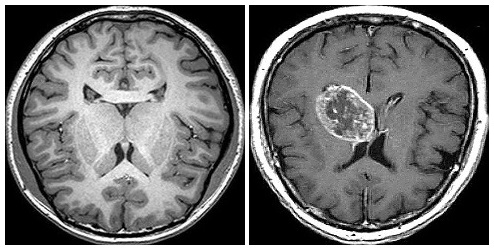
Cel głębokiego uczenia jest taki sam jak uczenia maszynowego, czyli stworzenie sieci neuronowej, która dla otrzymywanych danych maksymalizuje swoją skuteczność

# Problem klasyfikacji

Jednym z podstawowych zadań zarówno dla uczenia maszynowego jak i uczenia głębokiego jest problem klasyfikacji danych. Polega ono na przyporządkowaniu danych wejściowych do jednej klasy, pochodzącej ze zbioru predefiniowanych klas. Głównym celem samej klasyfikacji jest stworzenie formalnego modelu, nazywanego klasyfikatorem, którego zadaniem jest klasyfikowanie nowo otrzymanych danych i przydzielenie ich do konkretnej klasy.

Zadanie klasyfikacji posiada wiele zastosowań np.: klasyfikowanie spamu e-mail, klasyfikowanie opinii na podstawie recenzji, wykrywanie obiektów na zdjęciach. Jednym z popularnych sektorów , w których wykorzystywane jest to zadanie jest sektor medyczny. Dane z systemów medycznych mogą zostać użyte do stworzenia klasyfikatora zdolnego do bardzo szybkiego klasyfikowania schorzeń u dotychczasowych jak i nowych pacjentów.

W tej pracy skupiono się tylko na problemie klasyfikacji binarnej, charakteryzującej się występowaniem jedynie dwóch klas. Dane mogą zostać zaklasyfikowane jako klasa pozytywna (1) lub klasa negatywna (0).



Rys. 1 Przykład klasyfikacji binarnej na podstawie skanów MRI. Po lewej stronie znajduje się skan pacjenta bez guza mózgu - klasa negatywna (0). Po prawej stronie skan pacjenta z guzem mózgu - klasa pozytywna (1).

# Ocena jakości klasyfikatora

W celu zweryfikowania jakości klasyfikatora binarnego użyto różnych metryk, aby zmniejszyć ryzyko błędnej oceny działania. Wszystkie wykorzystane metryki bazują na przedstawionej w tabeli 123 macierzy pomyłek.

Tab. 1 Macierz pomyłek (ang. confusion matrix)

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Przewidziana** **wartość** | |
| **1** | **0** |
| **Rzeczywista**  **wartość** | **1** | **TP**  prawdziwie pozytywny | **FN**  fałszywie negatywny |
| **0** | **FP**  fałszywie pozytywny | **TN**  prawdziwie negatywny |

Tab. 2

Pierwszą z metryk jest skuteczność (ang. *accuracy*), mówiąca o stosunku poprawnie zaklasyfikowanych próbek względem całego zbioru.

Kolejną metryką jest precyzja (ang. *precision*), która jest określona następującym stosunkiem:

Przez wartość tej metryki rozumiemy odsetek prawdziwie pozytywnym przypadków z wszystkich, które klasyfikator określił jako pozytywne. Mała wartość precyzji sugeruje, że klasyfikator błędnie klasyfikuje negatywną klasę jako pozytywną.

Następną metryką jest czułość (ang. *recall*), czyli zdolność klasyfikatora do wykrywania przykładów pozytywnych

Kolejną metryką, która jest połączeniem dwóch poprzednich jest metryka F1. Jest ona średnią harmoniczną precyzji i czułości.

Ostatnią metryką jest ROC

ROZDZIAŁ 3 – PROJEKT SYSTEMU

# Opis projektu

Głównym zadaniem aplikacji ma być szybka i łatwa możliwość sklasyfikowania wybranego przez użytkownika skanu MRI mózgu przez wybrany przez użytkownika, wytrenowany klasyfikator. Aplikacja ma działać lokalnie, tak samo jak lokalnie ma być dokonywana predykcja. Użytkownik ma mieć możliwość ręcznego wyboru oraz załadowania modelu, oraz dowolnego skanu. Użytkownik powinien otrzymać wynik klasyfikacji w formie tekstowej, informującej go czy przesłany skan został zaklasyfikowany do klasy pozytywnej lub negatywnej. Proces uczenia oraz testowania modeli ma odbywać się lokalnie na komputerze. Cały projekt ma zostać zaimplementowany w języku Python w wersji 3.10.8.

# Wymagania funkcjonalne aplikacji

**F1** Wybór modelu z lokalnego komputera

Użytkownik ma mieć możliwość wyboru i załadowania modelu z lokalnego komputera poprzez wybór ścieżki do modelu.

**F2** Wybór zdjęcia z lokalnego komputera

Użytkownik ma mieć możliwość wyboru i załadowania dowolnego zdjęcia z lokalnego komputera poprzez wybór ścieżki do zdjęcia.

**F3** Wyświetlanie załadowanego zdjęcia

Aplikacja powinna wyświetlić wybrane i załadowane przez użytkownika zdjęcie.

**F4** Klasyfikowanie wybranego zdjęcia

Aplikacja ma po wybraniu modelu oraz zdjęcia dokonać klasyfikacji załadowanego zdjęcia wykorzystując załadowany model.

**F5** Wyświetlanie wyniku klasyfikacji

Użytkownik po wykonaniu klasyfikacji ma zostać poinformowany w formie tekstowej o wyniku zwróconym przez model. Tekst ma informować użytkownika czy wybrane zdjęcie według modelu pochodzi z jednej z dwóch klas: „tumor”, „no tumor”.

# Wymagania niefunkcjonalne

**NF1** Interfejs graficzny ma być wykonany w języku angielskim

**NF2** Aplikacja ma informować tekstowo o swoim statusie

# Diagram przypadków użycia

Przedstawiony na rysunku diagram przypadków użycia, który został stworzony opierając się na wymienionych wymaganiach funkcjonalnych, ma na celu wizualne opisanie wszystkich funkcjonalności aplikacji.



# Aplikacja ??desktopowa??

Aplikacja graficzna ma zostać wykonana przy pomocy dołączonej do języka Python biblioteki Tkinter. Cały interfejs

* 1. Prototyp wyglądu aplikacji