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.

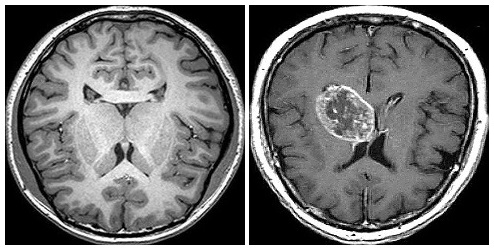
Cel głębokiego uczenia jest taki sam jak uczenia maszynowego, czyli stworzenie sieci neuronowej, która dla otrzymywanych danych maksymalizuje swoją skuteczność

# 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), tak jak zostało to zaprezentowane na Rysunku 1.



Rys. 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. 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 |

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 użytą metryką jest krzywa ROC (ang. *receiver operating characteristic curve).* Krzywa ta jest wykresem graficznym, który obrazuje zdolność klasyfikatora do oddzielania klasy pozytywnej od negatywnej. Metryka jest używana tylko w przypadku klasyfikatora binarnego. Tworzy się ją poprzez wykreślenie stosunku wskaźnika czułości opisanej wzorem (wzór recall), oraz specyficzności (ang. specificity) opisanej wzorem

dla różnej wielkości progu klasyfikacji. Przykładowy wygląd krzywej ROC został przedstawiony na Rysunku 69.



Do przypisania wartości dla metryki ROC wykorzystuje się pole pod powierzchnią krzywej (AUC, ang. *area under curve).* Wartość AUC równa 0.5 oznacza brak umiejętność klasyfikatora do rozróżniania klas, a tym bardziej wartość AUC jest bliższa 1, tym bardziej klasyfikator poprawnie oddziela klasy pozytywne od negatywnych. Wartość poniżej 0.5 oznacza, że klasyfikator jest gorszy niż losowe klasyfikowanie do jednej z klas.

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

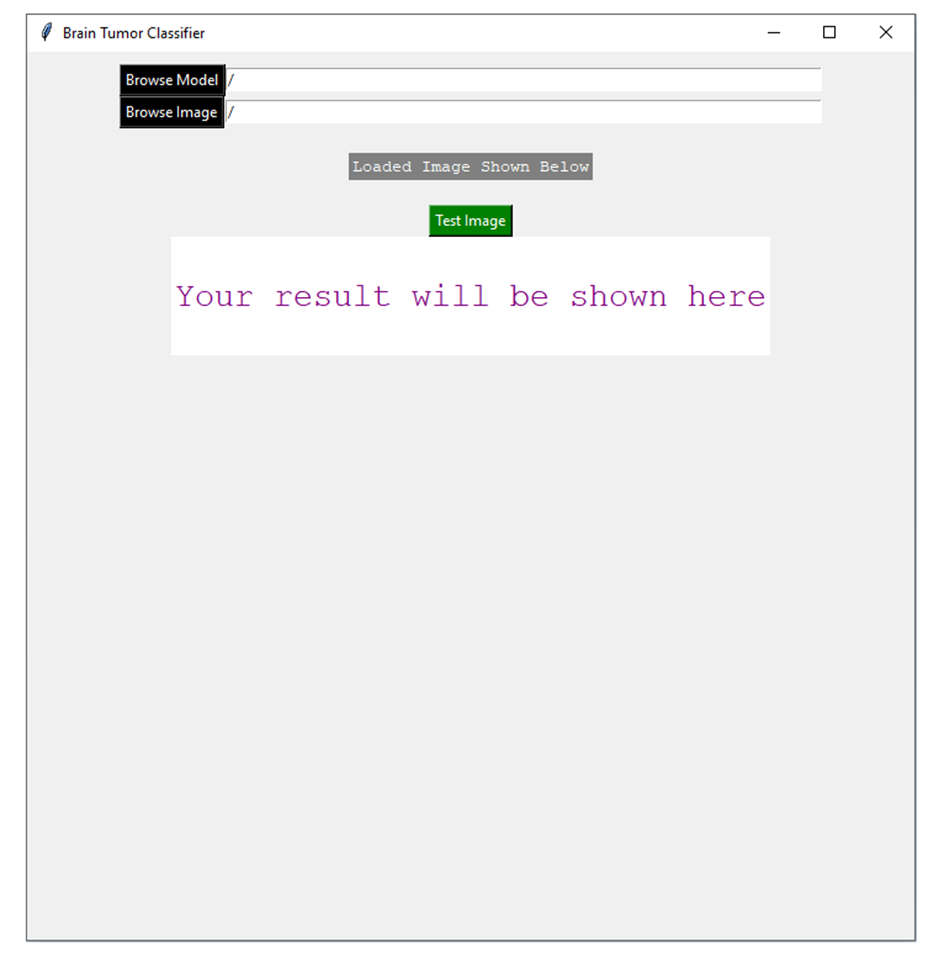
# Diagram przypadków użycia

Przedstawiony na Rysunku 69 diagram przypadków użycia, który został stworzony opierając się na wymienionych wymaganiach funkcjonalnych, ma na celu wizualne opisanie wszystkich funkcjonalności aplikacji.

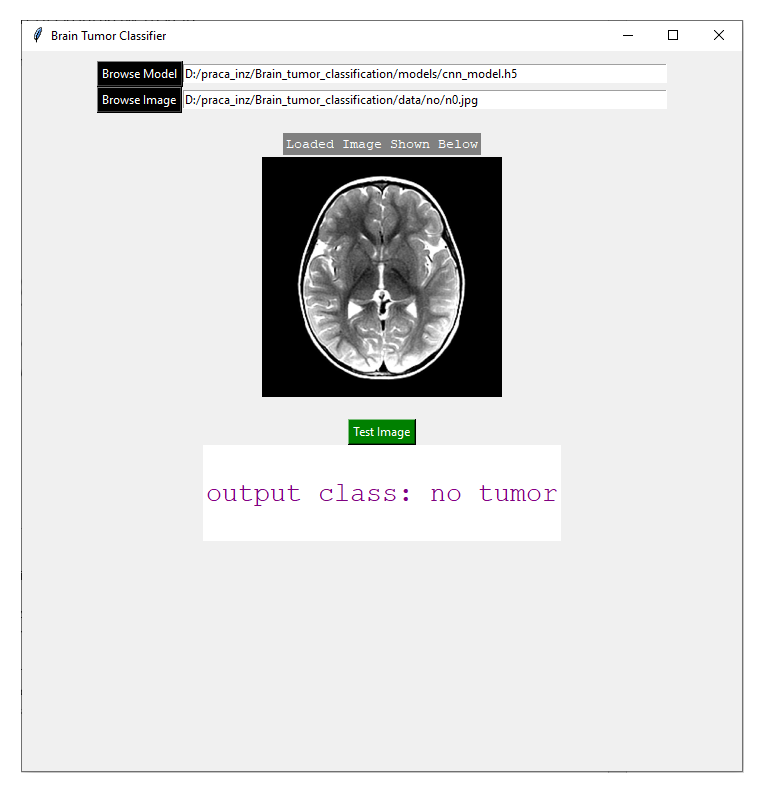


# Prototyp interfejsu graficznego

Aplikacja graficzna ma zostać wykonana przy pomocy dołączonej do języka Python biblioteki Tkinter. Cały interfejs będzie składać się z jednego widoku, w którym będą zawarte wszystkie funkcjonalności. Prototyp wyglądu, przed interakcją użytkownika został przedstawiony na Rysunku 911. Interfejs będzie składał się z trzech sekcji. W górnej sekcji znajdą się dwa przyciski odpowiedzialne kolejno za wybranie i wczytanie modelu, oraz za wybranie wczytanie zdjęcia. Dodatkowo obok przycisków zostanie pokazana ścieżka wybranego modelu i zdjęcia. W środkowej sekcji znajdować się będzie podgląd załadowanego zdjęcia, a także przycisk służący do jego klasyfikacji. Ostatnią część aplikacji zajmować będzie okno, w którym wypisywane będą wyniki klasyfikacji. Prototyp wyglądu po dokonaniu klasyfikacji przedstawia Rysunek 912.



Rys. Prototyp wyglądu interfejsu przed interakcją użytkownika



Rys. Prototyp interfejsu użytkownika po wykonaniu predykcji

# 69) Użyte technologie i biblioteki

W tym podrozdziale zostały wymienione i opisane główne technologie i biblioteki potrzebne do wykonania projektu:

1. Python – język programowania wysokiego poziomu, który posiada wiele zróżnicowanych bibliotek umożliwiających wykorzystanie go do wielu zadań. Python jest jednym z popularniejszych języków wykorzystywanych w dziedzinie uczenia maszynowego i głębokiego uczenia.
2. Tkinter – wbudowana biblioteka języka Python umożliwiająca tworzenie prostego interfejsu graficznego. Może być używana do tworzenia interfejsów aplikacji dla większości systemów Unix, w tym macOS i Windows.
3. TensorFlow – popularna otwarto źródłowa biblioteka języka Python wykorzystywana w procesie uczenia maszynowego lub uczenia głębokich sieci neuronowych. Cechuje się łatwością implementacji łącząc ją z wysokopoziomowym API, oraz szybkością działania w związku z użyciem języka C++ w samym silniku.
4. Keras – otwarto źródłowa biblioteka języka Python służąca jako interfejs biblioteki TensorFlow. Jej głównym celem jest zapewnienie łatwości obsługi dla użytkownika, przy zachowaniu dużej swobody działania.
5. Scikit-learn – otwarto źródłowa biblioteka języka Python posiadająca wiele algorytmów uczenia maszynowego. Biblioteka zawiera także dużo przydatnych narzędzi m.in. wstępnego przetwarzania danych
6. NumPy - otwarto źródłowa biblioteka pozwalająca na zaawansowaną obsługę wielowymiarowych tabel i macierz, a także na wykonywaniu na nich łatwych i szybkich obliczeń
7. Matplotlib – otwarto źródłowa biblioteka wykorzystywana do tworzenia wykresów oraz wizualizacji

Bibliografia

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[2] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362, 2020

[3] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007

[4] H.P. Langtangen, Python Scripting For Computational Science, Texts in Computational Science and Engineering (Springer Berlin Heidelberg, 2009).

[5] Hoo, Zhe Hui, Jane Candlish, and Dawn Teare. "What is an ROC curve?." *Emergency Medicine Journal* 34.6 (2017): 357-359. ( ROC curve )