

**Wydział Elektrotechniki, automatyki, informatyki i inżynierii biomedycznej.**

Praca dyplomowa

*Federacyjne uczenie maszynowe*

*Federated machine learning*

Autor: *Mikołaj Skrzyniarz*

Kierunek studiów: *Informatyka i Systemy Inteligentne*

Opiekun pracy: *dr inż. Piotr Szwed*

Kraków, 2022

Spis treści

[1. Wstęp 4](#_Toc105538556)

[1.1. Cel pracy 5](#_Toc105538557)

[1.2. Zakres pracy 5](#_Toc105538558)

[2. Wprowadzenie teoretyczne 6](#_Toc105538559)

[2.1. Uczenie maszynowe 6](#_Toc105538560)

[2.2. Rodzaje uczenia maszynowego 8](#_Toc105538561)

[Uczenie nadzorowane 9](#_Toc105538562)

[Uczenie nienadzorowane 10](#_Toc105538563)

[Pozostałe rodzaje uczenia maszynowego 11](#_Toc105538564)

[2.3. Sieci neuronowe 12](#_Toc105538565)

[Rodzaje sieci neuronowych 14](#_Toc105538566)

[2.4. Tradycyjne uczenie maszynowe 15](#_Toc105538567)

[2.5. Rozproszone uczenie maszynowe 16](#_Toc105538568)

[2.6. Federacyjne uczenie maszynowe 17](#_Toc105538569)

[2.7. Prywatność danych w uczeniu federacyjnym 19](#_Toc105538570)

[Prywatność różnicowa 19](#_Toc105538571)

[Szyfrowanie homomorficzne 20](#_Toc105538572)

[Bezpieczne obliczenia wielopartyjne 20](#_Toc105538573)

[2.8. Rodzaje federacyjnego uczenia maszynowego 20](#_Toc105538574)

[Uczenie federacyjne wertykalne 20](#_Toc105538575)

[Uczenie federacyjne horyzontalne 21](#_Toc105538576)

[Uczenie federacyjne transferowe 22](#_Toc105538577)

[Pozostałe rodzaje uczenia federacyjnego 23](#_Toc105538578)

[3. Projekt i implementacja oprogramowania 24](#_Toc105538579)

[Projekt oprogramowania 24](#_Toc105538580)

[Opis możliwości konfiguracyjnych oprogramowania 25](#_Toc105538581)

[Wykorzystane technologie oraz narzędzia 27](#_Toc105538582)

[4. Opis przebiegu wykonanych badan i eksperymentów 27](#_Toc105538583)

[5. Podsumowanie 28](#_Toc105538584)

[6. Bibliografia 29](#_Toc105538585)

# Wstęp

Dziedziny zajmujące się zagadnieniami związanymi z sztuczną inteligencją istnieją w świecie technologii i nauki od dłuższego czasu. Ich popularność stale rośnie a sztuczna inteligencja powiązana jest z takimi obszarami jak matematyka, statystyka, nauki o danych (ang. *data science)*, duże zbiory danych (ang. *big data)* oraz z oczywistych względów – informatyka (ang. *computer science*). Trudno się temu dziwić, wszystkie te zagadnienia niejako przeplatają się nawzajem – poruszają podobne problemy oraz wymagają zbliżonych umiejętności.

W przeciągu ostatnich dwudziestu lat termin sztuczna inteligencja ewoluował z nauki wzbudzającej ogromną ciekawość wśród ludzi do praktycznej technologii, powszechnie używanej w celach komercyjnych. Ten postęp spowodowany jest rosnącą dostępnością systemów umożliwiające prowadzenie różnego rodzaju skomplikowanych obliczeń, materiałów naukowych oraz świadomością jak i wiedzą uczonych, studentów jak i wszystkich pracowników branż technologicznych. Ta z kolei prowadzi do powstawania nowych, bardziej rozbudowanych i zaawansowanych algorytmów. W efekcie liczba firm stosujących rozwiązania bazujące na mechanizmach sztucznej inteligencji w ubiegłych czasach wzrosła diametralnie. Wizja komputerowa (ang. *computer vision)*, przetwarzanie języka naturalnego, rozpoznawanie mowy, wykrywanie schorzeń oraz różnego rodzaju anomalii to tylko niektóre z wielu przykładów użycia, a z prostych mechanizmów wspomagających podejmowanie decyzji korzystamy, świadomie lub nie, tak naprawdę na co dzień.

Uczenie maszynowe jest największym obszarem szeroko rozumianej sztucznej inteligencji. Może być rozumiany jako zajmujący się szukaniem rozwiązania problemu polegającego na stworzeniu urządzenia, które będzie osiągało lepsze wyniki w skali ustalonej wcześniej metryki poprzez naukę w oparciu o doświadczenie oraz decyzje podejmowane w przeszłości. Przykładem może być wykrywanie schorzenia na podstawie zdjęć rentgenowskich danego narządu. Celem jest poprawne przydzielenie etykiet „zdrowy” i „chory” dla każdego ze zdjęć. By to osiągnąć dany algorytm musi nauczyć się rozróżniać zdjęcia na podstawie określonych cech. To z kolei odbywa się w procesie uczenia, podczas którego dany model poddaje się próbom na zbiorze uczącym, zawierającym dane służące jako wzorzec, mające już przydzielone etykiety [2].

Tradycyjny proces uczenia z uwzględnieniem danych zbieranych urządzenia rozproszone polega na agregacji takich danych na wspólnym urządzeniu pełniącym rolę serwera, wytrenowaniu modelu oraz propagacji gotowego modelu między urządzeniami. Głównym problemem takiego rozwiązania jest fakt, że dane istnieją w formie odizolowanych obiektów oraz ich bezpieczeństwo jak i prywatność, które mogą być naruszone podczas procesu agregacji. Rozwiązaniem tego może być zastosowanie federacyjnego uczenia maszynowego, zaproponowanego przez firmę kilka lat temu przez firmę *Google*. Początkowo koncept ten był jedynie ideą jednak obecnie jest coraz częściej stosowany komercyjnie, np. przez wspomnianą wyżej organizację [3]. Ten typ uczenia działa nieco inaczej niż metoda tradycyjna. W takiej konfiguracji urządzenia lub organizacje trenują modele lokalnie, przy użyciu zgromadzonych danych. Następnie parametry modelów są wysyłane do jednostki centralnej, na której są agregowane i w oparciu o które tworzony jest model globalny. W teorii skuteczność wytrenowanego w ten sposób modelu powinna być zbliżona do tradycyjnej metody przy zachowaniu większego bezpieczeństwa całego procesu oraz niższego ryzyka wycieku danych.

## Cel pracy

Celem niniejszej pracy było zaprojektowanie oraz przeprowadzenie eksperymentów mających na celu symulację federacyjnego uczenia maszynowego. Wyniki powinny zostać porównane z wynikami uzyskanymi przy użyciu tradycyjnego sposobu uczenia. W wykorzystanym podejściu samodzielni agenci trenują swoje modele przy użyciu obserwowanych danych. Modele te są okresowo agregowane, a następnie dystrybuowane wewnątrz grupy.

W pracy należało rozważyć różne tryby działania algorytmu, a także zaproponować metodę agregacji wag oraz parametry sterujące. Docelowy system miał być przetestowany na dużym zbiorze danych z użyciem modeli o różnym stopniu złożoności. W przypadku zastosowania *transfer learning* rdzeń modelu nie powinien podlegać modyfikacji.

## Zakres pracy

TODO

# Wprowadzenie teoretyczne

Celem drugiego rozdziału pracy jest zaznajomienie czytelnika z wiedzą teoretyczną na temat poruszanych pojęć oraz zagadnień. W oparciu o wykorzystane pozycje literaturowe wytłumaczone zostanie czym tak naprawdę jest uczenie maszynowe, jakie są jego rodzaje oraz omówiony zostanie jeden z najbardziej znanych algorytmów uczenia maszynowego – sieci neuronowe. Następnie opisane zostaną sposoby uczenia maszynowego mające miejsce w praktycznym zastosowaniu. Są to najprostszy, nazywany przez autora uczeniem „tradycyjnym”, kolejno uczenie rozproszone oraz będące tematem badań niniejszej pracy - federacyjne uczenie maszynowe. Zostaną również przedstawione różnice pomiędzy wspomnianymi modelami oraz rodzaje uczenia federacyjnego.

## Uczenie maszynowe

Uczenie maszynowe wchodzi w skład zagadnień poruszanych przez obszar sztucznej inteligencji, która jest jednym z najbardziej dynamicznie rozwijających się obszarów stref branży technologicznej oraz nauk z nią związanych. Na podstawie dostępnej literatury oraz materiałów w sieci możemy wyznaczy 4 kategorie działań zajmujących się zagadnieniami związanymi z inteligencją i uczeniem maszyn, gdzie każda kolejna jest zawężeniem zakresu poprzedniej (patrz rys. 1.1). Są to:

1. sztuczna inteligencja (ang. *Artificial intelligence, AI) -* zawiera każdą z technik, która umożliwia urządzeniom naśladowanie ludzkiej inteligencji*.*
2. uczenie maszynowe (ang. *Machine learning, ML) -* ogranicza techniki definiowane przez *AI* do takich, które umożliwiają maszynom naukę poprzez doświadczenie pewnych problemów lub zadań*.*
3. uczenie głębokie (ang. *Deep learning, DL) -* zakres *ML* obejmujący techniki pozwalające urządzeniom samodzielną naukę w celu wykonania danego zadania*.*
4. sieci neuronowe (ang. *Neural networks, NN*) - rodzaj algorytmu wchodzący w skład *DL*, pozwalający na przetwarzanie danych wejściowych w konkretny, możliwy do przewidzenia wynik przy pomocy sieci wzorowanych na ludzkim układzie nerwowym [10].

Diagram

Description automatically generated

Rys. 2.1. Zależność między uczeniem maszynowym oraz innymi obszarami sztucznej inteligencji. Źródło: [8] #TODO: zrobić własne na podstawie tego.

Uczenie maszynowe jako podzbiór obszaru sztucznej inteligencji zajmuje się rozwiązywaniem problemów związanych z podejmowaniem decyzji, identyfikacją wzorców oraz wszelkiego rodzaju klasyfikacją wykonywanym prze maszyny. Odbywa się to poprzez wykonywanie różnego rodzaju obliczeń matematycznych, na przykład rozwiązywanie równań, formuł oraz algorytmów. W tym celu konstruowany w procesie uczenia jest model matematyczny, który w określony sposób może zostać następnie użyty do rozwiązania wspomnianych problemów.

Celem uczenia maszynowego jest uzyskanie takiego algorytmu, który poddany próbie na docelowym problemie znajdzie prawidłowe rozwiązanie, na podstawie którego możliwe będzie wyciągnięcie wniosków przez człowieka lub podrzędny system. Dobieranie odpowiednich rodzajów modeli do danego problemu, weryfikacja ich skuteczności oraz próby wynalezienia nowych rozwiązań jest procesem badań prowadzonych przez wielu specjalistów związanych z branżą technologiczną oraz naukową.

Element nauki modelu polega na stopniowej poprawie jego skuteczności rozwiązywania określonego wcześniej zdania. Odbywa się to w procesie uczenia przy użyciu zbioru uczącego zawierającego przygotowane wcześniej dane. Nauka w kontekście sztucznej inteligencji może być rozumiana na kilka sposobów:

* zdobycie wiedzy oraz zrozumienia danego problemu poprzez doświadczenie;
* naukę oraz zdobywanie umiejętności wymaganych do rozwiązania danego zadania;
* zapamiętywanie podejmowanych decyzji i otrzymanych w związku z tym rezultatów, następnie wykorzystanie tej wiedzy podczas podejmowania decyzji w przyszłości.

Uczenie maszynowe obejmuje także działanie związane z budowaniem systemów informatycznych różnego rozmiaru, które poprawiają swoją skuteczność wraz z doświadczeniem, w oparciu o zaimplementowany algorytm uczenia. Takie systemy mogą zostać wykorzystane do:

* rozpoznawania wzorców lub obiektów, np. rozpoznawania mowy, konkretnych obiektów, zachowań lub działań;
* przetwarzania języka naturalnego (ang. *Natural language processing, NLP*);
* diagnozowania chorób i schorzeń w medycynie;
* przewidywania przyszłych prognoz, trendów, np. sytuacji na giełdzie lub pogody na podstawie panujących warunków;
* konstrukcji robotów, systemów automatyki budynkowej, np. reakcja na panujące warunki w danym obiekcie;
* proponowanie użytkownikowi treści na podstawie historii jego wyszukań, np. na różnego rodzaju serwisach aukcyjnych i reklamowych

oraz wielu innych [9].

## Rodzaje uczenia maszynowego

Na podstawie charakteru, przebiegu oraz wyniku algorytmów uczenia maszynowego wyznaczono kilka kategorii uczenia, którym są one przypisywane. Każdy z nich charakteryzuje inna wydajność jak i komfort użytkowania wynikający z stopnia, w jakim dany algorytm wymaga poświęcenia uwagi człowieka. Mimo, że wymienione w dalszej części tego podrozdziału algorytmy uczenia maszynowego mogą zachowywać się inaczej w różnych warunkach środowiskowych, to każdy z nich może odnaleźć swoje zastosowanie praktyczne.

### Uczenie nadzorowane

Modele działające w oparciu o uczenie nadzorowane (ang. *Supervised learning*) przyjmują wartości wejściowe oraz docelowe wartości wyjściowe (patrz rys. 2.1). Oznacza to, że przed rozpoczęciem uczenia znany jest zbiór możliwych do uzyskania wartości wyjściowych – są one zdefiniowane na poziomie badanego zbioru danych. W oparciu o informacje podane na wejście algorytmu model generuje algorytm, który przekształca podane na wejście wartości w wartości docelowe. Stworzony w ten sposób algorytm nie odzwierciedla rzeczywistej funkcji z pełną dokładnością, a jedynie w drobnym przybliżeniu.

Diagram

Description automatically generated

Rys. 2.2. Schemat poglądowy struktury modelu zbudowanego w oparcie o uczenie nadzorowane. Źródło: opracowanie własne na podstawie [11]. #TODO: zrobić własne, jak np. wejście i wyjście kwadratu jakiegoś i wchodzi cos znane i wychodzi znane.

Ten rodzaj uczenia maszynowego jest najbardziej powszechnym. Używany dla problemów związanych z klasyfikacją – celem jest nauka już skonstruowanego systemu klasyfikacji, czyli przypisaniu danej próbce odpowiedniej klasy, np. wyznaczenie cyfry na podstawie zdjęcia ręcznie napisanego tekstu. Zagadnienie klasyfikacji najczęściej wiąże się z zastosowaniem takich metod jak sieci neuronowe lub drzewa decyzyjne, które w czytelny dla człowieka sposób określają etykietę dla każdego obiektu pochodzącego z zbioru wejściowego. Obie te metody są mocno zależne od informacji opisujących dany system klasyfikacji. Innym przykładem algorytmu będący przykładem uczenia nadzorowanego jest regresja. Celem jest przewidzenie przyszłego trendu na podstawie zebranych dotychczas danych.

Uczenie nadzorowane zakłada zachowany, wcześniej ustalony format danych wejściowych. Nie ma możliwości uzyskania informacji wyjściowych w sytuacji gdy, któraś z próbek różni się strukturą od określonej na poziomie architektury modelu struktury warstwy wejściowej. Dodatkowo możliwe jest „przetrenowanie” modelu (ang. *Overfitting*), czyli nauczenie się danych wejściowych na pamięć, a nie samych wzorców. Taki model będzie świetnie sprawdzał się podczas sesji uczenia, jednak jego zdolności generalizacyjne w rzeczywistych przypadkach użycia będą sporo niższe. Jest to bardzo częstym zjawiskiem, któremu można zapobiec stosując różne metody zwalniające proces uczenia lub jego całkowite przerwanie po zaobserowaniu początku tego zjawiska. Niemniej jednak zjawisko to dotyka tak naprawdę każdą z technik uczenia maszynowego. Zjawiskiem przeciwnym jest „niedotrenowanie” model (ang. *Underfitting*) wynikające z zbyt krótkiej sesji uczenia lub niewystarczających rozmiarów uczącego zbioru danych.

### Uczenie nienadzorowane

Uczenie nienadzorowane (ang. *Unsupervised learning*) różni się o uczenia nadzorowanego tym, że zbiór możliwych do uzyskania wartości wyjściowych nie jest znany przed przeprowadzeniem procesu uczenia. Takie algorytmy przyjmują wartości wejściowe i ich celem jest zaobserwowanie pewnych wzorców (patrz rys. 2.2) oraz różnic między próbkami oraz samodzielny podział na klasy lub grupy.

Diagram

Description automatically generated

Rys. 2.2. Schemat poglądowy struktury modelu zbudowanego w oparcie o uczenie nienadzorowane. Źródło: opracowanie własne na podstawie [11]. #TODO: zrobić własne jak wyżej.

W celu przeprowadzenia uczenia nienadzorowanego stosowane są między innymi dwa, główne podejścia. Pierwszym z nich „nagradzanie” modelu za prawidłowe rozwiązanie problemu. W takim podejściu celem modelu nie jest wykonanie danego zadania w sposób prawidłowy a jedynie uzyskanie jak największej nagrody. Takie działanie w sposób adekwatny odzwierciedla świat rzeczywisty, w którym prawidłowe zachowania są nagradzane a nieprawidłowe karane. Model skonstruowany w ten sposób nie musi mieć wiedzy na temat efektów swoich decyzji, pośrednio wynika to z uzyskanej nagrody. Drugim sposobem uczenia nienadzorowanego jest analiza skupień. Polega to na grupowaniu próbek zbioru danych w grupy posiadające wspólne cechy. Założenie może przypominać zagadnienie klasyfikacji, efektem działania algorytmu jest naturalny podział zbioru danych na grupy, symbolizujące różne klasy. Wyznaczone w ten sposób klasy nie mają jednak przypisanej etykiety. W oparciu o takie rozwiązanie działają chociażby systemy rekomendacji w sklepach internetowych takich jak *Allegro.pl*. Stosowane tam modele sztucznej inteligencji zbierają w grupy użytkowników przeglądających te same aukcje i na podstawie tej wiedzy wyświetlają użytkownikom inne, rekomendowane aukcje, które były także odwiedzane przez użytkowników danej grupy. Takie rozwiązanie znajdzie swoje zastosowanie w wielu problemach jednak systemy rekomendacji są bardzo dobrym przykładem, pozwalającym na zrozumienie zasady działania tego rodzaju uczenia.

Niestety uczenie nienadzorowane, mimo swojej zasady działania także jest podatne na przetrenowanie. Dodatkowo w celu uzyskanie sensownej dokładności takiego modelu najczęściej konieczne jest zastosowanie dużego zbioru danych, a sam proces uczenia pozwalający na osięgnięcie akceptowalnych wyników może być znacznie dłuższy w porównaniu do uczenia nadzorowanego. Nadal znanych jest masa przypadków będących dowodem skuteczności tego rodzaju uczenia – od nauki w gry komputerowe, poprzez detekcję anomalii w ruchu sieciowym, aż do stworzenia autopilotów sterujących pojazdami mechanicznymi [11].

### Pozostałe rodzaje uczenia maszynowego

Pozostałe rodzaje uczenia maszynowego są nadal w sporej części na etapie badań, jednak w przyszłości mogą się okazać najlepszym rozwiązaniem wspomagającym dla wielu systemów informatycznych. Na przykład - kompromisem pomiędzy uczeniem nadzorowanym a nienadzorowanym docelowo ma być uczenie częściowo nadzorowane (ang. *Semi-supervised learninig*). Taki rodzaj uczenia miałby być w stanie przetwarzać zbiory danych zawierających próbki z już przypisanymi etykietami oraz próbkami bez etykiet. Skonstruowany w ten sposób model mógłby skutecznie sobie radzić w praktyce niezależnie od danych użytych na jego wejściu. Co więcej oba rodzaje uczenia mogłyby się nawzajem wspierać w procesie uczenia – uzyskiwane wyniki byłyby wartościowe dla całego modelu, a nie jego części [12].

Uczenie wzmocnione (ang. *Reinforcement learning*) polega na podejmowaniu przez model działań, które wpływają na środowisko a następnie obserwacji efektów wywołanych tym danym działaniem. Jedno z podejść zakłada podejmowanie działań na zasadzie prób i błędów, obserwację ich wpływu na środowisko oraz wyciąganie wniosków pozwalające wyróżnić najbardziej efektywny zbiór akcji. Drugi ze sposobów polega na użyciu narzędzi programistycznych w celu wykonania estymacji wpływu danej akcji. Obecnie nie jest do końca jasne, jakie miałyby być konkretne zastosowania obu metod, ani w jakich warunkach jedna byłaby skuteczniejsza od drugiej [13].

## Sieci neuronowe

Ludzki układ nerwowy przypomina budową sieć, w której połączone ze sobą są miliardy komórek nerwowych zwanych również neuronami. Komórki te komunikują się poprzez krótkie sygnały elektryczne. Z kolei wymiana tych impulsów odbywa się przy użyciu węzłów zwanych synapsami, zlokalizowanych na gałęziach zwanych dendrytami. Każdy z neuronów jest łączy tysiącami gałęzi z innymi komórkami, w związku z czym nieustannie otrzymuje całą masę sygnałów. Chcąc uprościć zasadę działania neuronów jako przekaźników informacji możemy rozumieć następująco: na podstawie otrzymanych sygnałów, po osiągnięciu określonego stanu lub spełnieniu określonego warunku dana komórka „zapala się”. Z kolei taka aktywność każdej z komórek może świadczyć o zupełnie innym charakterze bodźca zewnętrznego. Taka kombinacja składająca się miliardów aktywnych i nieaktywnych komórek może określić w sposób jednoznaczny jakieś zjawisko lub sytuację w danej chwili.

Stworzona na przestrzeni lat swego rodzaju adaptacja ludzkiego układu nerwowego, jedna z najczęściej stosowanych metod uczenia maszynowego nazywana jest sieciami neuronowymi (ang. *Neural networks*). Tak jak i w układzie nerwowym sieci neuronowe składają się z węzłów zwanych neuronami (patrz rys. 2.3). Funkcję podobną do synaps pełnią wagi, od wartości których zależna jest ostateczna wielkość wejściowa komórki neuronowej.

Diagram

Description automatically generated

Rys. 2.3. Ideowy schemat prostej sieci neuronowej. Źródło: opracowanie własne.

Wyjściowa wartość w komórce neuronowej zależna jest także od funkcji aktywacji, która na podstawie danych wejściowych wyznacza ostateczną wartość w danym neuronie, która najczęściej znajduje w zakresie *<0, 1>* lub rzadziej *<-1, 1>.* Kilka podstawowych funkcji aktywacji zostało przedstawionych na rys. 2.4.

![Chart

Description automatically generated]()

Rys. 2.4. Najbardziej powszechne funkcje aktywacyjne. Źródło: opracowanie własne na podstawie [15-16].

Neurony budują połączone ze sobą warstwy, które z kolei tworzą całą sieć. Każda kolejna warstwa rozbiera dane wejściowe na mniejsze części, składające się na daną cechę badanej próbki. Każdy system zawierający komórki zwane neuronami nazywany jest siecią neuronową. W związku z tym prosty system, składający się z dwóch warstw – wejściowej i wejściowej nazywany jest siecią neuronową, to samo tyczy się bardzo skomplikowanych systemów mogących zawierać setki warstw i miliony komórek. Poziom skomplikowania sieci zależy od złożoności danych wejściowych, a ostatnia warstwa powinna zawierać prostą do zrozumienia informacje na temat wykrytych wzorców. W ramach przykładu możemy rozważyć wspomniany już w niniejszej pracy problem rozpoznawania ręcznie pisanych cyfr. Warstwa wejściowa będzie zależała od formatu danych wejściowych, np. jakości zdjęcia a dokładniej liczby pikseli. Warstwa wyjściowa z kolei w typowym przypadku zawierałaby 10 neuronów – po jednym na każdą z cyfr. Wynik może być jednoznaczny, tj. po dokonaniu procesu predykcji aktywny może być 1 neuron (ma wartość maksymalną wynikającą z funkcji aktywacji, np. 1), a pozostałe 9 zostają nieaktywne (mają wartość np. 0). Może też być niejednoznaczny – wartości neuronów warstwy wyjściowej zawierają wartości z określonego zakresu, przy czym największa wartość oznacza największe prawdopodobieństwo wykrycia danej cyfry, a wartość najmniejsza wskazuje najmniejsze prawdopodobieństwo wystąpienia danej cyfry. Uzyskany wynik jest również zależny od dobranej funkcji aktywacji [14].

### Rodzaje sieci neuronowych

Sieci neuronowe jako ogólny system dzielimy najczęściej na trzy typy. Charakteryzują się one różną budową, sposobem połączeń oraz kierunkiem przepływu. Wyróżniane rodzaje sieci opisano poniżej.

1. Sieci jednokierunkowe – składają się one z kilu warstwach o jednym kierunku przepływu. Każda kolejna warstwa połączona jest z następną – nie istnieją połączenia z warstwami poprzednimi. Z reguły taką sieć możemy podzielić na jedną warstwę wejściową, jedną warstwę wyjściową oraz co najmniej jedną warstwę ukrytą.
2. Sieci rekurencyjne – sieci, w których występuje jedno lub więcej sprzężeń zwrotnych. Oznacza to, że wielkości podane na wejście pierwszej warstwy zależą także od wartości wyjściowych ostatniej warstwy. Sama struktura sieci jest taka sama jak w przypadku sieci jednokierunkowych.
3. Sieci komórkowe – sieci w których informacje są wysyłane pomiędzy wszystkimi komórkami sąsiednimi, a sama sieć przypomina układ współrzędnych. Sieci te nazywane są też samoorganizującymi się mapami, a przykładem może być mapa Kohonena [17].

Wspomniano we wcześniejszej części pracy sieci neuronowe składają się z warstw. W zależności od zadanego problemu inne rodzaje warstw sieci neuronowych różnią się skutecznością. Poniżej znajduje się krótki opis kilku podstawowych typów warstw sieci neuronowych.

* Warstwa gęsta (ang. *Dense layer)* to najprostsza z dostępnych warstw. Sposób działania opiera się na obliczaniu prostej zależności wynikającej z funkcji aktywacji [18]. Najczęściej stosowana warstwa, jednak nie najlepiej sobie z bardziej złożonymi problemami, np. identyfikacją obiektów na podstawie obrazów dużej rozdzielczości.
* Warstwa konwolucyjna (ang. *Convolutional layer)* to warstwa używane najczęściej podczas problemów klasyfikacji przy użyciu zbiorów zawierających dane w postaci obrazów. Dzięki swojej architekturze umożliwiają podział zdjęcia na elementarne części, mniejsze od pojedynczych pikseli. W taki sposób każda z warstw może zawierać informacje o określonych cechach danego obrazu.
* Warstwa łącząca (ang. *Pooling layer)* to warstwa, która ma za zadanie ograniczenie liczby próbek w danym segmencie sieci, poprzez wybór maksymalnej wartości z zadanej wielkości części danych wejściowych [19].
* Warstwa spadająca (ang. *Dropout layer)* to warstwa, daje wartość równą 0 losowym wielkościom wejściowym. Prawdopodobieństwo zmiany jest głównym parametrem wejściowym [20].
* Warstwa normalizująca wsadowo (ang. *Batch normalization layer*) to warstwa normalizująca wartości wejściowe w celu uzyskania ustandaryzowanych zakresów wielkości [21].

## Tradycyjne uczenie maszynowe

Tradycyjny model uczenia maszynowego przewiduje istnienie wielowęzłowego systemu, gdzie każdy węzeł symbolizuje pojedyncze urządzenie lub organizację zbierającą dane. Taka struktura umożliwia przetwarzanie względnie dużych zbiorów danych, których powiększanie odbywa się poprzez dokładanie kolejnych węzłów.

Ten sposób uczenia przewiduje agregację danych z wielu urządzeń w jeden zbiór, który następnie służy do nauki modelu określonego typu. Tak skonstruowany model zostaje następnie wysłany do każdego urządzenia korzystającego z danej aplikacji. Samo urządzenie zbiera dane w czasie rzeczywistym, a proces ich wysyłania do jednostki na której ma miejsce proces uczenia odbywa się okresowo. Jest to jedną z wad tego rodzaju uczenia – nie ma możliwości by zbierane przez urządzenie dane na bieżąco aktualizowały używany model. Niemniej jednak cały proces przeprowadzony w opisywany sposób skutkuje rosnącą skutecznością danego algorytmu.

W obecnych czasach wszystkie urządzenia mobilne, IoT (ang. *Internet of things*) kolekcjonują całą masę danych, w związku z czym proces trenowania modelu poprzez pojedynczą jednostkę, mimo wysokiej mocy obliczeniowej, może trwać sporą ilość czasu. Dodatkowo dane wędrują między urządzeniami, co może narażać ich bezpieczeństwo. Między innymi z tych powodów zaczęto szukać innych, bardziej wydajnych i bezpiecznych metod szkolenia modeli sztucznej inteligencji a taki sposób uczenia nie jest powszechnie używany w celach komercyjnych.

Proces uczenia maszynowego na podstawie danych zebranych przez wiele rozproszonych urządzeń w oparciu o tradycyjną metodę możemy w uproszczeniu podzielić na cztery kroki (patrz rys. 2.1):

1. okresowe wysyłanie zgromadzonych przez urządzenia rozproszone danych do jednostki centralnej;
2. agregacja uzyskanych danych w jeden zbiór uczący;
3. przeprowadzenie procesu uczenia na jednostce centralnej;
4. dystrybucja zbudowanego modelu sztucznej inteligencji między urządzeniami korzystającymi z systemu [2].

**Diagram

Description automatically generated**

Rys. 2.1. Diagram przedstawiający koncept tradycyjnego uczenia maszynowego. Źródło: opracowanie własne.

## Rozproszone uczenie maszynowe

W związku z ogromnym tempem przyrostu danych zbieranych przez urządzenia u i organizacje zaczęto szukać sposobów, aby zoptymalizować proces uczenia maszynowego. Powstało wtedy takie pojęcia jak nauka na dużą skalę (ang. *large-scale learning*), opisujące zbiór zagadnień badających zagadnienia związane z uczeniem w oparciu o duże zbiory danych.

Rozproszone uczenie maszynowe (ang. *distributed machine learning*) jest jednym z proponowanych rozwiązań stale rosnących zbiorów danych i czasów uczenia. Tak jak w tradycyjnym sposobie uczenia zakłada istnienie wielowęzłowego systemu, składającego się z urządzeń lub organizacji zbierających dane. Operuje także na zagregowanych danych, jednak proces uczenia dystrybuowany jest pomiędzy wiele urządzeń. Dzięki temu możliwe jest wykonywanie obliczeń i algorytmów równolegle, w związku z czym moc obliczeniowa używana w jednostce czasu jest znacznie większa a sam czas uczenia jest znacznie mniejszy. Rozwiązanie to, tak jak i rozwiązanie tradycyjne, pozwala na bardzo dynamiczny wzrost agregowanych danych poprzez dodawanie dodatkowych urządzeń i organizacji. Pozwala też na prostą skalowalność części systemu odpowiedzialnej za udostępnienie mocy obliczeniowej, bowiem wystarczające jest dołożenie kolejnych urządzeń odpowiedzialnych za proces uczenia. W związku z tym rozwiązanie to skutecznie radzi sobie z problemem długiego czasu uczenia modeli sztucznej inteligencji. Niestety nie rozwiązuje problemu prywatności i bezpieczeństwa danych, ze względu na wspomnianą agregację i centralizację danych w postaci jednego zbioru uczącego [6].

Proces uczenia maszynowego z uwzględnieniem założeń proponowanych przez rozproszenie uczenie maszynowe odbywa się dokładnie w taki sam sposób jak przy wspomnianej metodzie tradycyjnej z tą różnicą, że uczenie odbywa się na stworzonej do tego platformie składającej się z wielu urządzeń odpowiedzialnych za wykonywanie obliczeń.

## Federacyjne uczenie maszynowe

Koncept federacyjnego uczenia maszynowe pierwszy raz światło dzienne ujrzał w 2016 roku, kiedy został zaprezentowany przez firmę *Google*. Ten nowy na tamte czasy model uczenia został użyty w aplikacji *Google keyboard,* gdzie w sposób kolaboracyjny trenowano model na przy użyciu kilku urządzeń z systemem *Android*. Jednak federacyjny sposób uczenia może zostać zaimplementowany użyciu każdego z urządzeń używanych w obszarze sztucznej inteligencji. Sama idea ma duży potencjał zrewolucjonizować rynek *AI*. Ciekawym przykładem może być sytuacja, gdzie technolodzy oraz pracownicy branży medycznej z całego świata w sposób federacyjny trenowali model mający na celu wykrywać chorobę COVID-19 na podstawie skanów klatki piersiowej [4]. Głównym celem dla którego podjęto próby wynalezienia kolejnego sposobu uczenia maszynowego było zapewnienie bezpieczeństwa oraz zniwelowanie ryzyka utraty lub wycieku danych, używając zestawów danych znajdujących się na wielu urządzeniach lub w wielu organizacjach.

Federacyjny sposób tworzenia modeli sztucznej inteligencji, w przeciwieństwie do obu wspomnianych wcześniej rodzajów uczenia, nie wymaga by zgromadzone przez urządzenia dane opuszczały pamięć danego urządzenia. Zamiast tego szkolenie przebiega lokalnie, przy użyciu zebranych danych. Odbywa się to, tak jak w przypadku modelu tradycyjnego, cyklicznie, jednak ze względu na brak konieczności wysyłania danych do serwera interwały są w większości przypadków dużo krótsze, a sam czas szkolenia jest mniejszy ze względu na mniejsze przyrosty zbioru danych. Wytrenowane w ten sposób modele są następnie wysyłane do platformy centralnej, gdzie poddawane są agregacji. W efekcie czego z wielu takich modeli uzyskiwany jest jeden, wyjściowy model, który jest dystrybuowany pomiędzy urządzeniami. Przy zastosowaniu konfiguracji, w której kilka organizacji wspólnie pracuje nad modelem sztucznej inteligencji sytuacja wygląda w sposób analogiczny – parametry modeli są agregowane na współdzielonym serwerze lub chmurze. Następnie model wynikowy jest wysyłany do organizacji, które dokonują aktualizacji lokalnie.

Docelowo przy zastosowaniu takiej metody zbiór danych znajdujący się na danym urządzeniunie powinien być w żaden sposób udostępniony pozostałym urządzeniom. Niemniej dopuszczalne są pewne odstępstwa od tej normy, przy czym muszę w takiej sytuacji zostać ustalone i zachowane odpowiednie procedury bezpieczeństwa. Skuteczność uzyskanego w sposób federacyjny algorytmu powinna być zbliżona do algorytmu stworzonego na tym samym zbiorze danych przy użyciu tradycyjnego sposobu uczenia.

Analogicznie do tradycyjnego oraz rozproszonego modelu uczenia maszynowego, proces federacyjnego uczenia możemy w uproszczeniu podzielić na cztery etapy (patrz rys. 2.2):

1. cykliczne przeprowadzanie nauki modelu na podstawie gromadzonych lokalnie danych;
2. okresowe wysyłanie wyszkolonych modeli do serwera;
3. agregacja uzyskanych modeli w jeden, wynikowy model;
4. dystrybucja zbudowanego modelu sztucznej inteligencji między urządzeniami korzystającymi z systemu [3].

Diagram

Description automatically generated

Rys. 2.2. Diagram przedstawiający koncept federacyjnego uczenia maszynowego. Źródło: opracowanie własne.

## Prywatność danych w uczeniu federacyjnym

Prywatność i bezpieczeństwo danych są kluczowymi właściwościami federacyjnego uczenia maszynowego. Ich zapewnienie wymaga istnienia różnego rodzaju modelów bezpieczeństwa i analizy. Poniżej wyróżniono kilka przykładowych modeli.

### Prywatność różnicowa

Prywatność różnicowa (ang. *Differential Privacy)* polega na dodaniu szumu do danych lub użyciu metod generalizacji co ma na celu ukrycia pewnych wrażliwych cech danego zbioru. W wyniku takiego działania pojedyncze próbki mogą być trudne do rozróżnienia, a sam algorytm uczenia działa na danych przybliżonych do rzeczywistych.

### Szyfrowanie homomorficzne

Szyfrowanie homomorficzne (ang. *Homomorphic Encryption)* jest to typ algorytmów szyfrujących, pozwalających na przeprowadzanie obliczeń przy użyciu zaszyfrowanych danych, bez konieczności ich deszyfrowania. Ze względu na fakt, że klucz deszyfrujący znany jest tylko przez urządzenie źródłowe, ryzyko wycieku danych jest wyjątkowo niskie. Powszechnie stosowane w algorytmach uczenia maszynowego. Zastosowanie tego modelu w większości przypadków skutkuje uzyskaniem kompromisu pomiędzy dokładnością a bezpieczeństwem i prywatnością danych.

### Bezpieczne obliczenia wielopartyjne

Bezpieczne obliczenia wielopartyjne (ang. *Secure Multi-party Computation, SMC)*. Tego typu modele w sposób naturalny angażują wiele obiektów, przy czym głównym założeniem jest, że każdy z obiektów zna tylko swoje dane wejściowe oraz wyjściowe. Zerowa wiedza jest szeroko pożądana w przypadku federacyjnego uczenia maszynowego, jednak często wymaga ona skomplikowanych protokołów obliczeniowych w związku z czym osiągniecie takiego stanu nie jest proste. Niemniej jednak w określonych warunkach częściowa wymiana wiedzy między danymi instancjami może być dopuszczalna pod warunkiem, że są zachowane odpowiednie działania mające na celu zachowanie bezpieczeństwa. Takie rozwiązanie w niektórych przypadkach może okazać się optymalnym kompromisem pomiędzy zapewnieniem wystarczającego bezpieczeństwa oraz wydajności danego systemu [3].

## Rodzaje federacyjnego uczenia maszynowego

Aktualnie jesteśmy w stanie zaobserwować dwa sposoby kategoryzacji typów uczenia federacyjnego:

* ze względu na wspólne aspekty udostępnianych przez urządzenia zbiorów danych,
* ze względu na charakter decentralizacji źródeł danych.

Aktualnie nie wszystkie z opisanych rozwiązań są powszechnie używane w środowiskach produkcyjnych jednak widoczna jest tendencja wzrostowa. Poniżej opisane zostały typy należące do pierwszej z kategorii.

### Uczenie federacyjne wertykalne

Uczenie federacyjne wertykalne (ang. *Vertical Federated Learning)* polega na trenowaniu modeli na podstawie danych pochodzących z różnych źródeł, opisujących różne dane kontekstowe dotyczące tych samych obiektów. Jako przykład posłużyć mogą przychodnia lekarska i lokalny bank znajdujące się w niewielkiej miejscowości. Najprawdopodobniej z obu tych instytucji korzysta spora część mieszkańców, zatem identyfikatory próbek, na przykład numer pesel, będą w obu przypadkach takie same. Jednak ze względu na różny charakter obu działalności badane atrybuty będą zupełnie inne. Wertykalny rodzaj uczenia maszynowego polega na agregacji parametrów modeli przeznaczonych do operowania na zbiorach danych opisujących różne cechy i zbudowaniu globalnego modelu, gdzie każda ze stron będzie brała czynny udział w procesie uczenia. Docelowo tak uzyskany model mógłby być z powodzeniem stosowany w każdej z tych organizacji. Ten rodzaj szkolenia zakłada, że agenci mogą zagrażać sobie nawzajem pod kątem bezpieczeństwa danych. Niemniej jednak każda ze stron jest niezależna ma ryzyka nawiązania współpracy przez jakiekolwiek z nich. Po zakończeniu procesu uczenia każda ze stron otrzymuje model posiadający parametry związane z cechami, które opisywał wysłany zbiór danych w związku z czym każda ze stron musi brać udział w podczas wnioskowania (ang. *inference time*).

A picture containing graphical user interface

Description automatically generated

Rys. 2.3. Uczenie federacyjne wertykalne. Źródło: opracowanie własne na podstawie [3].

### Uczenie federacyjne horyzontalne

Uczenie federacyjne horyzontalne (ang. *Horizontal Federated Learning)* jest to proces uczenia w sytuacji gdy zbiory danych pochodzących z różnych źródeł opisują te same dane kontekstowe, dotyczące jednak różnych zdarzeń lub obiektów. Przykładem mogą być dwie prywatne przychodnie lekarskie znajdujące się w różnych miastach. Z dużym prawdopodobieństwem można założyć, że będą one zbierać te same lub bardzo podobne dane na temat swoich pacjentów. Jednak w związku z położeniem w dwóch różnych miejscowościach w skład pacjentów obu przychodni będą wchodziły inne osoby. Przykładem mogą być wszystkie organizacje działające w tej samej branży, mające jednak inne grupy odbiorców. Podczas zastosowania tego typu uczenia możliwa jest też wymiana danych na temat danej grupy cech. Jest to najczęściej używany rodzaj uczenia, stosowany powszechnie przez *Google*, na przykład w aplikacji *Google Assistant* [7]. Pozwala on na zachowanie bezpieczeństwa oraz prywatności między użytkownikami, a jedynym obiektem w całym systemie mogącym stwarzać zagrożenie jest serwer. Po zakończeniu procesu uczenia wszystkie parametry modelu wynikowego udostępniane są dla każdego odbiorcy.

A picture containing timeline

Description automatically generated

Rys. 2.4. Uczenie federacyjne horyzontalne. Źródło: opracowanie własne na podstawie [3].

### Uczenie federacyjne transferowe

Uczenie federacyjne transferowe (ang. *Federated Transfer Learning*) może zostać zaimplementowany w sytuacji gdy dwie organizacje opisują zarówno różne cechy jak i obiekty. Przykładem mogą być bank oraz sklep z elektroniką znajdujące się w innych krajach. Ze względu na dużą odległość między obydwoma miejscami najprawdopodobniej grupa odbiorców będzie zupełnie inna. Ze względu na różne branże obu organizacji tylko bardzo mała część gromadzonych danych kontekstowych będzie miała taki sam charakter. Ten typ uczenia służy raczej jako rozszerzenie do wspomnianych wcześniej uczenia wertykalnego i horyzontalnego, aniżeli jako samodzielne rozwiązanie [3].

A picture containing table

Description automatically generated

Rys. 2.5. Uczenie federacyjne transferowe. Źródło: opracowanie własne na podstawie [3].

### Pozostałe rodzaje uczenia federacyjnego

Uczenie federacyjne z zastosowaniem wielu silosów (ang. *Cross-Silo Federated* Learning) oraz uczenie federacyjne z zastosowaniem wielu urządzeń (ang. *Cross-Device Federated Learning*) należą do drugiej kategorii według której dzielimy typy uczenia federacyjnego. Pierwszy z nich zakłada istnienie „silosów” gromadzących dane takich jak organizacje rób regiony geograficzne. Przy takim ustawieniu liczba klientów wynosi waha się zazwyczaj od 2 do 100. Z kolei drugi zakłada istnienie od 0 do nawet 1010 klientów, jednak w tej sytuacji mała część z nich bierze czynny udział w procesie budowania modelu wyjściowego. Uczenie z zastosowaniem wielu silosów jest coraz częściej stosowane, na przykład w takich branżach jak medyczna lub finansowa [5].

# Projekt i implementacja oprogramowania

Trzeci rozdział ma na celu przedstawienie wybranych podejścia oraz założeń towarzyszących autorowi podczas tworzenia oprogramowania umożliwiającego przeprowadzenie badań nad skutecznością federacyjnego uczenia maszynowego. Następnie zostaną przedstawione możliwości konfiguracyjne stworzonej aplikacji, umożliwiającej w pełni zautomatyzowane przeprowadzenie eksperymentów wraz zapisaniem informacji na ich temat oraz wyników wyjściowych jak i wykresów.

## Projekt oprogramowania

Celem części implementacyjnej wchodzącej w zakres praktyczny pracy było stworzenie aplikacji, która umożliwiałaby przeprowadzenie eksperymentów i badań obrazujących skuteczność federacyjnego uczenia maszynowego. Standardowym rozwiązaniem, najczęściej stosowanym w tego typu badaniach jest ich przeprowadzanie przy pomocy notatników umożliwiających przeprowadzanie obliczeń, ich wizualizację w sposób interaktywny przy bieżącym zapisywaniu uzyskiwanych wyników w pamięci podręcznej urządzenia. Najczęściej używanymi narzędziami są:

* *Jupyter Notebook*, umożliwiający uruchomienie notatnika na lokalnej maszynie;
* *Google Colab* uruchamiający kod źródłowy na zasobach maszyny wirtualnej firmy *Google*.

Oba rozwiązania bazują na języku *Python.* Takie podejście jest wygodne, ponieważ pozwala na uniknięcie uruchamiania całości kodu źródłowego po każdej jego zmianie. To dzięki możliwości podziału kodu na sekcje, gdzie każda z sekcji może zostać uruchomiona oddzielnie, mimo współdzielonej pamięci.

Autor zdecydował się na stworzenie pełnoprawnej aplikacji, mającej formę szeregu skryptów, umożliwiającej w pełni zautomatyzowane przeprowadzenie eksperymentów – od wyboru rodzaju uczenia, przez podział danego zbioru danych na podzbiory oraz uczenie modeli, na zebraniu metryk uczenia, stworzeniu wykresów oraz ich zapisaniu kończąc. Takie podejście pozwalało na zastosowanie odpowiedniej struktury katalogowej oraz podziału na modułu, odpowiedzialne za inne funkcjonalności systemu. Dzięki temu ma bardziej przejrzystą formę, jest także prostsza w zrozumieniu, zarządzaniu oraz utrzymaniu.

Celem takiego działania było uniknięcie konieczności ingerencji w kod źródłowy w celu zmiany parametrów uczenia – cała konfiguracja miała odbywać się poprzez plik konfiguracyjny, pełniący w tym przypadku rolę interfejsu między silnikiem aplikacji a człowiekiem. Umożliwiało to przeprowadzenie badań w sposób mało skomplikowany, nawet z punktu użytkownika nie mającego wiedzy na temat zawartości kodu źródłowego. Możliwe było także całkowite skupienie się na odpowiednim doborze parametrów i obserwacji uzyskanych wyników, a nie edycji kodu źródłowego.

Warto zaznaczyć, że celem części praktycznej nie było uzyskanie jak najlepszej skuteczności wynikowego modelu sztucznej inteligencji a zbadanie możliwości uczenia federacyjnego oraz wpływu poszczególnych parametrów na ten rodzaj uczenia.

## Opis możliwości konfiguracyjnych oprogramowania

Plik umożliwiający konfigurację procesu uczenia jak i innych istotnych parametrów z punktu widzenia aplikacji miałem formę pliku w formacie *JSON*. Takie rozwiązanie wydaje się przejrzyste nawet dla osób niezwiązanych z tworzeniem oprogramowania. Zrzut ekranu przedstawiający możliwości konfiguracyjne znajduje się na rys. 3.1. Poniżej znajduje się lista możliwych do parametryzacji wartości wraz z krótkim opisem.

*learning\_type* – wybrany przez użytkownika rodzaj uczenia. Możliwe przeprowadzenie uczenia:

* tradycyjnego – wartość *traditional;*
* federacyjnego – wartość *fedarated.*

Podanie innych wartości skutkuje wyświetleniem błędu o stosownym komunikacie.

*dataset* – sekcja odpowiedzialna za definicję parametrów odpowiedzialnego za wybór i wstępny podział zbioru danych. Dostępne pola to:

* *name* – nazwa zbioru danych, możliwe podanie jakiegokolwiek zbioru danych udostępnianego przez bibliotekę *Tensorflow*.
* *test\_data\_ratio* – część wybranego zbioru danych przeznaczona na stworzenie zbioru testowego.

*federated\_learning* – sekcja konfiguracji fedearcyjnego uczenia maszynowego. Dostępne podsekcje:

* *data\_distribution* – ustawienia podziału zbioru danych między uczestników w procesie uczenia federacyjnego. Dostępne pola to:
  + *clients\_number* – liczba klientów biorących udział w kolaboracyjnym trenowanie wyjściowego modelu.
  + *main\_classes\_per\_client\_number* – liczba klas głównych, które widzą klienci.
  + *main\_class\_ownership\_per\_client\_ratio* – stosunek próbek każdej z klas głównych widzianych przez klienta, do całości próbek danej klasy znajdującej się w wybranym zbiorze danych.
* *cycle* – ustawienia całego cyklu uczenia. Dostępne pola to:
  + *iterations* – liczba iteracji procesu uczenia;
  + *iterations\_to\_aggregate* – liczba iteracji pomiędzy agregacją modeli klientów w jeden model globalny.
* *client* – sekcja ustawień klientów. Dostępne pola oraz podsekcje:
  + *weights\_sending*  –ustawienia sposobu wysyłania wag modeli przez klientów do serwera. Dostępna pola to:
    - *send\_only\_changed\_weights* – informacja o tym, czy wysyłane powinny być jedynie wagi, które zmieniły się na przestrzeni ostatniej iteracji. Dostępne wartości:
      * *true*;
      * *false*.
* *server* – sekcja ustawień serwera, tj. jednostki globalnej w procesie uczenia. Dostępne pola oraz podsekcje:
  + *early\_stopping* – sekcja odpowiedzialna za zachowanie procesu uczenia w sytuacji, gdy wystąpiło zjawisko przetrenowania modelu globalnego. Dostępne pola to:
    - *enabled* – informacja o tym, czy proces uczenia powinien zostać przerwany w sytuacji, gdy dochodzi do zjawiska przetrenowania.
    - *metric\_type* – metryka, na podstawie zmian której ma być podjęta decyzja o przerwaniu uczenia. Dostępne wartości:
      * *loss* – uzyskana wartość funkcji strat na zbiorze testowym,
      * *accuracy* – uzyskana dokładność modelu na zbiorze testowym.
    - *patience* – maksymalna liczba epok bez poprawy wartości zadeklarowanej metryki.

*traditional\_learning* - sekcja konfiguracji tradycyjnego uczenia maszynowego. Dostępne pola oraz podsekcje:

* *epochs* – liczba epok procesu uczenia;
* *early\_stopping* – sekcja odpowiedzialna za zachowanie procesu uczenia w sytuacji, gdy wystąpiło zjawisko przetrenowania modelu. Dostępne pola to:
  + *enabled* – informacja o tym, czy proces uczenia powinien zostać przerwany w sytuacji, gdy dochodzi do zjawiska przetrenowania.
  + *metric\_type* – metryka, na podstawie zmian której ma być podjęta decyzja o przerwaniu uczenia. Dostępne wartości:
    - *loss* – wartość funkcji strat na zbiorze uczącym,
    - *accuracy* – uzyskana dokładność modelu na zbiorze uczącym.
    - *val\_loss* – wartość funkcji strat na zbiorze walidacyjnym,
    - *val\_accuracy* – uzyskana dokładność modelu na zbiorze walidacyjnym.
  + *patience* – maksymalna liczba epok bez poprawy wartości zadeklarowanej metryki.
  + *restore\_best\_weights* – informacja o tym, czy po przerwaniu uczenia powinny zostać przywrócone wagi z najlepszej epoki. Dostępne wartości:
    - *true;*
    - *false.*

Text

Description automatically generated with low confidence

Rys. 3.1a. Plik konfiguracyjny umożliwiający parametryzację procesu uczenia. Źródło: opracowanie własne.

## Text Description automatically generated with low confidence

Rys. 3.1b. Plik konfiguracyjny umożliwiający parametryzację procesu uczenia. Źródło: opracowanie własne.

## Wykorzystane technologie oraz narzędzia

W celu stworzenia opisanej aplikacji wykorzystane zostały następujące technologie oraz narzędzia:

* język programowania *Python* w wersji 3.9.0 wraz z bibliotekami:
  + *TensorFlow* w wersji 2.8.0 wraz z wszystkimi zależnościami;
  + *Keras* w wersji 2.8.0 wraz z wszystkimi zależnościami;
  + *Matplotlib* w wersji 3.5.1 wraz z wszystkimi zależnościami;
  + *pandas* w wersji 1.4.2 wraz z wszystkimi zależnościami;
  + *Numpy* w wersji 1.22.3 wraz z wszystkimi zależnościami.
* *Makefile*

# Opis przebiegu wykonanych badan i eksperymentów

# Podsumowanie

# Bibliografia

**Opracowania książkowe**

1. TODO – podzielić na **Opracowania książkowe**, **Dokumenty**, **Źródła internetowe**, pozmieniać kolejność zgodnie z występowaniem w tekście. Sformatować.

**Dokumenty**

1. *Machine learning: Trends, perspectives, and prospects*, M.I. Jordan, T.M. Mitchell, **artykuł**
2. *Federated Machine Lraning: Concept and Applications,* Yang Liu, Tianjian Chen, Yongxing Tong, **artykuł**
3. *Federated learning: Opportunities and Challenges,* Priyanka Mary Mammen, , **artykuł**
4. *Cross-silo federeated training in the cloud with diversity scalind and semi-supervised learning,* Kishore Nandury, Anand Mohan, Frederick Weber, , **artykuł**
5. *A surver of methods for distributed machine learning,* Diego Peteiro-Barral, BerthaGuijarro-Berdinas, , **artykuł**

**Źródła internetowe**

1. Google hey google [dostęp: 05.06.2022], <https://support.google.com/assistant/answer/10176224?hl=en>
2. Czym jest uczenie maszynowe (Machine Learning)? – SAP [dostęp: 05.06.2022] <https://www.sap.com/poland/insights/what-is-machine-learning.html>
3. *Machine Learning Overview*, Taiwo Oladipupo Ayodele, **artykuł?**
4. Deep learning vs. Machine learning – Azure Machine Learning [dostęp: 05.06.2022], <https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning>
5. *Types of Machine Learning Algorithms ,*Taiwo Oladipupo Ayodele, , **artykuł**
6. *A survey on semi-supervised learning*, Jesper E. van Engelen, Holger H. Hoos, **artykuł**
7. *Reinforcement learning: A survey*, Leslie Pack Kaebling, **artykuł**
8. *An itroduction to neural networks*, Kevin Gurney, University of Sheffield, 1997 **książka**
9. *Funkcja aktywacji – Wikipedia, wolna encyklopedia* [dostęp 07.06.2022] <https://pl.wikipedia.org/wiki/Funkcja_aktywacji>
10. *Activation functions in Python* [dostęp 08.06.2022] <https://www.nbshare.io/notebook/751082217/Activation-Functions-In-Python/>
11. *Wstęp do sieci neuronowych* [dostęp 08.06.2022] <https://home.agh.edu.pl/~vlsi/AI/wstep1/sieci.html>
12. *Dense layer* [dostęp 08.06.2022] <https://keras.io/api/layers/core_layers/dense/>
13. *An Introduction to Convolutional Neural Networks*, Keiron O`Shea and Ryan Nash **artykuł**
14. *Dropout layer* [dostęp 08.06.2022] <https://keras.io/api/layers/regularization_layers/dropout/>
15. *BatchNormalization layer* [dostęp 08.06.2022] <https://keras.io/api/layers/normalization_layers/batch_normalization/>