**Monika Etrych**

Projekt Sztuczne Sieci Neuronowe

klasyfikacja zdatności wód do spożycia

# Opis danych:

1. ph (0 to 14)

2. Twardość - zawartość magnezu i wapnia [mg/L]

3. Ciała stałe – całkowicie rozpuszczone ciała stałe – np. sole, minerały

4. Chloraminy

5. Siarki

6. Przewodnictwo elektryczne [μS/cm].

7. Węgiel organiczny

8. Trihalometany [μg/L].

9. Mętność - miara przepuszczlności światła

10. Zdatność do spożycia: 1 – zdatna, 0 – niezdatna dla człowieka

# Przegląd literatury:

Dane zostały pobrane z Kaggle, a w komentarzach pojawiło się wiele dowodów na to, że dane są losowe, tak więc interpretowanie ich w jakikolwiek sposób nie ma sensu.

Inni użytkownicy Kaggle uzyskiwali następujące wyniki dokładności:

Najwięcej 81%, a średnio: 69%

Natomiast mi po wszystkich próbach udało się osiągnąć około 70% dokładności

W trakcie badania, które osiągnęło najwyższy wynik został zastosowany **Random decision forests**, który pracuje na kilku drzewach równocześnie oraz dobiera parametry modelu na podstawie najczęstszego wyniku z drzew.

# Opis problemu:

Na postawie 9 zmiennych wejściowych:

ph, twardość, całkowicie rozpuszczone ciała stałe, chloraminy, siarki, przewodnictwo, węgiel organiczny, trihalometany i mętność

stworzyć sieć neuronową, która będzie klasyfikować wodę do dwóch grup: 0 – niezdatna do spożycia i 1 - zdatna do spożycia. Celem projektu jest przeanalizowanie wpływu różnych czynników:

- liczby warstw

- liczby neuronów

- rodzaju funkcji aktywacji

# Budowanie modelu:

**Początkowe parametry:**

liczba warstw: 3

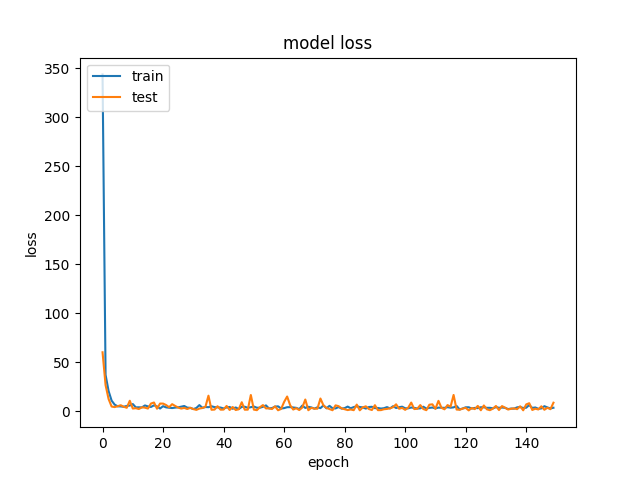
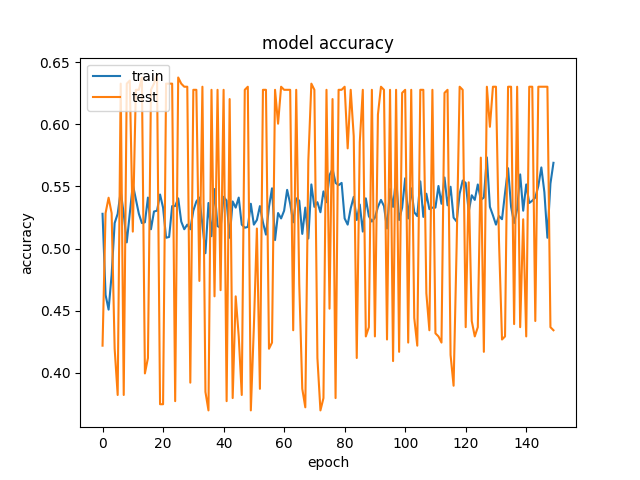
warstwy ukryte: 1

liczby neuronów: 9, 12, 1 (22)

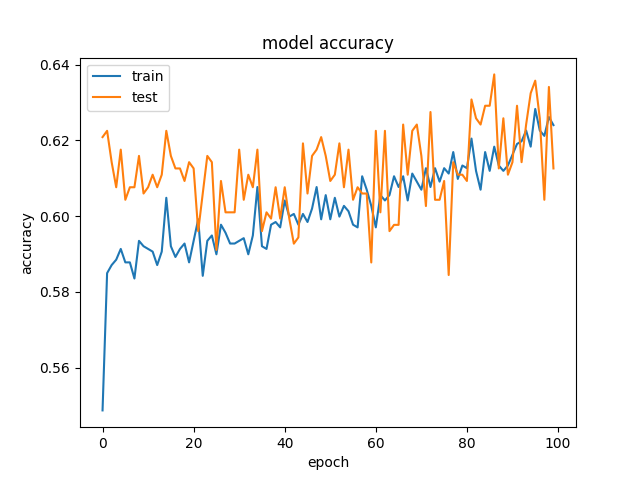
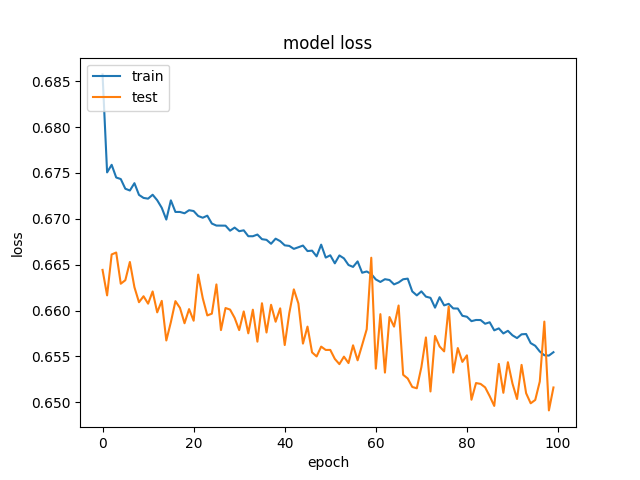
metody uczenia: **Uczenie nadzorowane** (Supervised Learning)

rodzaj funkcji aktywacji: relu, sigmoid

rodzaj sieci: jednokierunkowa

wielkość i sposób doboru próby uczącej i testowej: testowa 30%, ucząca – 70%, losowanie

Po stworzeniu modelu dla pierwszych przykładowych parametrów model prezentuje się w następujący sposób. Na pierwszy rzut oka widać bardzo duży chaos, ponieważ dane nie zostały odpowiednio przekształcone dla różnych jednostek. Po zestandaryzowaniu model prezentuje się tak:

Teraz widać już tendencję do wzrostu dopasowania oraz spadku błędów, jednak na prawym rysunku widać, że dane testowe są znacznie oddalone od uczących. Może to wskazywać, że model łatwiej dopasowuje się do danych testowych niż uczących, bo gdy pracuje na testowych, jest już wytrenowany. Z wykresów można również odczytać, że model nie jest jeszcze dobrze dopasowany - można go dłużej uczyć, na lewym rysunku dopasowanie cały czas wzrasta – nie stabilizuje się. Na prawym błędy cały czas maleją – nie stabilizują się. Może to być wskazówką do wykonania większej liczby prób uczenia – epoch. Do tej pory zastosowana była jedna warstwa przez co widać liniowy trend uczenia się. Być może

# Analiza liczby warstw

Wszystkie ukryte warstwy zostały stworzone takie same: ReLU, 12 węzłów

## 2 warstwy:

Accuracy: 69%

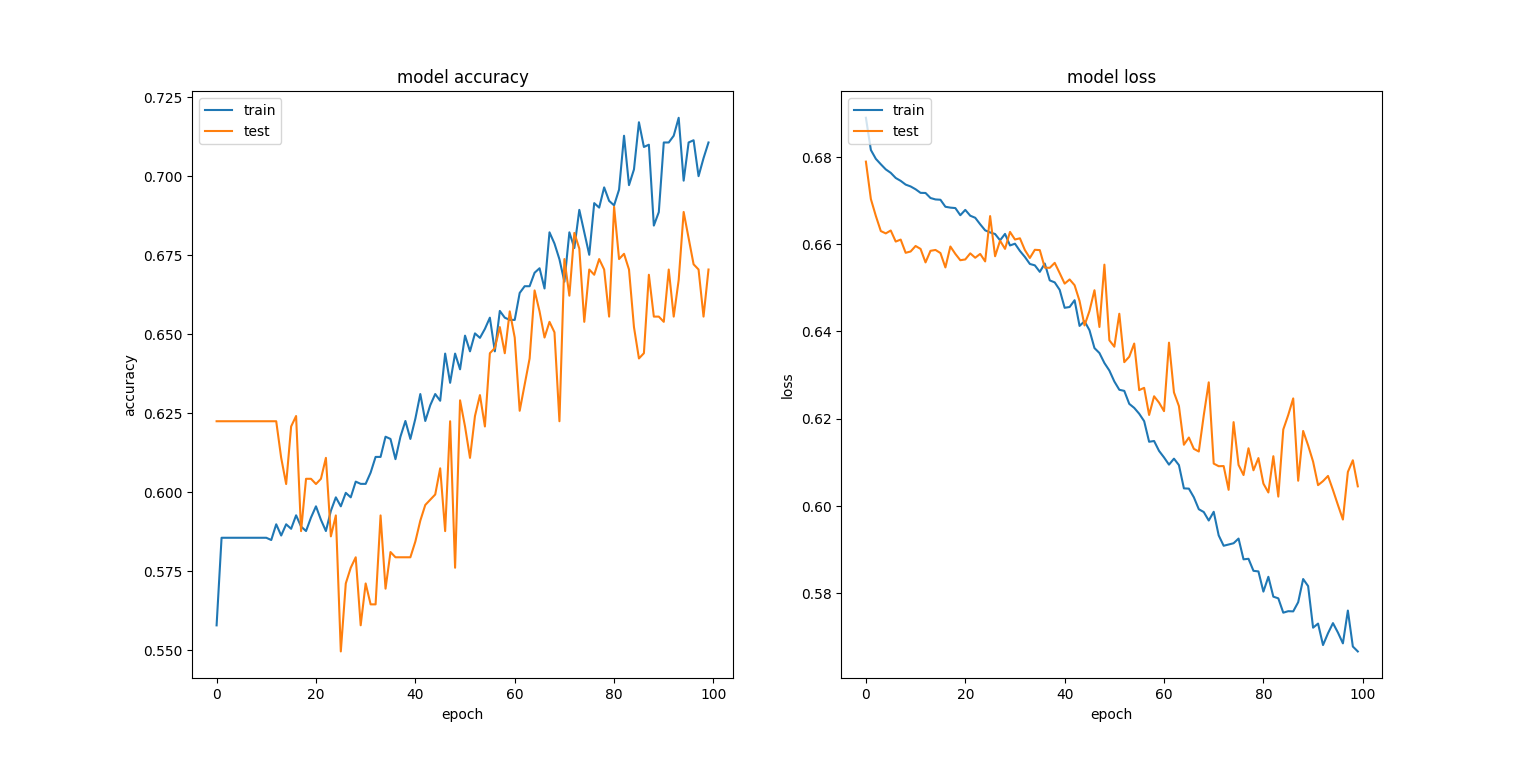
## 

Jak widać wraz ze zwiększeniem liczby warstw dopasowanie wzrasta, a wykresy stają się bardziej okrągłe, mniej liniowe. Model szybciej dopasowuje się.

Accuracy: 70%

## 3 warstwy:

## 4 warstwy:



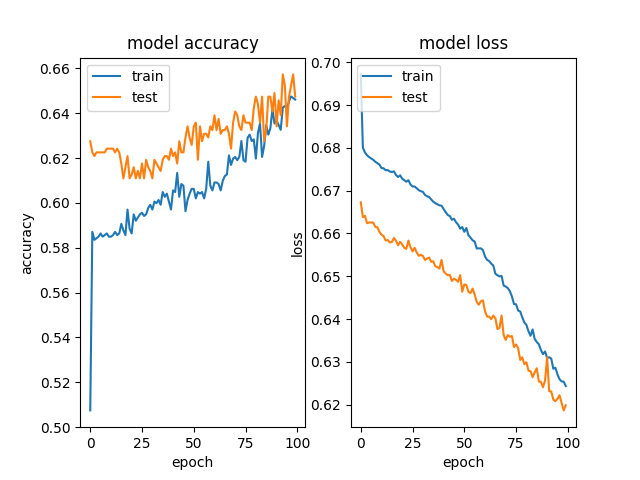
Jednak dla 3 warstw następuje lekki overfitting. Dla 4 jest on już bardzo widoczny. Dla danych testowych jest gorsze dopasowanie niż dla uczących oraz błąd predykcji jest większy. Accuracy: 70%

Optymalna liczba warstw ukrytych to: 2

# Analiza – liczba neuronów

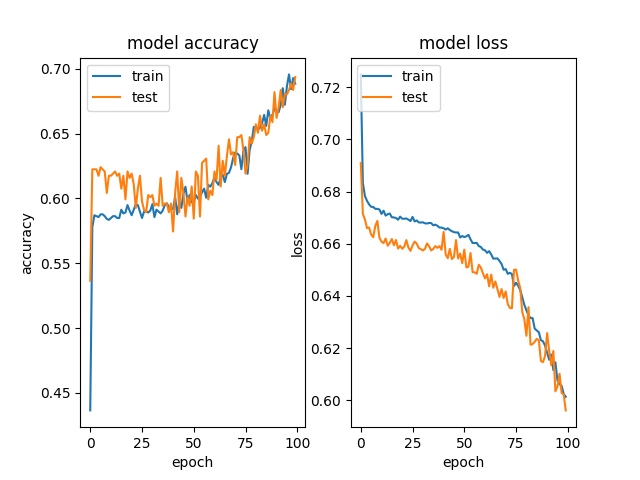
2 warstwy ukryte; relu, relu, wyjściowa: sigmoid

1. Liczba neuronów: 9, 10, 10, 1

Accuracy: 62, 64, 65 -> 63%

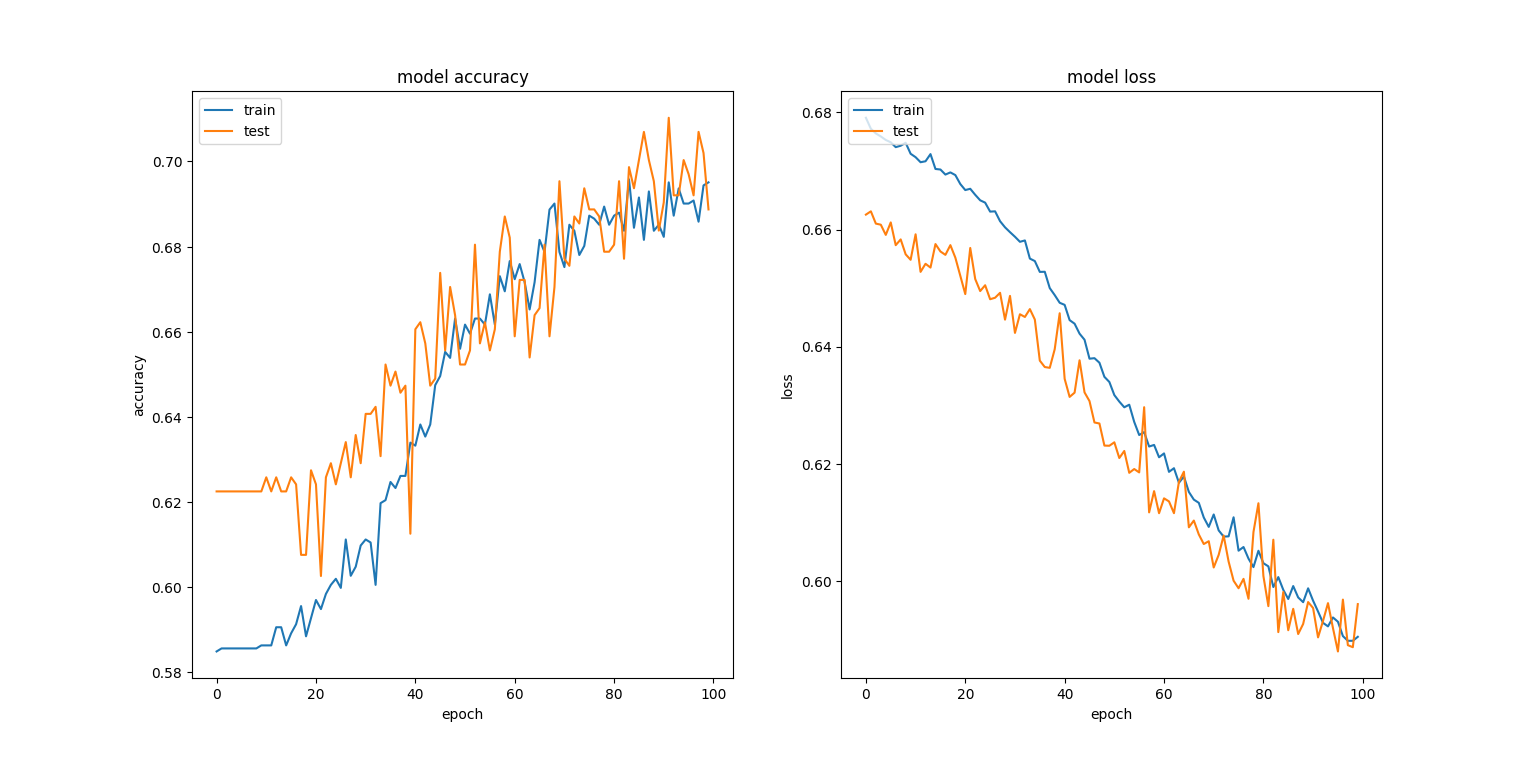
1. Liczba neuronów: 9, 15, 15, 1

Accuracy: 69, 70, 70 -> 70%

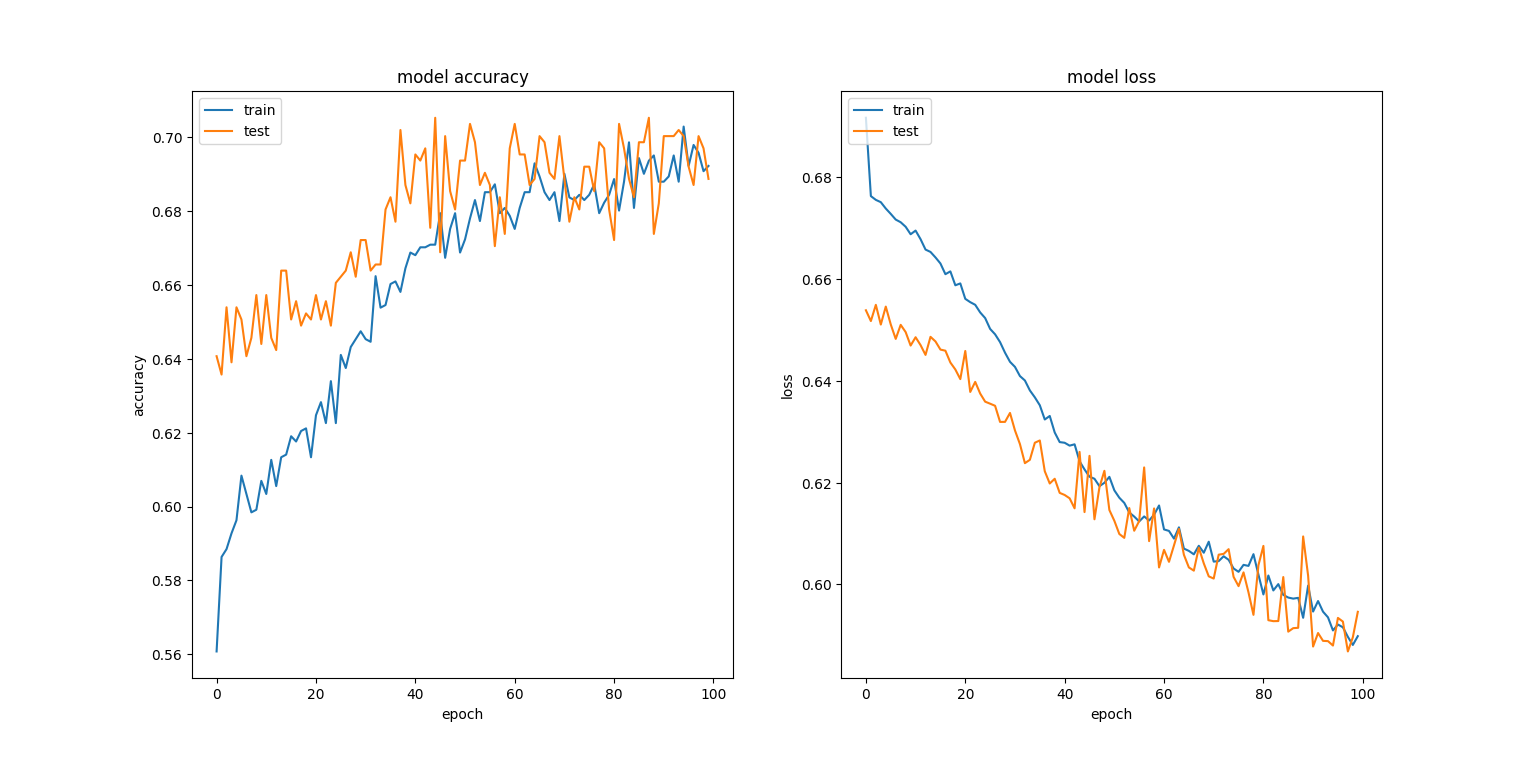


1. Liczba neuronów: 9, 12, 15, 1

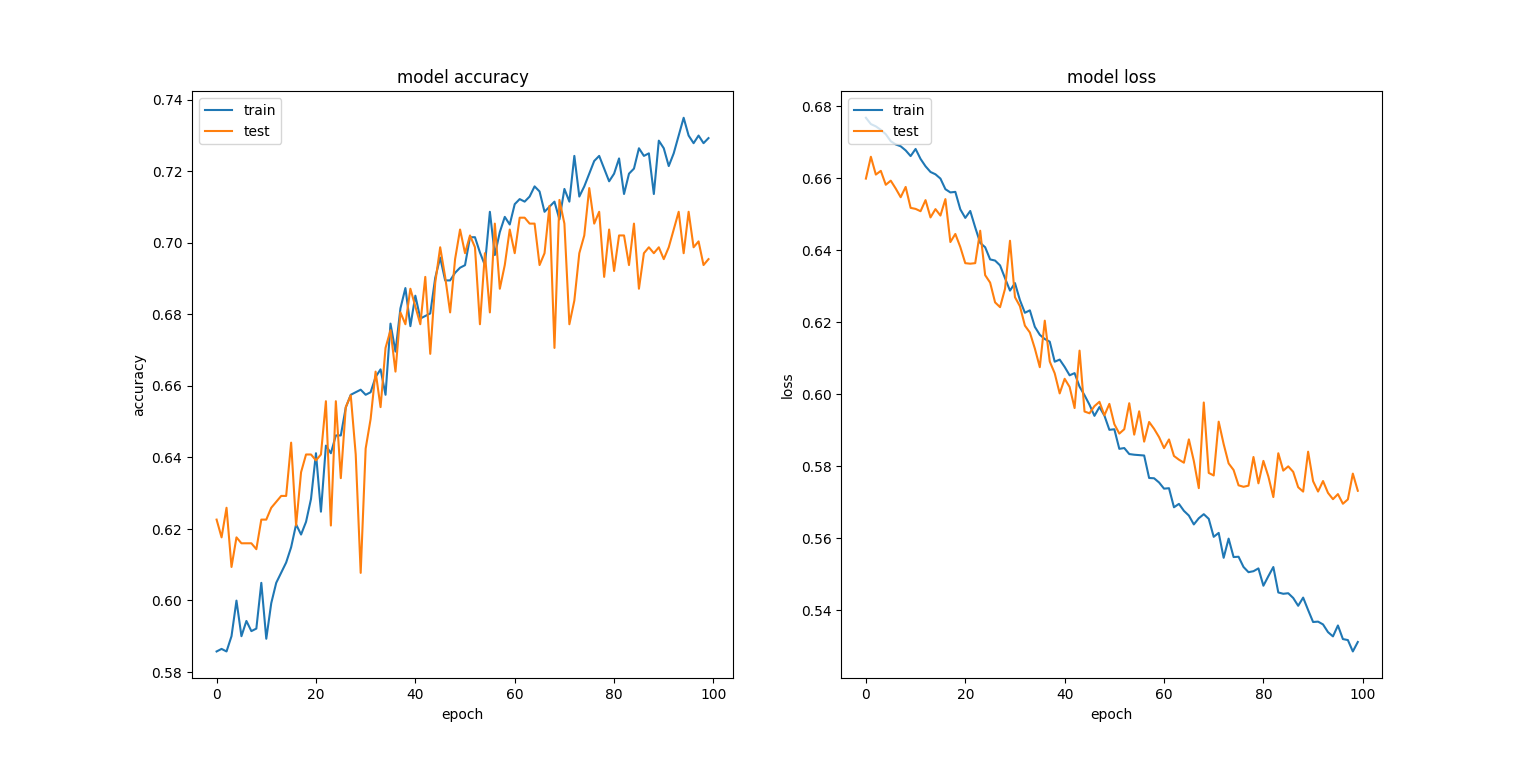
Accuracy: 70, 68, 69 -> 69%



1. Liczba neuronów: 9, 15, 12, 1

Accuracy: 69, 68, 69 -> 69%

1. Liczba neuronów: 9, 30, 30, 1

Accuracy: 72, 73, 72 -> 72%

Nieznaczna zmiana liczby węzłów nie wpływa znacząco na dokładność modelu. Jeżeli zwiększymy liczbę neuronów do 30, to widać zwiększenie dokładności do 72% oraz widać wystarczające wyuczenie dla danych testowych, w pewnym momencie wykres stabilizuje się.

Analiza – funkcje aktywacji

Funkcje aktywacji w sieciach neuronowych definiują w jaki sposób dane wejściowe są przekształcane w dane wyjściowe w danym węźle.

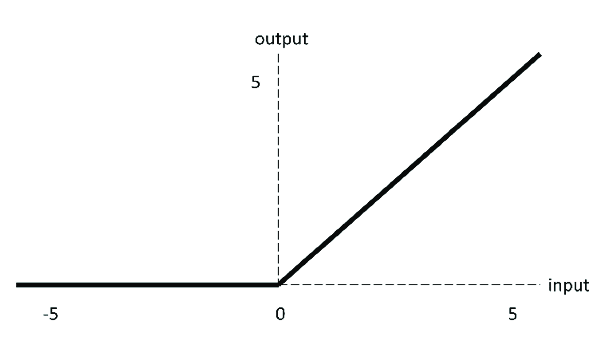
epochs = 100, batch=25, 2 warstwa ukryta, liczba węzłów: 9, 12, 12, 1

## Warstwy ukryte:

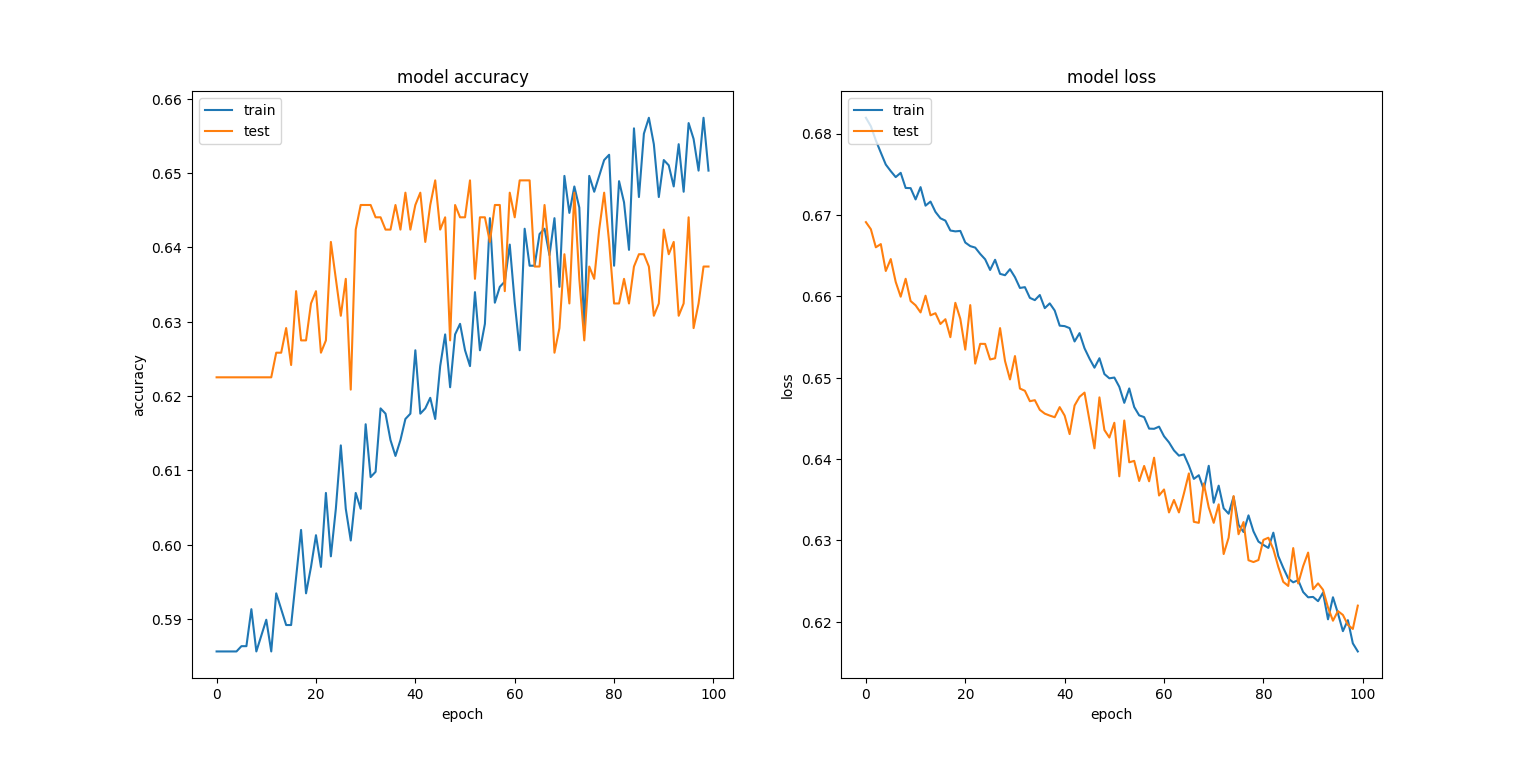
* ReLU
* Sigmoid (logistic)
* Tanh

1. ReLU

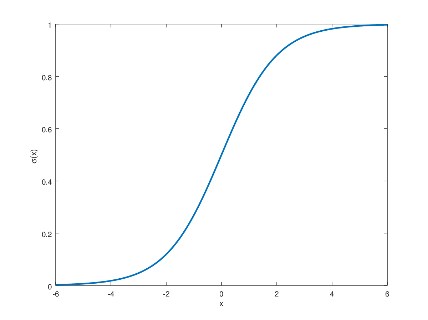
Ujemne wartości zamienia na 0, a pozostałe po prostu zwraca.



Accuracy około 70%

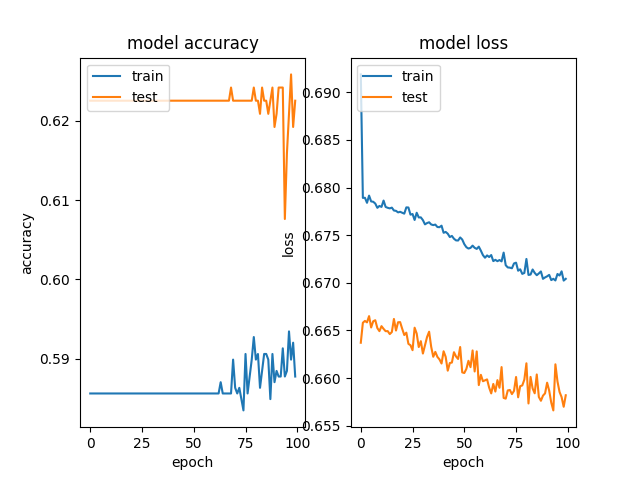


1. Sigmoid (logistic)

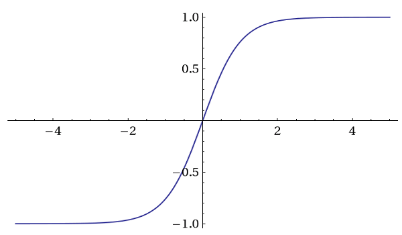


Ta funkcja przekształca dowolną rzeczywistą liczbę w liczbę z przedziału [0, 1]

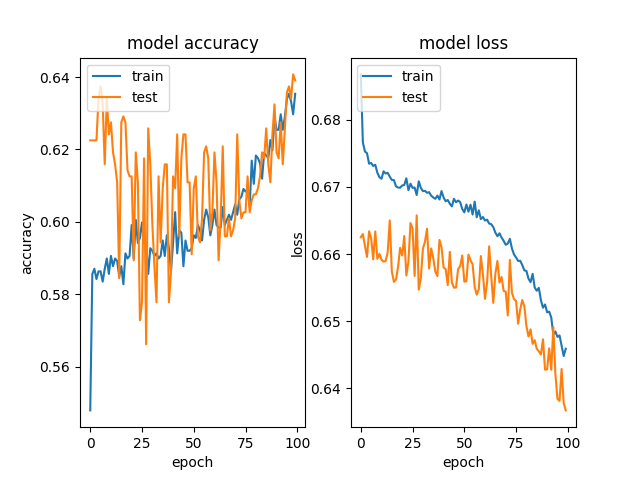
Accuracy około: 60%



1. Tanh



Ta funkcja przekształca dowolną rzeczywistą liczbę w liczbę z przedziału [-1, 1]



Accuracy około 64%

Najlepsze dopasowanie około 70% wyszło dla funkcji ReLU. Środkowy wynik wyszedł dla funkcji tanh – 64% i najniższy dla sigmoid – 60%. Model na podstawie funkcji sigmoid praktycznie nie uczy się, nie ma dopasowanie jest cały czas na stałym poziomie.

## Funkcje aktywacji - output

* Liniowa
* Logistic
* Softmax

1. Liniowa

Nie zmienia wartości

1. Softmax

Zwraca wektor prawdopodobieństw od [0,1].

W tym przypadku, gdy mam model, który szacuje wartość binarną (1 – woda jest zdatna do picia, 0 – woda nie jest zdatna do picia), jedyną rozsądną funkcją aktywacji na wyjściu jest użycie funkcji logistic, tak aby zapewnić wynik zerojedynkowy.