Warszawa
10.05.2022r.
Galicki Cyprian 300008
Mikołaj Prażmo 302679

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych

Sieci neuronowe w zastosowaniach biomedycznych (SNB)
Dr. Inż. Bogumił Konarzewski

Etap #3: Optymalizacja + testy końcowe
Nr. Zespołu 23
Nr. Tematu 23
Klasyfikacja stanu płodu na podstawie badania KTG za pomocą sieci MLP (katalog: Cardiotocography_MLR)

Oświadczam, że niniejsza praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu Sieci neuronowe w zastosowaniach biomedycznych, została wykonana przeze mnie samodzielnie.
Galicki Cyprian 300008
Mikołaj Prażmo 302679

Spis treści

Treść zadania

Sprawozdanie z etapu 3 powinno zawierać opis działań zmierzających do optymalizacji struktury sieci oraz algorytmu jej uczenia. Jako minimum wymagane jest przeanalizowanie nauki dla wszystkich kombinacji następujących cech sieci:

o 3 różnych wartości współczynnika szybkości uczenia (ang. *learning rate*)

o 3 różnych struktur sieci (o innej liczbie neuronów w warstwach)

Dla każdej kombinacji należy przedstawić wykresy obrazujące proces uczenia sieci (przebieg błędu w zależności od liczby iteracji), wartości czułości i specyficzności.

Wpływ różnych ilości neuronów warstwy ukrytej

MLP = Multi Layer Perceptron / Perceptron wielowarstwowy

Liczba neuronów na wejściu, jeden neuron na zmienną, w naszym przypadku M = 14.

Na wyjściu jeden neuron, płód: normalny/chory, więc N = 1.

Ilość w warstwie ukrytej wyznaczamy na podstawie poniższego wzoru:

Jako dobre przybliżenie można przyjąć liczbę neuronów w warstwie ukrytej jako średnią geometryczną wymiarów wejść i wyjść sieci:

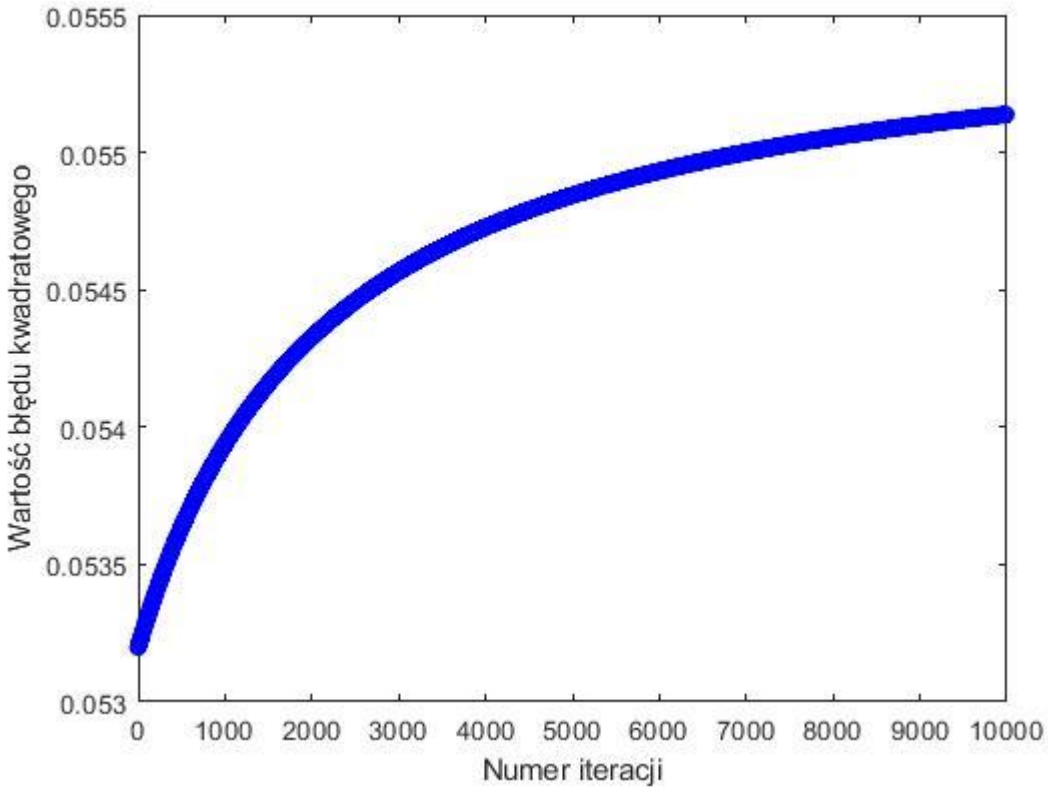
$$K \approx \sqrt{NM}$$

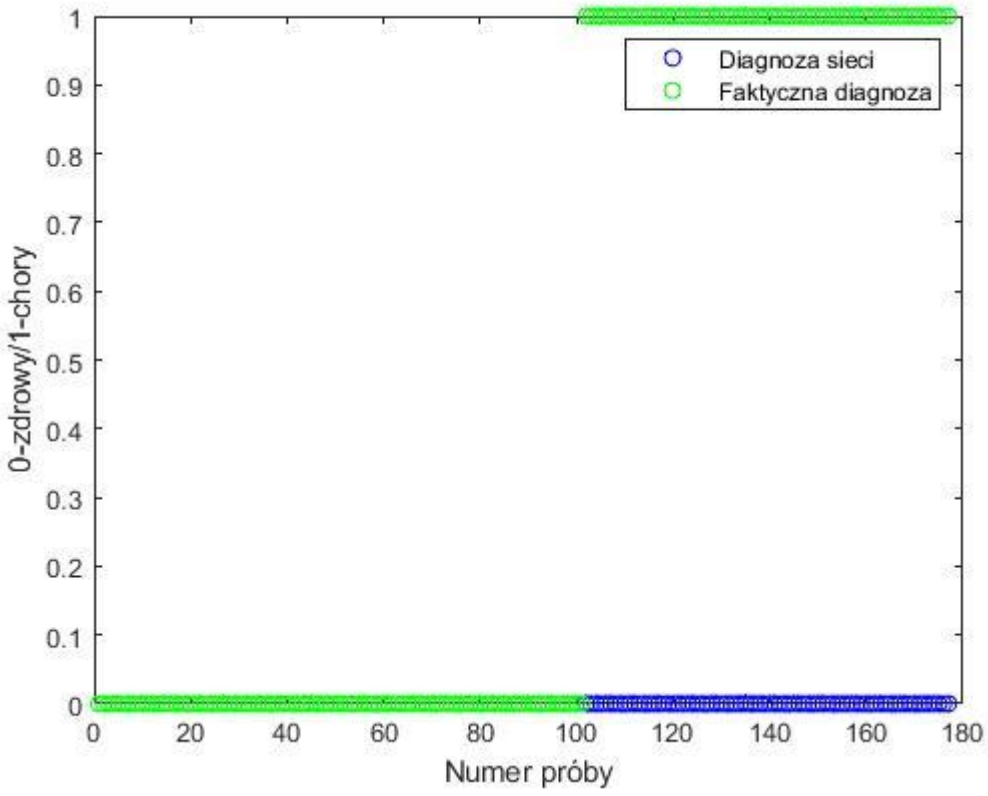
W naszym przypadku $K = \sqrt{\mathrm{MN}} = \sqrt{1 * 14} \approx 4$

Współczynnik prędkości uczenia	Ilość neuronów w warstwie ukrytej	Czułość	Specyficzność
n = 0.001	2	0	1
n = 0.001	3	1	0
n = 0.001	4	0,92	0,73
n = 0.001	5	0,97	0,69
n = 0.001	6	0	0,99
n = 0.001	7	0	1
n = 0.001	8	0,78	0,95

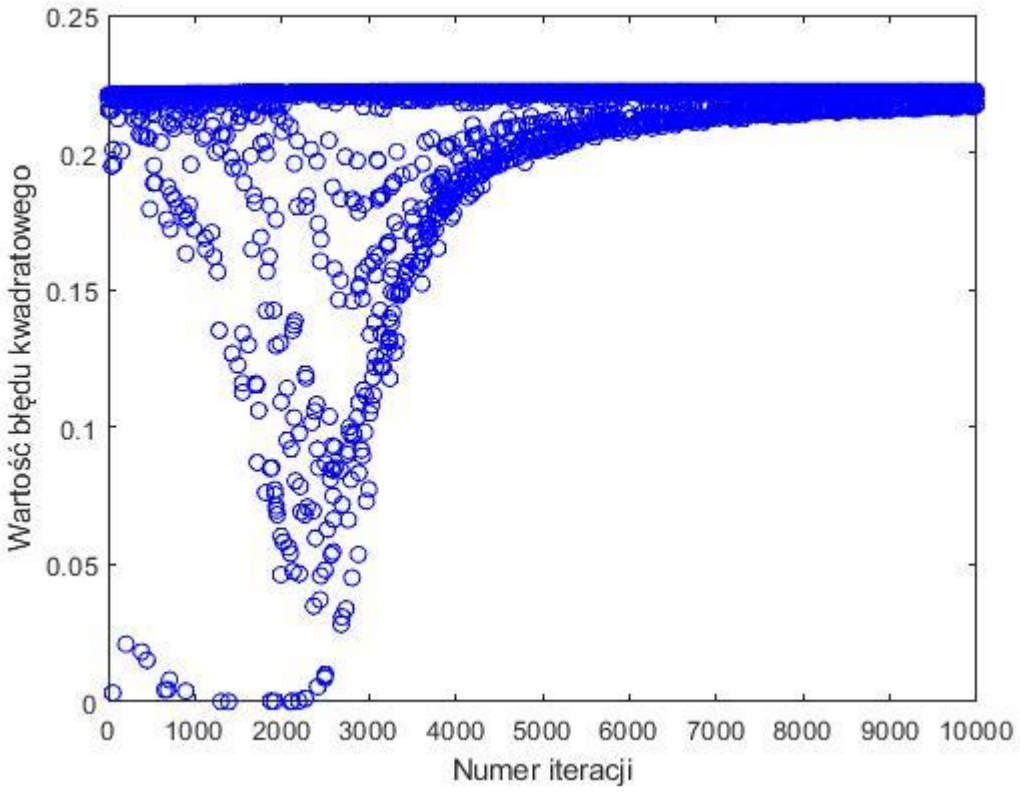
Najlepsze wyniki sieć osiągnęła dla 5 neuronów w warstwie ukrytej. Dla 4 i 8 neuronów wyniki nadal były bliskie prawdy, więc sieć jak najbardziej była użyteczna. Dla 2, 3, 6 i 7 neuronów, sieć nie potrafiła dokonać poprawnej klasyfikacji raz grupy chorej raz zdrowej. W takiej sytuacji staje się ona bezużyteczna.

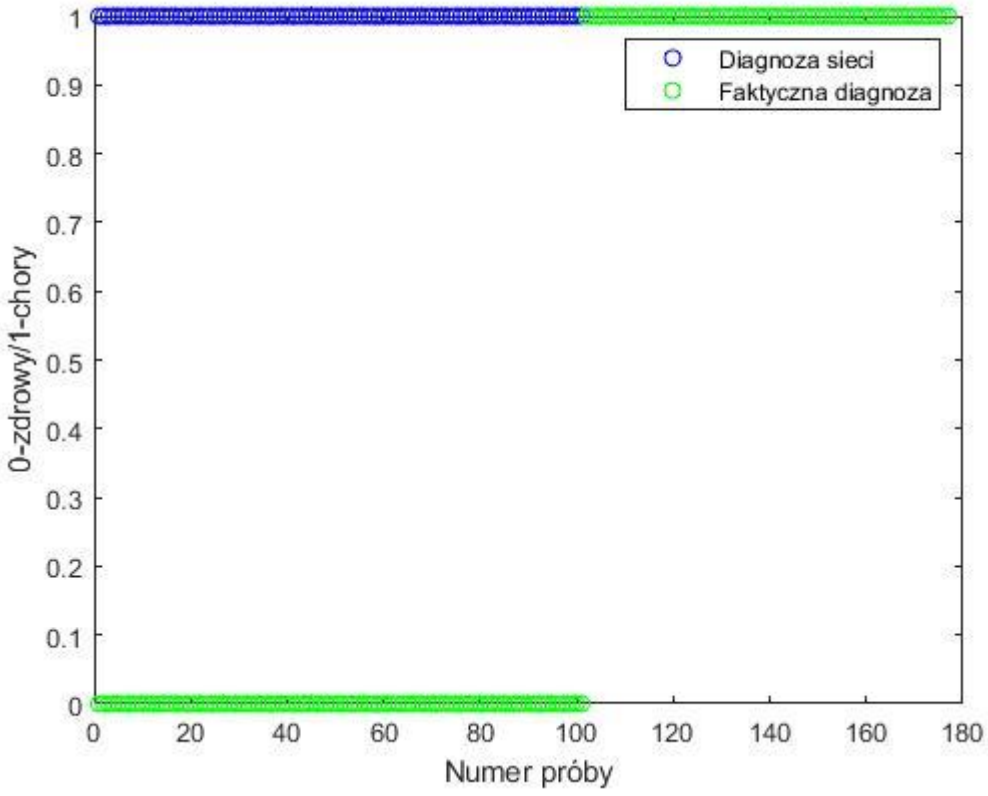
Ilość neuronów warstwie ukrytej: 2



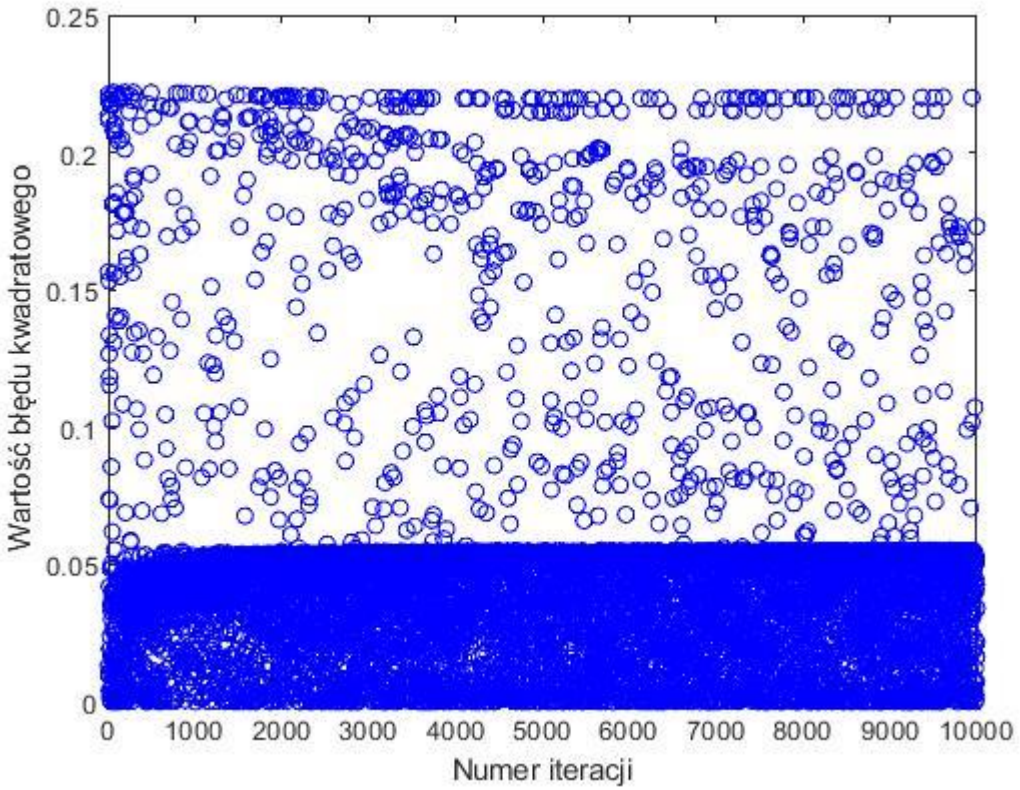


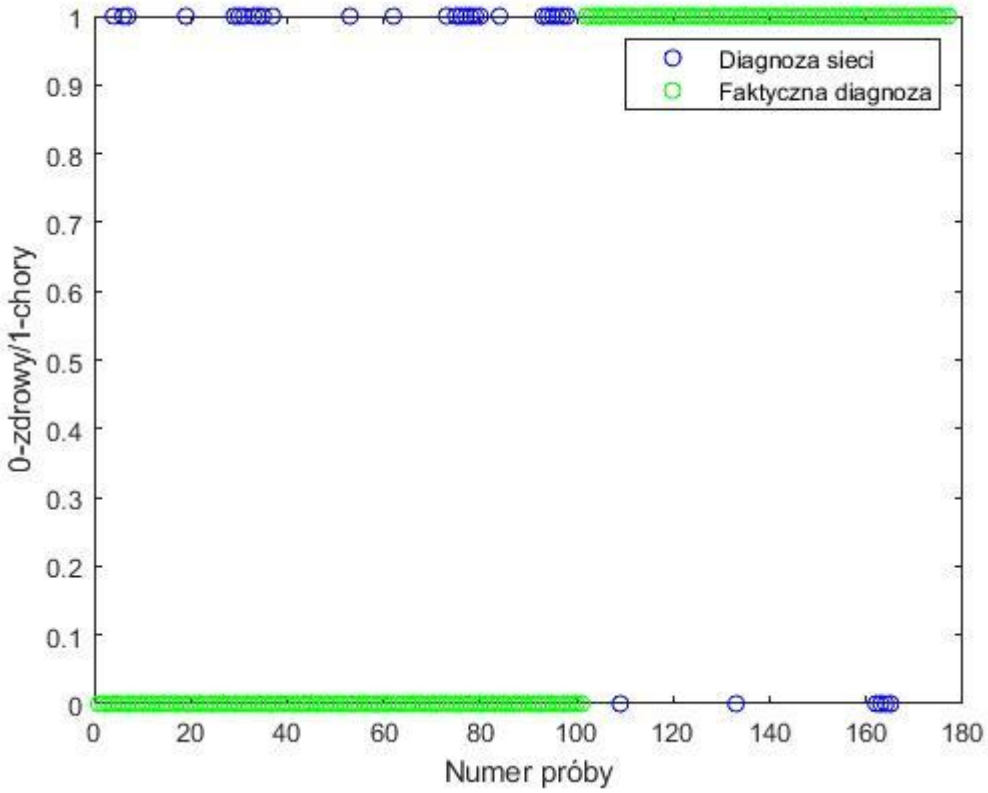
Ilość neuronów warstwie ukrytej: 3



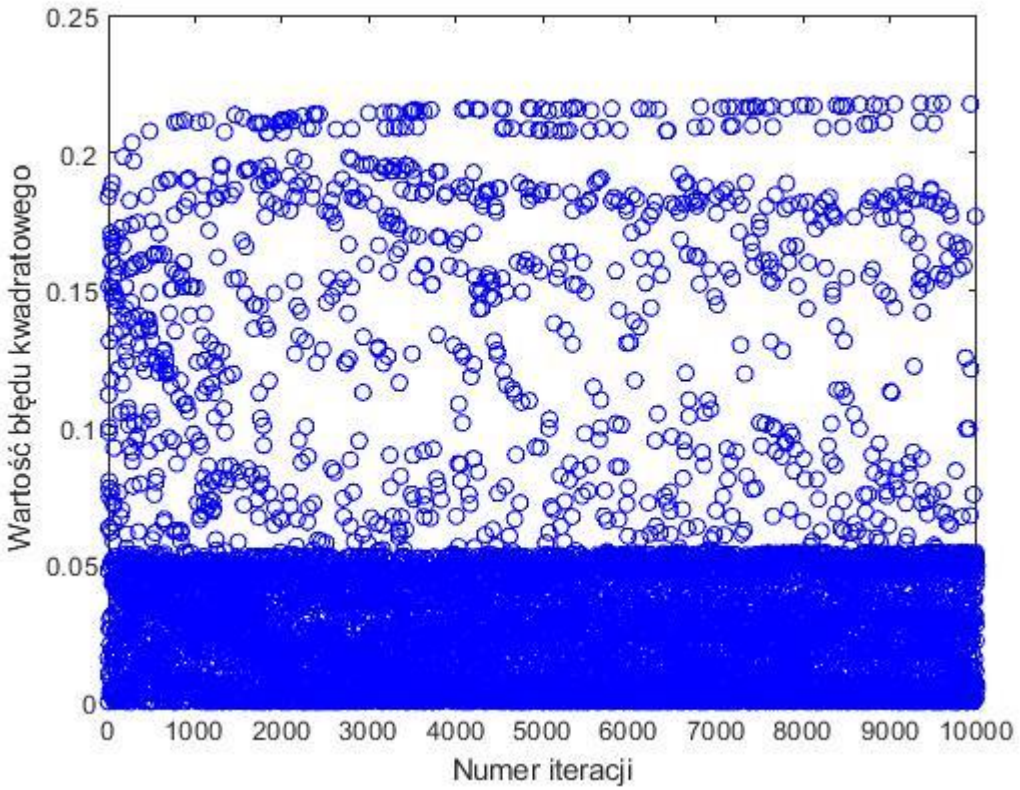


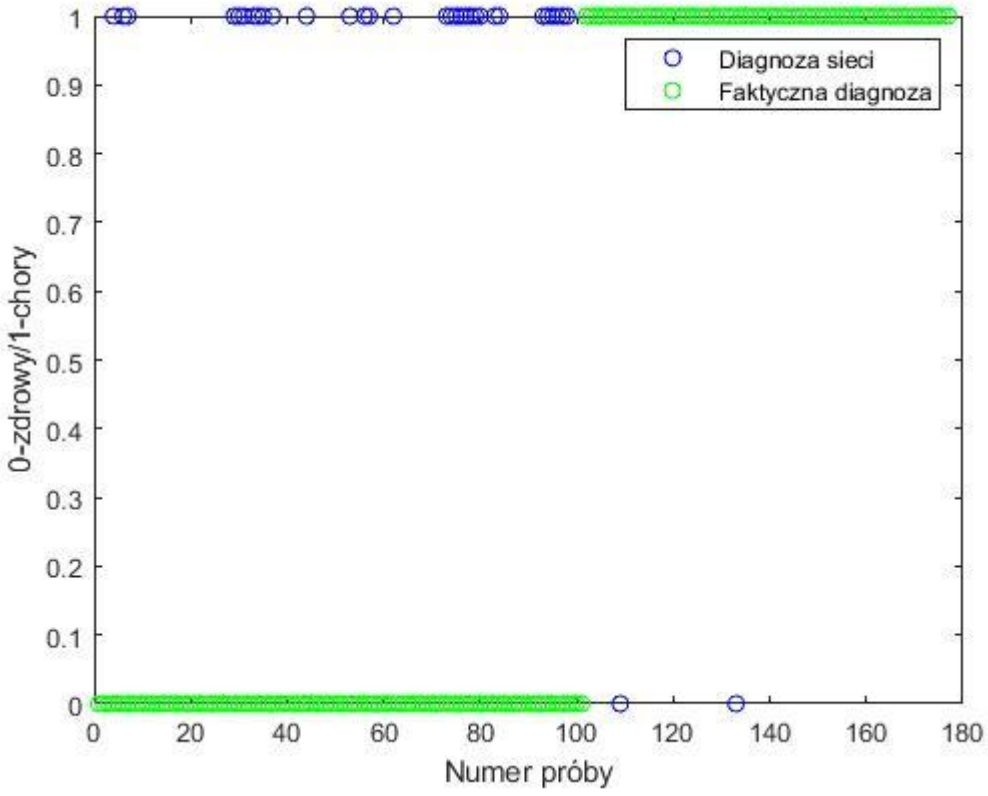
Ilość neuronów warstwie ukrytej: 4



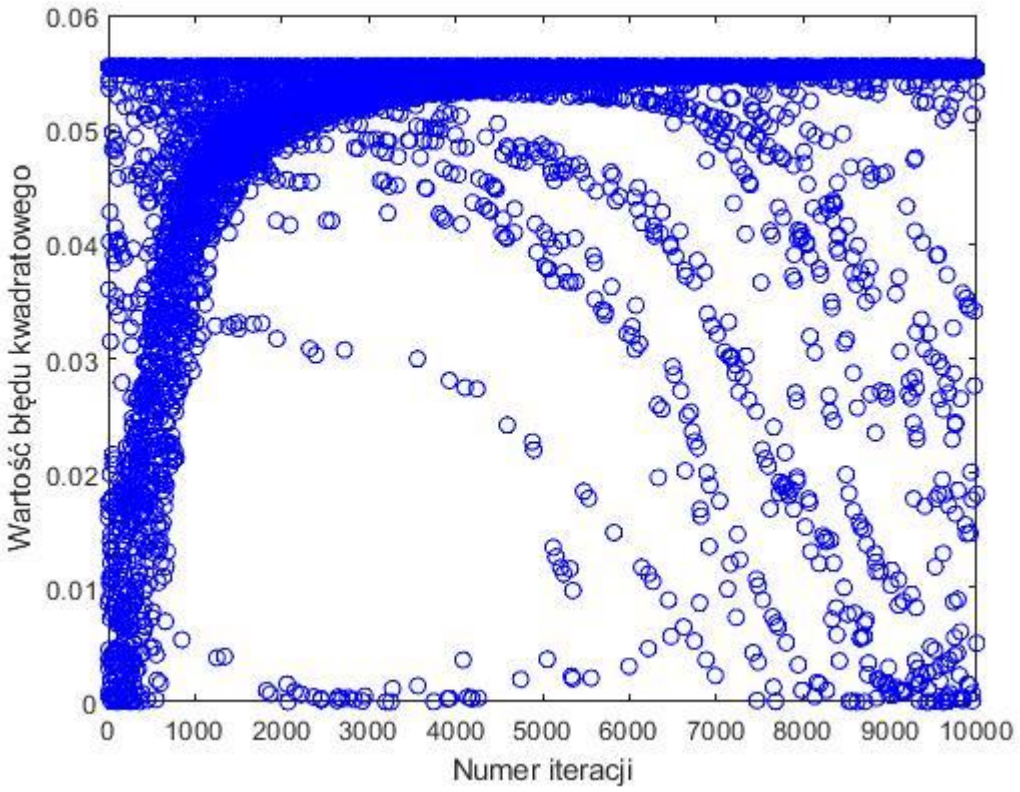


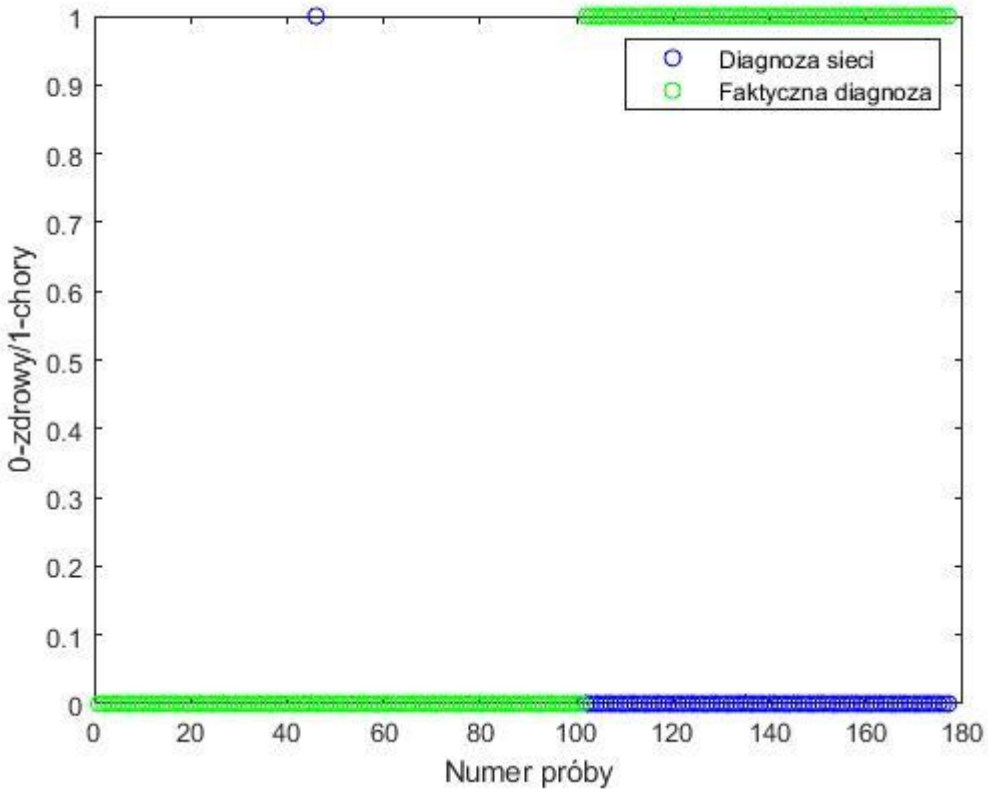
Ilość neuronów warstwie ukrytej: 5



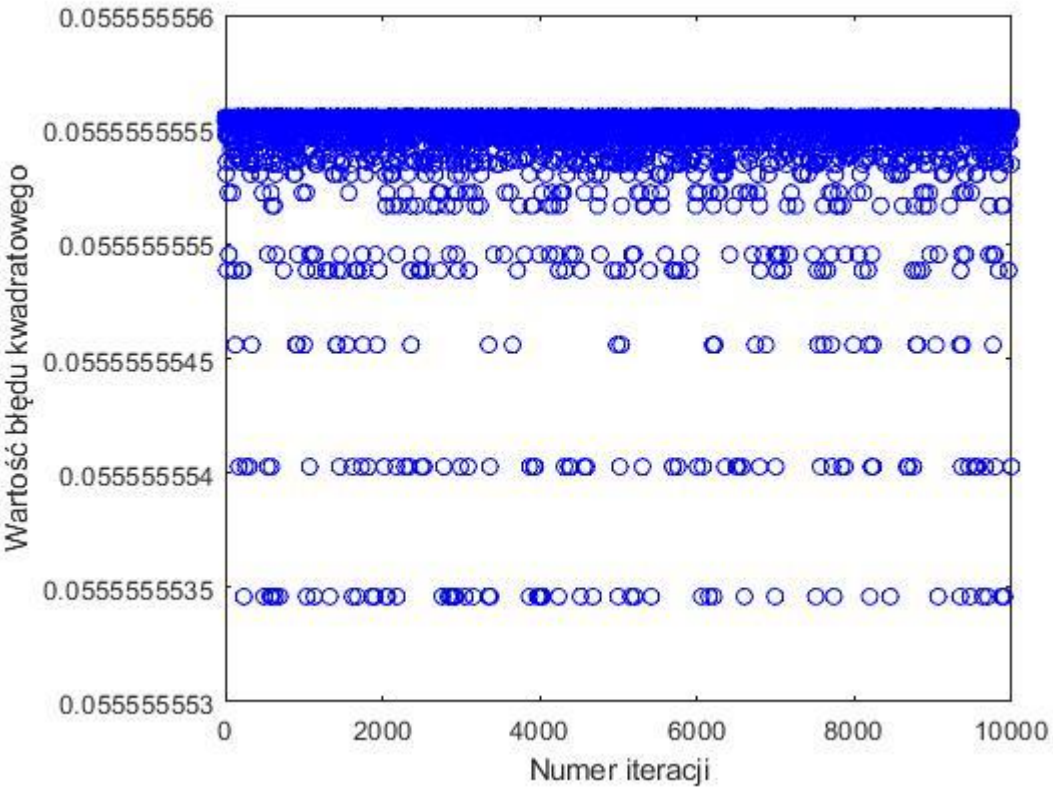


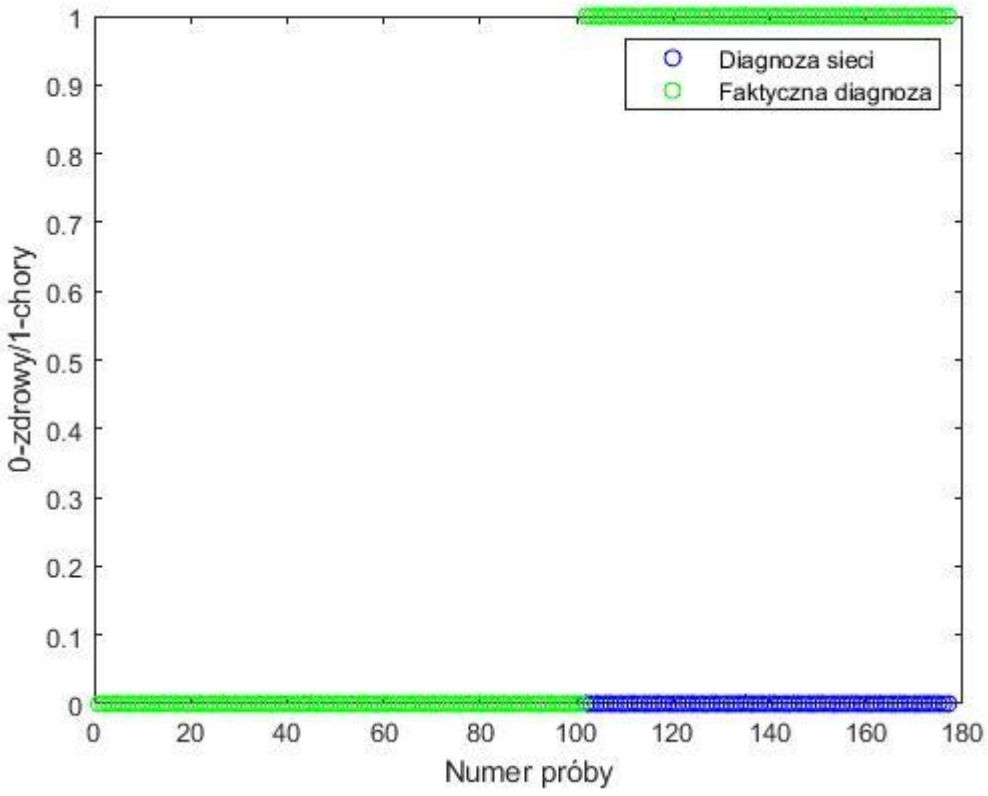
Ilość neuronów warstwie ukrytej: 6



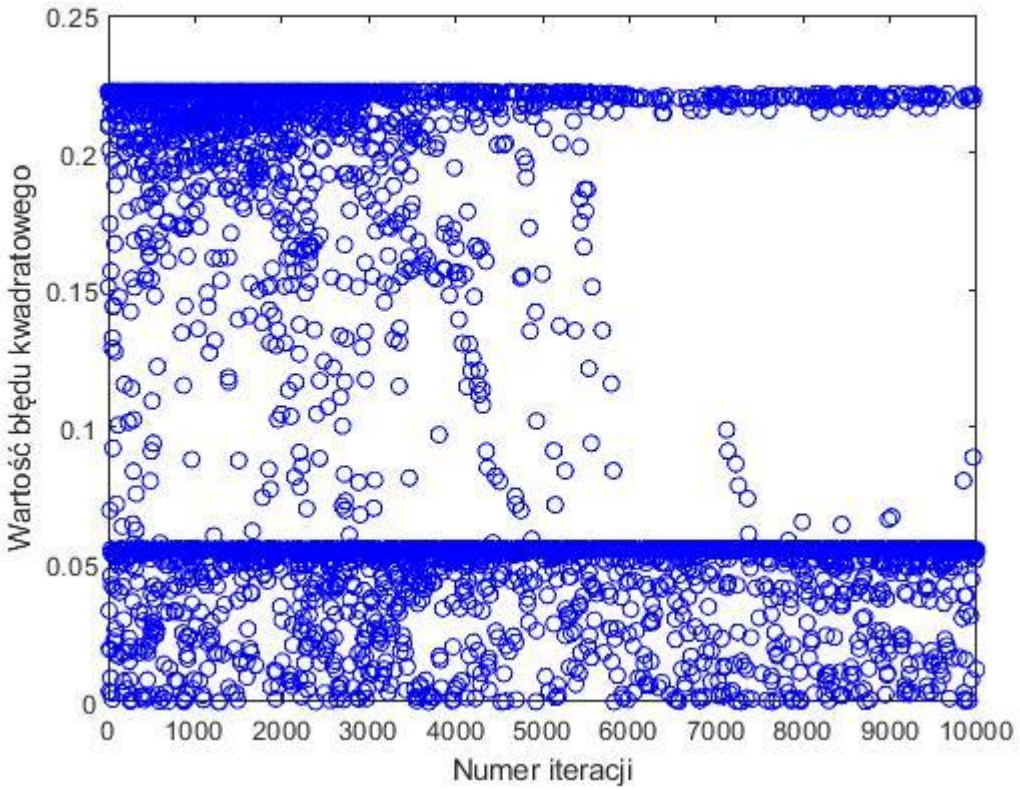


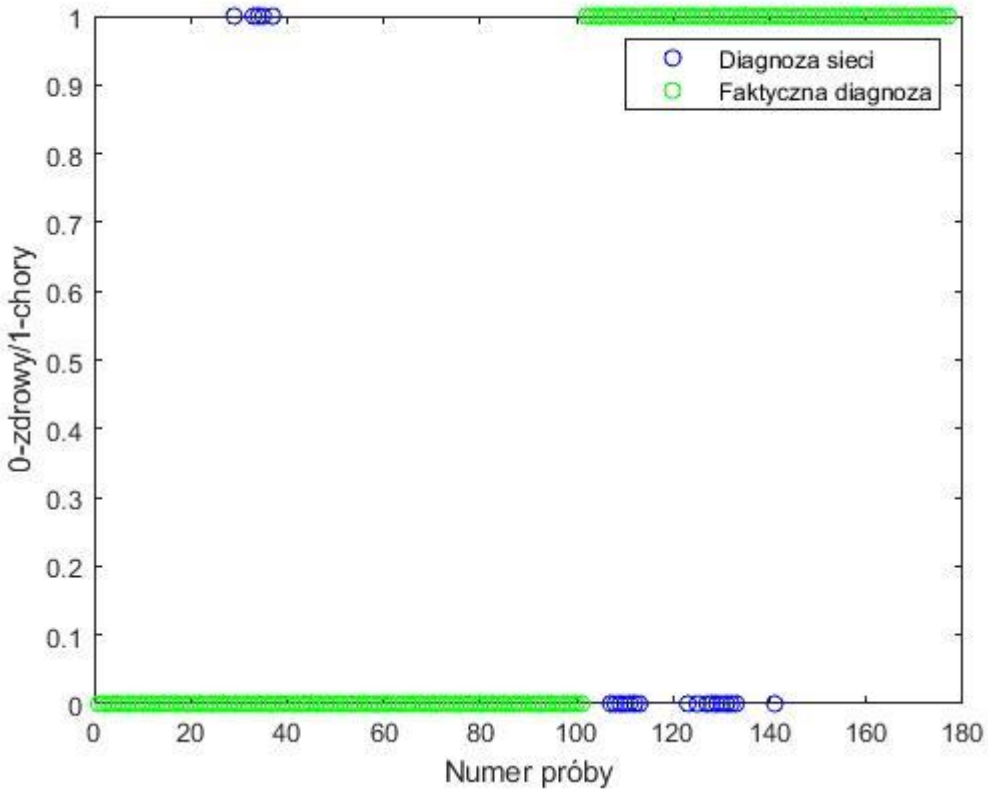
Ilość neuronów warstwie ukrytej: 7





Ilość neuronów warstwie ukrytej: 8





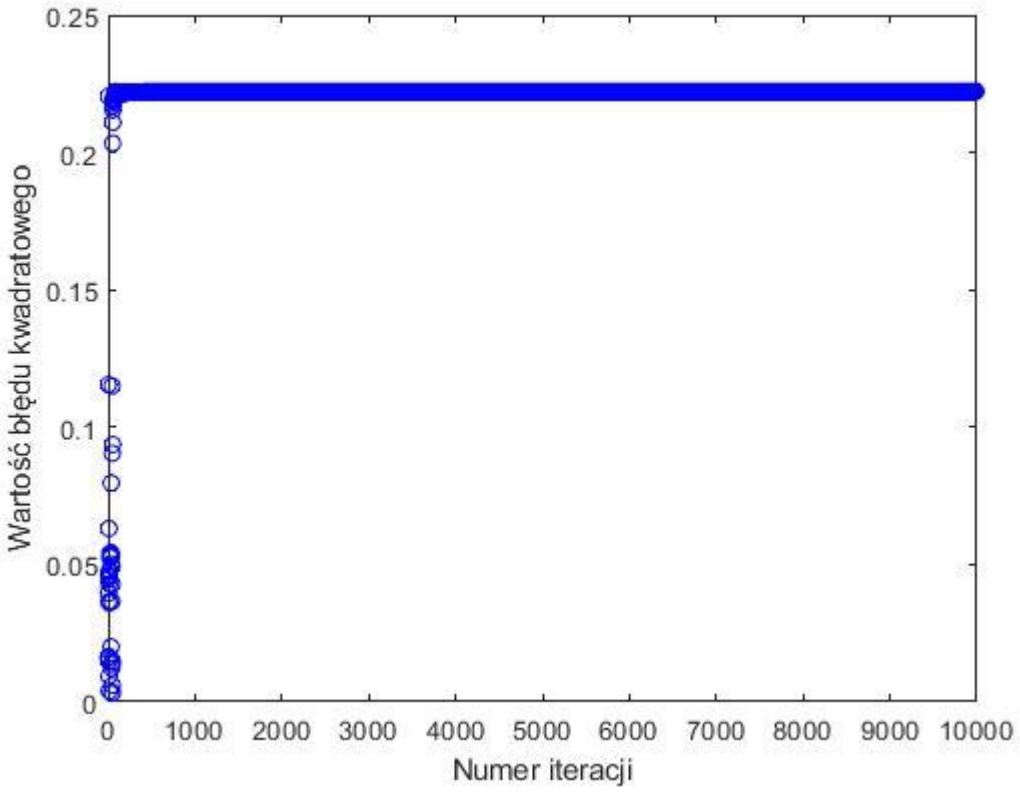
Wpływ różnych wartości współczynnika szybkości uczenia

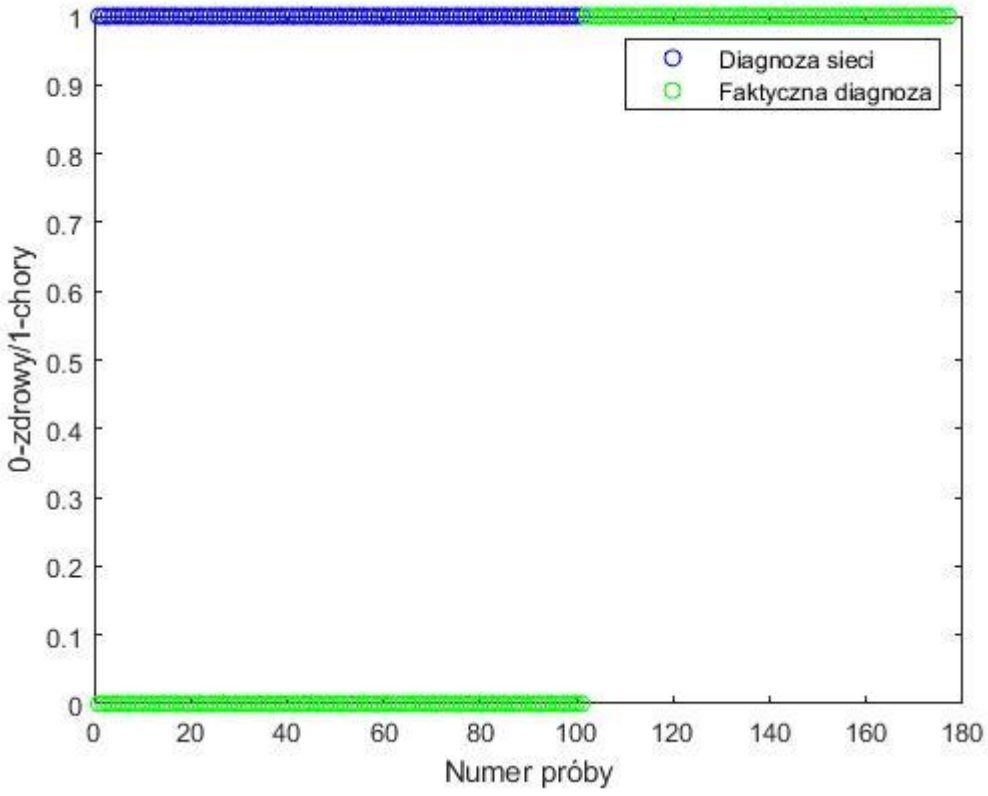
Na podstawie wyników uzyskanych w poprzednim punkcie, postanowiliśmy pozostać przy 5 neuronach w warstwie ukrytej.

Współczynnik prędkości uczenia	Ilość neuronów w warstwie ukrytej	Czułość	Specyficzność
n = 0.1	5	1	0
n = 0.01	5	0,82	0,93
n = 0.001	5	0,97	0,69
n = 0.0001	5	1	0,70
n = 0.00001	5	1	0,51

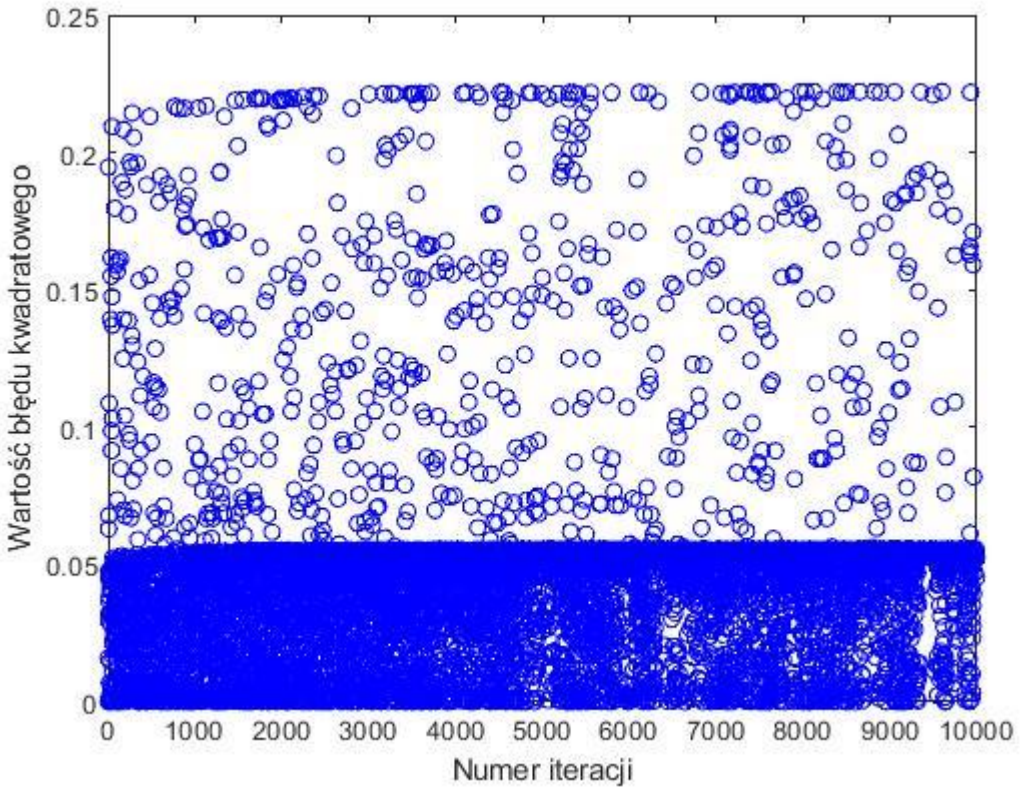
Najlepsze wyniki sieć osiągnęła dla n = 0.0001. Dalsze zmiany n w obu kierunkach, nie poprawiały wyników dlatego sprawdziliśmy skuteczność dla 5 wartości. Mały współczynnik uczenia może przynieść lepsze efekty, jednak jego nadmiernie mała wartość, może przedłużać proces uczenia. Natomiast wyższe jego wartości mogą być pomocne przy szybkim testowaniu większych sieci, gdyż wyniki pomimo że mniej dokładne, nadal nie będą znacząco odbiegały od kierunku naszych poszukiwań.

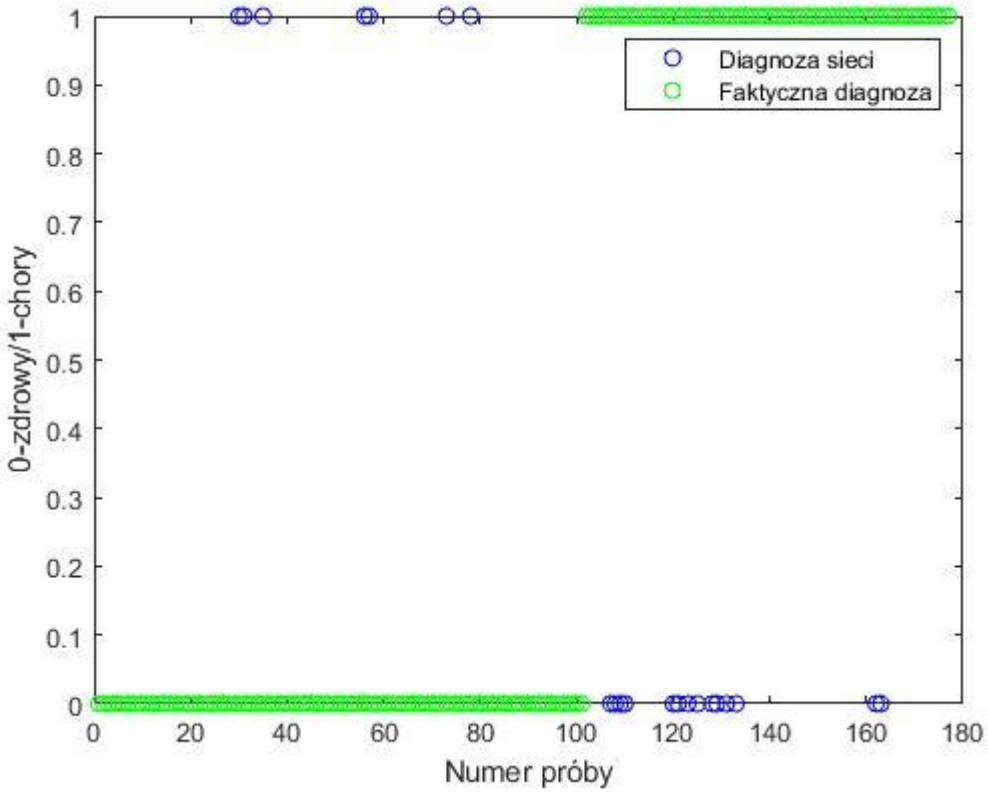
Wartość współczynnika prędkości uczenia: n = 0.1



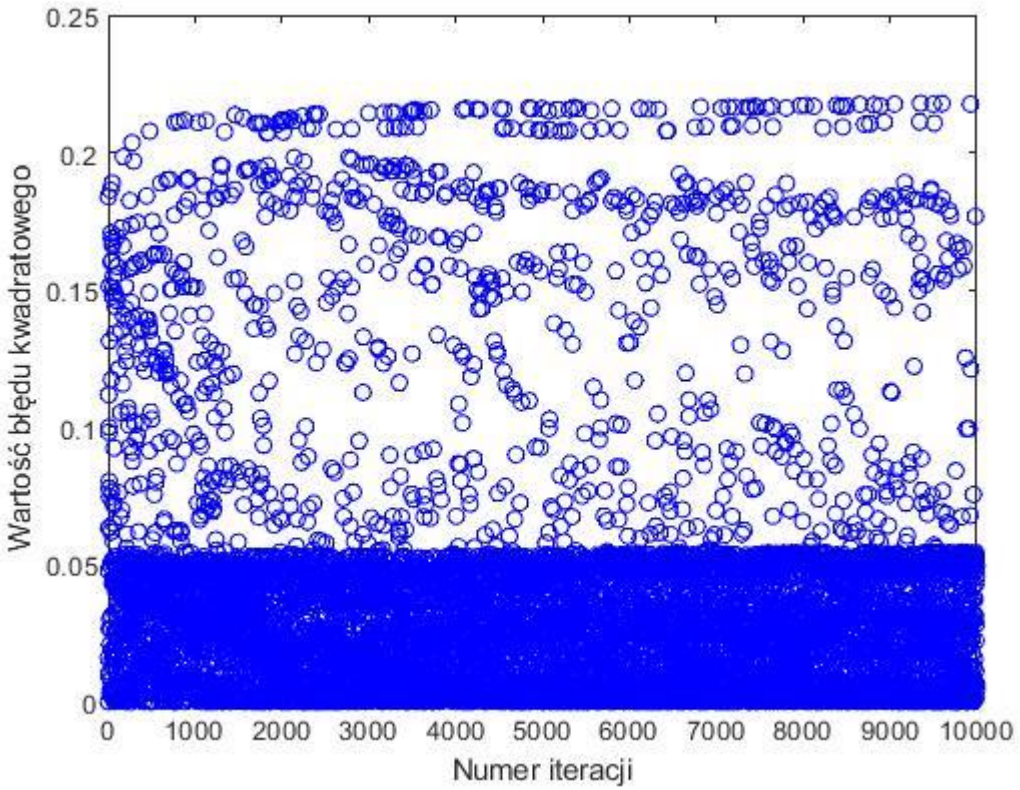


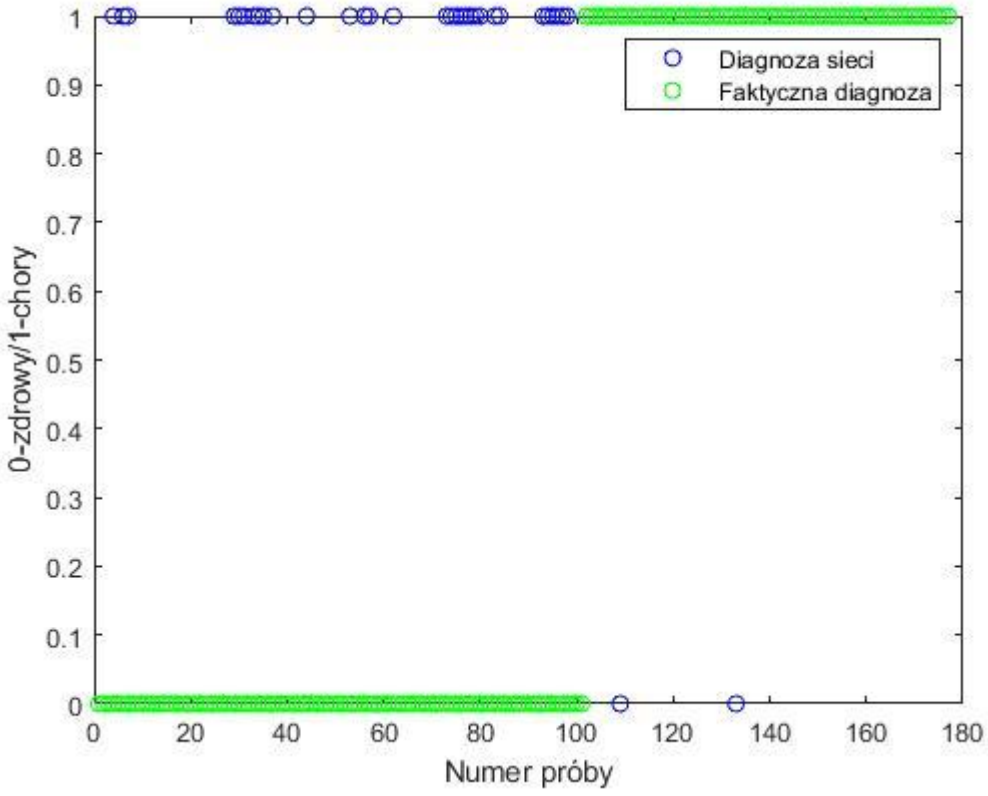
Wartość współczynnika prędkości uczenia: n = 0.01



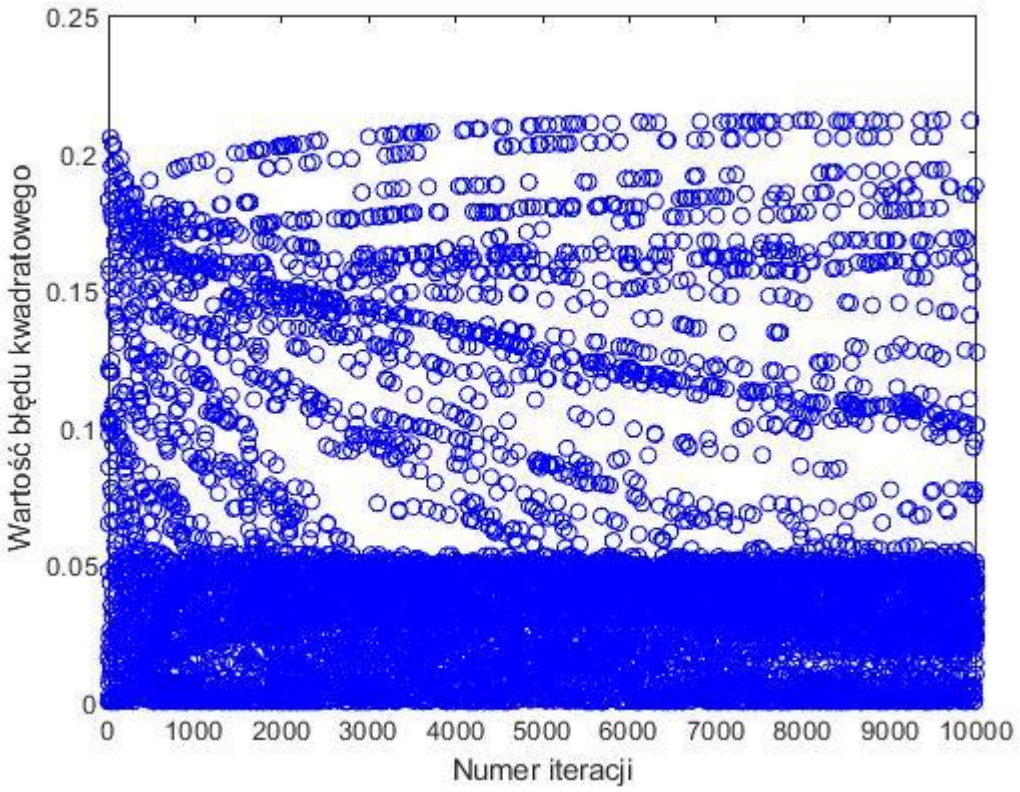


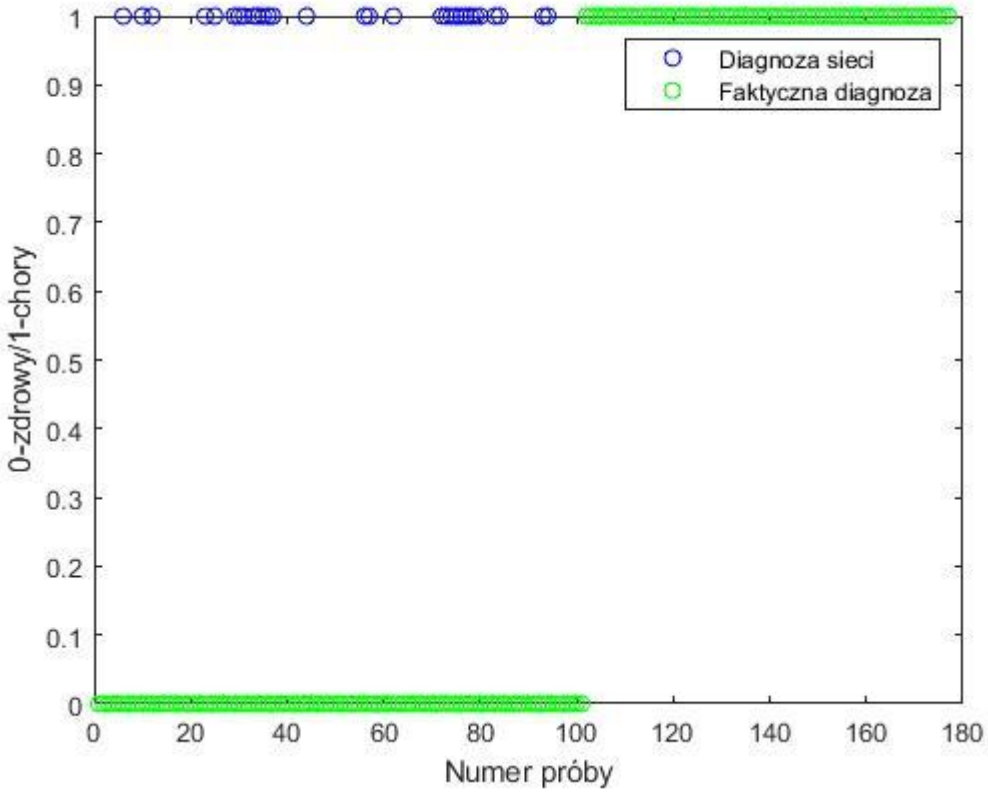
Wartość współczynnika prędkości uczenia: n = 0.001



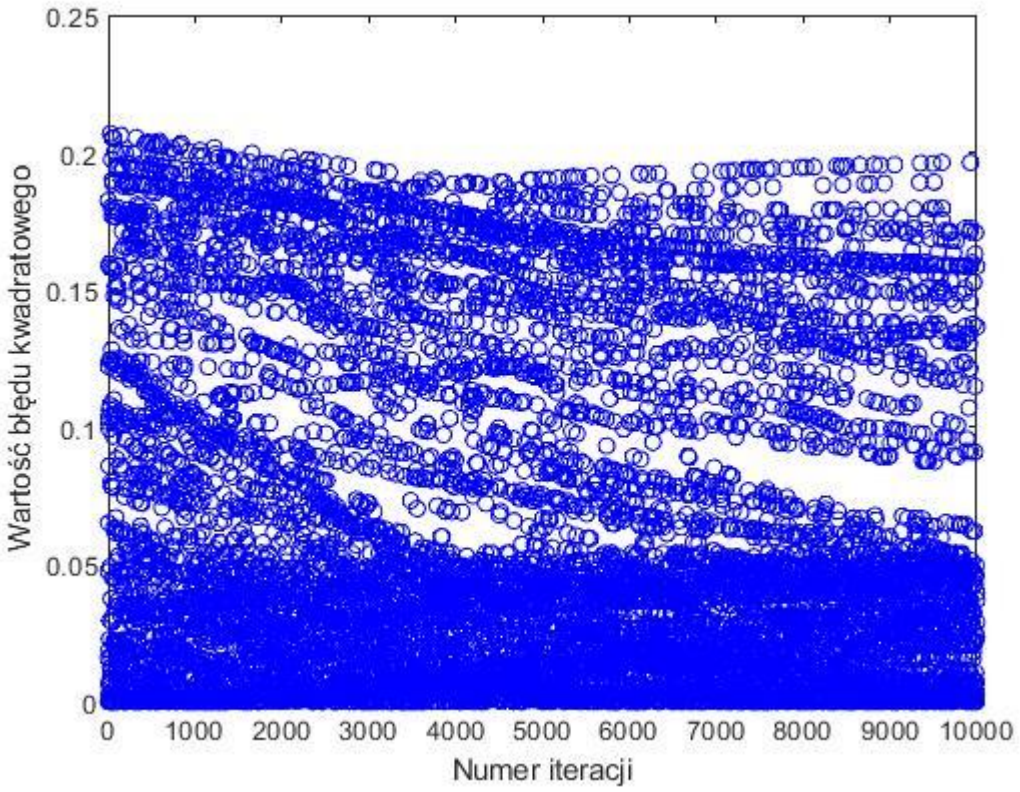


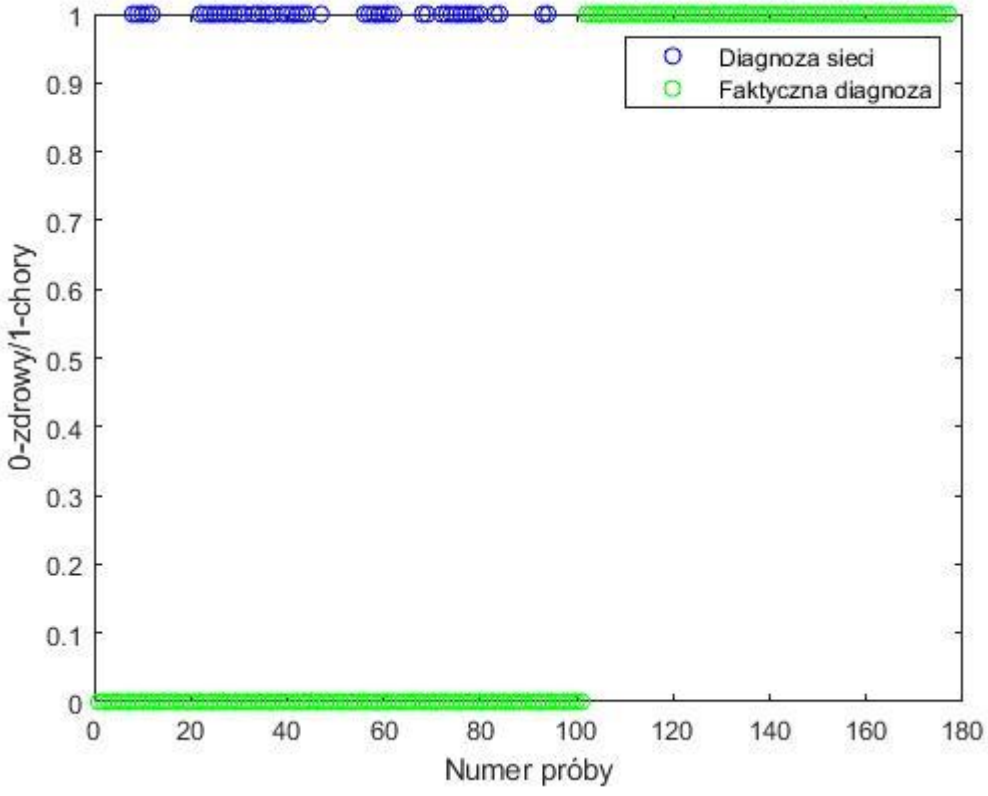
Wartość współczynnika prędkości uczenia: n = 0.0001





Wartość współczynnika prędkości uczenia: n = 0.00001





Listing Kodu

```
clc;
clear all;
close all;
%zczytanie macierzy danych z excela
M = readmatrix('proba.xls','Sheet','Arkusz3','Range','G2:AN2128');
%normalizacja kolumn macierzy
for i=1:34
    maximum=max(M(:,i));
    for j=1:2127
        X(j,i)=M(j,i)/maximum;
    end
end
%przydzial cech jako kolumn macierzy znormalizowanej wraz z transpozycj¹
Nazwy_cech=["LB", "AC", "FM", "UC", "DL", "DP", "Width", "Min", "Max", "Nmax", "Nzeros",
"Mode", "Mean", "Variance"];
h=300;
ill=2051;
%wektor danych treningowych zdrowych
tr_in_1=[X(1:h,1) X(1:h,2) X(1:h,3) X(1:h,4) X(1:h,9) X(1:h,11)  X(1:h,13) X(1:h,14) X(1:h,15)
X(1:h,16) X(1:h,17) X(1:h,18) X(1:h,19) X(1:h,21)];
%wektor danych treningowych chorych
tr_in_2=[X(1951:ill,1) X(1951:ill,2) X(1951:ill,3) X(1951:ill,4) X(1951:ill,9) X(1951:ill,11)
X(1951:ill,13) X(1951:ill,14) X(1951:ill,15) X(1951:ill,16) X(1951:ill,17) X(1951:ill,18)
X(1951:ill,19) X(1951:ill,21)];
%scalony wektor treningowy
tr_in=[tr_in_1;tr_in_2];
training_input=transpose(tr_in);
h_1=h+100;
tst_in_1=[X(h:h_1,1) X(h:h_1,2) X(h:h_1,3) X(h:h_1,4) X(h:h_1,9) X(h:h_1,11)  X(h:h_1,13)
X(h:h_1,14) X(h:h_1,15) X(h:h_1,16) X(h:h_1,17) X(h:h_1,18) X(h:h_1,19) X(h:h_1,21)];
tst_in_2=[X(ill:2126,1) X(ill:2126,2) X(ill:2126,3) X(ill:2126,4) X(ill:2126,9) X(ill:2126,11)
X(ill:2126,13) X(ill:2126,14) X(ill:2126,15) X(ill:2126,16) X(ill:2126,17) X(ill:2126,18)
X(ill:2126,19) X(ill:2126,21)];
tst_in=[tst_in_1;tst_in_2];
test_input=transpose(tst_in);
% wektor informacji zdrowy/chory
output_1=X(1:h,34);
output_2=X(1951:ill,34);
output=[output_1;output_2];
test_output_1=X(h:h_1,34);
test_output_2=X(ill:2126,34);
test_output=[test_output_1;test_output_2];
%ile przypadków chorych w grupie testowej - do wyliczenia czu³oœci sieci
count_test_ill=2126-ill+1;
%ile przypadków zdrowych w grupie testowej - do wyliczenia specyficznoœci
%sieci
count_test_healthy=h_1-h+1;
for i=1:length(test_output)
    if  (test_output(i) < 0.4)
       test_output(i) = 0;
   else
       test_output(i) = 1;
   end
end
num_of_input_neurons=14;
num_of_hidden_neurons=4;
num_of_output_neurons=1;
rng(0,"twister")
for i=1:num_of_hidden_neurons
    for j=1:num_of_input_neurons
        w_hidden(i,j)=normrnd(0,1);
    end
end
fprintf("Wagi po³¹czeñ miêdzy input a hidden\n");

disp(w_hidden);
for i=1:num_of_output_neurons
    for j=1:num_of_hidden_neurons
        w_output(i,j)=normrnd(0,1);
    end
end
fprintf("Wagi po³¹czeñ miêdzy hidden a output\n");
disp(w_output);
%wspó³czynnik uczenia
```

```
n=0.001;
%liczba iteracji
Z=10000;
for z=1:Z
        i=randi([1,200]);
        %suma wa¿ona warstwy ukrytej
        v_1=w_hidden*training_input(:,i);
        %sygna³y na wyjœciu warstwy ukrytej
        y_1=sigmoid(v_1);
        %suma wa¿ona warstwy wyjœciowej
        v=w_output*y_1;
        %sygna³y na wyjœciu warstwy wyjœciowej
        y=sigmoid(v);
        %obliczenie b³êdu
        e=output(i)-y;
        derivative_v=derivative(v);
        delta=derivative_v.*e;
        e_1=(w_output)'*delta;
        derivative_v1=derivative(v_1);
        delta_1=derivative_v1.*e_1;
        %obliczenie wartoœci delta warstwy wyjœciowej
        delta_w_output=n*delta*(y_1)';
        %aktualizacja wartoœci wag warstwy wyjœciowej
        w_output=w_output-delta_w_output;
        %obliczenie wartoœci delta warstwy ukrytej
        delta_w_hidden=n*delta_1*(training_input(:,i))';
        %aktualizacja wartoœci wag warstwy ukrytej
        w_hidden=w_hidden+delta_w_hidden;
        error(z,:)=0.5*(output(i)-y).^2;
end
figure();
plot(1:Z,error(1:Z),'bo');
title('Wartoœæ b³êdu w zale¿noœci od iteracji');
xlabel('Numer iteracji');
ylabel('Wartoœæ b³êdu kwadratowego');
% plot(x,z(1,:));
% figure;
% plot(x,z(2,:),'-bo');
% figure;
% plot(x,z(3,:),'-bo');
fprintf("Wagi po³¹czeñ miêdzy hidden a output po nauce\n");
disp(w_output);
fprintf("Wagi po³¹czeñ miêdzy input a hidden po nauce\n");
disp(w_hidden);
%podanie danych do sprawdzenia poprawnoœci dzia³ania sieci neuronowej
count_ill_correct=0;
count_healthy_correct=0;
for i=1:size(test_input,2)
    v_1_t=w_hidden*test_input(:,i);
    y_1_t=sigmoid(v_1_t);
    v_t=w_output*y_1_t;
    y_t(i)=sigmoid(v_t);
    diagnosis(i)=prog(y_t(i));
    if diagnosis(i)==1
        fprintf("Badany %d jest chory\n",i);
        if (diagnosis(i)==test_output(i))
        count_ill_correct=count_ill_correct+1;
        end
    elseif diagnosis(i)==0
        fprintf("Badany %d jest zdrowy\n",i);
        if (diagnosis(i)==test_output(i))
            count_healthy_correct=count_healthy_correct+1;
        end
    end
end
figure;
plot(diagnosis,'bo');
hold on;
plot(test_output,'go');
title('Porównanie diagnoz prawdziwych z diagnozami postawionymi przez sieæ');
legend('Diagnoza sieci','Faktyczna diagnoza');
xlabel('Numer próby');
ylabel('0-zdrowy/1-chory');
%czu³oœæ
czul=count_ill_correct/count_test_ill;
fprintf("Wartoœæ czu³oœci sieci to %.2f\n",czul);
%specyficznoœæ
```

```
spec=count_healthy_correct/count_test_healthy;
fprintf("Wartoœæ specyficznoœci sieci to %.2f\n",spec);
%funkcja pomocnicza obliczaj¹ca wartoœæ funkcji aktywacji - sigmoidalnej
function sigmoid=sigmoid(v)
    alfa=10;
    sigmoid=1./(1+exp(-alfa*v));
end
%funkcja pomocnicza obliczaj¹ca wartoœæ pochodnej funkcji aktywacji
function derivative=derivative(v)
       alfa=10;
       derivative=(alfa*exp(-alfa*(v)))./(1+exp(-alfa*(v))).^2;
end
function prog=prog(y)
    if y<0.5
        prog=0;
    elseif y>=0.5
        prog=1;
    end
end
```

Bibliografia

Informacje o czułości i specyficzności

https://brain.fuw.edu.pl/edu/index.php/Uczenie_maszynowe_i_sztuczne_sieci_neuronowe/Wyk%C5%82ad_Ocena_jako%C5%9Bci_klasyfikacji

Symulator sieci neuronowej

https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=regplane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.46723&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false