Универзитет у Београду

Електротехнички факултет


# Неуралне мреже

# Извештај првог пројектног задатка

Студенти:

Јован Спасојевић 0118/2017

Алекса Богдановић 0578/2017

# Садржај

## Задатак 1 : Решавање проблема регресије применом неуралне мреже

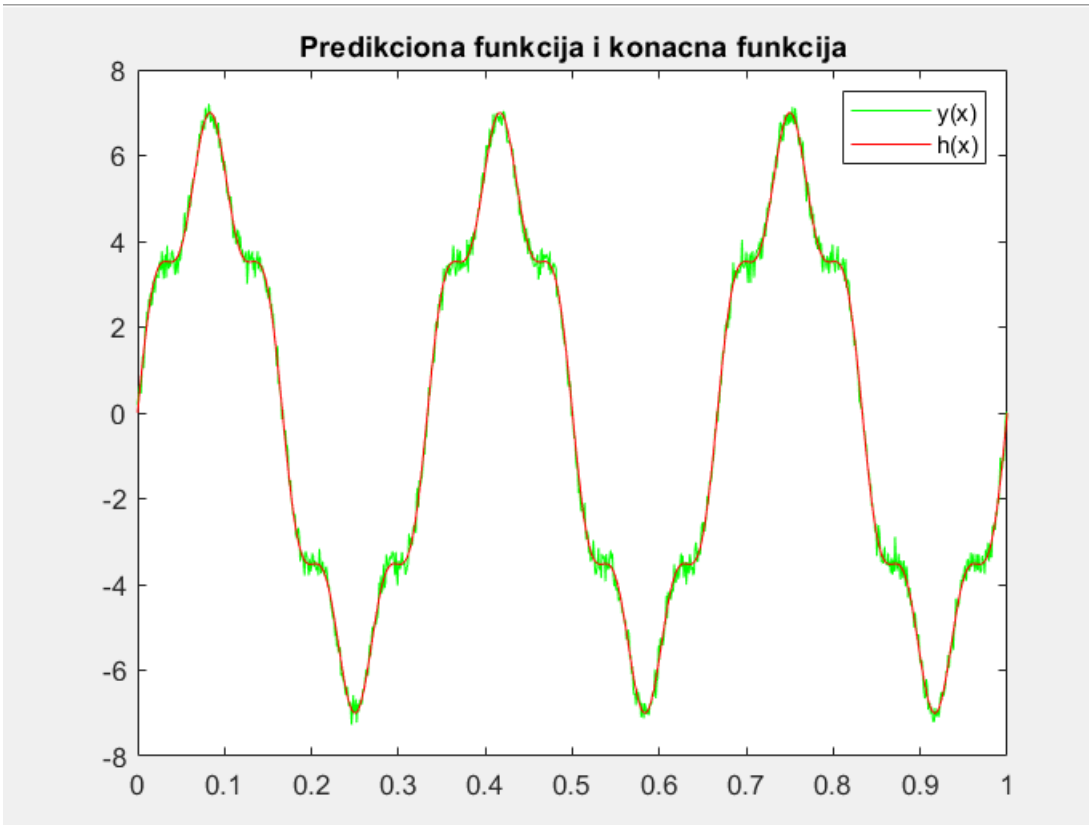
Потребно је креирати неуралну мрежу са предикцијом функције

$$\mathrm{h(x)} = \mathrm{Asin}(2f_1\mathrm{x}) + \mathrm{Bsin}(2f_2\mathrm{x})$$

Вредности параметара су:

| Параметар | Формула | Вредност |
|---|---|---|
| A | $mod((118 + 2017),7) + 1$ | 1 |
| B | $mod((578 + 2017),4) + 3$ | 6 |
| $f_1$ | $5 * (mod((1 + 1 + 8),4) + 1)$ | 15 |
| $f_2$ | $3 * (mod((5 + 7 + 8),4) + 1)$ | 3 |

Дата предикциона функција са датим параметрима изгледа као на слици 1. Вредности улаза ће бити бројеви између 0 и 1 са кораком од 0,001 и имаће 3 периоде. Вредности y-осе представља амплитуду, а вредности x-осе представљају вредност променљиве x. Функција y(x) представља збир предикционе функције и случајног шума. Функција h(x) представља предикциону функцију.




*Слика 1 : Предикциона функција(црвена) и зашумљена функција(зелена)*

Бројимо колико имамо вредности улаза како би могли да поделимо вредности за тренирање и тестирање неуралне мреже. Имамо 3001 вредности између 0 и 3 са кораком 0,001 и од тога 80% користимо за тренирање и 20% за тестирање.

Пре него што поделимо вредности за тренирање и тестирање радимо Suffile(насумично промешане вредности) над вредностима улаза.

```
% Suffile
idx = randperm(number);
```

Са овим вредностима креирамо неуралну мрежу.

Изабрао сам да се ради са 50 неурона који се налазе у скривеном слоју.

```
NN = feedforwardnet(50);
```

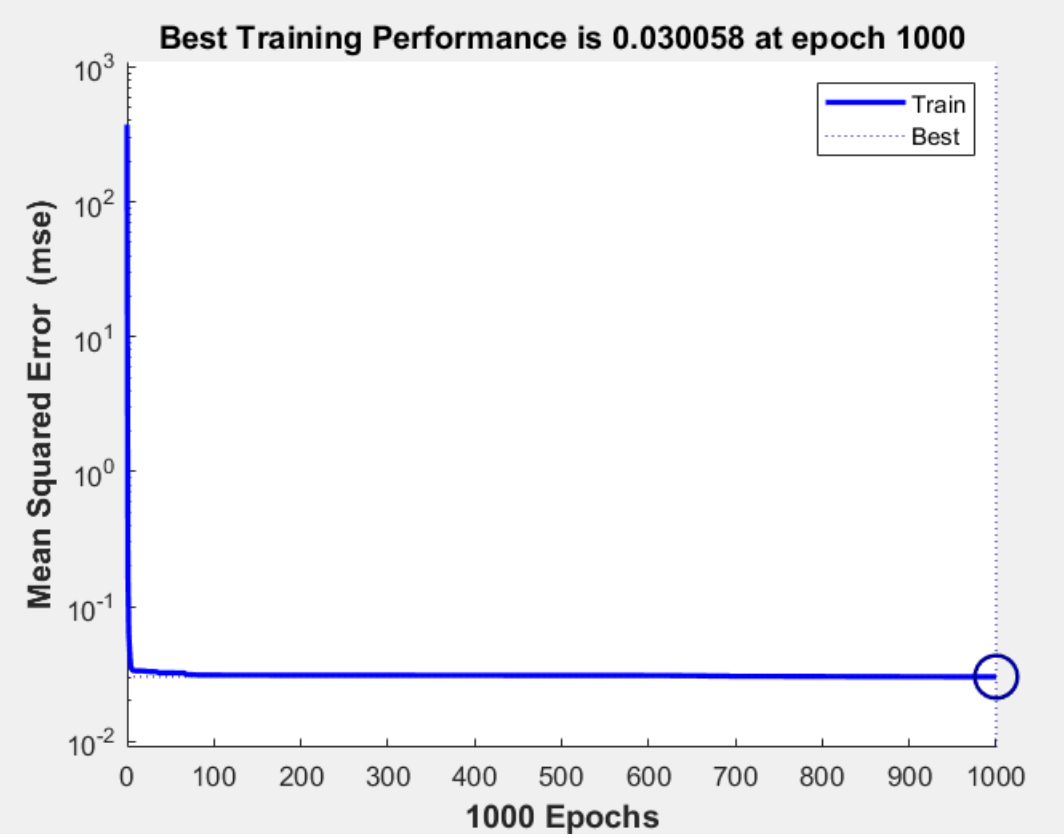
Искључена је заштита од преобучавања.

```
NN.divideFcn = "";
```

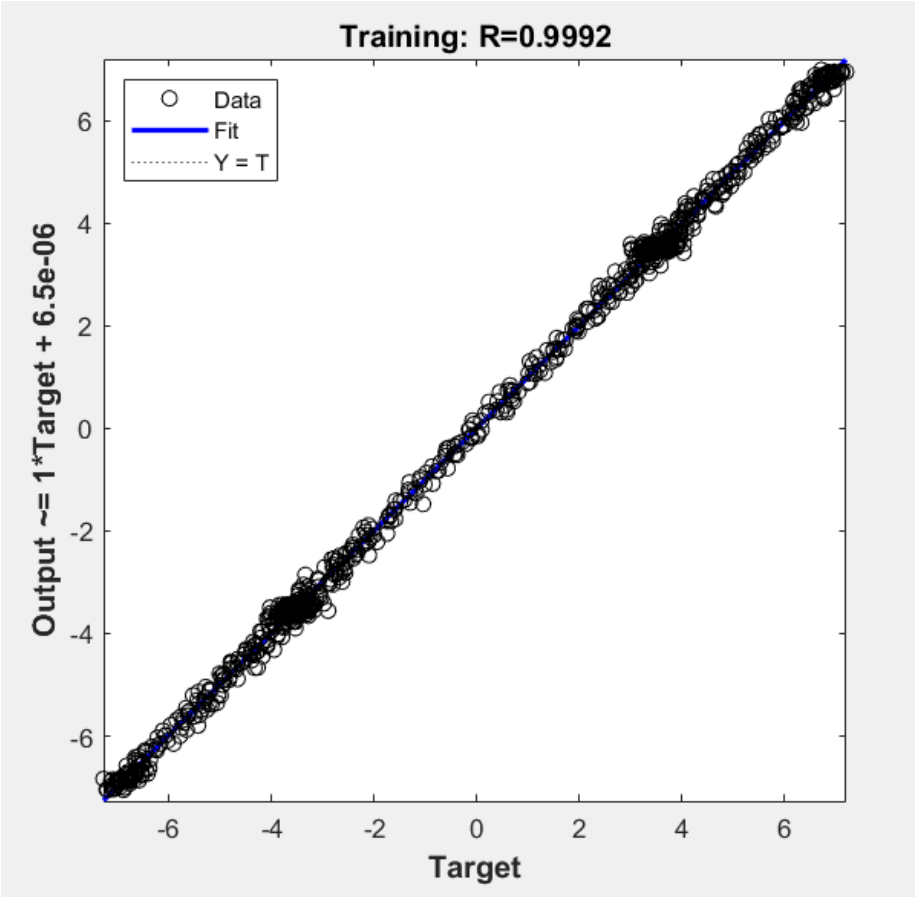
Тренирамо нашу неуралну мрежу са 50 неурона над 80% података које смо узели за тренирање наше мреже.

```
% Treniranje neuralne mreze
NN = train(NN,inputs_train,finaly_train);
```

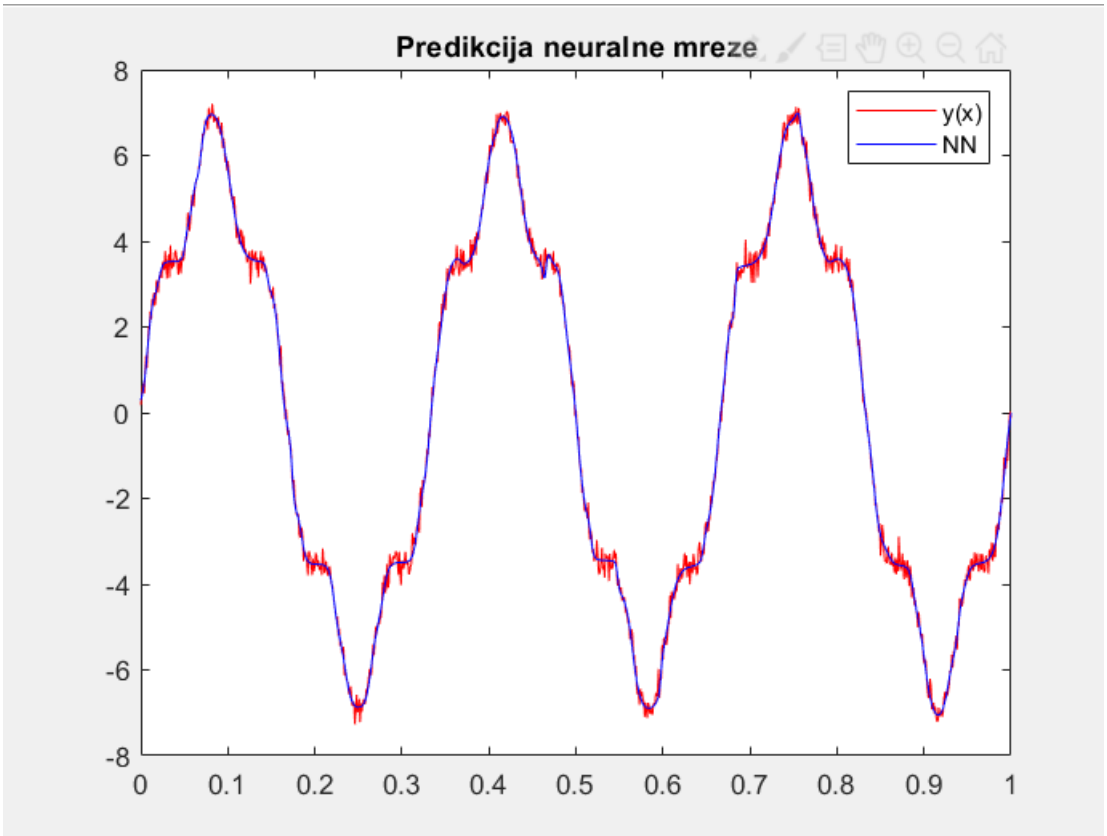
На следећим сликама ће бити приказана преформанса неуралне мреже у 1000 епоха и регресија неуралне мреже.




*Слика 2 : Преформанса неуралне мреже*




*Слика 3 : Регресија неуралне мреже*




*Слика 4 : Успешност предикције неуралне мреже*

На основу последње слике можемо рећи да је неуралнa мрежа успешно истренирана и да наша неуралнa мрежа добро предвиђа.

**Код овог задатка у Matlab-у**

```
function zadatak1

A = mod((118+2017),7)+1;
B = mod((578+2017),4)+3;
f1 = 5*(mod((1+1+8),4)+1);
f2 = 3*(mod((5+7+8),4)+1);

inputs = 0:0.001:1;

outputs = A*sin(2*pi*f1*inputs)+B*sin(2*pi*f2*inputs);

std =(0.2 * min(A,B)).*randn(1,length(inputs));

finaly = outputs + std;

plot(inputs, finaly, 'g', inputs, outputs, 'r' );
title('Predikciona funkcija i konacna funkcija');
legend('y(x)','h(x)');

number = size(finaly, 2);

number_train = 0.8*number;
number_test  = 0.2*number;

idx = randperm(number);

inputs_train = inputs(:, idx(1 : number_train));
finaly_train = finaly(:, idx(1 : number_train));

inputs_test = inputs(:, idx(number_train+1 : number));
finaly_test = finaly(:, idx(number_train+1 : number));

NN = feedforwardnet(50);

NN.divideFcn = "";

NN = train(NN,inputs_train,finaly_train);

a = sim(NN,inputs_test);

figure;
plot(inputs, finaly, 'r', inputs, NN(inputs), 'b');
title('Predikcija neuralne mreze');
legend('y(x)','NN');
```

## Задатак 2 : Решавање проблема класификације применом неуралне мреже

Треба да се креира неуронска мрежа која ће служити за класификацију вештачки генерисаних података уз помоћ *.mat* датотеке која садржи податке које треба класификовати уз помоћ неуралне мреже.

Вредност потребног параметара(за избор *.mat* датотеке)

| Параметар | Формула | Вредност |
|---|---|---|
| Dataset_number | $mod\big((118+578),3\big)+1$ | 1 |

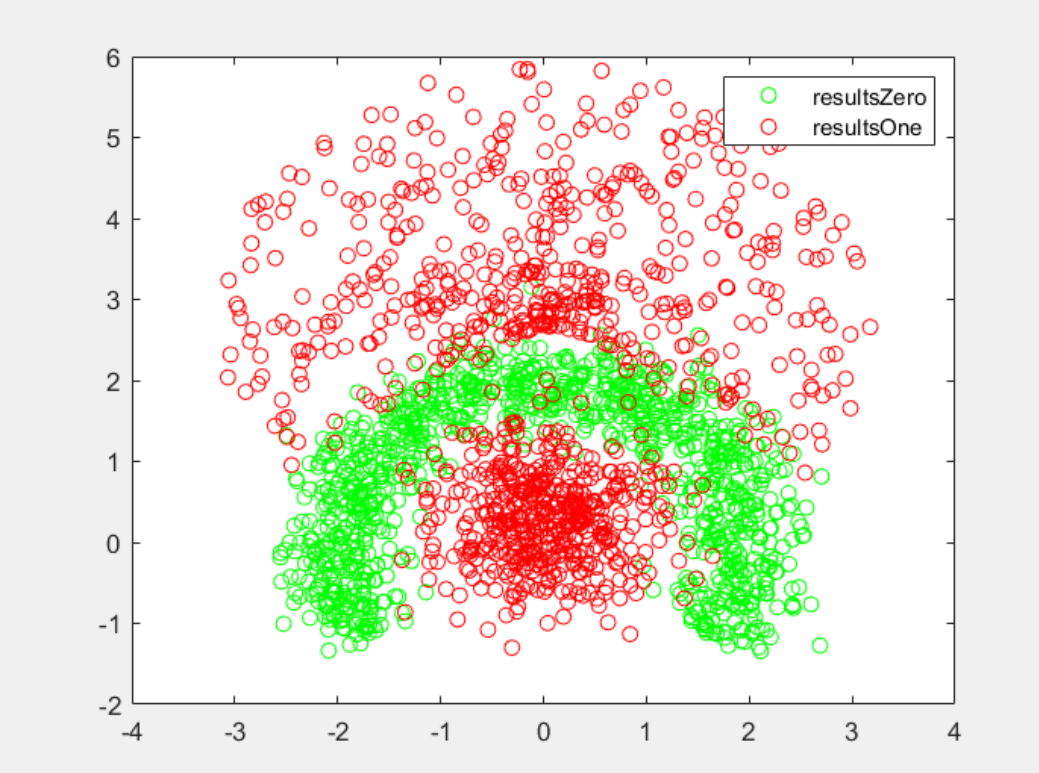
Учитавамо податке из *dataset1.mat* и параметар постављамо у облику матрице.

```
values = load('dataset1.mat');
matrix = values.data;
```

Матрица има 3 колоне. Прве две колоне садрже обележја. Трећа колона садржи припадност класи. И то ћемо поделити у неколико параметара, одвојићемо припадности тамо где има 0 и тамо где има 1 и узећемо вредности из прве и друге колоне.

```
resultsZero = matrix(1:1000,1:2)';
resultsOne = matrix(1001:2000,1:2)';
```

На основу ових вредности, на следећој слици, видећемо приказ података према *resultsZero* и *resultsOne*.




*Слика 5 : Приказ података*

На основу ових података, као и у претходнднном задатку, морамо да поделимо на 80% и 20% за тренирање и тестирање. Наравно радимо и Suffile пре поделе вредности.

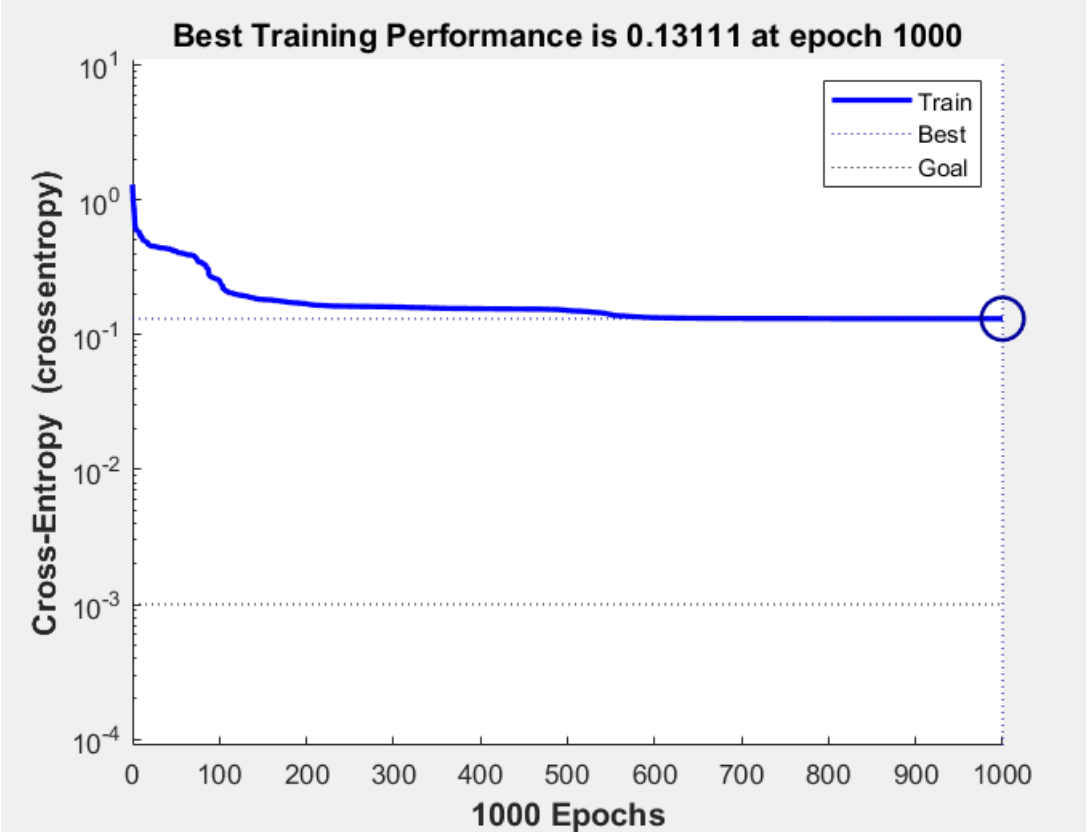
```
number = length(inputs(1,:));
% Suffile
idx = randperm(number);
```

Креирамо три врсте неуралне мреже по захтевима задатка

**1) Оптимална неуралне мрежа**

У овој неуралној мрежи сам користио 2 скривена слоја са 3 неурона у првом слоју и 4 у другом слоју. Искључена је заштита од преобучавања и постављено је да се ради у 1000 епоха.

```
NN = patternnet([3 4]);
NN.divideFcn = '';
NN.trainParam.epochs = 1000;
```

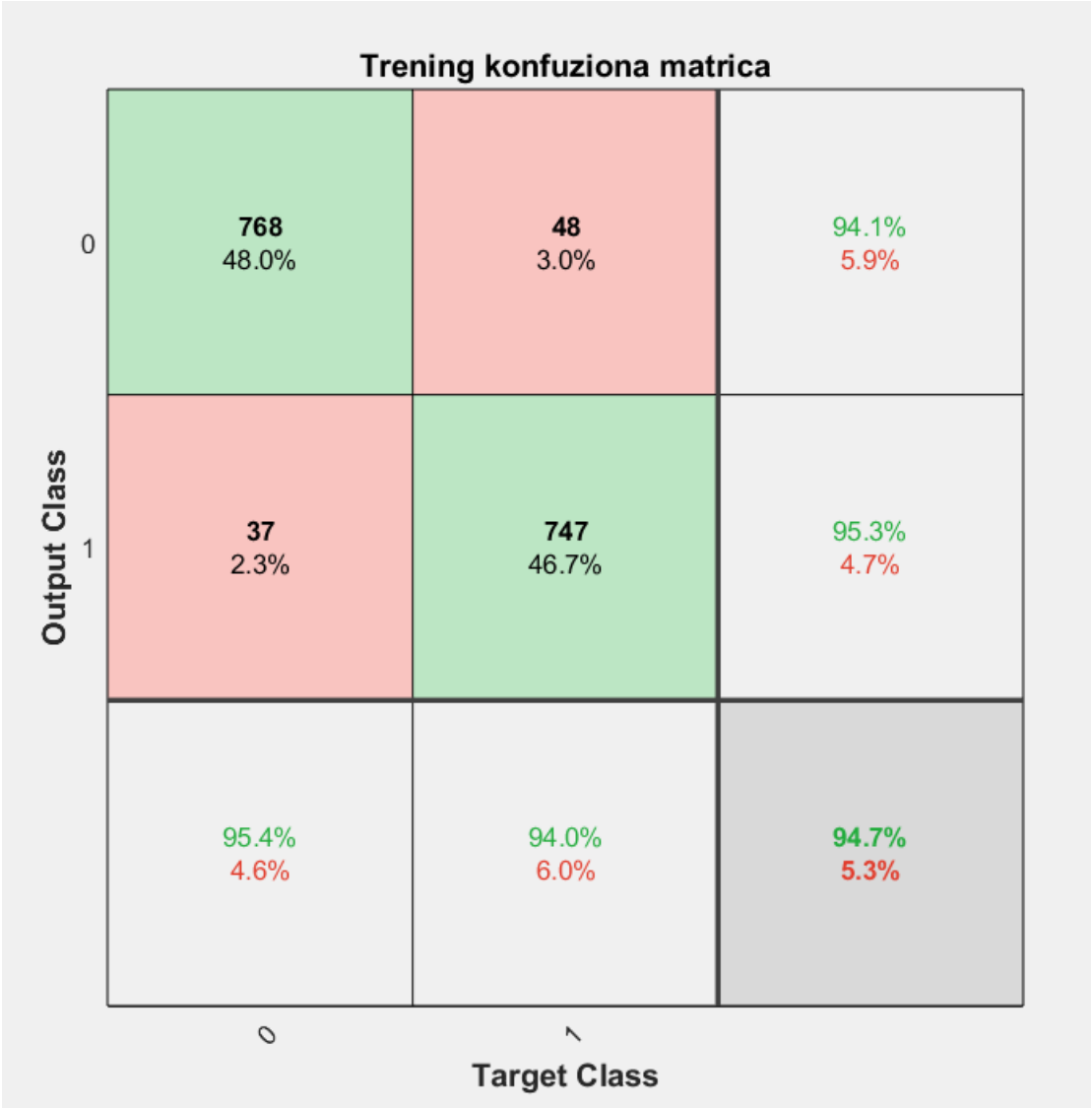



*Слика 6 : Крива преформансе оптималне неуралне мреже*

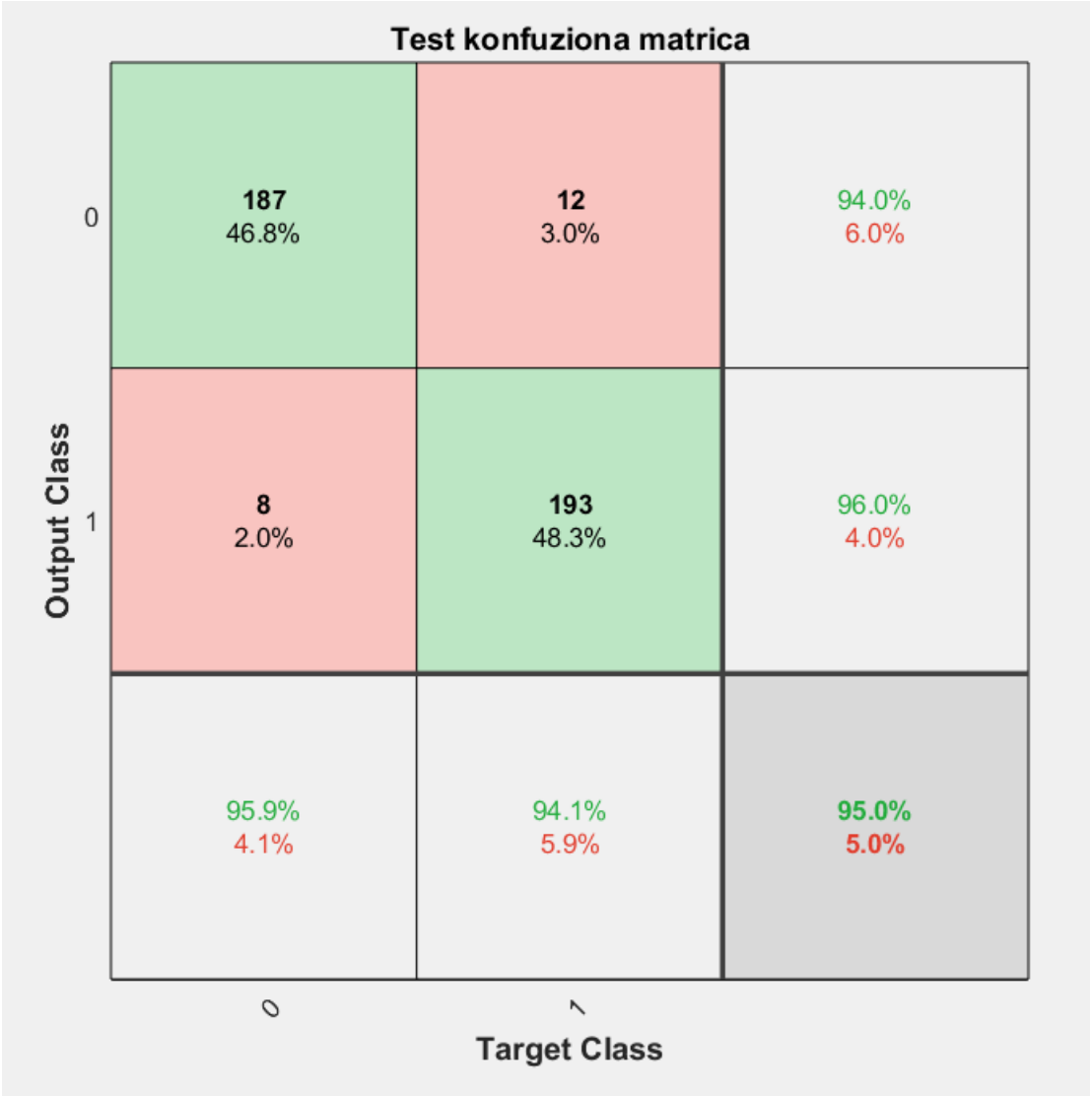
Параметри прецизности и осетљивости за оптималну неуралну мрежу

| Параметар | Формула | Вредност |
|---|---|---|
| precisionTrain | $\frac{trainTransp(1,1)}{sum(trainTransp(1,:))}$ | 0.941176470588235 |
| precisionTest | $\frac{testTransp(1,1)}{sum(testTransp(1,:))}$ | 0.939698492462312 |
| recallTrain | $\frac{trainTransp(1,1)}{sum(trainTransp(1,:))}$ | 0.954037267080745 |
| recallTest | $\frac{testTransp(1,1)}{sum(testTransp(1,:))}$ | 0.958974358974359 |

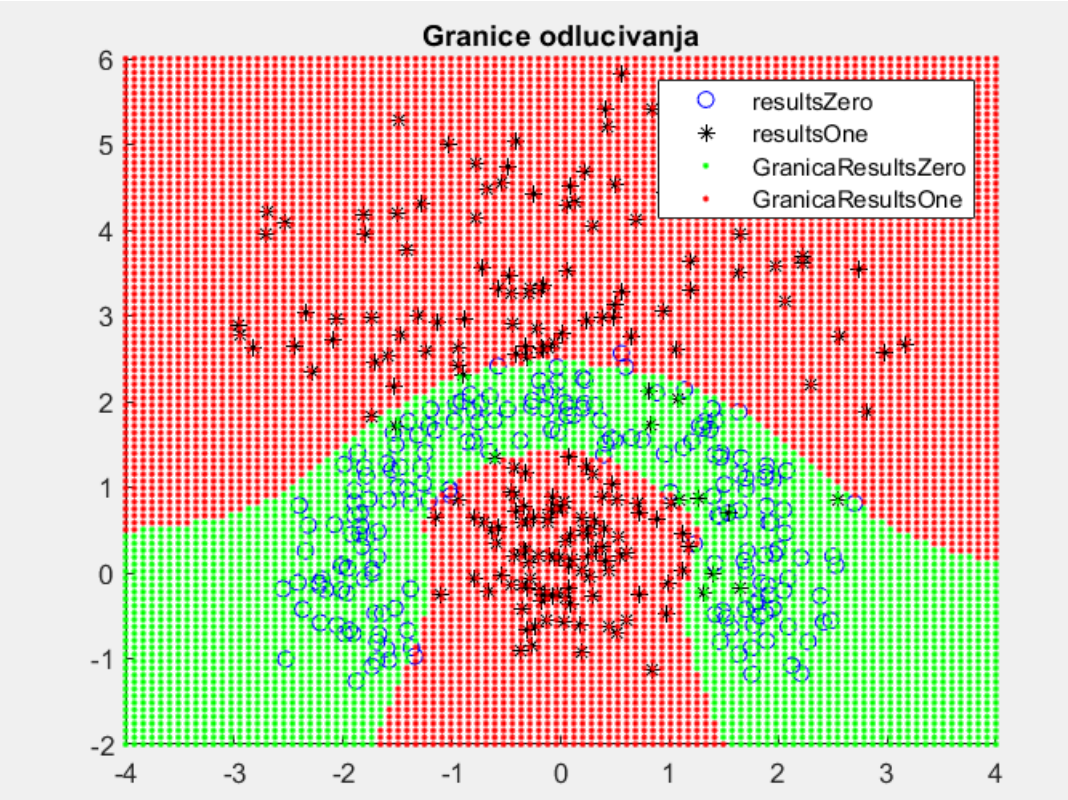
Конфузионе матрице за ову неуралну мрежу:




*Слика 7 : Тренинг конфузиона матрица оптимално обучене неуралне мреже*




*Слика 8 : Тест конфузиона матрица оптимално обучене неуралне мреже*



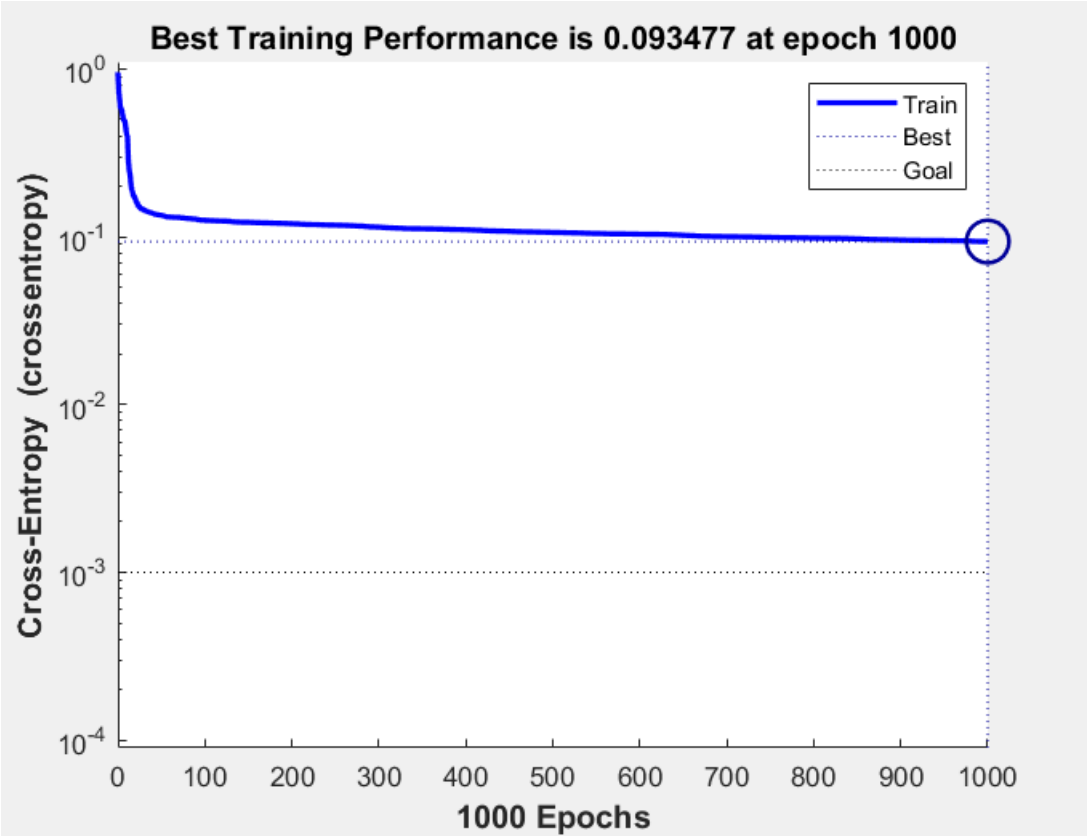

*Слика 9 : Граница одлучивања оптимално обучене мреже*

На основу ове границе одлучивања видимо да је добро обучена мрежа, увек се мора пазити на број неурона, ако ставимо мали број неурона граница ће бити јасна и види се јасно да је ово *appropriate-fitting*, ако ставимо велики број неурона границе ће бити конфузне и неће бити скроз јасне.

**2) *Overfitting* неуралнa мрежа**

У овој неуралној мрежи сам користио 3 скривена слоја по 6 неурона. Искључена је заштита од преобучавања и постављено је да се ради у 1000 епоха.

```
NN = patternnet([6 6 6])
NN.divideFcn = '';
NN.trainParam.epochs = 1000;
```

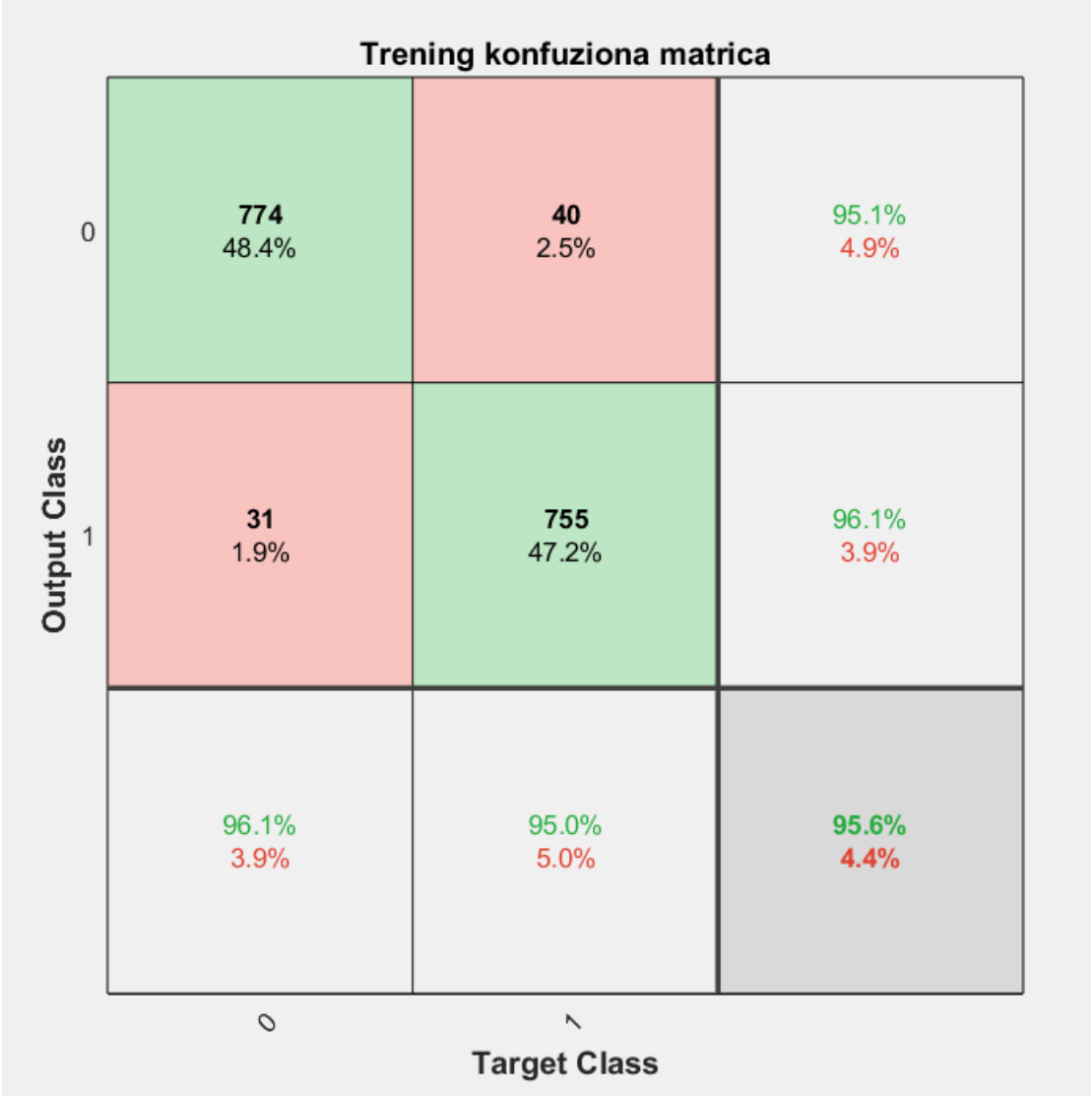



*Слика 10 : Крива преформансе Overfitting неуралне мреже*

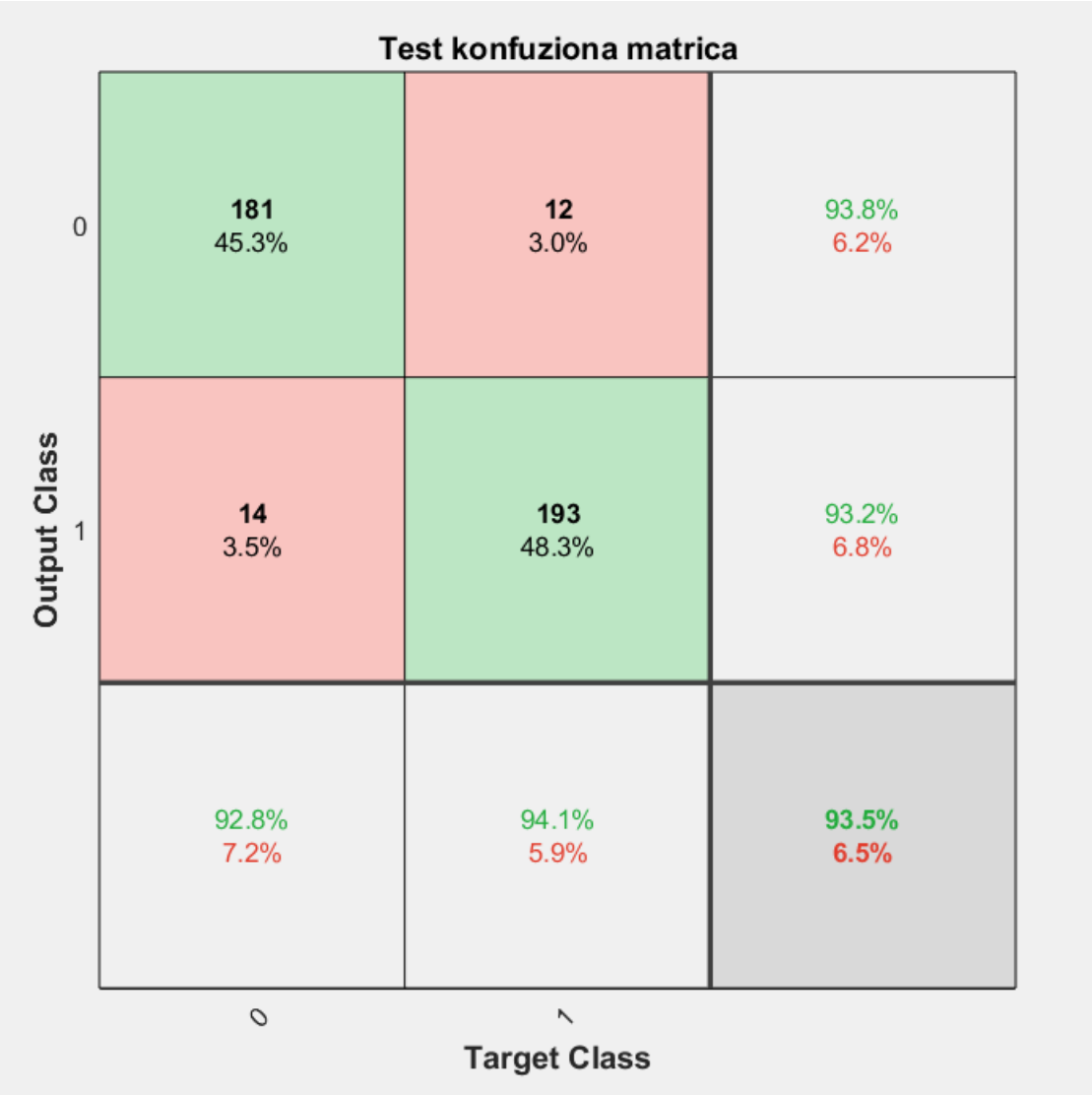
Параметри прецизности и осетљивости за *Overfitting* неуралну мрежу

| Параметар | Формула | Вредност |
|---|---|---|
| precisionTrain | $\frac{trainTransp(1,1)}{sum(trainTransp(1,:))}$ | 0.950859950859951 |
| precisionTest | $\frac{testTransp(1,1)}{sum(testTransp(1,:))}$ | 0.937823834196891 |
| recallTrain | $\frac{trainTransp(1,1)}{sum(trainTransp(1,:))}$ | 0.961490683229814 |
| recallTest | $\frac{testTransp(1,1)}{sum(testTransp(1,:))}$ | 0.928205128205128 |

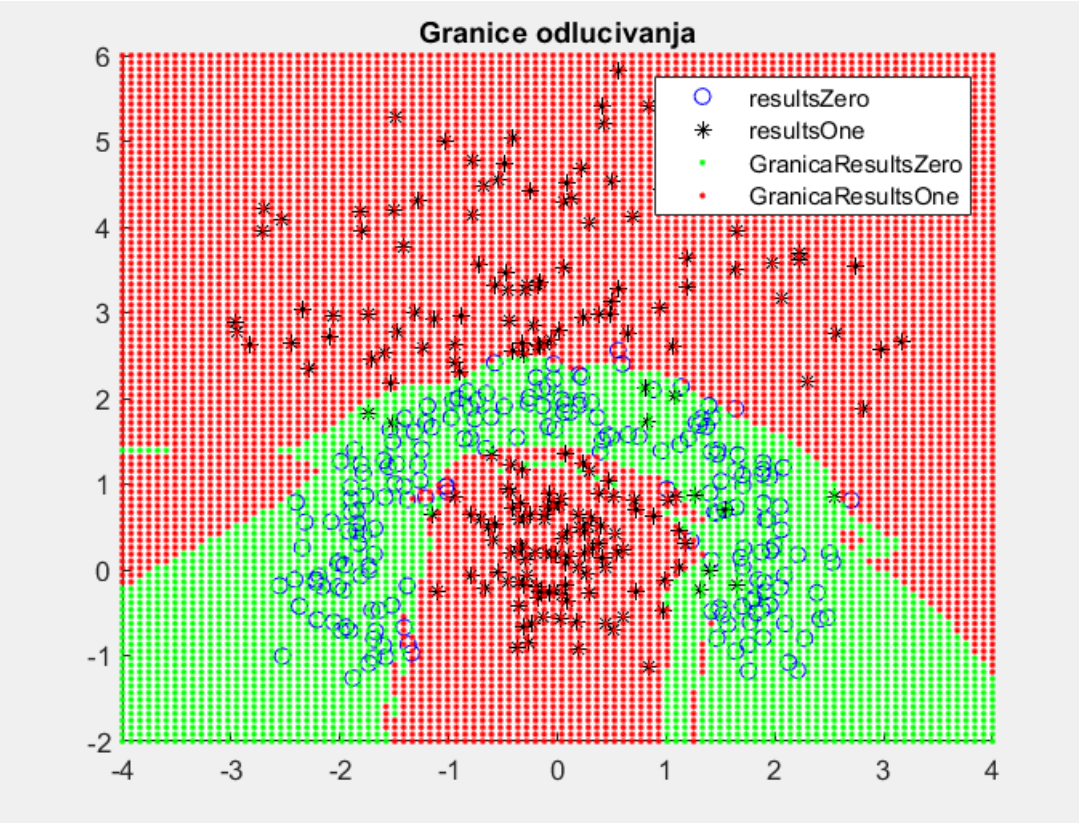
Конфузионе матрице за ову неуралну мрежу:




*Слика 11 : Тренинг конфузиона матрица Overfitting обучене неуралне мреже*




*Слика 12 : Тест конфузиона матрица Overfitting обучене неуралне мреже*



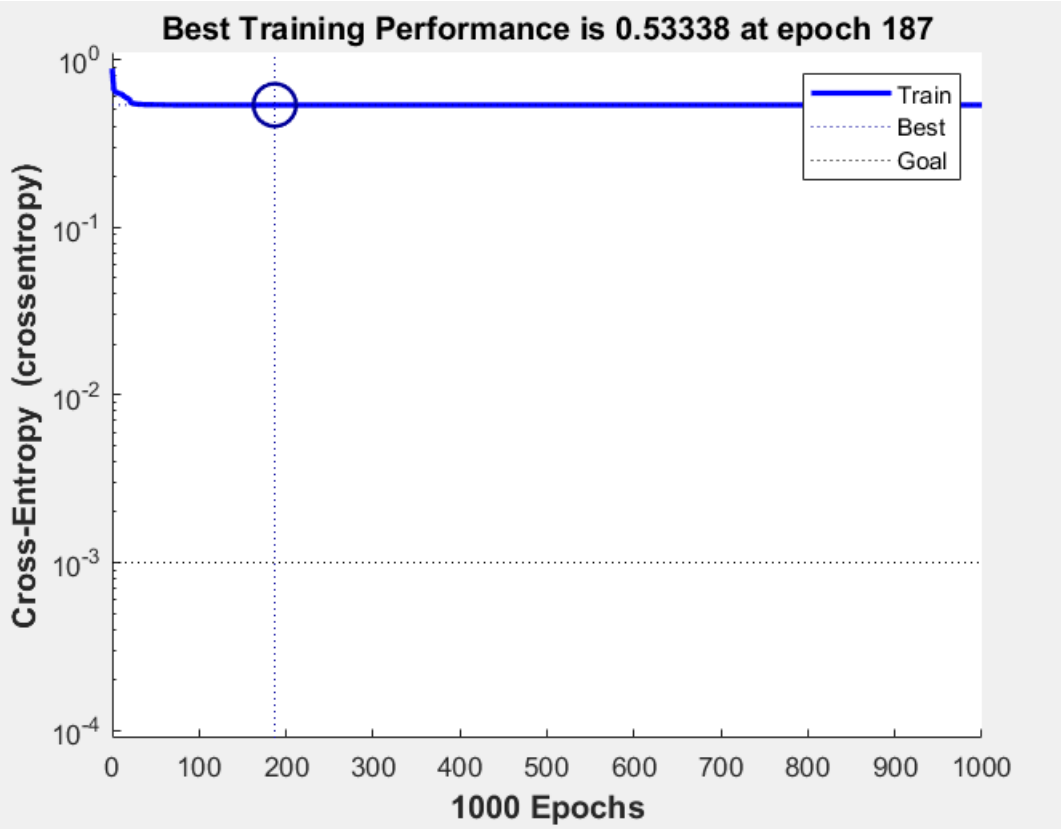

*Слика 13 : Граница одлучивања Overfitting обучене мреже*

На основу ове границе одлучивања видимо да је добро обучена *over-fitting* мрежа, мада као и у претходној неуралној мрежи мора се пазити на број неурона, исто важи ако је превелики број неурона биће лоше границе. За нијансу је боља од претходне мреже.

**3) *Underfitting* неуронска мрежа**

У овој неуралној мрежи сам користио један скривени слој са 1 неуроном. Искључена је заштита од преобучавања и постављено је да се ради у 1000 епоха.

```
NN = patternnet(1);
NN.divideFcn = '';
NN.trainParam.epochs = 1000;
```

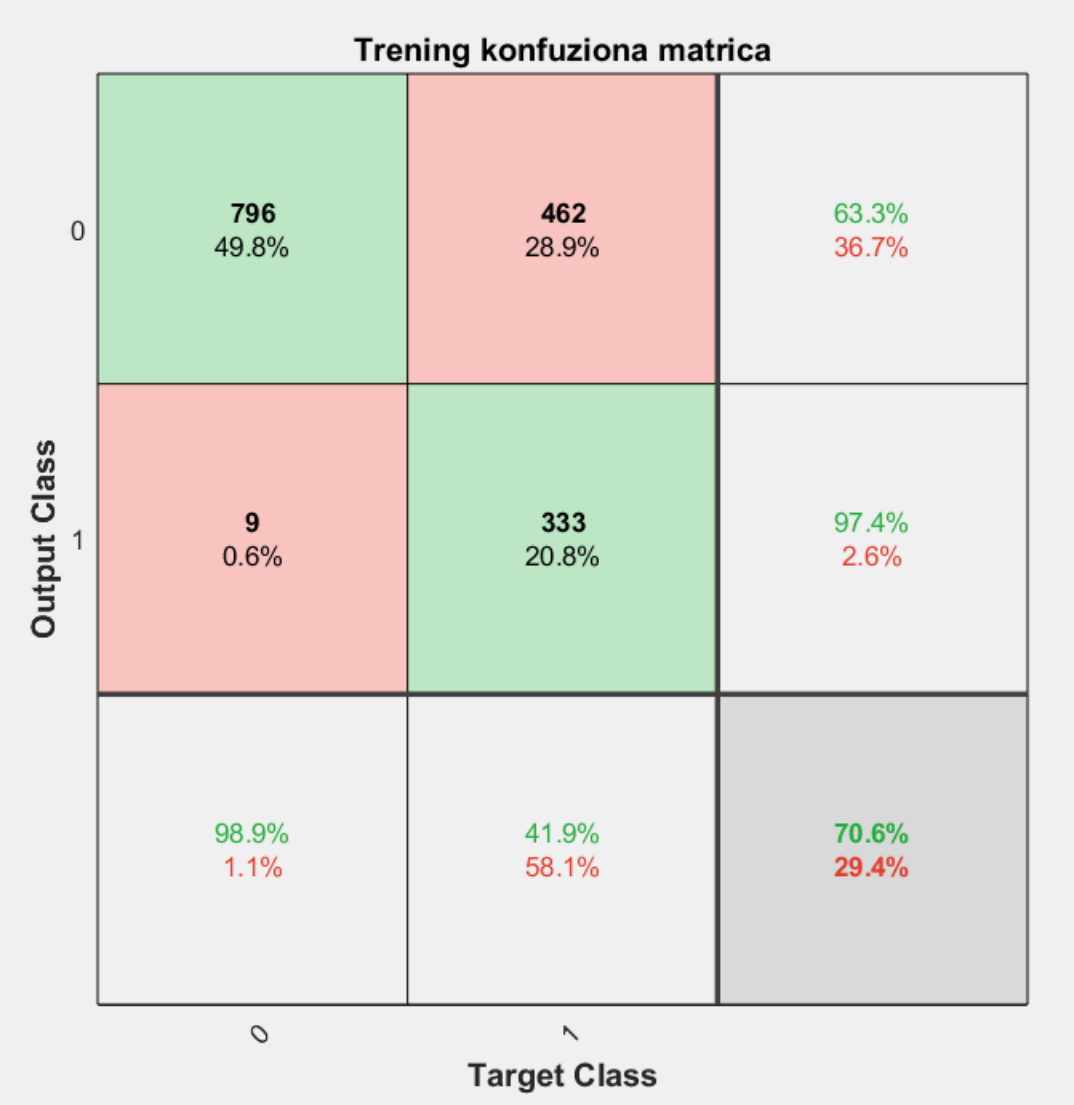



*Слика 14 : Крива преформансе Underfitting неуралне мреже*

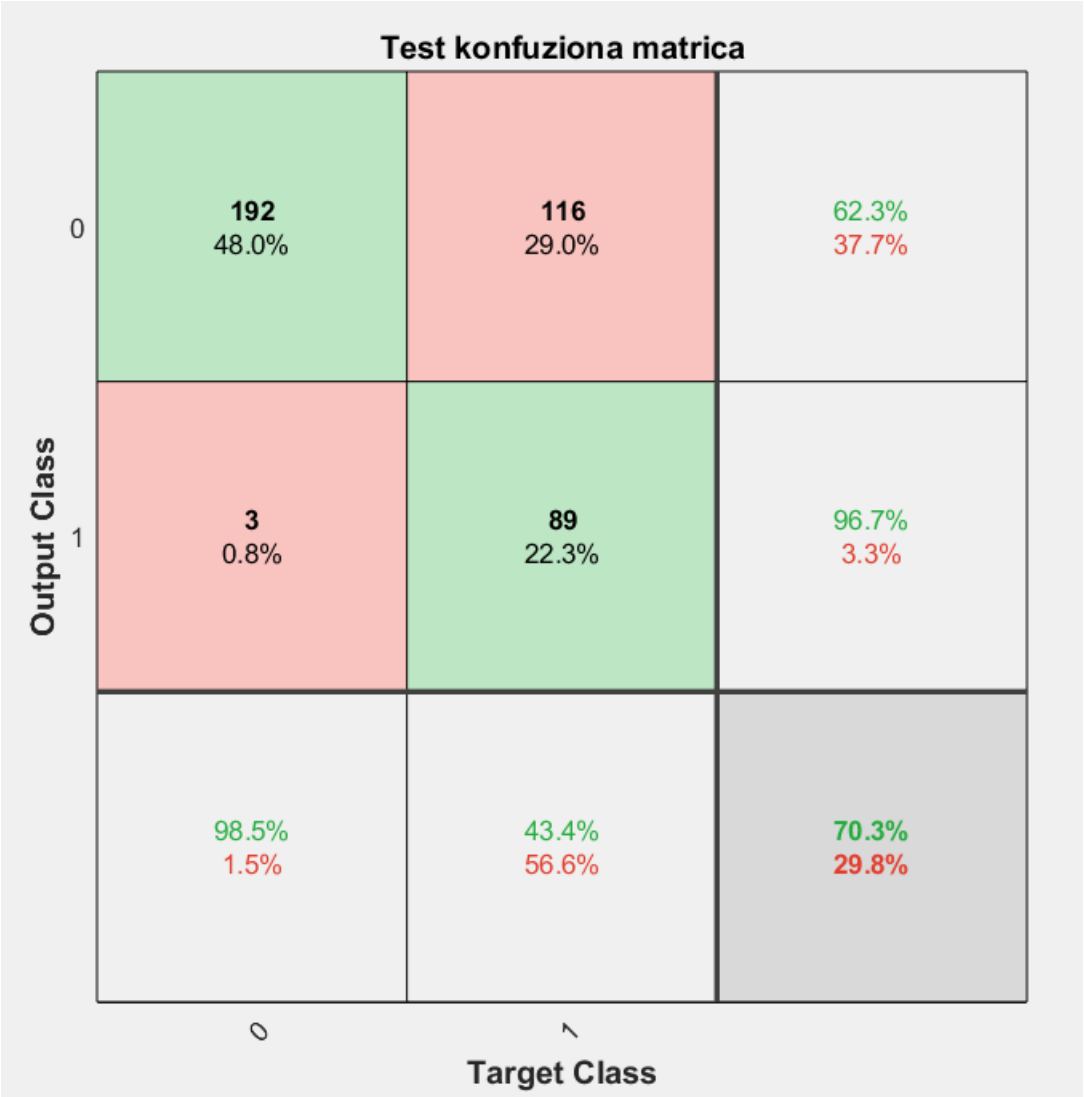
Параметри прецизности и осетљивости за *Underfitting* неуралну мрежу

| Параметар | Формула | Вредност |
|---|---|---|
| precisionTrain | $\frac{trainTransp(1,1)}{sum(trainTransp(1,:))}$ | 0.632750397456280 |
| precisionTest | $\frac{testTransp(1,1)}{sum(testTransp(1,:))}$ | 0.623376623376623 |
| recallTrain | $\frac{trainTransp(1,1)}{sum(trainTransp(1,:))}$ | 0.988819875776398 |
| recallTest | $\frac{testTransp(1,1)}{sum(testTransp(1,:))}$ | 0.984615384615385 |

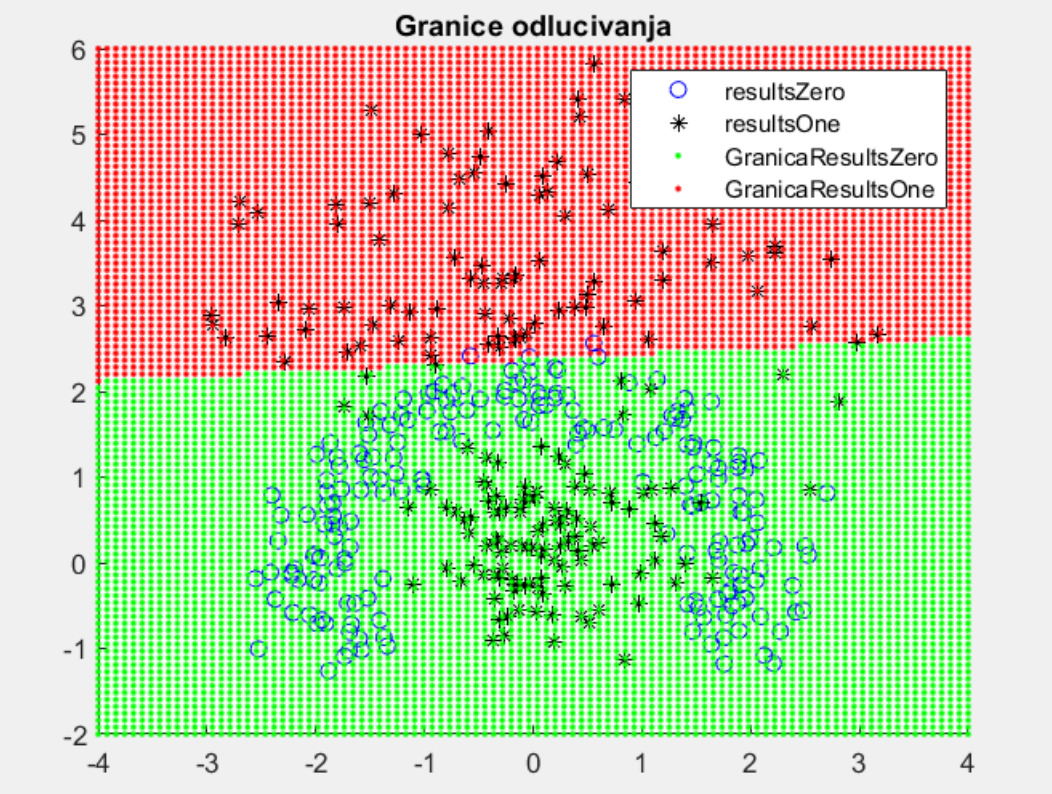
Конфузионе матрице за ову неуралну мрежу:




*Слика 15 : Тренинг конфузиона матрица Underfitting обучене неуралне мреже*




*Слика 16 : Тест конфузиона матрица Underfitting обучене неуралне мреже*




*Слика 17 : Граница одлучивања Underfitting обучене мреже*

На основу ове границе одлучивања видимо да је добро обучена *under-fitting* мрежа, као што је речено у претходна два случаја мора се пазити на број неурона, ако ставимо 1, 2 или 3 неурона ове границе ће изгледати као *under-fitting* мреже, ако се стави више границе ће изгледати исто као и у првом задатку па већ није *under-fitting* мрежа. Ова мрежа је лошија у односу на претходна 2 примера.

**Код овог задатка у Matlab-у**

```
function zadatak2

values = load('dataset1.mat');
matrix = values.data;

inputs = matrix(:,1:2)';
resultsZero = matrix(1:1000,1:2)';
resultsOne = matrix(1001:2000,1:2)';
finaly = matrix(:,3)';

figure;
    plot(resultsZero(1,:),resultsZero(2,:),'go');
    hold on;
    plot(resultsOne(1,:),resultsOne(2,:),'ro');
    legend('resultsZero','resultsOne');

number = length(inputs(1,:));

number_train = 0.8*number;
number_test  = 0.2*number;
```

```
idx = randperm(number);

inputs_train = inputs(:, idx(1 : number_train));
finaly_train = finaly(:, idx(1 : number_train));

inputs_test = inputs(:, idx(number_train+1 : number));
finaly_test = finaly(:, idx(number_train+1 : number));

NN = patternnet([3 4]);

    NN.divideFcn = '';
    NN.trainParam.epochs = 1000;
    NN.trainParam.goal = 1e-3;
    NN.trainParam.min_grad = 1e-9;
    NN.trainParam.min_grad = 1e-8;

    NN = train(NN,inputs_train,finaly_train);

    a = sim(NN, inputs_test);

    output_a = sim(NN,inputs_train);

figure;
    plotconfusion(finaly_train,output_a);
    title('Trening konfuziona matrica');

figure;
    plotconfusion(finaly_test, a);
    title('Test konfuziona matrica');

    correctZero = inputs_test(:,finaly_test==0);
    correctOne = inputs_test(:,finaly_test==1);

    numbersPoint = 100;
    xLimit = linspace(-4, 4, numbersPoint);
    yLimit = linspace(-2, 6, numbersPoint);

    input_test_limit = [];

    for i = xLimit
        input_test_limit = [input_test_limit
[i*ones(size(yLimit)); yLimit]];
    end

    outTest = sim(NN, input_test_limit);
    limitResultsZero = input_test_limit(:,outTest<0.5);
    limitResultsOne = input_test_limit(:,outTest>=0.5);
```

```
figure;
    hold all;
        plot(correctZero(1,:),correctZero(2,:),'bo');
        plot(correctOne(1,:),correctOne(2,:),'k*');
        scatter(limitResultsZero(1,:),
limitResultsZero(2,:),'g.');

scatter(limitResultsOne(1,:),limitResultsOne(2,:),'r.');
        title('Granice odlucivanja');

legend('resultsZero','resultsOne','GranicaResultsZero','
GranicaResultsOne');

    [~,trainT,~,~] = confusion(finaly_train,output_a);
    trainTransp = trainT';
    [~,testT,~,~] = confusion(finaly_test,a);
    testTransp = testT';

    precisionTrain =
trainTransp(1,1)/sum(trainTransp(1,:));
    precisionTest =
testTransp(1,1)/sum(testTransp(1,:));

    recallTrain =
trainTransp(1,1)/sum(trainTransp(:,1));
    recallTest = testTransp(1,1)/sum(testTransp(:,1));

NN = patternnet([6 6 6]);

    NN.divideFcn = '';
    NN.trainParam.epochs = 1000;
    NN.trainParam.goal = 1e-3;
    NN.trainParam.min_grad = 1e-9;
    NN = train(NN,inputs_train,finaly_train);

    a = sim(NN, inputs_test);

%% Ponavljaju se delovi koda sa pravljenjem confusion
matric i parametra preciznosti i osetljivosti
```

```
NN = patternnet(1);

    NN.divideFcn = '';
    NN.trainParam.epochs = 1000;
    NN.trainParam.goal = 1e-3;
    NN.trainParam.min_grad = 1e-9;
    NN = train(NN,inputs_train,finaly_train);

    a = sim(NN, inputs_test);

%% Ponavljaju se delovi koda sa pravljenjem confusion
```

## Задатак 3: Тражење оптималних хипермараметара применом методе унакрсне валидације

| Параметар | Формула | Вредност |
|---|---|---|
| Q | $mod((1+1+8+5+7+8),8)+1$ | 7 |

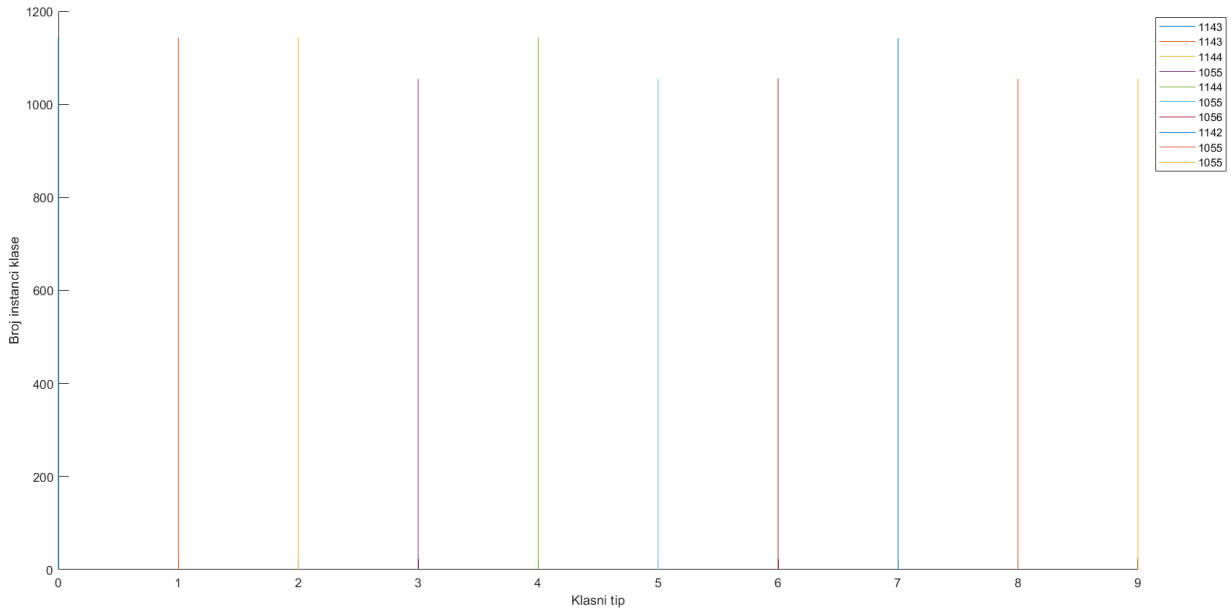
Потребно је направити неуралну мрежу за препознавање цифара писане руком.

Имаће 16 улаза који представљају 8 тачака(x и y) , а излаз треба да представља цифру која је препозната.

За решавање овог проблема сам користио **patternet** мрежу која има 16 улаза и 10 излаза.

Излази ће имати вредности 0 или 1(само један излаз може бити 1) који ће се користити за класификацију.

Скуп података је већ подељен у два фајла "**pendigits.tes**" тест скуп и "**pendigits. tra**" тренинг скуп. Учитавањем оба фајла добијамо овакав приказ број одбирака по класама:




*Zad3_Slika 1 Приказ броја одбирака по класама*

Користи се унакрсна валидација да би се нашли најбољи хиперпарамтери мреже и њене структура. Избор хиперпараметара у унакрсној валидацији ће се вршити по **Accuracy** зато што су подаци донекле балансирани.

**Accuracy** се рачуна као количник збира TP и TN и укупног броја сета података.

TP је број одбирала који су коректно класификовани као тражена класа, а TN број одбирака који су коректно класификовани као остале класе.

Вредности хиперпараметара:

1. Структура: [3 5],[10 10],[2 16],[12 8 12]

Овако су изабране вредности како би се мрежа тестирала на различитим величинама скривених слојева.(некад мали први слој па велики други, велики и један и други слој итд)

2. Функција активације: “**logsig**”,”**tansig**”,”**poslin**”

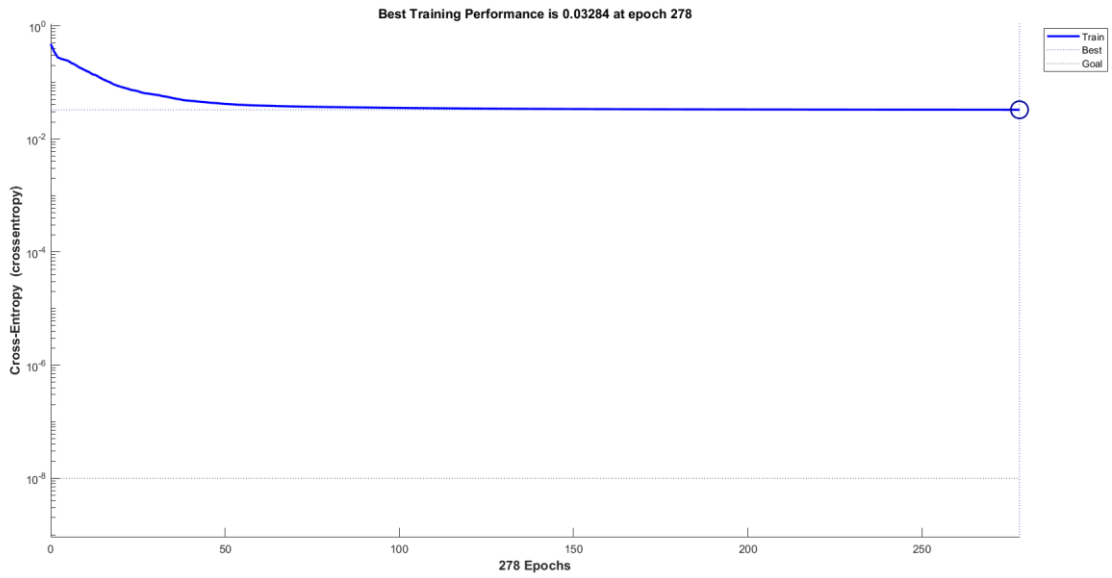
Изабране су ове функције да би се тестирало на две нелинеарне и једном линеарном функцији активације.

3. Коефицијент регуларизације: 0.01,0.02,0.05,0.1,0.02

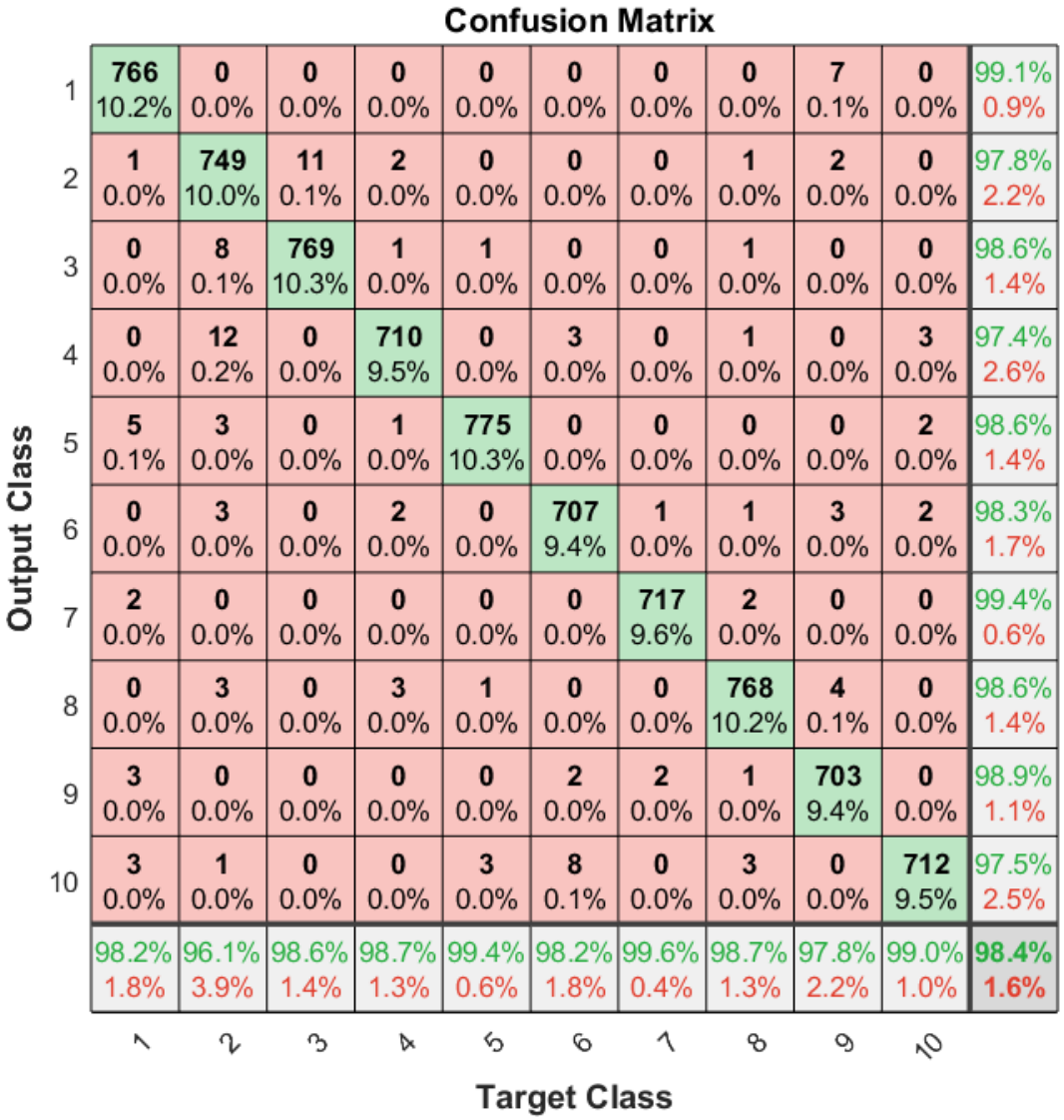
Изабране су ове вредности како би се тестирало са мањим и већим коефицијентима регуларизације.

После унакрсне валидације одабрани су хиперпараметри су [10 10] за структуру, “**logsig**” за функцију активације, 0.01 за коефицијент регуларизације.

Ови параметри се онда користе да се креира мрежа која се тренира на тренинг скупу.

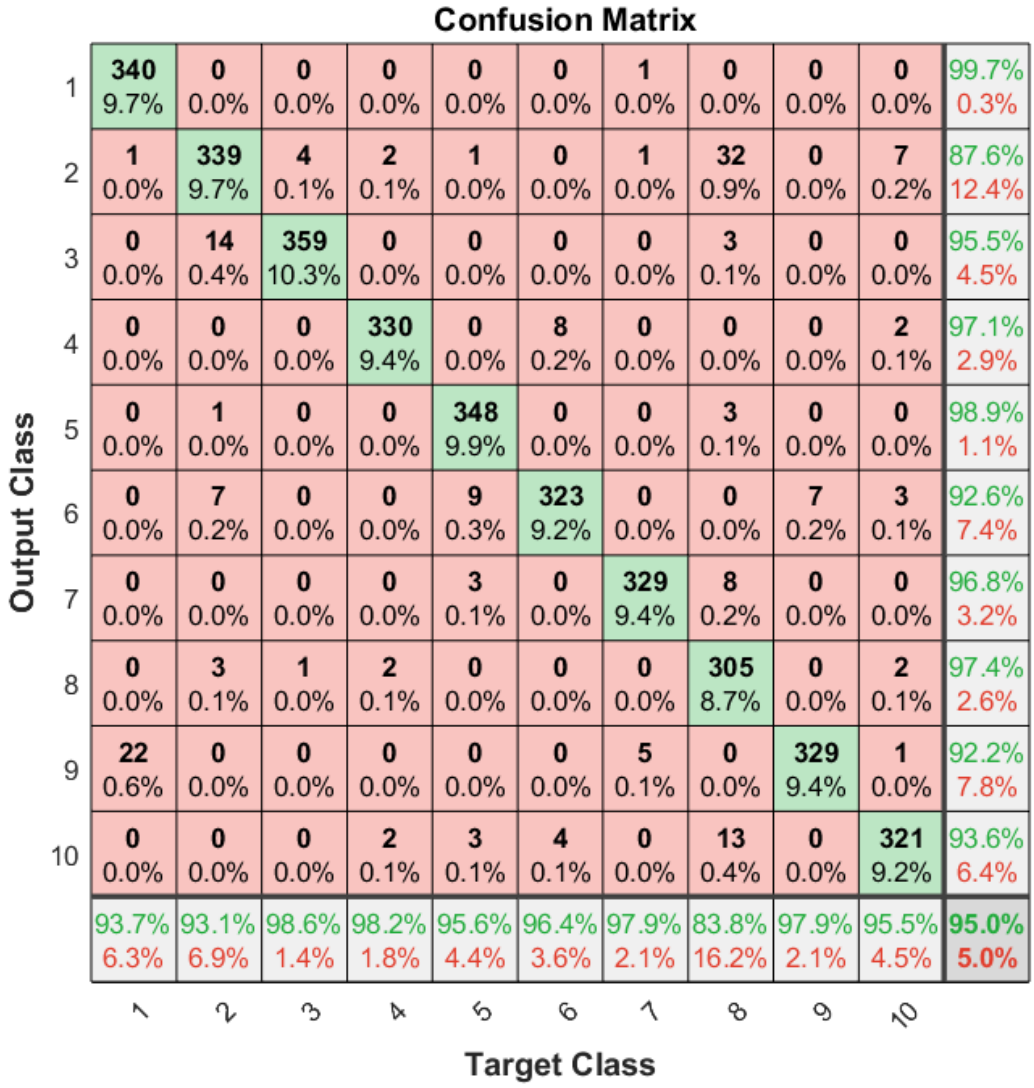


После тренирања цртамо конфузионе матрице.




*Zad3_Slika 2 Конфузиона Матрица Тренинг Скуп*

| Класа | Прецизност(P) | Осетљивост(R) |
|---|---|---|
| 0 | 0.99094 | 0.98205 |
| 1 | 0.97781 | 0.96149 |
| 2 | 0.9859 | 0.9859 |
| 3 | 0.97394 | 0.98748 |
| 4 | 0.98601 | 0.99359 |
| 5 | 0.98331 | 0.98194 |
| 6 | 0.99445 | 0.99583 |
| 7 | 0.98588 | 0.98715 |
| 8 | 0.98875 | 0.97775 |
| 9 | 0.97534, | 0.99026 |




*Zad3_Slika 3 Конфузиона матрица Тест скуп*

| Класа | Прецизност(P) | Осетљивост(R) |
|---|---|---|
| 0 | 0.99707 | 0.93664 |
| 1 | 0.87597 | 0.93132 |
| 2 | 0.95479 | 0.98626 |
| 3 | 0.97059 | 0.98214 |
| 4 | 0.98864 | 0.95604 |
| 5 | 0.9255 | 0.96418 |
| 6 | 0.96765 | 0.97917 |
| 7 | 0.97444 | 0.83791 |
| 8 | 0.92157 | 0.97917 |
| 9 | 0.93586 | 0.95536 |

Прецизност(P): Представља однос свих одбирака који су тачно смештени у класу и укупног броја одбирака смештеног у ту класу.

Осетљивост(R): Представља однос свих одбирака који су тачно смештени у класу и укупног броја одбирака те класе.