SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Diplomski studij računarstva

Laboratorijska vježba 5

Neuronske mreže

Klasifikacija vrste cvijeta Irisa pomoću neuronske mreže

Denis Lazor, DRB

Osijek, 2020.

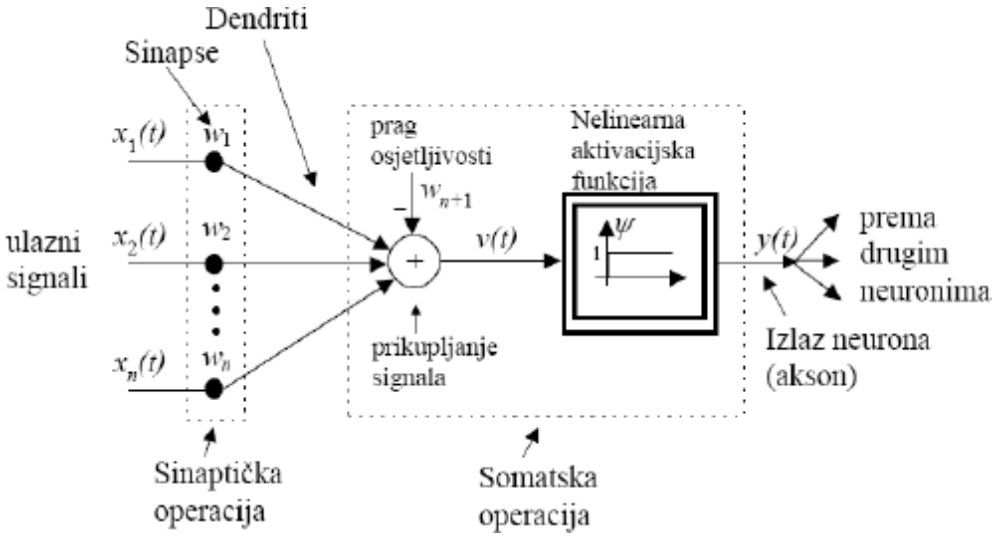
SADRŽAJ

1. UVOD

Zadatak 5. vježbe je bio inicijalizirati klasifikacijsku neuronsku mrežu sa određenim kombinacijama parametara te pomoću nje klasificirati ulazne podatke o dimenzijama cvijeta Irisa u jednu od 3 klase. Također trebalo je proučiti utjecaj određenih parametara na točnost klasifikacije.

2. NEURONSKE MREŽE

Neuronske mreže su sustavi za procesuiranje informacija koji su inspirirani biološkim živčanim sustavom kao što je mozak. Sastoji se od velikog broja međusobno povezanih procesnih elemenata tzv. neurona. Svaki neuron je sumirajući element povezan sa aktivacijskom funkcijom. Prvi model neurona koji je bio osmišljen još 1943. od strane McCollocha i Pittsa se zvao „perceptron“ i njegova jedina razlika od modernijih modela neurona je bila ta što je kao prijenosnu funkciju koristio diskontinuiranu step funkciju. Kasnije su se počee koristiti kontinuirane funkcije. Najveća revolucija se dogodila uvođenjem višeslojnih mreža i njihovih algoritama za učenje.



Slika 2.1. Shematski prikaz perceptrona

Iz slike 2.1 je vidljivo kako svaki neuron prikuplja signale od prethodnog sloja(pomnožene sa težinama), te uz dodatak praga osjetljivosti dolazi do prijenosne funkcije odnosno nelinearne aktivacijske funkcije. Izlaz iz te funkcije potom odlazi do svakog neurona u idućem sloju gdje se proces ponavlja. Neuroni se najčešće dijele na statičke i dinamičke, gdje statički neuroni ovise isključivo o trenutnim vrijednostima signala i težina, dok kod dinamičkih postoje određene povratne veze i promjenjive aktivacijske funkcije.

Da bi se neuronska mreža definirala, pored osnovnih parametara koji opisuju oblik i tip mreže, odnosno arhitekturu, potrebno je odrediti i algoritam učenja.
Proces učenja je u biti proces optimizacije pomoću algoritma gdje se pronalaze težine između neurona koje najbolje opisuju rješenje odnosno aproksimaciju problema. Proces učenja najčešće uključuje slijedeće korake:

- Dovođenje na ulaz neuronske mreže niz slučajeva (uzoraka) koje želimo naučiti raspoznavati.
- Odrediti pogrešku između dobivenog izlaza i željenog izlaza.
- Promijeniti težine da bi se izlaz bolje aproksimirao

Imamo 3 osnovna tipa učenja neuronskih mreža:

- Nadzirano učenje – učenje na temelju poznatih uzoraka i rezultata
- Učenje pojačavanjem – uključuje povratnu vezu iz okoline
- Nenadzirano učenje – učenje iz pravilnosti ulaznih podataka.

Najčešće se koristi nadzirano učenje, a najčešće korišteni algoritam učenja je sa povratnom propagacijom pogreške (eng. backpropagation).

Neuronske mreže zbog svoje sposobnosti učenja i aproksimacije se najčešće koristi za slijedeće primjene:

- Raspoznavanje znakova teksta (i analiza slike),
- Prepoznavanje govora,
- Adaptivno uklanjanje šuma,
- Predviđanje cijena dionica(financije),
- Medicinska dijagnostika.

Parametri neuronske mreže:

- hidden_layer_sizes – predstavlja broj neurona u skrivenim slojevima neuronske mreže kao ntorka, u gore navedenom primjeru (100,) je navedeno da mreža ima jedan skriveni sloj s 100 neurona.
- activation – predstavlja aktivacijsku funkciju neurona, te može biti odabrano:

  -> ‘identity’, f(x) = x.

  ->‘logistic’, f(x) = 1 / (1 + exp(-x)).

  ->‘tanh’, f(x) = tanh(x).

  -> ‘relu’, f(x) = max(0, x).
- solver – predstavlja metodu koja će se koristiti za optimizaciju težina neuronske mreže, te može biti odabrano:

  -> ‘lbfgs’, metoda iz porodice kvazi-Newton metoda.

  -> ‘sgd’, engl. Stochastic Gradient Descent.

  -> ‘adam’ novija metoda 'sgd'-a
- alpha – L2 regularizacijski parametar
- max_iter – maksimalni broj iteracija učenja neuronske mreže

3. PROGRAMSKI KOD

U postojeći kod dodana je funkcija za prikaz rezultata na ekranu. Funkcija je prikazana na slici 3.1

```
# Printing results -- Added by Denis Lazor
def print_results(Neurons,Layers,Alpha,Activation,Solver,MaxIterations):
    print("\nNeurons per layer: " + str(Neurons) + ", Layers: " + str(Layers) + ", Alpha: " + str(Alpha) +
          ", MaxIterations: " + str(MaxIterations) + ", Activation: " + str(Activation) + ", Solver: " + str(Solver))
```

Slika 3.1. Funkcija za prikaz rezultata

Kod prikazan na slici 3.2 predstavlja funkcije za dohvaćanje podataka iz interaktivnog prozora. Prvo se uzima naziv datoteke sa uzorcima kao string i izdvaja se dio sa veličinom i 'cast-a' se u varijablu tipa int. Nazivi aktivacijske funkcije i funkcije za učenje mreže se dohvaćaju preko *currentText()* funkcije(ranije *currentIndex()*).

```
ACTIVATION_F = self.comboActivation.currentText()  # --Altered by Denis Lazor // ...currentIndex() --> ...currentText()
SOLVER = self.comboSolver.currentText()  # -||-
MAX_ITER = int(self.tbxMaxIter.text())
LAYERS = self.tbxNeuroPerLayer.text().count(',') + 1  # Counting the number of layers --  Added by Denis Lazor
```

Slika 3.2. Prikaz funkcija za dohvaćanje podataka iz interaktivnog prozora

Kod prikazan na slici 3.3. predstavlja inicijalizaciju regresijske neuronske mreže.

```
    #Create neural network -- Added by Denis Lazor
    mlp = MLPClassifier(hidden_layer_sizes=N_PER_LAYER, activation=ACTIVATION_F, solver=SOLVER, alpha=ALPHA, max_iter=MAX_ITER)
```

Slika 3.3. Inicijalizacije neuronske mreže

Na slici 3.4. je prikazan poziv funkcije *print_results().*

```
#Printing results -- Added by Denis Lazor
print_results(N_PER_LAYER, LAYERS, ALPHA, ACTIVATION_F, SOLVER, MAX_ITER)
```

Slika 3.4. Poziv print_results() funkcije

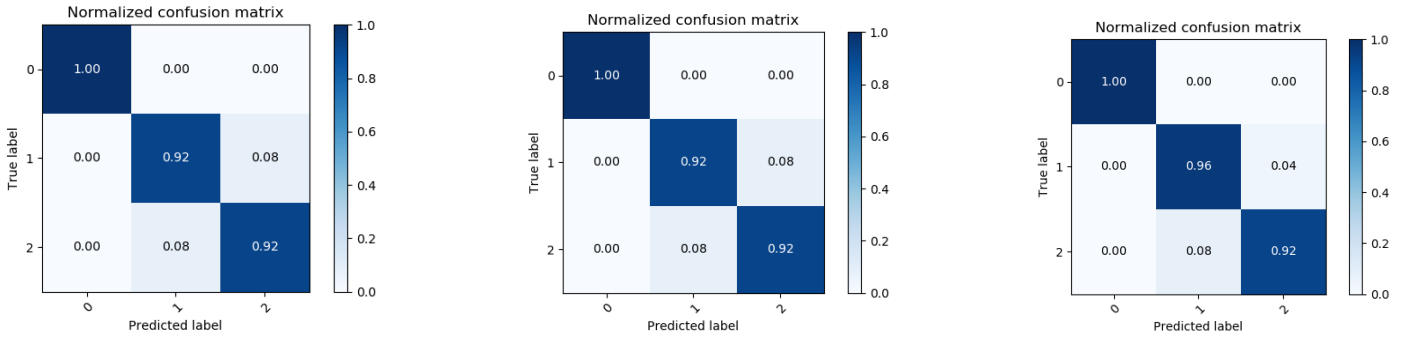
4. REZULTATI

Eksperimenti su provedeni na način da su se testirale različit kombinacije aktivacijskih funkcija i algoritama za učenje. Za svaku kombinaciju se mijenjao broj neurona u sloju i broj slojeva te se spremao prikaz konfuzijske matrice.

4.1. Ovisnost o broju neurona

Tablica 1 Adam – Identity kombinacija

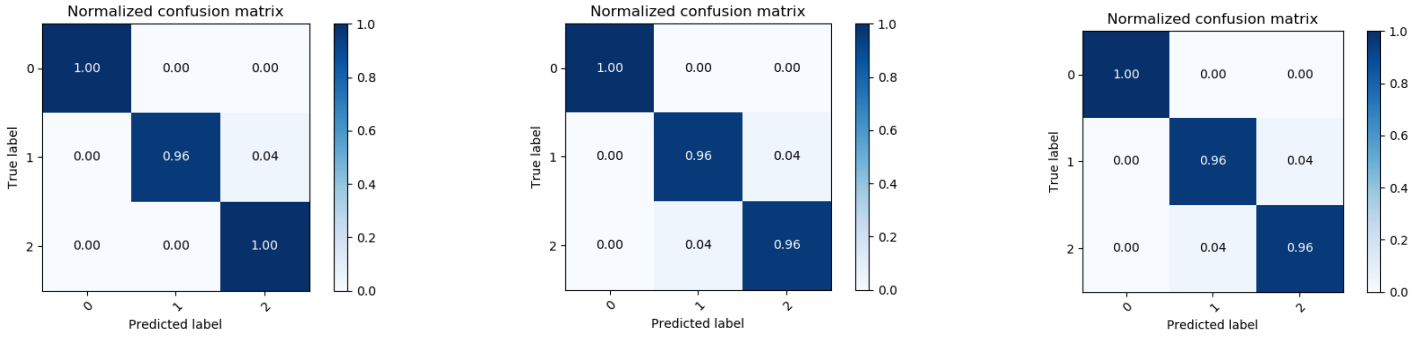
Učenje - aktivacija	Adam - Identity		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	94.67%	94.67%	96%



Slika 4.1.1. Adam – Identity kombinacija

Tablica 2 Adam – Logitsic kombinacija

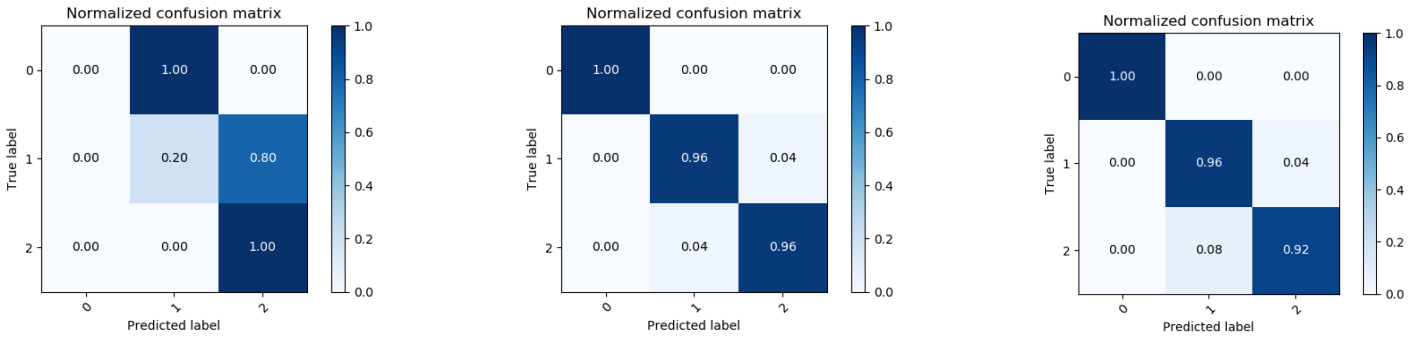
Učenje - aktivacija	Adam - Logistic		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	98.67%	97.3%	97.3%



Slika 4.1.2 Adam –Logistic kombinacija

Tablica 3 Adam – Relu kombinacija

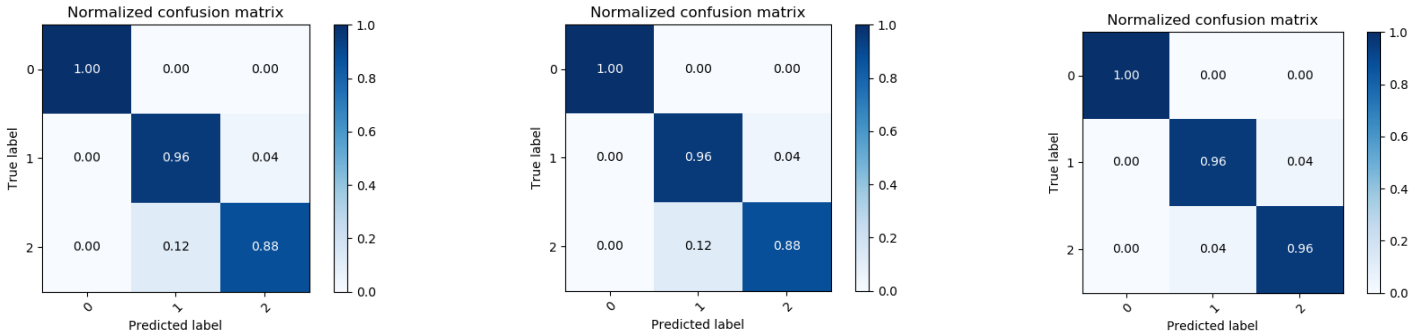
Učenje - aktivacija	Adam - Relu		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	40%	97.3%	96%



Slika 4.1.3.. Adam – Relu kombinacija

Tablica 4 Adam – Tanh kombinacija

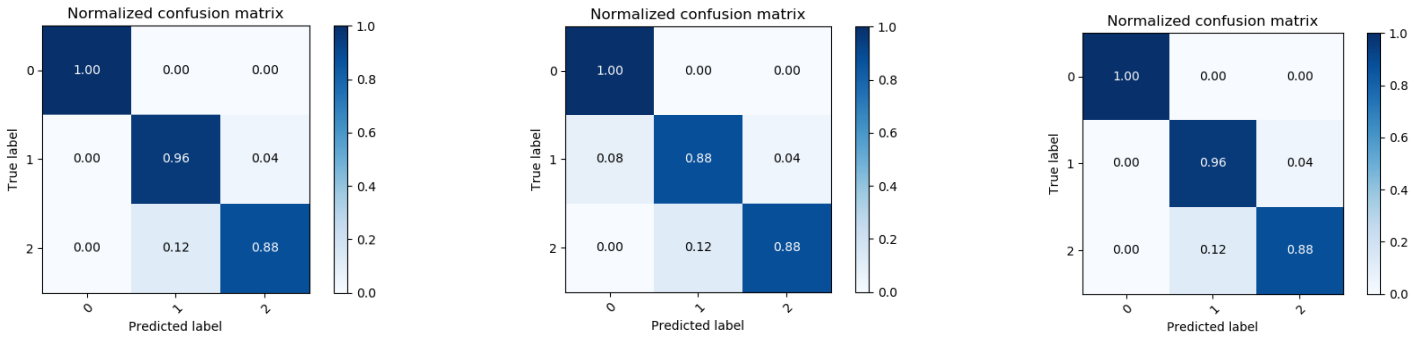
Učenje - aktivacija	Adam - Tanh		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	94.67%	94.67%	97.3%



Slika 4.1.4. Adam – Tanh kombinacija

Tablica 5 Lbfgs – Identity kombinacija

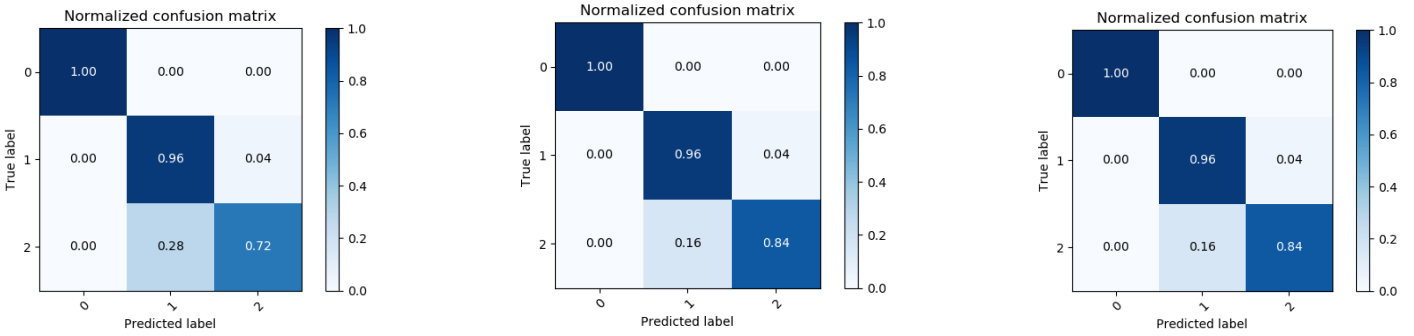
Učenje - aktivacija	Lbfgs- Identity		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	94.67%	94.67%	94.67%



Slika 4.1.5. Lbfgs – Identity kombinacija

Tablica 6 Lbfgs - Logistic kombinacija

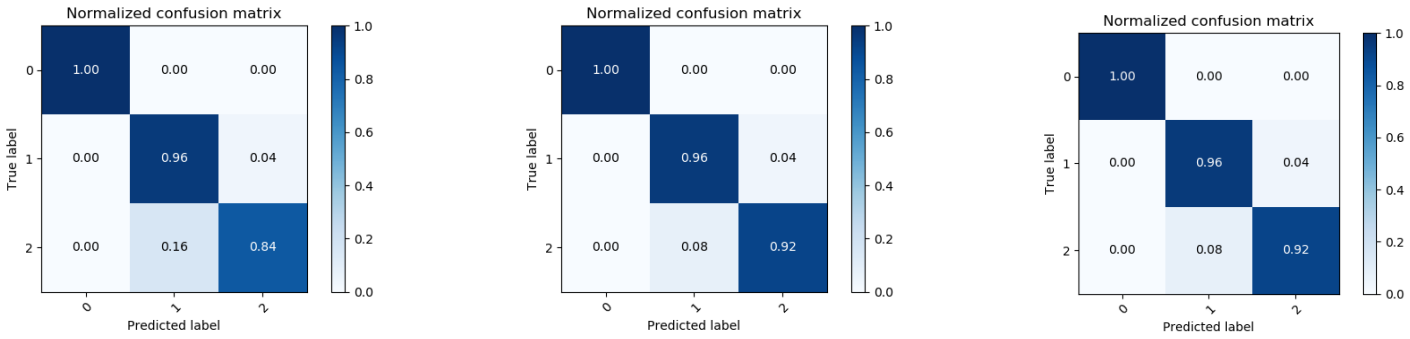
Učenje - aktivacija	Lbfgs - Logistic		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	89.33%	93.33%	93.33%



Slika 4.1.6. Lbfgs - Logistic kombinacija

Tablica 7 Lbfgs -Relu kombinacija

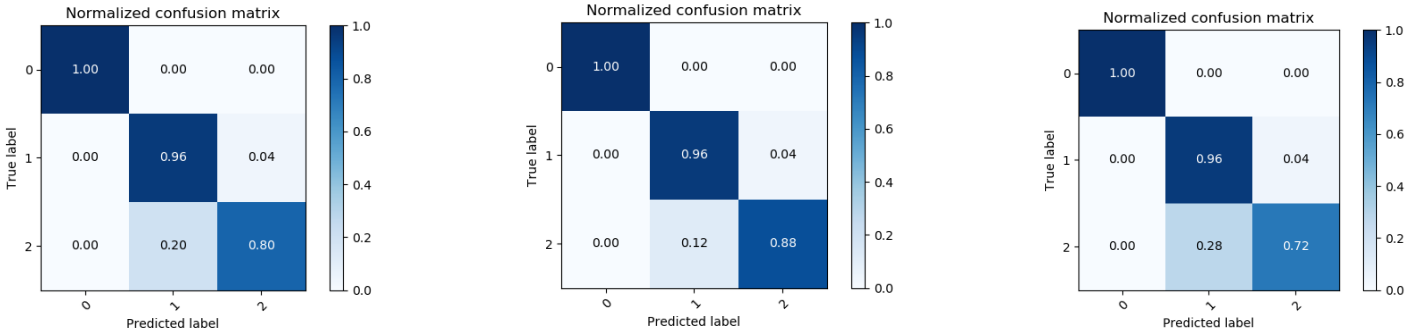
Učenje - aktivacija	Lbfgs -Relu		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	93.33%	96%	96%



Slika 4.1.7. Lbfgs -Relu kombinacija

Tablica 8 Lbfgs - Tanh kombinacija

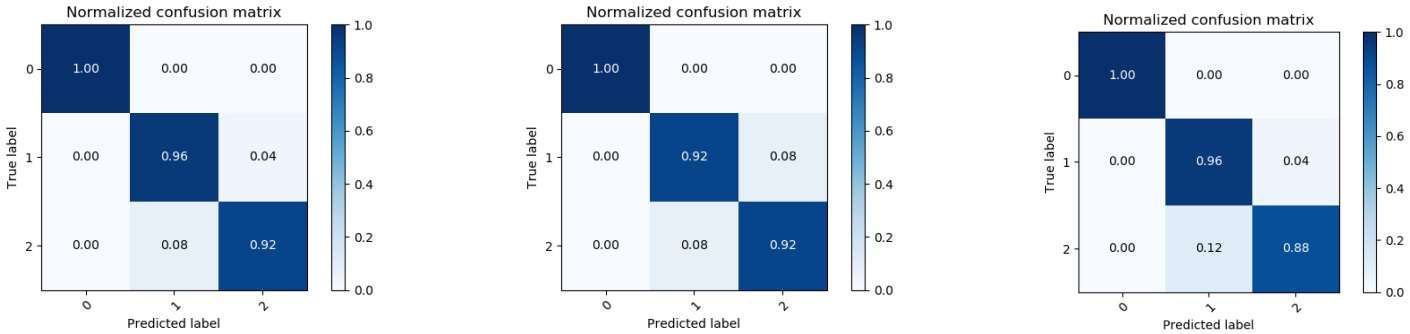
Učenje - aktivacija	Lbfgs - Tanh		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	91.33%	94.67%	89.33%



Slika 4.1.8. Lbfgs - Tanh kombinacija

Tablica 9 Sgd - Identity kombinacija

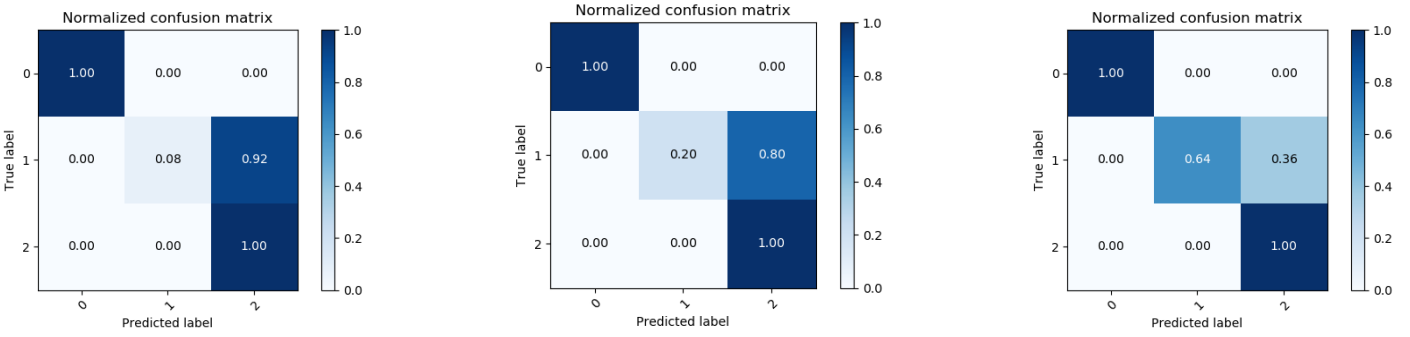
Učenje - aktivacija	Sgd - Identity		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	96%	94.67%	94.67%



Slika 4.1.9. Sgd - Identity kombinacija

Tablica 10 Sgd - Logitsic kombinacija

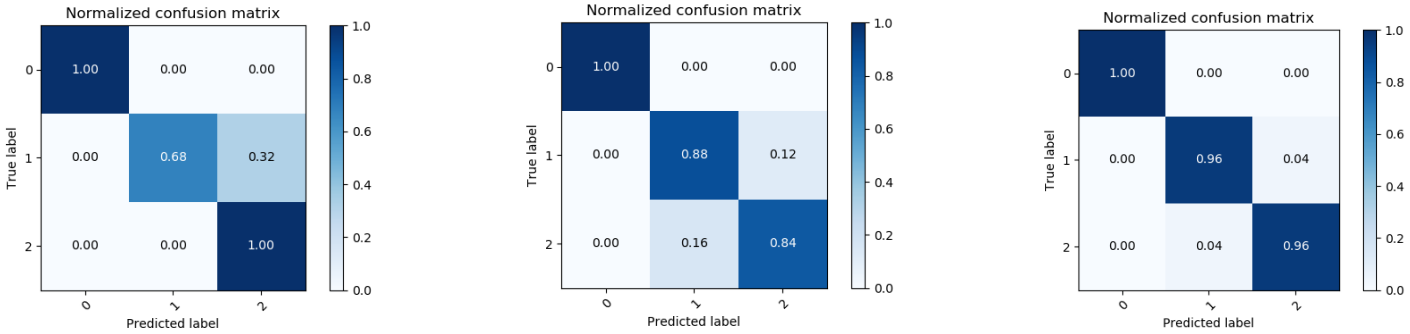
Učenje - aktivacija	Sgd - Logitsic		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	69.33%	73.33%	88%



Slika 4.1.10 Sgd - Logitsic kombinacija

Tablica 11 Sgd - Relu kombinacija

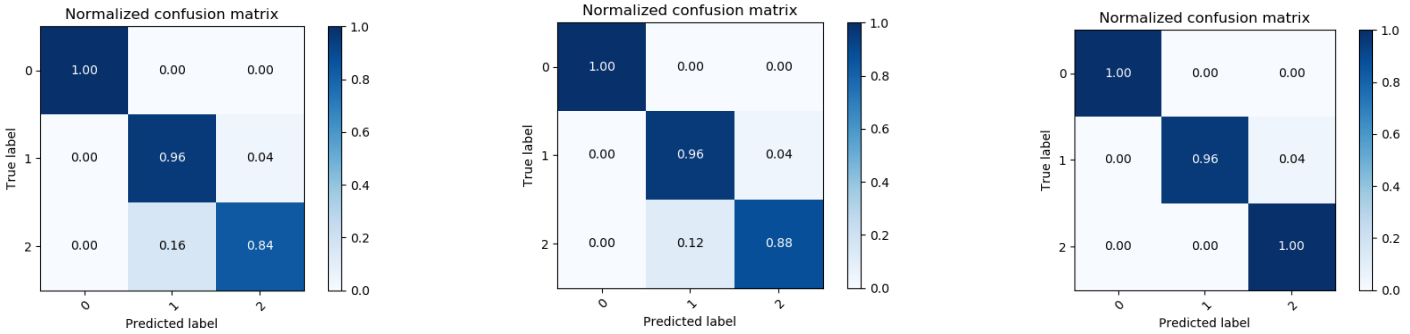
Učenje - aktivacija	Sgd - Relu		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	96 %	90.67%	97.33%



Slika 4.1.11. Sgd - Relu kombinacija

Tablica 12 Sgd - Tanh kombinacija

Učenje - aktivacija	Sgd - Tanh		
Broj slojeva	1		
Broj neurona	5	10	30
Preciznost	93.33%	94.67%	98.67%

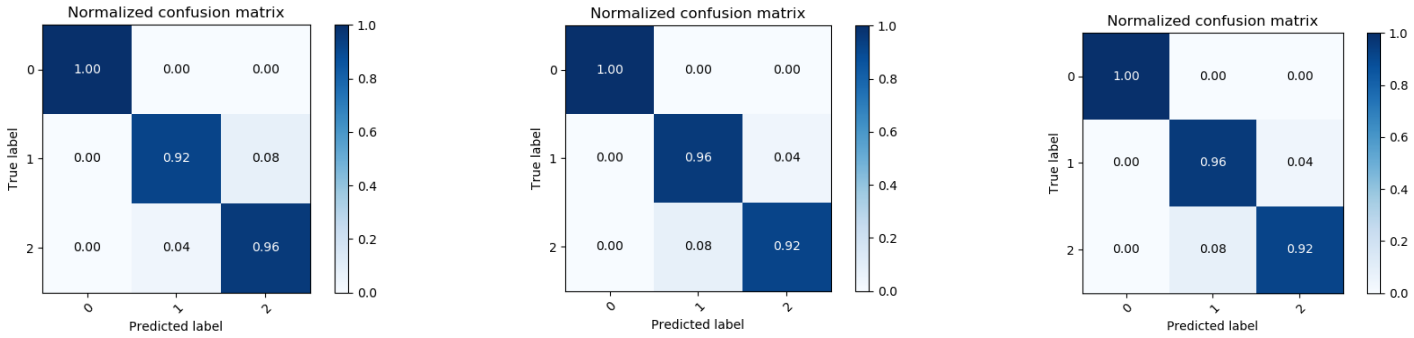


Slika 4.1.12. Sgd - Tanh kombinacija

4.2. Ovisnost o broju slojeva

Tablica 13 Adam – Identity kombinacija

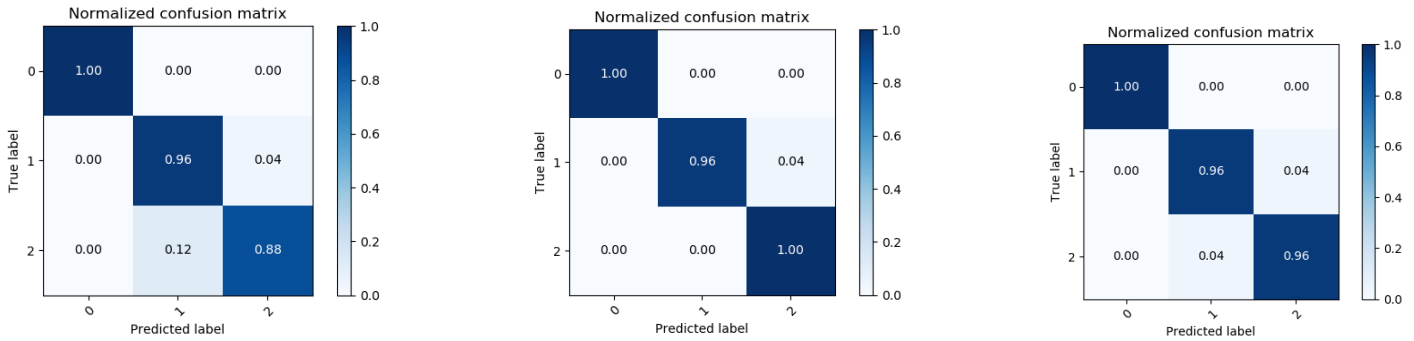
Učenje - aktivacija	Adam - Identity		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	96%	96%	96%



Slika 4.2.1. Adam – Identity kombinacija

Tablica 14 Adam – Logitsic kombinacija

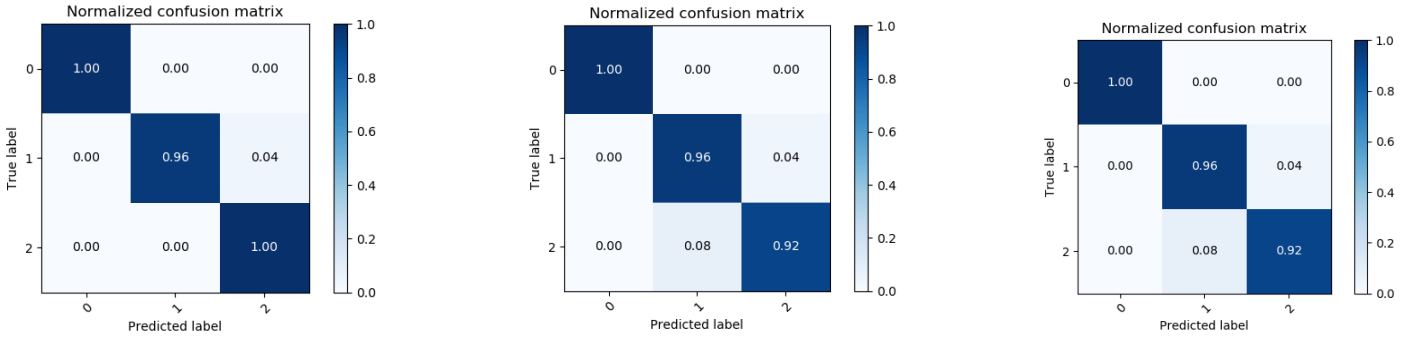
Učenje - aktivacija	Adam – Logitsic		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	94.67%	98.67%	97.3%



Slika 4.2.2 Adam –Logistic kombinacija

Tablica 15 Adam – Relu kombinacija

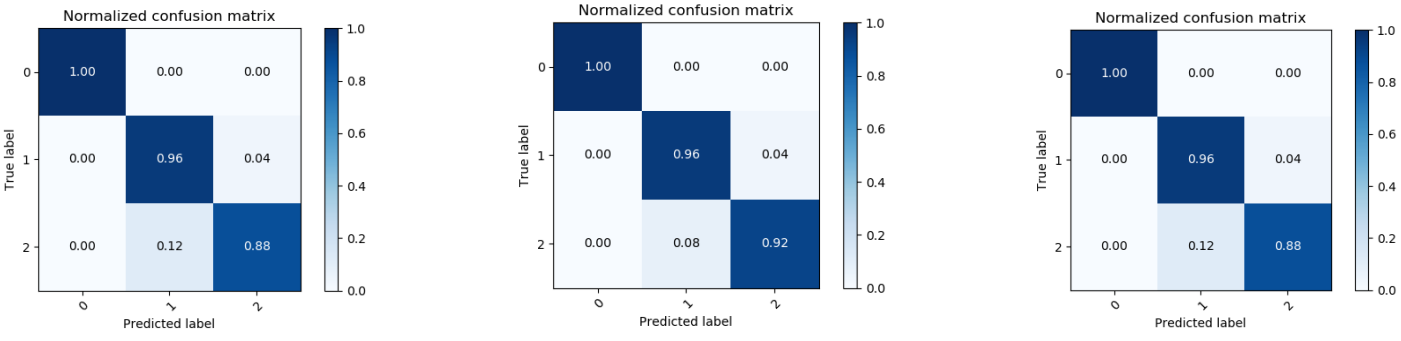
Učenje - aktivacija	Adam – Relu		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	98.67%	96%	96%



Slika 4.2.3. Adam – Relu kombinacija

Tablica 16 Adam – Tanh kombinacija

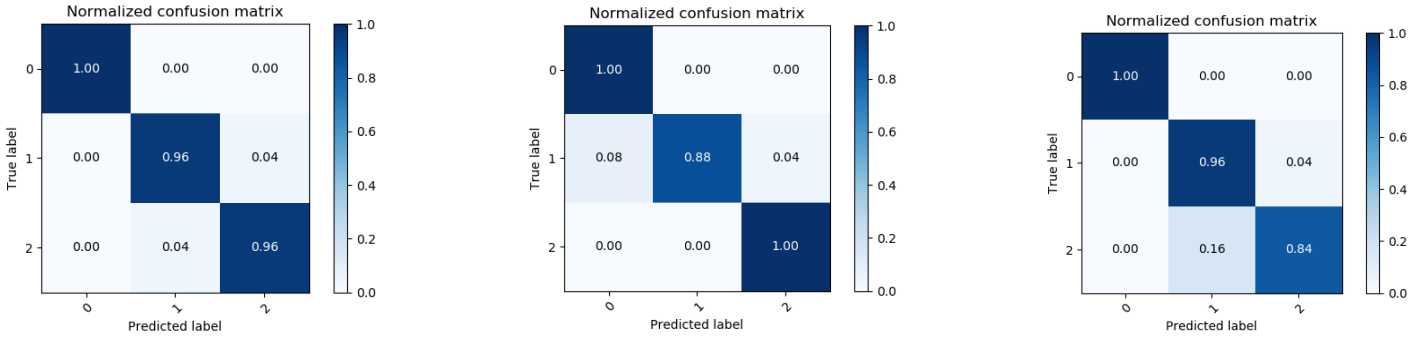
Učenje - aktivacija	Adam – Tanh		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	94.67%	96%	94.67%



Slika 4.2.4. Adam – Tanh kombinacija

Tablica 17 Lbfgs – Identity kombinacija

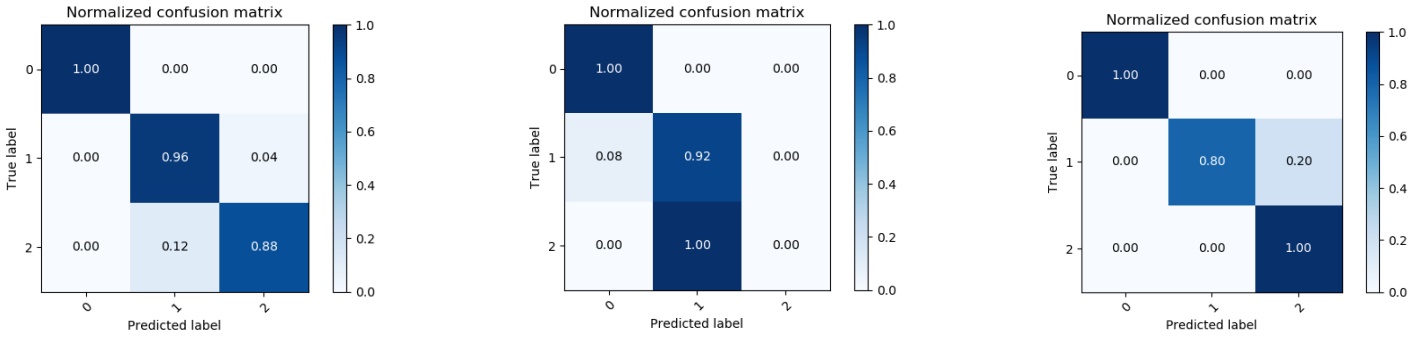
Učenje - aktivacija	Lbfgs – Identity		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	97.3%	96%	93.33%



Slika 4.2.5. Lbfgs – Identity kombinacija

Tablica 18 Lbfgs - Logistic kombinacija

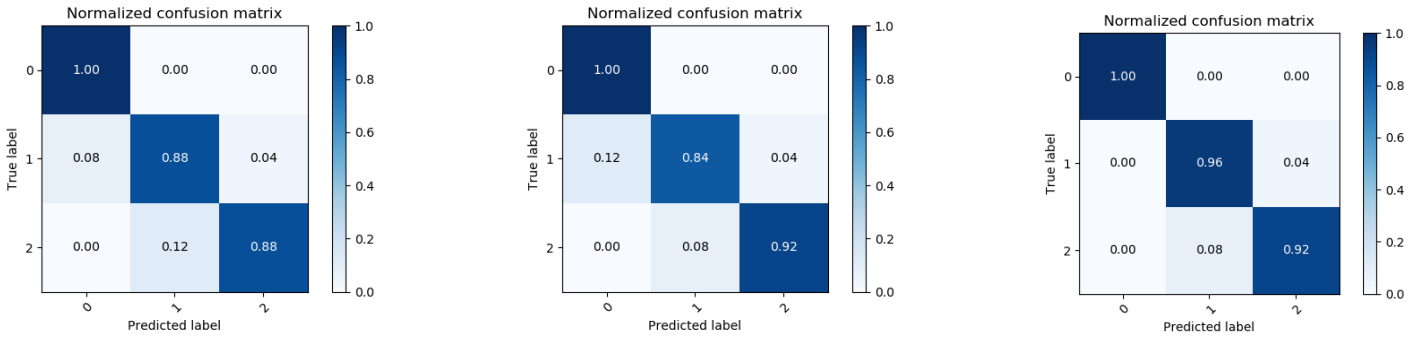
Učenje - aktivacija	Lbfgs - Logistic		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	94.67%	64%	93.33%



Slika 4.2.6. Lbfgs - Logistic kombinacija

Tablica 19 Lbfgs -Relu kombinacija

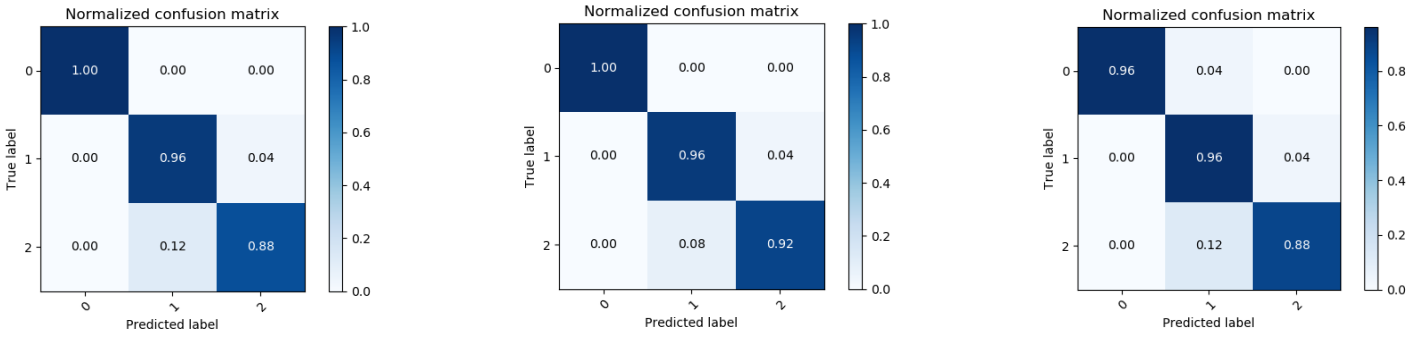
Učenje - aktivacija	Lbfgs -Relu		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	92%	92%	96%



Slika 4.2.7. Lbfgs -Relu kombinacija

Tablica 20 Lbfgs - Tanh kombinacija

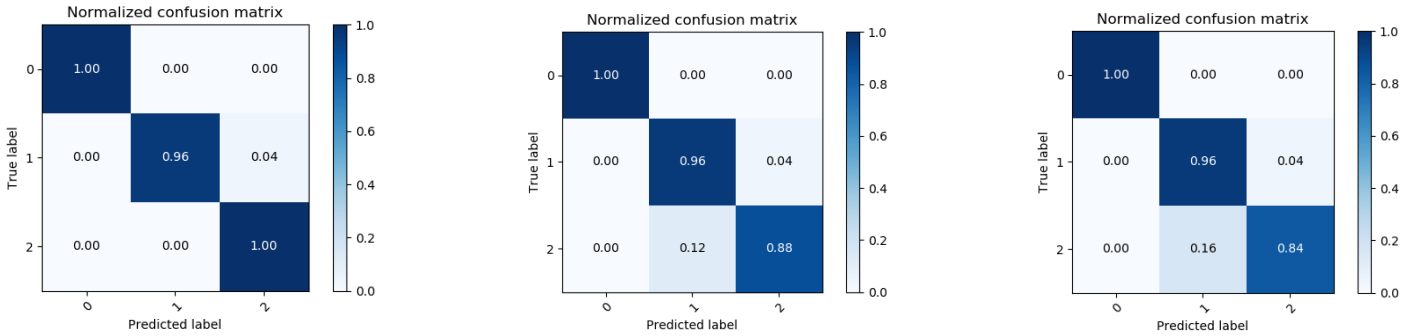
Učenje - aktivacija	Lbfgs - Tanh		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	94.67%	96%	93.33%



Slika 4.2.8. Lbfgs - Tanh kombinacija

Tablica 21 Sgd - Identity kombinacija

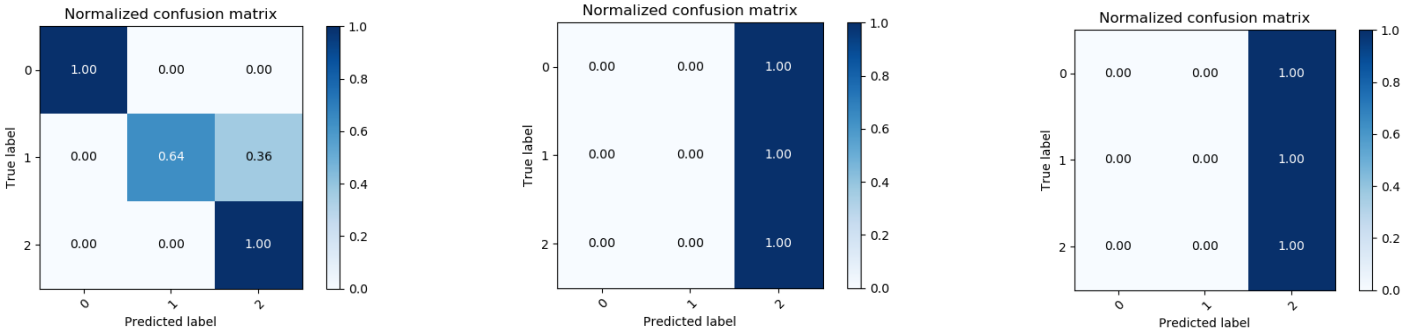
Učenje - aktivacija	Sgd - Identity		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	98.67%	94.67%	93.33%



Slika 4.2.9. Sgd - Identity kombinacija

Tablica 22 Sgd - Logitsic kombinacija

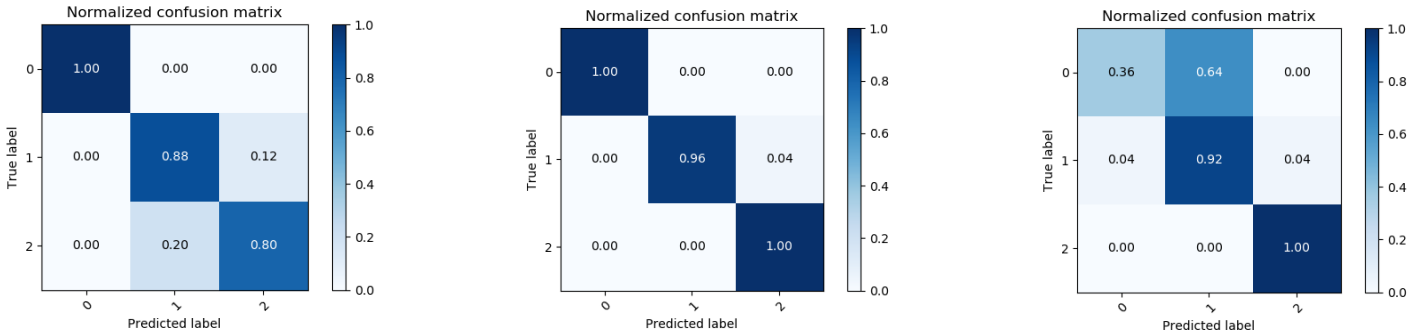
Učenje - aktivacija	Sgd - Logitsic		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	88 %	33.33%	33.33%



Slika 4.2.10 Sgd - Logitsic kombinacija

Tablica 23 Sgd - Relu kombinacija

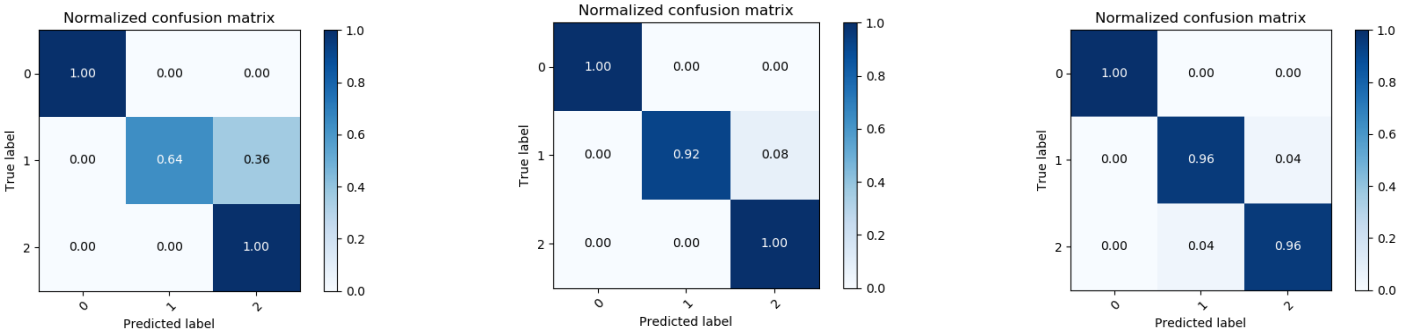
Učenje - aktivacija	Sgd - Relu		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	89.33%	98.67%	76%



Slika 4.2.11. Sgd - Relu kombinacija

Tablica 24 Sgd - Tanh kombinacija

Učenje - aktivacija	Sgd - Tanh		
Broj neurona	5	10, 5	30, 10, 5
Broj slojeva	1	2	3
Preciznost	88 %	97.3%	97.3%



Slika 4.2.12. Sgd - Tanh kombinacija

5. Zaključak

Iz grafova dobivenih provođenjem eksperimenata različitih kombinacija može se vidjeti da dodavanjem neurona u sloj ne dobivamo puno na preciznosti, zapravo negdje uopće ne dobivamo, a negdje jako malo. Dok uvođenje novih slojeva čak u nekim slučajevima smanjuje preciznost modela. Ali te su razlike u preciznostima između različitih kombinacija jako male i mijenjaju se tijekom višestrukog izvođenja istog eksperimenta, pa jedino što sa sigurnošću možemo utvrditi je da na ovakav jednostavan problem uvođenje novih neurona i slojeva nema veliki utjecaj i da će vremenom narušit preciznost modela zbog problema prilagođavanje podatcima. Najgore rezultate je dala kombinacija algoritma za učenje i aktivacijske funkcije *'SGD - Logistic'* dok je najbolje rezultate dala kombinacija *'Adam - Relu'*. *'SGD – Logistic'* kombinacija je za mreže sa više od jednog sloja sve cvjetove klasificirala uvijek u jednu klasu. Problem proizlazi iz aktivacijske funkcije koja mapira ulaz između 0 i 1 čime se automatski smanjuju gradijenti koji su potrebni algoritmu za učenje kako bi podesio težine. Za svaki novi sloj ti gradijenti su sve manji pošto se ulazi mapiraju u sve manji prostor i dolazi do pojave nestajaćeg gradijenta zbog čega ta kombinacija jedino radi za jednoslojnu mrežu. *'Adam'* i *'Lbfgs'* algoritmi nemaju toliki problem jer koriste i druge derivacije i na neki način koriste prijašnje gradijente tako da za ovaj jednostavna model rade. '*Relu*' nema problem s nestajućim gradijentom i on je na ovom modelu pokazao zašto je popularniji od trenutno korištenih aktivacijskih funkcija