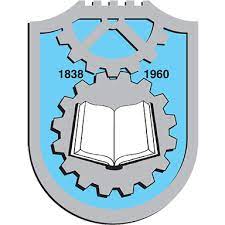
Факултет инжењерских наукаУниверзитет у Крагујевцу

****

VREŠTAČKA INTELIGENCIJA- Dermatology –

Student: Predmetni nastavnik:

Ilija Todorović 313/2023 Dr. Vesna Ranković

Kragujevac, 2023/24

Sadržaj

[1. Uvodna razmatranja 2](#_Toc155817977)

[2. Objašnjenje koda 4](#_Toc155817978)

[3. Korišćeni algoritmi 9](#_Toc155817979)

[4. Zaključak 10](#_Toc155817980)

[5. Literatura 11](#_Toc155817981)

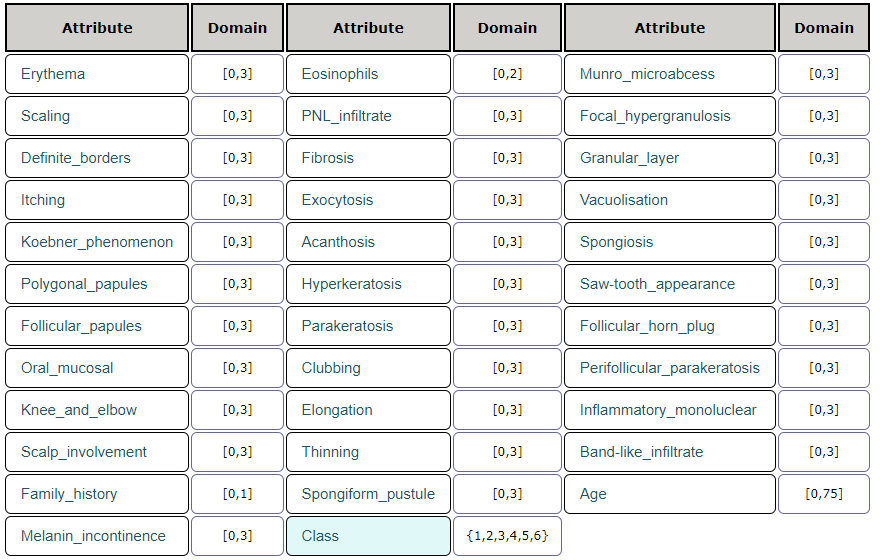
# Uvodna razmatranja

**Data Set**

U ovom seminarskom se koristi data set koji se odnosi na dermatologiju. Dijagnoza eritemato-skvamoznih bolesti prestavlja problem u dermatologiji. Pacijenti su prvo procenjeni sa 12 karakteristika a potom su uzeti uzorci sa koze za procenu 22 histološke karakteristike.

U skupu podataka koji je korišćen karakteristike porodične istorije su prestavljene sa vrednostima 0 i 1 tj. 1 ako je bilo koja od bolesti primećena u porodici prethodno i 0 ako nije. Tu se nalazi i starost pacijenta koja je predstavljena prosto brojem godina tog pacijenta. Svi drugi parametri (klinički i histološki) prestavljeni su vrednostima od 0 do 3 koje označavaju[1]:

* 0 - označava da parametar nije bio prisutan
* 1,2- označavaju relativne srednje vrednosti
* 3 – označava najveću moguću prisutnost



Slika 1 - Parametri koji se nalaze u data setu[1]

Ovaj data set je klasifikacionog tipa sa problemom iz stvarnog sveta. Ima šest klasa, 34 parametara(slika 1) i 366 instanci. Takođe ima nedostajuće vrednosti.

Analizom data seta ustanovljeno jeda su nedostajuće vrednosti godine pacijenta. S obzirom da je ovo problem iz medicine i da se u vecini medicinskih studija kada se predstavljaju rezultati uzimaju srednje vrednosti godina ideja jebila da se nedostajuće vrednosti zamene prosečnom starosti pacijenata iz data seta. Kod kako je to uradjeno biće prikazan u nastavku.

# Objašnjenje koda

**Biblioteke**

Za početak definisane su sve biblioteke(Dodatak A) koje su korišćene za rad na ovom projektu.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from keras.models import Sequential

import matplotlib.pyplot as plt

from keras.layers import Dense

import numpy as np

import keras.utils

import sklearn

from sklearn.metrics import confusion\_matrix

**Dodatak A**

**from Sklearn.model\_selection import train\_test \_split** se koristit da kasnije podeli data set na train i test delove.

**from sklearn.preprocessing import LabelEncoder** – će se koristiti pošto data set ima 6 klasa da te klase pretvori u binarne matrice

**from keras.models import Sequential** i **from keras.layers import Dense** – koriste se kako bi se definisala višeslojna mreža neurona

**import matplotlib.pyplot as plt** – koristi se kako bi se kasnije vizualizovali podaci iz dataseta

**import numpy as np –** koristi se za rad sa arrayevima

**import keras.utils** –pruža razne druge pomoćne funkcije za rad sa neuronskim mrežama poput to\_categorical itd..

**from sklearn.metrics import confusion\_matrix** – omogućava rad sa konfuzionom matricom

**Učitavanje data seta i umetanje nedostajućih vrednosti**

Kada su ispisane sve biblioteke koje će se koristiti učitava se i čita se data set(Dodatak B).

k = open("derm.txt", "r").read().split("\n")

**Dodatak B**

U narednom delu koda prolazi(Dodatak C) se kroz svaki red data seta i i gleda se da li su vrednosti numericke, ako jesu onda se pretvaraju u integere pošto su prethodno bili stringovi, a ako nisu numeričke (pošto su u data setu nedostajuće vrednosti upisane sa „?“) onda se pretvaraju u nulu. To je jako bitno zbog narednog koraka gde se računa prosek godina. Nakon toga se ulazi u petlju kako bi se pšroverilo na kojim mestima u data setu je broj godina 0 i na tom mestu se menja vrednost sa prosekom.

l = [[int(y) if y.isnumeric() else 0 for y in x.split(",")] for x in k]

prosek = sum([x[-2] for x in l]) / len(l)

for x in l:

  if x[-2] == 0:

    x[-2] = int(prosek)

**Dodatak C**

Potom se deli data set na ulaze i izlaze i onda se pretvara u float(Dodatak D).

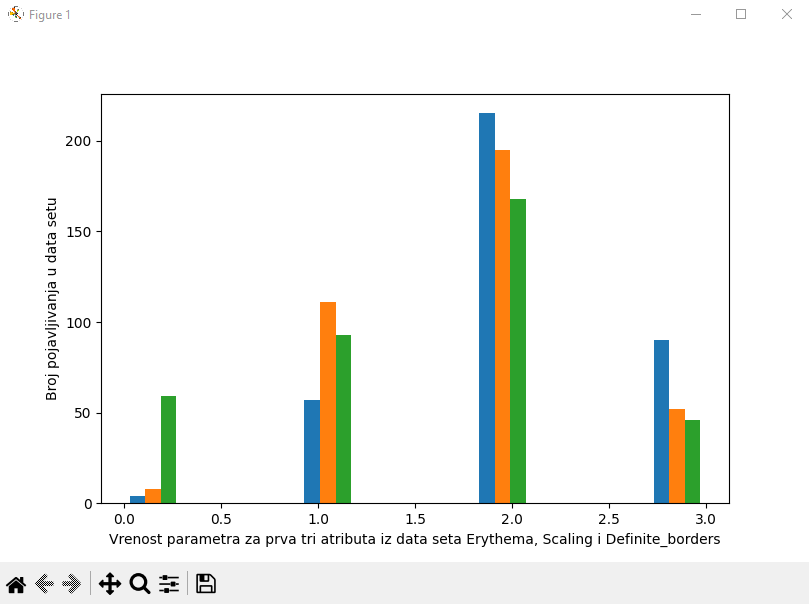
ulazi = np.array([x[:-1] for x in l] , dtype = np.float32)

izlazi = np.array([x[-1] for x in l] , dtype = np.float32)

**Dodatak D**

**Vizuelizacija**

Što se vizuelizacije tiče, prvobitna zamisao je bila da se napravi korelacija izmedju starosti pacijenata i nekog od parametra ali s obzirom da drugi parametri imaju vrednosi od nula do tri ovaj prikaznije bio najadekvatniji. Zbog toga je bila ideja da se prikažu vrednosti prvih nekoliko parametara iz data seta i broj pojavljivanja u data setu tako se može prikazati koliko pacijenata je u kom stadijumu(slika 2).



Slika 2 - Vizuelizacija za prva tri parametra iz data seta

plt.hist([ulazi[:,i] for i in range(3)])

plt.xlabel('Vrenost parametra za prva tri atributa iz data seta Erythema, Scaling i Definite\_borders')

plt.ylabel('Broj pojavljivanja u data setu')

plt.show()

**Dodatak E**

Broj parametara koji će se prikazati se veoma jednostavno može promeniti promenom broja u for petrlji u prvoj liniji. Ovde su stavljena tri radi lepšeg prikaza. Korišćena je plt.hist funkcija kako bi dijagram imao ovakaz oblik “pravougaonika” tj. podaci su predstavljeni u histogramu(Dodatak E).

x\_train ,x\_test ,y\_train ,y\_test  = sklearn.model\_selection.train\_test\_split(ulazi, izlazi, test\_size=0.2, train\_size=0.8, random\_state=42, shuffle=True, stratify=None)

**Dodatak F**

Zatim je potrebno podeliti data set na trening I test skup. U ovom slučaju taj odnos je bio 80/20. I primenjeno je shuffle-ovanje radi boljeg obučavanja neuronske mreže(Dodatak F).

Nakon ovog koraka potrebno je enkodirati izlaze tj. pretvoriti ih u binarne matrice korišćenjem funkcije LableEncoder()(Dodatak G).

one\_hot = LabelEncoder()

one\_hot.fit(y\_train)

one\_hot\_1 = one\_hot.transform(y\_train)

one\_hot\_2 = keras.utils.to\_categorical(one\_hot\_1)

test\_one\_hot = LabelEncoder()

test\_one\_hot.fit(y\_test)

test\_one\_hot\_1 = test\_one\_hot.transform(y\_test)

test\_one\_hot\_2 = keras.utils.to\_categorical(test\_one\_hot\_1)

**Dodatak G**

Potom se pristupa definisanju I optimizaciji višeslojne neuronske mreže. Ova mreža će imati 5 neurona u prvom sloju sa 34 ulaza I aktivaciona funkcija je “relu”. Takođe imaće jedan skriveni sloj sa n neurona koji takodje koristi “relu” kao aktivaionu funkciju. Relu je funkcija koja ide od nula pa “naviše” I zato je smatrana pogodnom s obzirom da u data setu ne postoje vrednosti manje od 0. U izlaznom sloju se nalazi 6 neurona I aktivationa funkcija je “softmax”. Definisane su po tri vrednosti na osnovu kojih će se vršiti optimizacija parametara. To su batch size, learning rate i broj neurona u skrivenom sloju. Potrebno je iz biblioteke keras.optimizers pozvati Adam optimizer kako learning rate ne bi bio defoltni, već kako bi postojala mogućnost izmene(Dodatak H).

bs = [8,16,32]

lr = [0.01, 0.001, 0.0003]

br\_neu = [10,15,30]

sve = []

for b in bs:

  for l in lr:

    for n in br\_neu:

      adam = keras.optimizers.Adam(learning\_rate=l)

      model = Sequential()

      model.add(Dense(5, input\_shape = (34,) , activation="relu"))

      model.add(Dense(n, activation="relu"))

      model.add(Dense(6, activation = "softmax"))

**Dodatak H**

Potom se model kompajlira u istoj for petlji I za funkciju gubitka se koristi “categorical\_crossentropy” zato što se pojavljuje višeklasna klasifikacija. Za optimizator se koristi “adam” I kao metrika se računa tačnost(Dodatak I).

model.compile(loss = "categorical\_crossentropy" , optimizer=adam , metrics = ["accuracy"])

      model.fit(x\_train , one\_hot\_2 , epochs = 100 , batch\_size = b , verbose=0)

**Dodatak I**

Broj epoha je postavljen na 100 I batch\_size na b(čije su vrednosti definisane u Dodatku H).

predictions = model.predict(x\_test)

predicted\_labels = np.argmax(predictions, axis=1)

true\_labels\_one\_hot = keras.utils.to\_categorical(test\_one\_hot\_1)

accuracy = np.sum(predicted\_labels == np.argmax(true\_labels\_one\_hot, axis=1)) / len(y\_test)

**Dodatak J**

U narednom koraku se radi evaluacija tačnosti modela na testnom skupu podataka. Za početak se traže predikcije na testnom skupu x\_test. Potom se koristi numpy funkcija argmax koja će vratiti index najveće vrednosti u svakom redu matrice predictions. Dakle promenljiva predicted\_labels će sadržati predviđene klase za svaki uzoraku testnom skupu. U sledećem redu se stvarne vrednosti pretvaraju u binarne matrice. I u poslednjoj liniji se računa tačnost modela gde se uporedjuju nizovi “predicted\_labels” I “true\_labels\_one\_hot”. Ovim se vraćaju vrednosti “true” ili “false” I uz pomoć np.sum se broje “true” vrednosti. Potom se ta suma deli sa dužinom testa kako bi se dobila vrednost za tačnost. Za kraj se štampa tačnost kako bi se mogli uporediti rezultati(Dodatak J).

Pošto postoje 3 for petlje sa po tri različita parametra optimizacije to znači da će se neuronska mreža trenirati čak 27 puta (tri na treći). Zato je potrebno štampati posle svakog treniranja koji parametri su korišćeni I rezultat koji su dali(Dodatak K). Takodje svaki rezultat je pakovan u prazan niz “sve” koji je definisan u Dodatku H.

sve.append(((b,l,n) , accuracy))

      print(f"bs={b} ,lr={l} , br\_neu={n} ,accuracy={accuracy}")

**Dodatak K**

Nakon toga da bi se ustanovilo koji parametric su najbolji potrebno je štampati ih u redu po metrici tačnosti(Dodatak M).

print(f"sortirano po acc")

print(sorted(sve , key = lambda x: x[-1]))

**Dodatak M**

Na kraju štampa se konfuziona matrica(Dodatak N)(Slika 4).

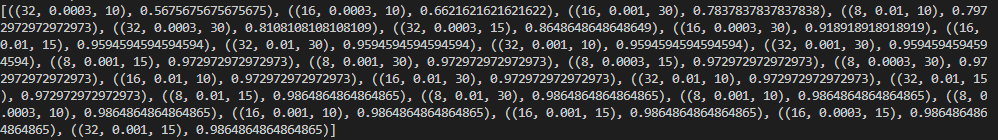
conf\_matrix = confusion\_matrix(np.argmax(true\_labels\_one\_hot, axis=1), predicted\_labels)

print("Confusion Matrix:")

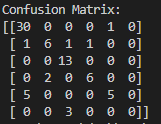
print(conf\_matrix)

**Dodatak N**

Dobijeni rezultati obučavanja su sledeći(slika 3):



Slika 3 - Rezultati obučavanje mreže



Slika 4 - Konfuziona matrica

# Korišćeni algoritmi

U mašinskom učenju, jedan od ključnih algoritama za obuku veštačkih neuronskih mreža je algoritam propagacije unazad (backpropagation). Ovaj algoritam se često koristi nakon faze propagacije unapred, gde se ulazni podaci šalju kroz mrežu kako bi se generisale predikcije. Kada imamo predikcije, koristimo neku funkciju gubitka kako bismo izračunali grešku između predviđenih vrednosti i stvarnih vrednosti. Funkcija model.compile poziva propagaciju unazad gde se izračunavaju gradijenti nakon čega se ažuriraju težine neuronske mreže. Taj process se ponavlja kroz vise iteracija tj. epoha dok se ne dodje do prihvatljive vrednosti funkcije gubitka I ostalih metrika mreže. Ovaj iterativni ciklus igra ključnu ulogu u optimizaciji performansi neuronskih mreža u mašinskom učenju.

Prednosti ove metode:

* Efikasno učenje težina
* Automatsko izračunavanje gradijenta(TensorFlow I PyThorch)
* Može se koristiti u raznolikim vrstama problema
* Efikasno minimizuje funkciju gubitka

Mane ove metode:

* Potreba za velikim skupom podataka
* Problemi sa nestajućim ili eksplodirajućim gradijentima

# Zaključak

Rezultati su bili poprilično iznenadjujući s obzirom da je tačnost dostigla veoma visoke nivoe za neke parametre. Parametri koji su ostvarili najbolje rezultate su bili batch size=32, learning rate=0.001 I broj neurona u skrivenom sloju =15. Prvobitno je primer uradjen sa 150 epoha ali pošto je model veoma brzo dolazio do zadovoljavajuće tačnosti smatrano je da je to previše I radi uštede vremena broj epoha je smanjen na 100. Pretpostavka je da ako bi se povećao broj epoha I vratio na 150 da bi rezultati bili još bolji I možda dostigli ideal. Algoritam sa propagacijom mreže unazad je generalno algoritam koji se može koristiti za razne vrste problema I svoju primenu je našao I u ovom primeru I pokazao veoma korisnim. Nedostajuće vrednosti su zamenjene srednjom vrednosti jer se to činilo kao najbolja opcija. U kodu kako je napisan trenutno konfuziona matrica je odstampana za poslednji pokušaj treniranja I njeni rezultati bi trebalo da se poklapaju sa rezultatima treniranja, s obzirom da je tačnost za poslednje treniranje bila oko 0.8.

# Literatura

[1] – Dataset - <https://sci2s.ugr.es/keel/dataset.php?cod=60> (pristpljeno 03.01.2024.)

[2] – Moodle portal, Predavanja iz predmeta Veštačka inteligencija