Klasifikacija Breast Cancer Winsconsin skupa podataka korišćenjem potpuno povezane neuronske mreže

1.Uvod

Rak dojke je jedan od vodećih uzroka smrti kod žena. U isto vreme, ovaj tip karcinoma je u velikom procentu izlečiv ukoliko je dijagnostikovan u ranijim stadijumima. Prema istraživanjima Svetske zdravstvene organizacije(World Health Organization), godišnje od raka dojke oboli 1.2 miliona žena. Rana i precizna dijagnostika su zahtevni, ali bitni zadaci za smanjenje smrtnosti od raka dojke. Iz navedenih razloga, za klasifikaciju benignih i malignih tumora se koriste algoritmi zasnovani na mašinskom učenju, koji omogućavaju veliku preciznost dijagnostike.
U okviru ovog rada prikazan je model klasifikacije podataka breast cancer Winsconsin korišćenjem potpuno povezane neuronske mreže, implementirani pomoću python biblioteka za dubinsko učenje - tensorflow i keras.

2.Breast cancer Winsconsin (Diagnostic) skup podataka

2.1 Opis

Skup podataka je preuzet sa UCI Machine Learning Repository Breast Cancer Wisconsin (Diagnostic) Data Set. Skup podataka je kreiran od stane Dr. William H. Wolberg na Winsconisn Univerzitetu. Sastoji se od 569 kliničkih instanci, od čega je 357 (62,7%) benignih i 212 (34,5%) malignih dijagnoza. Svaka instanca se sastoji od 32 atributa. Prvi atribut predstavlja ID broj pacijenta, drugi atribut izlaznu binarnu klasu koja indukuje benignu ili malignu dijagnozu. Ostalih 30 atributa su izračunati analizom digitalizovanih slika dobijenih aspiracionom biopsijom tankom iglom (Fine needle aspiration - FNA) tkiva dojke.
Za svako jedro ćelije opisuje se deset osobina: *radijus*(srednja vrednost rastojanja od centra do oboda oblasti), *tekstura* (standardna devijacija vrednosti sive skale), *granica oblasti, površina, varijacija veličine u radijusu, kompaktnost, konkavnost, konkavne tačke, simetrija i fraktalna dimenzija.* Za svaku od navedenih osobina se koriste tri realne vrednosti koji ih opisuju: srednja i maksimalna vrednosti, i standardna greska. Detaljan pregled atributa se nalazi u tabeli 1.1.

NAZIV	*RASPON*
id	id pacijenta
diagnosis	B, M
radius_mean	6.98 - 28.11
texture_mean	9.31-39.28
perimiter_mean	43.79-188.50
area_mean	143.50-2501.0
smoothness_mean	0.05-0.16
compactness mean	0.02-0.25
concavity mean	0.00 -0.43
concave_points_ mean	0.00-0.20
symmetry_mean	0.11-0.30
fractal_dimension_ mean	0.05-0.10
radius_se	0.11-2.87
texture_se	0.36-4.88
perimiter_se	0.76-21.98
area_se	6.80-542.2

NAZIV	RASPON
smoothness_se	0.001-0.03
compactness_se	0.002-0.14
concavity_se	0.0-0.3
concave_points_se	0.0-0.05
symmetry_se	0.008-0.08
fractal_dimension_se	0.0-0.03
radius_worst	7.93-36.04
texture_worst	12.02-49.54
perimeter_worst	50.41-251.2
area_worst	185.2-4254
smoothness_worst	0.07-0.223
compactness_worst	0.03-1.058
concavity_worst	0.0-1.252
concave_points_ worst	0.0-0.29
symmetry_worst	0.16-0.66
fractal_dimension_ worst	0.06-0.20

1.1 Detaljniji opis podataka

2.2 Preprocesiranje podataka

Pandas modul nam omogućava lak rad sa višedimenzionalnim podacima. Podatke učitavamo iz .csv fajla pomoću read_csv metoda. Atribut ID izbacujemo iz podataka pošto prestavlja jedinstvenu identifikaciju pacijenta, pa nije od značaja za neuronsku mrežu. U tabeli 1.1 se moze primetiti da se vrednosti atributa ne nalaze u istom rasponu, pa ih je poželjno standardizovati, da bi se ubrzalo treniranje neuronske mreze. Koristimo StandardScaler, dostupan u okviru Preprocessing modula. Standardizacija se vrši po formuli:

$$z = \frac{X - \mu}{\sigma}$$

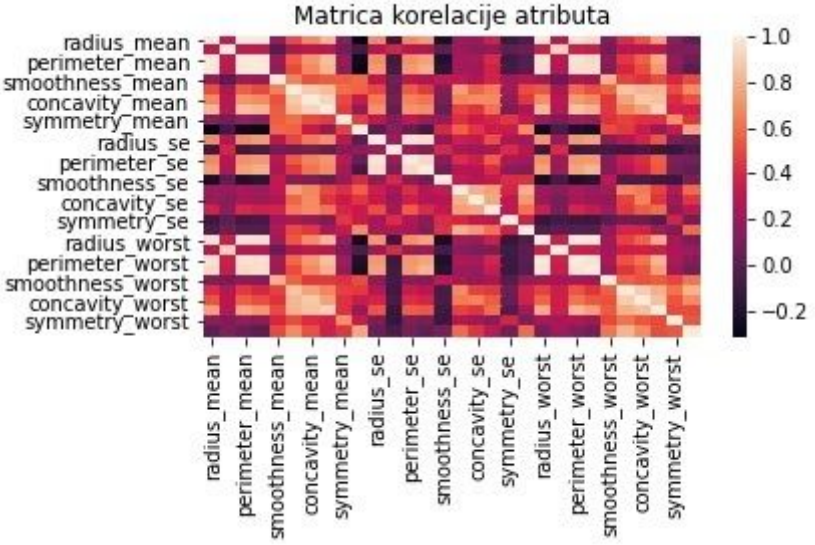
gde je X - vrednost atributa,
μ -srednja vrednost
σ - standardna devijacija.

Jako je bitno da se prilikom standardizacije za računanje srednje vrednosti i standardne devijacije koriste samo vrednosti atributa instanci koje pripadaju trening skupu, da ne bi došlo do kompromitovanja test skupa. Nakon standardizacije, vrednosti svih atributa se nalaze u intervalu [-1, 1].

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
data_train = scaler.fit_transform(data_train)
data_test = scaler.transform(data_test)
```

Dakle, nad trening podacima računamo vrednosti potrebne za izračunavanje formule, i zatim ih transformišemo(fit_transform metod), dok trening podatke samo transformišemo(transform metod).



Na slici 1.2 je prikazana matrica korelacije atributa zadatog skupa.

Uočljivo je da postoji jaka korelacija između određenih atributa. Na primer, korelacija izmedju elemenata perimeter-worst i radius-mean je 0.97. Kako je dobijena vrednost blizu jedinice, izmedju njih postoji približno linearna zavisnost, pa nam oba atributa daju sličnu informaciju. Takođe, u okviru samog opisa podataka je pomenuto da je prilikom uzimanja uzoraka analizirano deset osobina, za svaku po tri vrednosti (mean, se, worst), pa je očekivano da se medju atributima javlja redundantnost.

Uz pomoć Analize glavnih elemenata(Principal component analysis) ćemo pokušati da smanjimo broj atributa. Analiza glavnih komponenti (Principal Component Analysis) koristi se kao alat u ispitivanju podataka i pri pravljenju modela. Ovom statističkom procedurom se, korišćenjem linearnih transformacija, početni skup atributa transformiše u novi skup nekorelisanih atributa. Pritom se indentifikuju oni atributi koji su najbitniji za podatke tj. koji opisuju najviše varijabilnosti, i oni koji nisu toliko bitni. Ideja je da se transformišu atributi tako da se prvom glavnom komponentom opiše najveći deo disperzije podataka. Da to bude pravac, jedinični vektor u n-dimenzionom prostoru po kom se najviše prostiru podaci. Svaki sledeći pravac tražimo tako da bude normalan na prethodne i da ima drugu najveću moguću disperziju, treću, itd. Takvim postupkom dobijamo jednu ortonormiranu bazu u prostoru podataka. Ovim se otkriva struktura u podacima kojom se najbolje opisuje varijabilnost.

U okviru modula decomposition, biblioteke scikit-learn postoji metod koji vrši analizu glavnih elementa. Kao argument se može proslediti željeni broj atributa.

```
from sklearn.decomposition import PCA
pca = PCA()
data_pca = pca.fit_transform(data_train)
```

Bitno je da se podaci standardizuju pre upotrebe pca algoritma.

Rezultati analize se nalaze u tabeli 1.3

1	0.449	11	0.964	21	0.997
2	0.633	12	0.972	22	0.998
3	0.728	13	0.980	23	0.998
4	0.794	14	0.984	24	0.998
5	0.848	15	0.987	25	0.999
6	0.890	16	0.990	26	0.999
7	0.912	17	0.992	27	0.999
8	0.923	18	0.994	28	1.000
9	0.943	19	0.995	29	1.000
10	0.954	20	0.996	30	1.000

1.3 PCA

U *i*-toj vrsti tabele se nalazi vrednost objašnjne varijanse sa prvih *i* atributa. Možemo primetiti da je sa prvih deset novoformiranih atributa objašnjeno preko 95% varijanse, a sa dvadeset preko 99%. U ovom primeru je dimenzija smanjena sa trideset na deset atributa.

2.3 Rešavanje problema nebalansiranih klasa

Kako smo već naglasili, u skupu podataka se nalazi 62,7% benignih i 37,3% malignih dijagnoza. Dakle, klase su blago nebalansirane. Pošto se prilikom treniranja neuronske mreže, za izbor modela koristi preciznost klasifikacije, model može pokazivati precenjenu tačnost ukoliko su klase nebalansirane (ukoliko je ukupna preciznost dobra, ali se manje zastupljene klase loše klasifikuju). Takođe, sama priroda problema nam nagoveštava da se situacija loše klasifikacije malignih instanci treba minimizovati. Gore navedeni problemi su rešeni definisanjem težina klasa.

```
from sklearn.utils import class_weight

class_weights = class_weight.compute_class_weight('balanced',
np.unique(diagnosis_train),diagnosis_train)
```

Na mestu prvog parametra se mogu naći tri vrednosti: balanced, dict ili None.Ukoliko je 'balanced', težine klasa će biti računate formulom `n_samples/(n_classes* np.bincount(y))`. Funkcija *bincount* određuje broj pojavljivanje svakog elementa u nizu. Ukoliko je prosleđen rečnik, ključevi su nazivi klasa, a vrednosti su odgovarajuće težine klasa. Ukoliko je prosleđen None onda su težine klasa jednake.

3. Arhitektura neuronske mreže

3.1 Mreža

Kreirana je višeslojna mreža propagacije unapred(Multilayer feed-forward neural network), jer je povezanost neurona usmerena ka izlazu iz mreže. U ovom primeru je implementirana neuronska mreža sa dva sloja- jednim skrivenim koji se sastoji od šest neurona, i jednim izlaznim slojem.

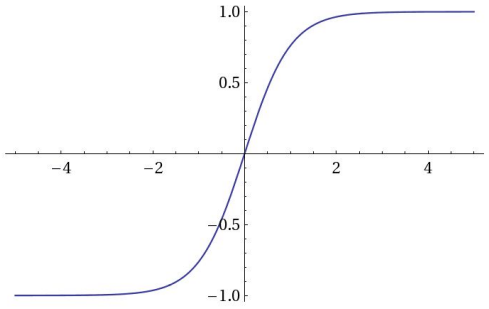
```
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
         keras.layers.Dense(6, activation='tanh', input_shape=(10,)),
         keras.layers.Dense(1, activation='tanh')
])
```

Mreža sa propagacijom unapred se implementira pomoću objekta Sequential. U okviru njega definisemo potpuno povezani skriveni sloj, i izlazni sloj.

3.2 Funkcija aktivacije

Aktivaciona funkcija predstavlja metamtičku formulu koja određuje izlaz iz neuronske mreže. Ova funkcija se primenjuje na svaki neuron, i određuje da li će se aktivirati ili ne, na osnovu značajnosti ulaza neurona za predikciju modela. U našem primeru je korišćen hiperbolički tangens kao funkcija aktivacije.



$$tanh(x) = \frac{e^x - e}{e^x + e}$$

1.4 Grafik i formula hiperboličkog tangensa

Prednost hiperboličkog tangensa je u tome da se negativni ulazi mapiraju u strogo negativne i ulazi koji su jednaki nuli se mapiraju u vrednosti bliske nuli. Predstavlja jednu od najpopularnijih aktivacionih funkcija za binarnu klasifikaciju.

3.3 Optimizator

Optimizator predstavlja jedan od dva neophodna parametra za prevođenje keras modela. Optimizatori dostupni u okviru kerasa se zasnivaju na *gradijentom spustu*. U ovom primeru je korišćen Adam optimizator koji prestavlja stohastičnu verziju gradijentnog spusta.

Bitna osobina optimiztora je korak učenja(learning rate). Korak učenja predstavlja hiperparametar koji se koristi prilikom treniranja neuronskih mreža. Predstavlja realan broj koji se najčešće nalazi u rasponu od 0.0 do 1.0. Korak učenja kontroliše koliko brzo se model adaptira određenom problemu. Što je korak manji to je učenje sporije i zahteva veći broj epoha, sobzirom da se prave manje promene na parametrima prilikom treniranja mreže. Ukoliko je korak veći, potrebno je manje epoha za treniranje, jer su promene u parametrima veće. Takođe, previše mali korak može dovesti do prespore konvergencije, dok preveliki može brzo izkonvergirati u lokalni minimum. Korak učenja zato mora biti pažljivo izabran, i zavisi od konkretnog problema koji se rešava.

```
lr_schedule = keras.optimizers.schedules.PolynomialDecay(
  initial_learning_rate=1e-2,
  decay_steps=10000, end_learning_rate=0.9, power=0.75)

opt = keras.optimizers.Adam(learning_rate= lr_schedule)
```

Korak učenja može biti statički (konstantan tokom celog treninga), ili dinamički. U okviru kerasa promena koraka se može vršiti nakon odredjenog broja epoha. Takođe, korak može opadati eksponencijano, polinomijalno ili se funkcija promene može definisati kao inverzna funkcija vremena. U ovom primeru je korišćena *polinomijlna promena koraka učenja*. Ovaj pristup primenjuje polinomijalnu funkciju opadanja pri svakom koraku optimizatora. Potrebno je obezbediti inicijalnu vrednosti koraka učenja(initial_learning_rate) i krajnje vrednosti(end_learning_rate), kao i stepen raspada(decay_step).

3.4 Treninranje mreže

Da bi se potpuno poveza mreža trenirala, prvo je potrebno prevesti napravljeni model. Pored optimizatora, neophodan parametar je funkcija gubitka(loss). Prilikom optimizacije zadata funkcija se minimizuje.

```
model.compile(loss=keras.losses.Binary_Crossentropy(),  optimizer=opt,
metrics=['accuracy'])
```

Pošto obrađujemo problem binarne klasifikacije, izabrana je funkcija binarne entropije, koja se računa po sledećoj formuli:

$$loss = y * (-log(y_p)) + (1-y) * (-log(1-y_p)),$$

gde je y klasa kojoj instanca pripada, a y_p klasa koju joj model dodeljuje.

Mrežu treniramo nad 70% podataka, a nad ostalim podacima vršimo testiranje.

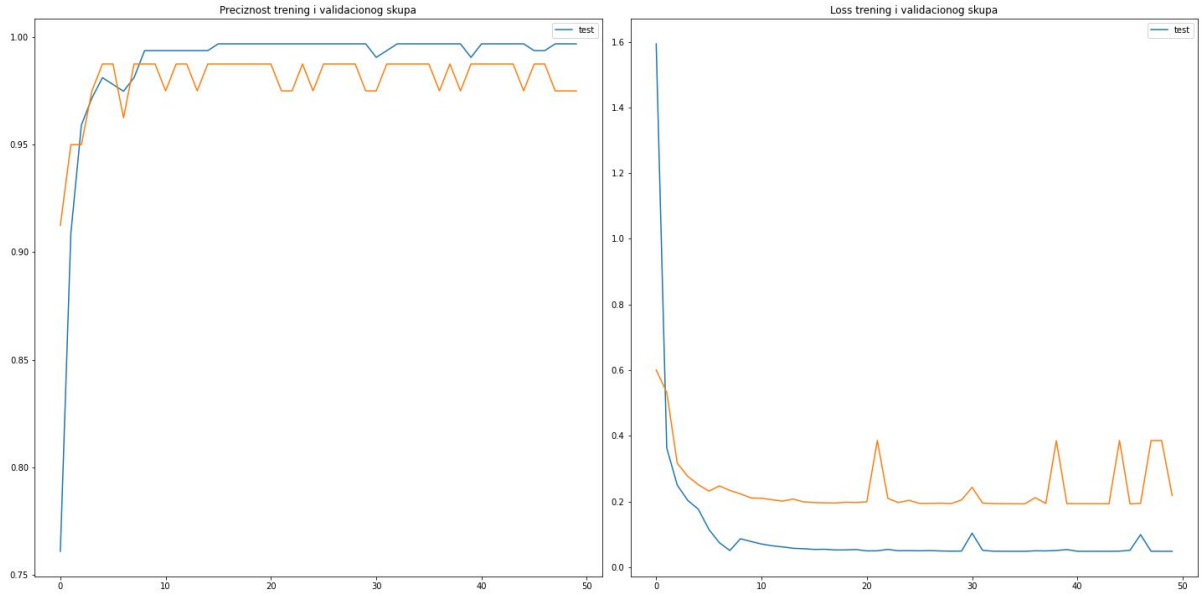
```
from sklearn.model_selection import train_test_split

data_train,    data_test,    diagnosis_train,    diagnosis_test    =
train_test_split(data, diagnosis, test_size=0.3, stratify=diagnosis)
```

Napomena: Podelu na trening i test skup je potrebno izvršiti pre standardizacije podataka.

Ovim smo precizirali sve potrebne parametre za definisanje modela. Kako je skup podataka skroman, nakon malog broja epoha može doći do preprilagođavanja, pa uvodimo i dodatni validacioni skup podataka, koji će nam pomoći da uočimo preprilagođavanje ukoliko do njega dođe. Validacioni skup dobijamo odvajanjem 20% instanci iz trening skupa. Dakle, mreža će se trenirati sa 80% instanci trening skupa(koji predstavlja 70% ukupnog skupa podataka), a na validacionom skupu se evaluira. Rezultati nakon treniranje mreže kroz 50 epoha se mogu naći na slici 1.5

```
history = model.fit(x=data_pca, y=diagnosis_train, epochs=50, class_weight=class_weights, validation_split=0.2)
```



1.5 Rezultati treniranja na 50 epoha

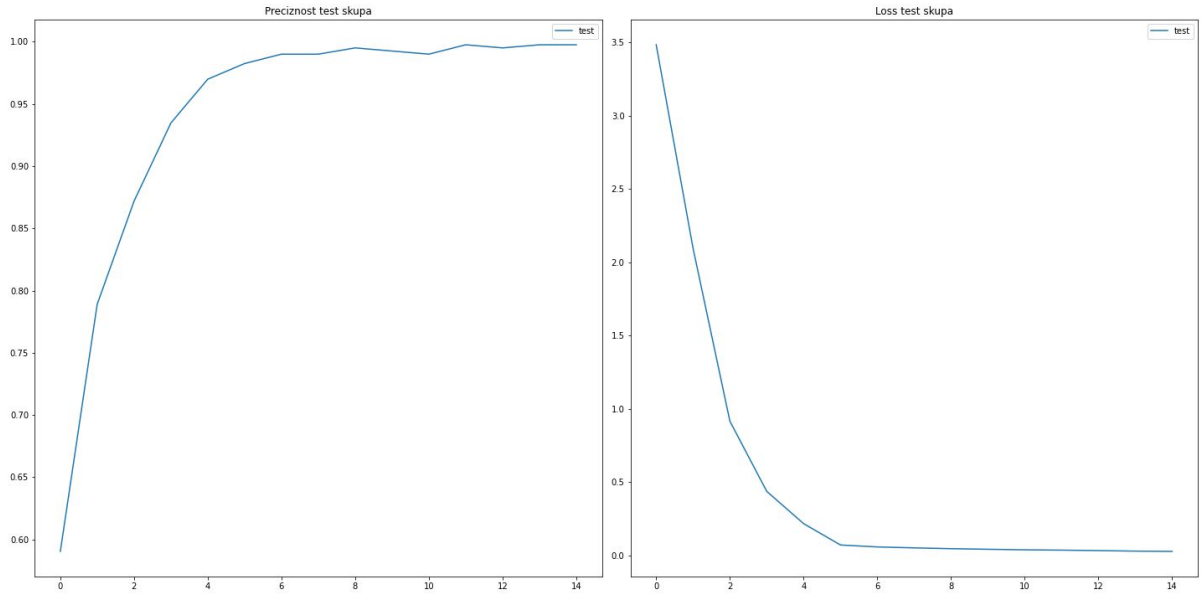
Na osnovu grafika se može zaključiti da je broj od pedeset epoha preveliki, i da već oko dvadesete epohe vrednost validacionog skupa počinje da opada, i znatno više da varira od vrednosti nad ostatkom trening instanci, što je znak preprilagođavanja. Takođe, preciznost ne raste nakon određenog trenutka, što nam govori da je potrebno smanjiti broj epoha. Izabrano je da se mreža trenira na petnaest epoha.

4. Rezultati

Validacioni skup je bitan prilikom konstrukcije arhitekture neuronske mreže, a nakon tog procesa nam više nije neophodan, već i te podatke želimo da koristimo za treniranje, a evaluaciju modela vršimo nad test instancama.

```
history = model.fit(x=data_pca, y=diagnosis_train, epochs=15, class_weight=class_weights)
```

Na slici ispod se nalaze vrednosti preciznosti i gubitka tokom treniranja.



1.6 Preciznost i gubitak nad trening skupom

Već na osnovu grafika se može zaključiti da je preciznost blizu jedinice. Da bismo detaljnije imali u uvid klasifikacije svake od klasa, posmatramo *matrice konfuzije.*

	B	M
B	*250*	*0*
M	*2*	*146*

Preciznost dobijena na trening skupu je 0.9949748516082764. Dakle, preko 99% instanci se dobro klasifikuje. Dve instance malignog tumora su loše klasifikovane.

Matrica konfuzije za trening skup

	B	M
B	103	4
M	3	61

Preciznost dobijena na test skupu je 0.9590643274853801. Dakle, oko 96% instanci se dobro klasifikuje. Sedam instanci se loše klasifikuje, od čega tri maligne i četri benigne dijagnoze.

Matrica konfuzije za test skupu

5. Zaključak

Rak dojke je jedan od vodećih uzroka smrti kod žena. U ovom primeru je analiziran problem klasifikacije skupa podataka Breast cancer Winsconsin. U programskom jeziku Python implementirana je neuronska mreža sa propagacijom unapred i dobijena je preciznost od 96%.