|  |  |
| --- | --- |
| **NASLOV PROJEKTNOG ZADATKA:** | *Predikcija dijabetesa* |
| **KOLEGIJ:** | *Primijenjeno strojno učenje* |
| **IME I PREZIME STUDENTA:** | *Mario Golemović, Srđan Lazić* |
| **AKADEMSKA GODINA:** | *2021./2022.* |
| **NAPOMENA:** |  |

**SADRŽAJ**

1. [UVOD 3](#_Toc107268642)

2. [DEFINIRANJE PROBLEMA 4](#_Toc107268643)

2.1. [Uključivanje funkcija 4](#_Toc107268644)

2.2. [Učitavanje csv filea 4](#_Toc107268645)

2.3. [Provjeravanje korelacije 5](#_Toc107268646)

2.4 [Provjeravanje dataseta 5](#_Toc107268647)

2.5 [Provjeravanje parametara 6](#_Toc107268648)

2.6 [Provjeravanje šansi za dijabetes 7](#_Toc107268649)

3. [PREDLOŽENO RJEŠENJE ZA PREDVIĐANJE DIJABETESA 8](#_Toc107268650)

3.1. [Izvlačenje podataka iz dataseta 8](#_Toc107268651)

3.2. [Skaliranje podataka 9](#_Toc107268652)

3.3 [Train i test skupovi 9](#_Toc107268653)

3.4. [Konfiguracija mreže 9](#_Toc107268654)

3.5. [Provedba učenja modela 10](#_Toc107268655)

3.6. [Evaluacija modela 10](#_Toc107268656)

4. [EVALUCIJA PREDLOŽENOG RJEŠENJA 11](#_Toc107268657)

5. [ZAKLJUČAK 14](#_Toc107268658)

[LITERATURA 15](#_Toc107268659)

1. ****UVOD****

Cilj ovog projekta je što točnije utvrditi odgovara li određena medicinska slika o osobi koja ima dijabetes ili ne. Kao parametre smo uzeli neke od najbitnijih stavki koje utječu na dijabetes kao što su trudnoća, razina glukoze, krvni tlak ,debljina kože ,inzulin, BMI te godine. Podatke koje koristimo u ovom projektu predikcije dijabetesa, skinuli smo s Keggle datasets stranice u obliku csv file-a. Pomoću funkcija u Pythonu odredili smo tipove podataka koji se nalaze u datasetu, vidjeli njihove rezultate različitih matematičkih formula i provjerili postoje li nepotpuni podaci u datasetu poput vrijednosti NULL. Da bismo nastavili s izradom modela za predikciju, morali smo provjeriti korelaciju i ispravnost podataka. Također morali smo se upoznati s određenim parametrima koji će utjecati na rezultat našeg modela, odnosno na to ima li osoba dijabetes ili ne. U ovom dokumentu ćemo detaljno opisati programski kod ovog projekta te svaki dio koda objasniti zasebno.

1. ****DEFINIRANJE PROBLEMA****

U ovom poglavlju ćemo detaljno objasniti kako smo pristupili problemu i započeli sa stvaranjem našeg programskog koda pomoću kojega ćemo moći predviđati dijabetes.

* 1. ****Uključivanje funkcija****

Na slici koja se nalazi ispod, prikazali smo sve funkcije koje smo uključili u programski kod odnosno koje su nam bile potrebne za rješavanje problema. Pomoću “matplotlib.pyplot as plt“ uključujemo funkcije za crtanje grafičkog prikaza.“ import numpy as np“ nam omogućava funkcije za rad s matricama i poljima, dok nam “ import seaborn as sns“ služi za crtanje grafova. Kako bismo izradili, čitali i spremali datasetove koristili smo “ import pandas as pd“.Također su nam bili potrebni slojevi unutar biblioteke keras koji služe za duboko učenje mreže, te smo to učinili s “ from keras.layers import Dense, Dropout“. Pomoću “from keras.models import Sequential“ uključili smo vrstu modela koju koristimo za duboko učenje u kojima se nalaze Dense i Dropout. Uključivanje funkcija za podjelu podataka na testni skup i skup za treniranje smo uključili pomoću “ from sklearn.model\_selection import train\_test\_split“. Još jedna od bitnih stvari su funkcije za skaliranje podataka koje smo uključili pomoću “ from sklearn.preprocessing import StandardScaler“, te smo iz biblioteke sklearn uključili funkcije za crtanje i računanje confusion matrica, matrica za računanje koliko je puta pogodio i promašio rezultat program pomoću “ from sklearn.metrics import confusion\_matrix“.

|  |
| --- |
| Slika koda  import matplotlib.pyplot as plt  import numpy as np  import seaborn as sns  import pandas as pd  from keras.layers import Dense, Dropout  from keras.models import Sequential  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler, MinMaxScaler, QuantileTransformer  from sklearn.metrics import confusion\_matrix |

Sl. 2.1. Prikaz svih funkcija i biblioteka

* 1. ****Učitavanje csv filea****

Kod smo započeli s pozivanjem pandas funckije read\_csv koja će pročitati csv, dok je df varijabla za dataset. Pomoću “print(df.info())“ ispisujemo par informacija o tom datasetu koje će se pojavljivati u konzoli odnosno ispisati se. Preostale 2 linije koda “ print(df.describe())“ i “ print(df.isnull().sum())“ također imaju istu svrhu s tim da sa zadnjom linijom koda u ovom dijelu prebrojavamo podatke koji su prazni ili nepotpuni.

|  |
| --- |
| Slika koda  df = pd.read\_csv('diabetes.csv')  print(df.info())  print(df.describe())  print(df.isnull().sum()) |

Sl. 2.2. Primjer učitavanja filea

* 1. ****Provjeravanje korelacije****

Provjeravamo korelaciju između varijabli podataka. Pozivamo matplotlib datoteku za crtanje grafova koja se piše kao “plt.figure(figsize=(15,10))“, figsize je veličina prozora. Nakon toga ćemo pozvati seaborn datoteku “ sns.heatmap(df.corr(), annot=True, fmt =".2f", cmap="coolwarm")“, funkcija heatmap za crtanje mape, “df.corr()“ izračunava korelaciju između podataka, “annot=true“ prikazuje nazive kolona podataka, “ fmt =".2f"“ će prikazivati float tipove podataka sa 2 decimalna razreda i “ cmap="coolwarm"“ je tema crteža. Na kraju pozivamo “ plt.show()“ funkciju za prikazivanje.

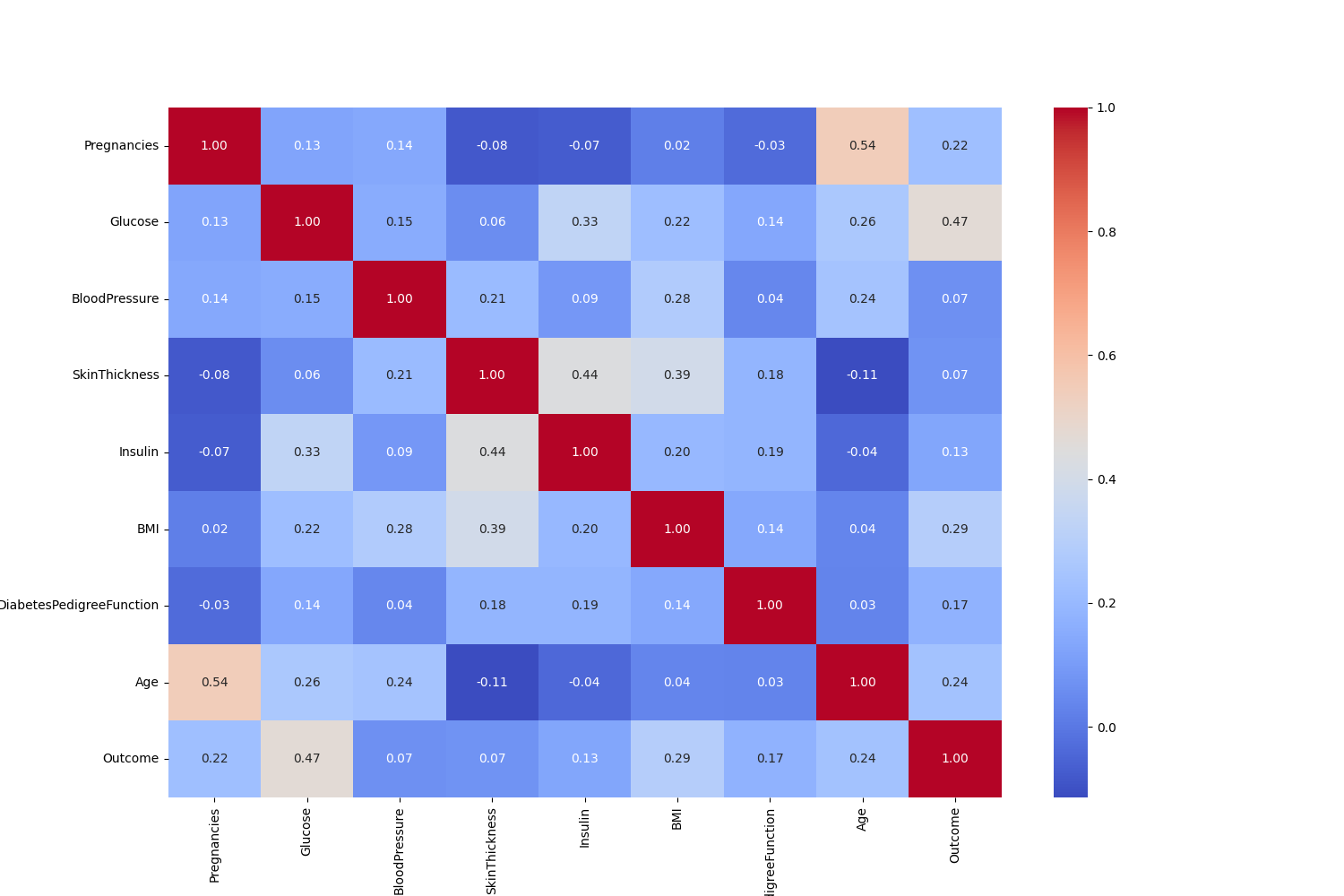
Slika koda

plt.figure(figsize=(15,10))

sns.heatmap(df.corr(), annot=True, fmt =".2f", cmap="coolwarm")

plt.show()

Sl. 2.3-1 Primjer koda za prvi graf



Sl. 2.3-2 Slika prvog grafa

* 1. ****Provjeravanje dataseta****

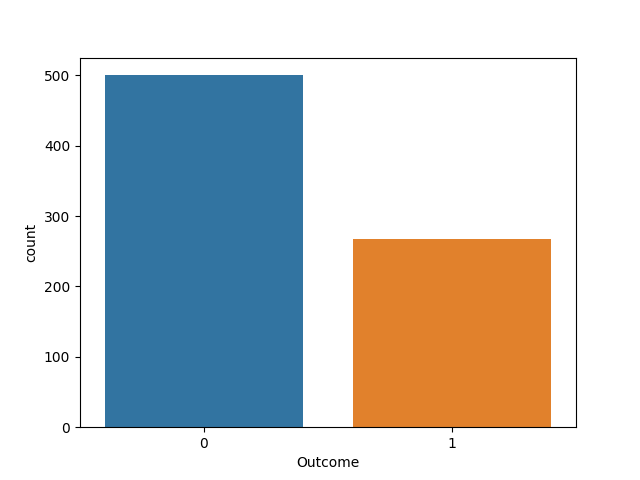
Provjeravamo dataset, koliko osoba ima dijabetes, a koliko ne. Pomoću “ sns.countplot('Outcome', data=df)“ pozivamo seaborn funkciju za crtanje countplota, “outcome“ prebrojava stupac outcome te “dana=df“ označava iz koje varijable to čitamo. “plt.show()“ na kraju prikazuje graf.

Slika koda

sns.countplot('Outcome', data=df)

plt.show()

Sl. 2.4-1 Primjer koda za drugi graf



Sl. 2.4-2 Slika drugog grafa

* 1. ****Provjeravanje parametara****

Provjeravamo odnos razine glukoze u krvi na BMI i broj godina. “ plt.figure(figsize=(20,10))“ ima posao da odredi veličinu prozora. Iz datoteke seabron pomoću “ sns.scatterplot(data=df, x="Glucose", y="BMI", hue="Age", size="Age")“ crtamo scatterplot, točkasti graf,“ data=df“ iz varijable df na x osi ćemo prikazati razinu glukoze, a na y osi ćemo prikazati razinu BMI, točkice prikazuju broj godina, a pomoću “size=age“ točkice će se grupirati u veličinama prema godinama.

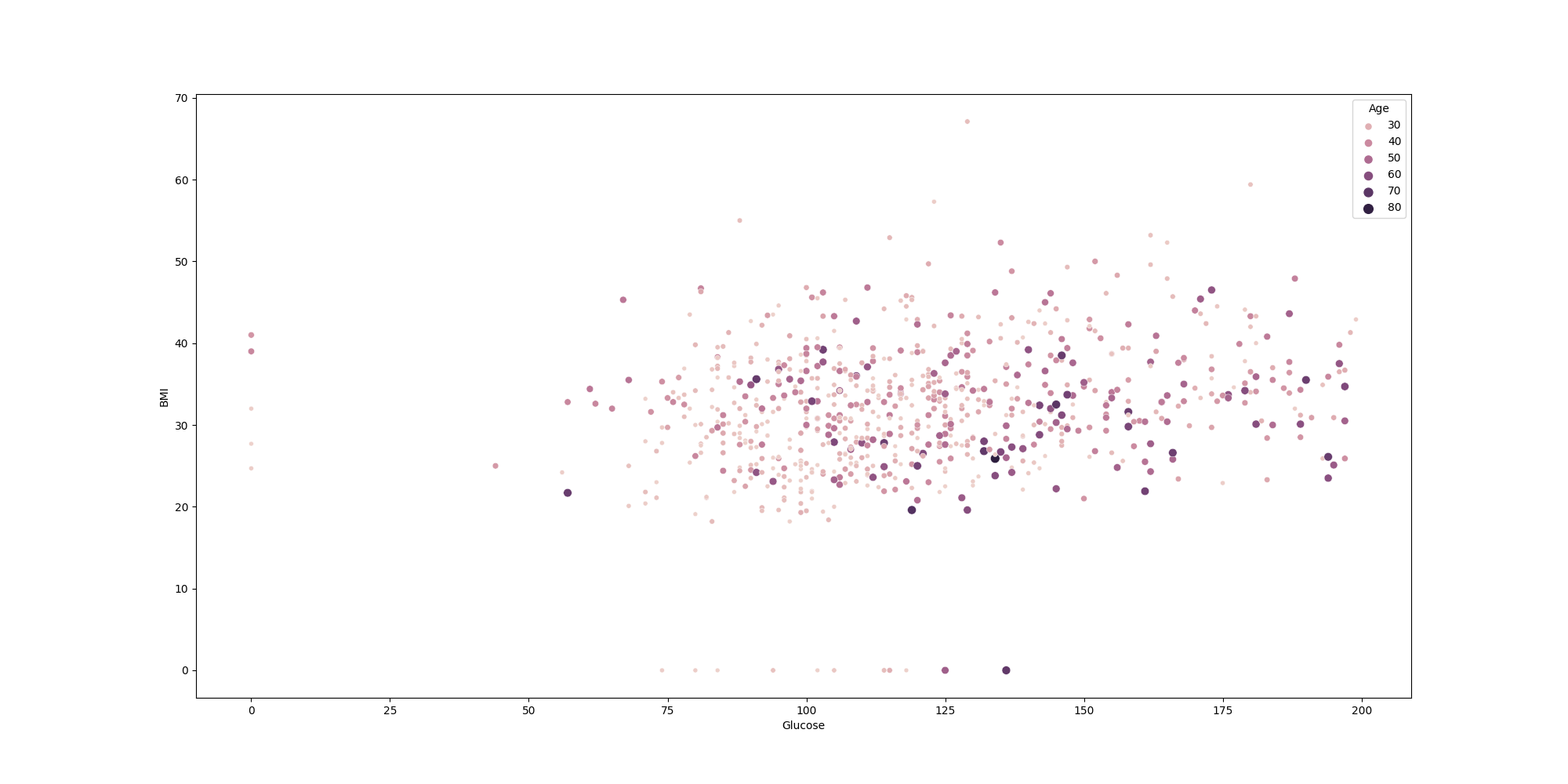
Slika koda

plt.figure(figsize=(20,10))

sns.scatterplot(data=df, x="Glucose", y="BMI", hue="Age", size="Age")

plt.show()

Sl. 2.5-1 Primjer koda za treći graf



Sl. 2.5-2 Slika trećeg grafa

* 1. ****Provjeravanje šansi za dijabetes****

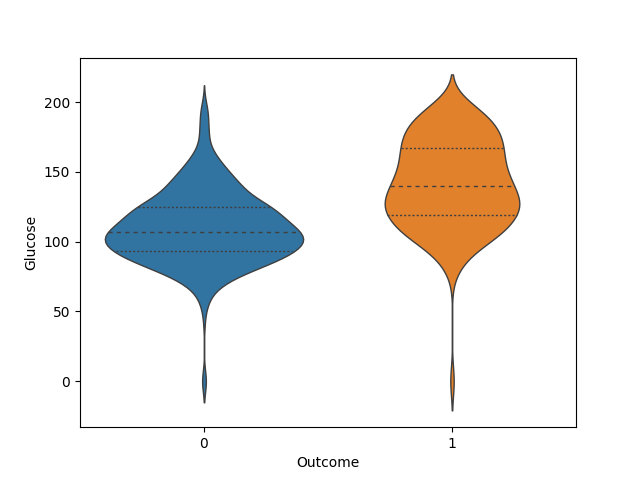
Normalna razina glukoze prema naim istraživanjima je ispod 140mg/dL krvi. Preko toga je visoka šansa za uzrok dijabetesa. Pomoću “ sns.violinplot(data=df, x="Outcome", y="Glucose", split=True, inner="quartile", linewidth=1)“ pozivamo seabron, violinplot, odnosno graf violine, “data=df“ iz varijable df na osi x je prikazan outcome, y os prikazuje razine glukoze,“split=True“, “inner="quartile"“, “linewidth=1“ su zajedno i oni će samo ukrasiti izgled grafa. Nama će pobliže prikazati kako se osobe grupiraju, split true će ih bolje grupirati. Inner označava na koliko grupa će ih podijeliti i to smo odabrali na četvrtine, a debljina linije je linewidth 1.

Slika koda

sns.violinplot(data=df, x="Outcome", y="Glucose", split=True, inner="quartile", linewidth=1)

plt.show()

Sl. 2.6-1 Primjer koda za četvrti graf



Sl. 2.6-2 Slika četvrtog grafa

1. ****PREDLOŽENO RJEŠENJE ZA PREDVIĐANJE DIJABETESA****

U sljedećim fazama našeg koda, pokušati ćemo na osnovu dobivenih podataka iz dataseta dobiti što točnije rješenje naše predikcije.

* 1. ****Izvlačenje podataka iz dataseta****

Dataset je nova varijabla te u nju samo spremamo brojčane vrijednosti iz dataseta koji smo učitali preko pandasa. U varijablu x ćemo pomoću “ x = dataset[:, 0:8]““ učitavati prvih 7 stupaca, ne uključujući 8. stupac i sve retke. Dok ćemo pomoću “ y = dataset[:, 8]“ učitati svaki redak i samo 8. stupac.

Slika koda

dataset = df.values

x = dataset[:, 0:8]

y = dataset[:, 8]

Sl. 3.1. Primjer koda za izvlačenje podataka

* 1. ****Skaliranje podataka****

Skaliramo podatke da bi program lakše i brže obradio te podatke. Program sebi napravi manji interval podataka, da kroz neuronsku mrežu kad podatci prolaze, ne množi previše puta i uspori program. StandardScaler je vrsta skalera koju ćemo koristiti za skaliranje te ćemo ga nazvati skaler. Pomoću fit\_transform toj funkciji predajemo podatke iz dataseta na kojima će se on istrenirati te će ih na kraju sa print prikazati.

Slika koda

scaler = StandardScaler()

x = scaler.fit\_transform(x)

print(x)

Sl. 3.2. Primjer koda za skaliranje

* 1. ****Train i test skupovi****

Dijelimo skup izvučenih podataka na train i test skupove. “train\_test\_split(x, y, random\_state=42)“ je funkcija koju pozivamo da podijelimo te podatke. Random\_state 42 je koeficijent na osnovu kojeg se na slučajni odabir izvlače podaci. Iz y se podaci guraju u y train i test , te isto tako i za x podatke. Train kroz neuronsku mrežu provlači te podatke te trenira taj model da on nauči kako izgleda kada netko ima dijabetes na osnovu parametara i izbaci rezultat. Pomoću testa provjeravamo koliko je kod naučio provjeravati podatke tako da mu predamo parametre koje nije provukao kroz train.

Slika koda

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=42)

Sl. 3.3. Primjer koda train i test

* 1. ****Konfiguracija mreže****

Pokušavamo definirati kako će naš model za predikciju izgledati. Model koji smo koristili za naš projekt će biti sequential model mreže, sekvencijalni red slojeva. Dodajemo dense sloj mreže sa 8 ulaznih čvorova. Kernel initializer definira način na koji se inicijaliziraju vrijednosti težina u keras slojevima. Aktivacija relu je najoptimalnije rješenje za obradu podataka kroz sloj za učenje. Relu konkretno pokazuje kako će se podaci matematički izračunavati kroz sloj. On će 4 podatka preračunati kroz sloj na relu način te ih šalje dalje. Nakon toga, ponovno dodajemo 2 dense sloja te jedan dropout sloj koji prema koeficijentu 0.25 slučajnim odabirom izbacuje podatke iz mreže. Na kraju, dodajemo još jedan dense sloj sa sigmoid aktivacijom te pomoću modela summary ispisujemo informacije o modelu kojega smo izradili.

Slika koda

model = Sequential()

model.add(Dense(8, input\_dim=8, kernel\_initializer='normal', activation='relu'))

model.add(Dense(units=200, input\_dim=8, kernel\_initializer='normal', activation='relu'))

model.add(Dense(units=64, input\_dim=200, kernel\_initializer='normal', activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(units=1, activation='sigmoid'))

model.summary()

Sl. 3.4. Primjer koda konfiguracije mreže

* 1. ****Provedba učenja modela****

Pomoću model.compile podešavamo karakteristike procesa učenja, to su dodatne postavke za model, dok je model.fit funkcija koja uči model. Pomoću batch\_size uzimati ćemo 5 po 5 linija skupa podataka i predavati ih dok ih ne predamo sve, te ćemo to učiniti 200 puta.

Slika koda

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

model.fit(x\_train, y\_train, epochs=200, batch\_size=5)

Sl. 3.5. Primjer koda provedbe učenja

* 1. ****Evaluacija modela****

Y\_pred je nova varijabla i u nju spremamo rezultate predikcije modela nad testnim skupom podatka, ti rezultati predikcije će biti neki float brojevi: “ y\_pred = (y\_pred > 0.5).astype('int')“ je formula pomoću koje ćemo te vrijednosti raščlaniti na 0 i 1 binarne vrijednosti te će to biti npr. int. Cf matrix je varijabla u koju spremamo rezultate proračuna funkcije confusion matrix koja će usporediti točnost predikcije. Na kraju dobivamo sliku rješenja programa.

Slika koda

y\_pred = model.predict(x\_test)

y\_pred = (y\_pred > 0.5).astype('int')

cf\_matrix = confusion\_matrix(y\_test, y\_pred)

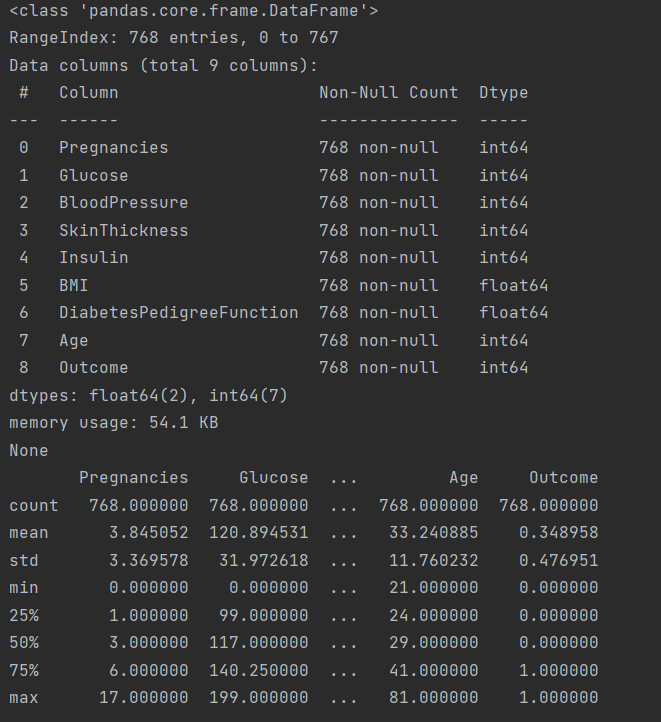
sns.heatmap(cf\_matrix, annot=True, cmap='Blues')

plt.show()

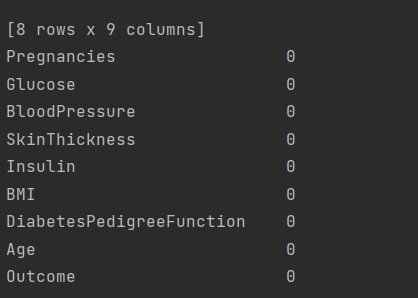
Sl. 3.6. Primjer koda evaluacije modela

1. ****EVALUCIJA PREDLOŽENOG RJEŠENJA****

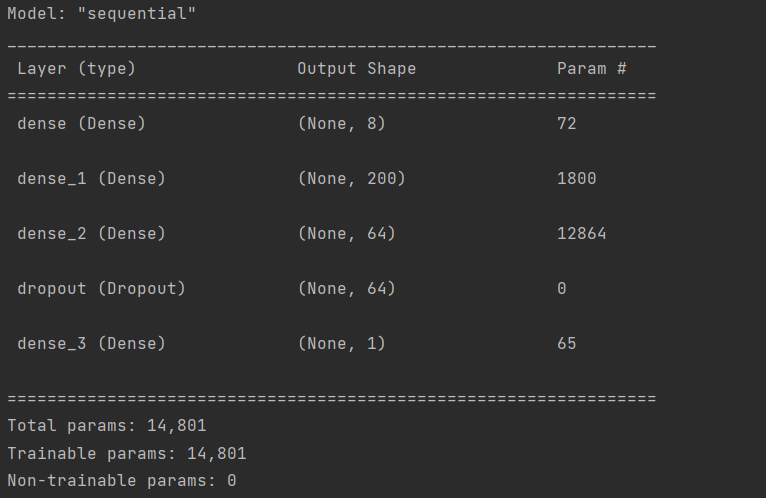
U sljedećim primjerima ćemo pokazati konkretne rezultate koje smo dobili potkrijepljene grafom i slikama nakon pokretanja programskog koda.



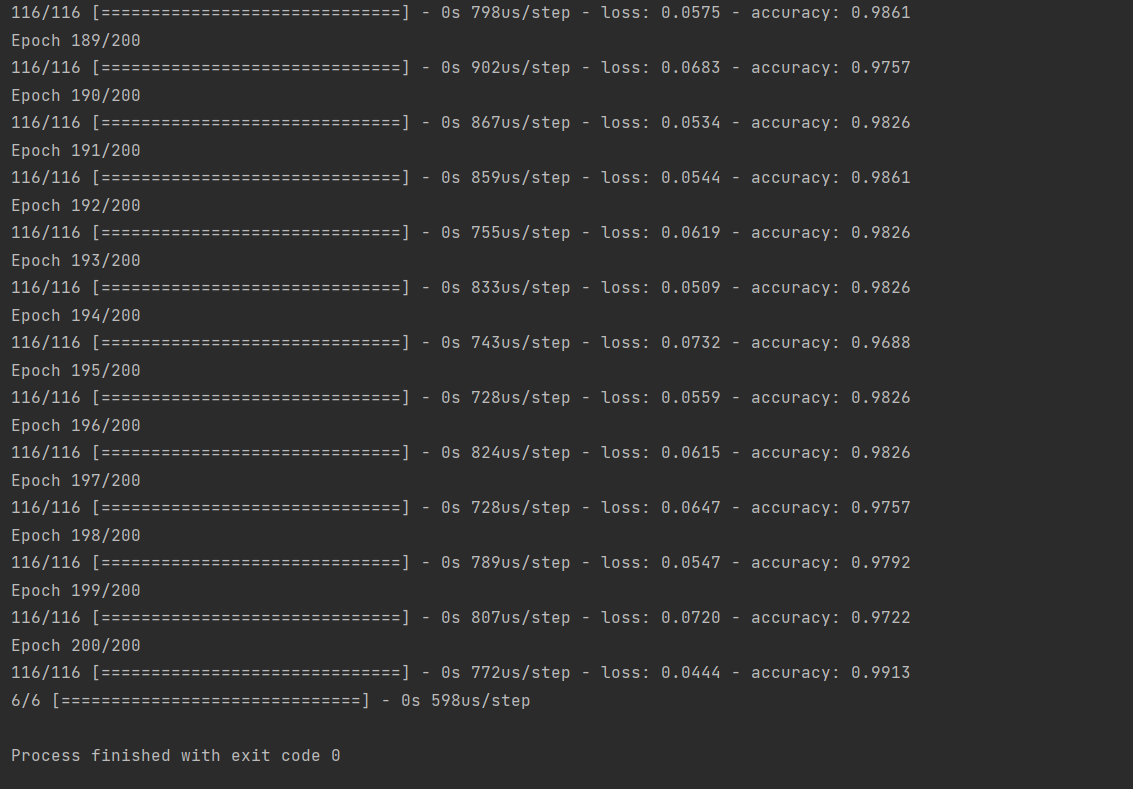
Sl. 4.1. Izlazni screenshot 1.



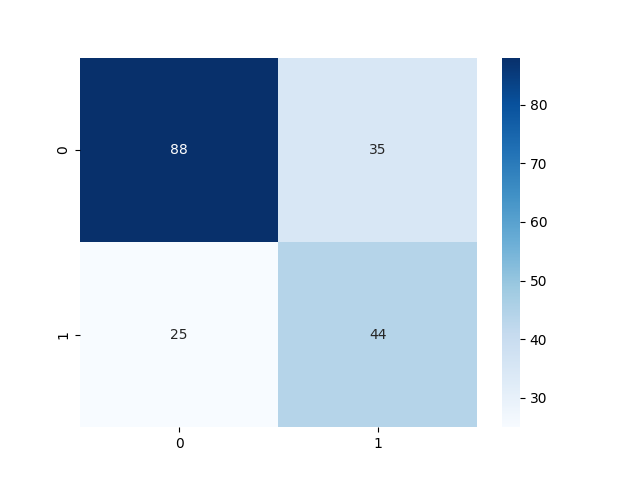
Sl. 4.2. Izlazni screenshot 2.



Sl. 4.3. Izlazni screenshot 3.



Sl. 4.4.Izlazni screenshot 4.



Sl. 4.5.Izlazni graf

1. ****ZAKLJUČAK****

Iz dobivenih rezultata koje smo potkrijepili grafovima vidimo da je accuracy traininga 0.9913, a loss 0.0444. Za testne podatke vidimo iz dobivenog grafa da program vrlo točno predviđa situacije kada osoba nema dijabetes, od 123 osobe koje nemaju dijabetes za 88 je točno pogodio, dok za 35 nije. S druge strane za situacije kada osoba ima dijabetes rezultati nisu baš najbolje predviđeni, od 69 osoba koje imaju dijabetes program je na testnim podacima pogodio da njih 44 ima, a čak 25 pogriješio.

LITERATURA

[1] Prezentacije s predavanja, laboratorijske vježbe.