Programska aplikacija za predviđanje bolesti

Benc, Lucija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Lucija Benc

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Studentica:

izv. prof. dr. sc. Tomislav Stipančić Lucija Benc

Zagreb, 2023.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru izv. prof. dr. sc. Tomislavu Stipančiću na pruženoj pomoći i stručnim savjetima, te uloženom vremenu, trudu i posvećenosti tijekom izrade ovog rada.

Posebno zahvaljujem svojoj obitelji na strpljenju, podršci i vjerovanju u mene, te prijateljima čiji mi je smijeh, ohrabrivanje i dijeljenje postignuća uljepšalo studentske dane.

Lucija Benc

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 23 - 6 / 1**	
Ur.broj: **15 - 23 -**	

DIPLOMSKI ZADATAK

Student: **Lucija Benc** JMBAG: 0035215933

Naslov rada na hrvatskom jeziku: **Programska aplikacija za predviđanje bolesti**

Naslov rada na engleskom jeziku: **Software application for disease prediction**

Opis zadatka:

Modeli strojnog i dubokog učenja djeluju temeljem podataka u različitim sferama ljudskog djelovanja. Kao takve moguće ih je koristiti za različita predviđanja temeljena na podacima. U radu je potrebno razviti model strojnog učenja koji može učinkovito predvidjeti bolesti čovjeka na temelju simptoma (za dijabetes, srčane bolesti i Parkinsonovu bolest), zdravstvenog stanja i dobi. Podaci koji povezuju bolesti sa simptomima trebaju biti definirani u tekstualnoj formi. Koraci razvoja modela trebaju uključivati:

- sakupljanje podataka za učenje i trening modela,
- čišćenje i prilagodbu podataka za učenje i trening modela,
- izradu modela strojnog učenja za predviđanje bolesti, te
- fazu evaluacije u kojoj se osnovni model čini robusnijim i točnijim.

Model je potrebno implementirati u sklopu web aplikacije kao korisničkog sučelja kroz koje korisnik komunicira s modelom. Razvijeno cjelovito rješenje potrebno je eksperimentalno evaluirati u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
28. rujna 2023.	30. studenoga 2023.	4. – 8 . prosinca 2023.

Zadatak zadao:

Izv. prof. dr. sc. Tomislav Stipančić

Predsjednik Povjerenstva:

Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	Jedinica	Opis
ω	/	težinski faktor
reLU	/	ispravljena linearna jedinica (*rectified linear unit*)
x	/	ulazni signal

SAŽETAK

Tema ovog rada je razvoj korisničkog sučelja koje služi kao komunikacija korisnika i modela za predviđanje. Sučelje, kao i model, razvijeni su koristeći programski jezik *Python*, a uz pomoć *Pythonovih* biblioteka otvorenog koda, *Streamlit* i *Anaconda*. U programskom jeziku *Python* su prvo izrađeni modeli predviđanja bolesti koje će biti implementirane u sučelju, a zatim je izrađen program koji definira korisničko sučelje u obliku programske aplikacije. U početku rada objašnjena je teorijska osnova za razvoj modela i povezivanje modela s programskom aplikacijom, te medicinska osnova koja stoji iza rada, a u drugom dijelu primjenom strojnog učenja i neuronskih mreža razvijeno je korisničko sučelje koje korisniku omogućuje upisivanje simptoma i stanja, na temelju čega ga obavještava o stanju, a ovisno o bolesti. Za kraj rada, izrađena je eksperimentalna evaluacija u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava.

Ključne riječi: neuronske mreže, model predviđanja, duboko učenje, *Python,* programska aplikacija, *Streamlit*

SUMMARY

The topic of this paper is the development of a user interface that serves as a communication tool between the user and a predictive model. Both the interface and the model are developed using the *Python* programming language with the help of *Python*'s open-source libraries, *Streamlit*, and *Anaconda*. Disease prediction models are first created in the *Python* programming language, and later implemented in the interface. The programm is developed so that it defines user interface in the form of software application. The beginning of the paper explains the theoretical foundation for model development and the integration of the model with the software application, as well as the medical basis underlying the work. In the second part, a user interface is developed through machine learning and neural networks. It allows the user to input symptoms and conditions, and the model provides information about the user's health status, depending on the disease. Finally, an experimental evaluation is conducted within the framework of Laboratory for Manufacturing and Assembly System Planning.

Key words: neural network, prediction model, deep learning, Python, software application, *Streamlit*

1. UVOD

Posljednjih 10 godina smo svjedoci sve bržeg napretka umjetne inteligencije. Spominje se kroz primjenu u medicini, industriji, ekonomiji, matematici, pa i umjetnosti. Zbog spomenutog napretka i razvoja, možemo primijetiti da sve više tehnologija, onih starijih, ali i modernih, u nekom dijelu i aspektu uvode umjetnu inteligenciju. Razlog tome je brzina i preciznost. S razvojem umjetne inteligencije, došlo je do napretka u mnogim poljima znanosti, primjerice medicini.

Posebno korištena metoda umjetne inteligencije na temelju koje dolazi do razvoja modela su umjetne neuronske mreže. To je metoda nastala inspirirana živčanim sustavom čovjeka.

Jedna od primjena umjetne inteligencije su modeli predviđanja. Uz primjenu modela predviđanja, u ovom će se radu izraditi programska aplikacija koja omogućuje korisniku upisivanje stanja i simptoma, na temelju kojih dobiva povratnu informaciju o svojem stanju. Takva primjena vodi do brže dijagnostike, te boljih ishoda liječenja bolesnika. Kako bi model bio što točniji, potrebno je izraditi detaljni skup podataka koji obuhvaća simptome bolesti i analizu stanja. Cilj treninga modela je da algoritam što točnije i preciznije prepoznaje dijagnozu na temelju unosa korisnika.

2. TEORIJSKA OSNOVA RADA

2.1 Umjetna inteligencija

Iako se čini da umjetna inteligencija postaje popularna posljednjih desetak godina, njen početak i prvo spominjanje seže u daleku 1956. godinu. Te je godine održana konferencija o umjetnoj inteligenciji na Sveučilištu *Dartmouth* u Sjedinjenim Američkim Državama. *Dartmouth* konferencija o umjetnoj inteligenciji smatra se formalnim početkom umjetne inteligencije. Na konferenciji su se okupili znanstvenici iz polja strojarstva i računarstva. Neki od njih bili su John McCarthy, Marvin Minsky i Nathaniel Rochester. Oni su dva dana raspravljali o izazovima tadašnjih industrijskih strojeva, te o ideji strojeva koji mogu razmišljati i inteligentno izvoditi zadatke. Mnogi ovaj događaj smatraju službenim početkom umjetne inteligencije kao znanstvene discipline. Fotografija zabilježena na konferenciji prikazana je na slici. [Slika 1.] [1]

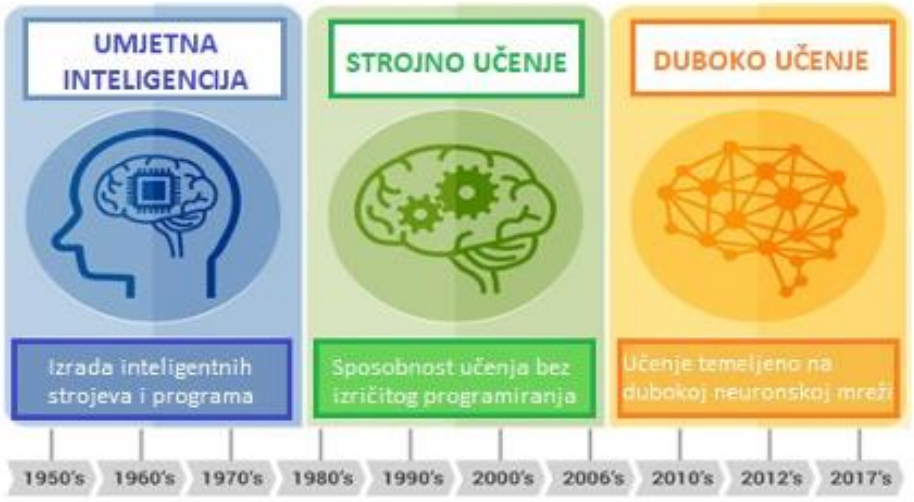
Slika 1. Dartmouth konferencija [Pogreška! Izvor reference nije pronađen.**]**

Iako je *Dartmouth* konferencija unijela velik entuzijazam i zanimanje, narednih godina nije zabilježen jači razvoj, te se razdoblje do 70.-ih godina naziva zimsko razdoblje umjetne inteligencije. Razlog tome bile su velike ratne, energetske i ekonomske krize koje su bile prisutne u cijelom svijetu.

Ponovno zanimanje raste 80.-ih godina s razvojem neuronskih mreža i ekspertnih sustava. Pojam ekspertnog sustava predstavlja inteligentni računalni sustav čije je inteligencija osnovana na bazi podataka koja obuhvaća određeno područje. Jednostavnije rečeno, ekspertni

sustav sadrži bazu znanja, odnosno podataka, te mehanizam koji mu omogućuje da donese zaključak.

Trenutak koji označava prekretnicu i eksploziju u razvoju umjetne inteligencije je razvoj strojnog, a potom i dubokog učenja o čemu će biti objašnjeno u sljedećim odlomcima. Vremenski razvoj umjetne inteligencije prikazan je na slici. [Slika 2.]



Slika 2. Vremenski prikaz razvoja umjetne inteligencije

Opširnu definiciju umjetne inteligencije dao je D. W. Patterson 1990. godine.

„*Umjetna inteligencija je grana računarske znanosti koja se bavi proučavanjem i oblikovanjem računarskih sustava koji pokazuju neki oblik inteligencije. Takvi sustavi mogu učiti, mogu donositi zaključke o svijetu koji ih okružuje, oni razumiju prirodni jezik te mogu spoznati i tumačiti složene vizualne scene te obavljati druge vrste vještina za koje se zahtijeva čovjekov tip inteligencije.*"

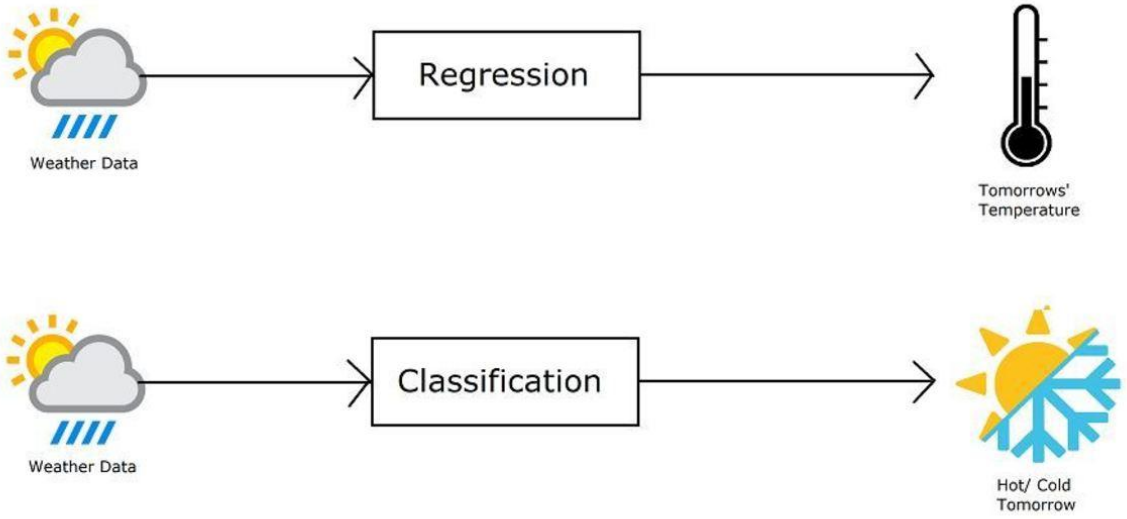
Danas je umjetna inteligencija uspješno područje s mnogim praktičnim primjenama i aktivnim temama istraživanja, a njen razvoj donosi brojne prednosti i prilike, ali isto tako postavlja značajne izazove. Etički problemi, sigurnosna pitanja, regulacija, i utjecaj na zaposlenje i društvo predstavljaju ključne aspekte. Postizanje ravnoteže između tehnološkog napretka i etičkog upravljanja ključno je za uspješno integriranje ove tehnologije u našu svakodnevicu. To ćemo postići regulacijom, obrazovanjem i odgovornim razvojem.

2.2 Strojno učenje

Strojno učenje je područje umjetne inteligencije koje se bavi razvojem algoritama, te njihovim poboljšanjem u smislu obavljanja zadatka. Ovakav tip učenja smatra se jednim od područja umjetne inteligencije. [4]

2.2.1 Podjela strojnog učenja

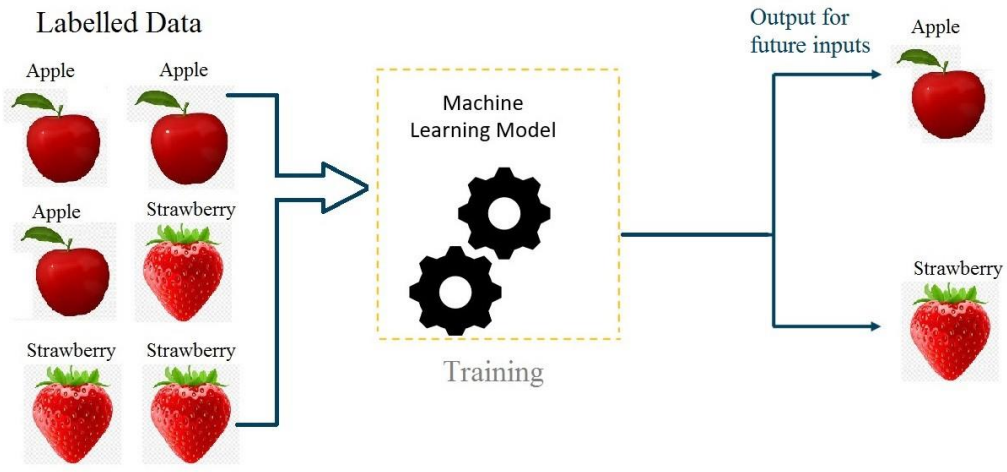
Strojno učenje dijele se u tri skupine koje su detaljnije opisane u nastavku rada. To su: nadzirano, nenadzirano i pojačano učenje [5]. Kod nadziranog učenja (engl. *supervised learning*) algoritam ima označene podatke za trening (eng. *labeled training data*). Označeni podatci podrazumijevaju da su nam poznati ulazni podatci, ali i željeni izlazi. Cilj algoritma je naučiti način preslikavanja iz ulaza u izlaz, te na taj način omogućiti predviđanje na temelju novih podataka. Nadzirani oblik učenja obuhvaća probleme regresije i klasifikacije. Podaci u obliku *(x,y),* gdje *x* označava ulaznu vrijednost, a *y* izlaznu vrijednost, nailaze na funkciju *f(x)=y*. Pritom funkcija *f(x)=y* za svaku novu ulaznu vrijednost *x*, predviđa izlaznu varijablu *y*. S obzirom na vrijednost izlazne varijable *y,* koriste se metode klasifikacije i regresije. Izlazna varijabla *y* može biti diskretna i kontinuirana. Diskretna varijabla može poprimiti samo određene vrijednosti iz diskretnog skupa. To znači da varijabla može imati samo vrijednosti izbrojive na temelju nekog skupa (obično cijeli brojevi). Primjer za diskretne varijable može biti broj stanovnika u gradu, broj djece u obitelji, broj automobila na parkingu… Možemo primijetiti da je to većinom cijeli broj. Ukoliko je *y* diskretna varijabla, radi se o klasifikaciji (eng. *Classification*). Kontinuirana varijabla može poprimiti bilo koju vrijednost unutar određenog raspona, što znači da varijabla može imati beskonačno mnogo mogućih vrijednosti unutar tog raspona. Primjeri kontinuiranih varijabli su dob ljudi, težinu, visinu, brzinu automobila. Za razliku od cijelih brojeva koji su slučaj kod diskretnih varijabli, kontinuirane varijable mogu poprimati decimalne vrijednosti. Primjerice, visina čovjeka može biti 181.6 *cm*. Kad je *y* kontinuirana varijabla (cjeloviti broj) koristi se metoda regresije (eng. *Regression*). Odličan primjer klasifikacije i regresije prikazan je na slici. [Slika 3.]



Slika 3. Razlika metode regresije i klasifikacije [4]

Ako kao ulazne podatke unesemo podatke o temperaturi, metoda regresije dala bi nam odgovor o točno predviđenoj temperaturi. S druge strane, uz iste ulazne podatke, metoda klasifikacije dat će nam podatak hoće li sutra biti hladno ili toplo. [1]

Neki od algoritma koje koristi nadzirano učenje su drvo odluke (eng. *decision tree*), SVM (eng. *Support Vector Machines*), Bayesov zaključak (engl. *Bayesian learning*) i neuronske mreže, o kojima će biti riječ u kasnijim poglavljima ovog rada. Primjer rada nadziranog učenja prikazan je na slici. [Slika 4.] Vidimo da su zadani ulazni podatci, ali i izlazni.

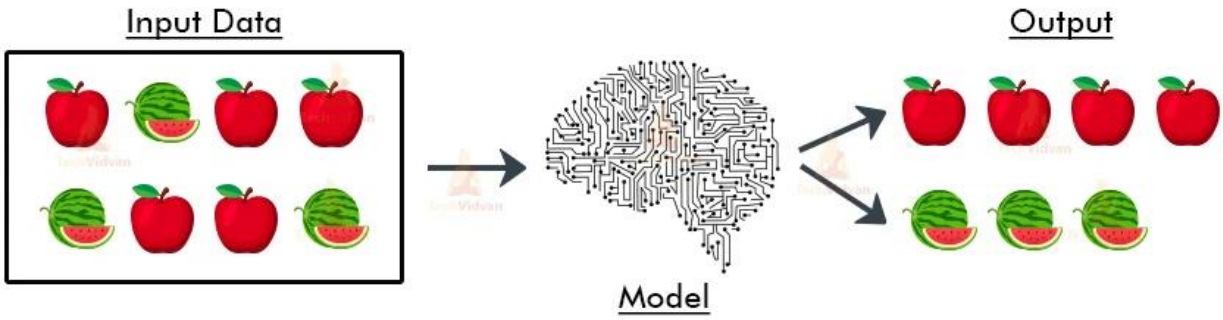


Slika 4. Nadzirano učenje [5]

Nenadzirano učenje (engl. *unsupervised learning*) koristi algoritme za donošenje zaključaka o neoznačenim podacima. Taj tip učenja pretražuje skrivene uzorke ili strukture u podatcima. Njime se grupiraju uzorci i otkrivaju strukture. Skup podataka sastoji se od ulaznih podataka, ali bez odziva, tj. izlaznih podataka, te se na temelju toga pronalaze i izvode zaključci. Najčešća tehnika kojim se koristi nenadzirano učenje je grupiranje (eng. *clustering*). Mogućnost pronalaska sličnosti i razlika u podacima omogućuje ovom tipu učenja široku primjenu. Tako se nadzirano učenje primjenjuje u različite svrhe: [32]

1. grupiranje – klasifikacija podataka po grupama ili klasama na temelju sličnosti (npr. *k-means* algoritam grupira podatke po kategorijama na temelju sličnosti)
2. smanjenje dimenzionalnosti (eng. *Dimensionality reduction*) – korištenje tehnike poput *Principal Component Analysis* (PCA) u svrhu projektiranja podataka u prostor niže dimenzionalnosti
3. analiza anomalija (eng. *Anomaly detection*) – identifikacija odstupanja ili neobičnih uzoraka u podatcima korisna je za otkrivanje prijevara ili grešaka u sustavu
4. prepoznavanje oblika (eng. *Pattern Recognition*)

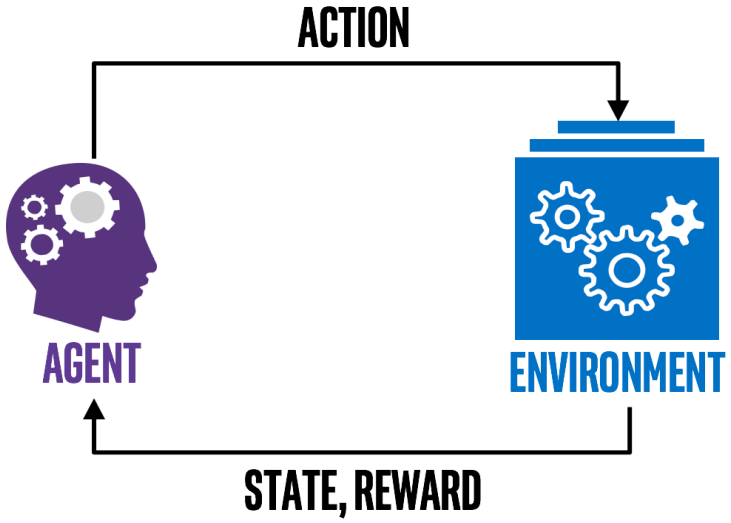
S obzirom na brojne mogućnosti primjene, nadzirano učenje često koristimo u medicini (npr. analiza sekvenca krvi), ekonomiji (npr. analiza tržišta) i policiji (npr. prepoznavanje objekta). Primjer nenadziranog učenja prikazan je na slici. [Slika 5.]



Slika 5. Nenadzirano učenje [6]

Pojačano učenje (engl. *reinforcement learning*) je model strojnog učenja kojem je temelj donošenje odluka. Kao izvor podataka, pojačano učenje koristi povratne informacije koje dobiva. Cilj je usavršiti i optimizirati model kako bi isti postigao najveću nagradu. Kad model donese točnu i dobru odluku, dobiva pozitivnu povratnu informaciju. Zbog toga, ulazna i izlazna vrijednost ne ovise jedna o drugoj. Kroz proces isprobavanja i pogrešaka, model

usavršava strategiju donošenja odluka, te uči činiti bolje izbore u različitim situacijama. Pojačano učenje se koristi u igricama, robotici i autonomnim sustavima, gdje model uči poduzimati akcije. To je moćan pristup rješavanju problema u kojima najbolji niz akcija nije unaprijed poznat, već se uči interakcijom s okolinom. [7]



Slika 6. Prikaz pojačanog učenja [8]

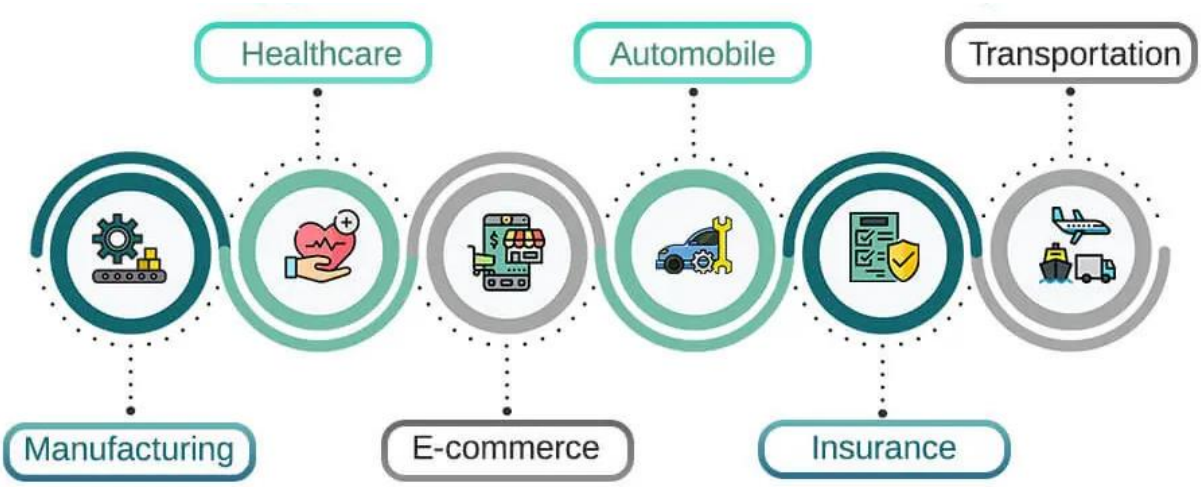
2.3 Duboko učenje

Pojam "duboko učenje" postao je popularan sredinom 2000-ih godina. Tada počinje snažan razvoj arhitekture neuronskih mreža, a duboko učenje svoju popularnost stječe zbog pristupnosti velikom broju skupova podataka, poboljšanju postojećih algoritama, te napretku u području hardvare-a, primjerice grafičkih procesora koji su omogućili bržu obradu podataka. Duboko učenje (engl. *deep learning*) je jedna od metoda strojnog učenja koje se bavi učenjem računala da obrađuje podatke po uzoru na ljudski mozak. Na taj način je strojevima omogućeno učenje iz podataka i donošenje odluka bez eksplicitnog programiranja. Duboko učenje koristi različite slojeve neurona i metode neuronskih mreža za analizu i obradu podataka. To su primjerice duboke neuronske mreže (eng. *deep neural networks*), konvolucijske neuronske mreže (eng. *convolution neural networks*) i duboke mreže vjerovanja (eng. *deep belief network*). Primjena dubokog učenja je raznovrsna. [7] Zahvaljujući razvitku i napretku umjetnih neuronskih mreža, iste pronalaze primjenu u područjima gdje su podatci velike dimenzionalnosti.

2.3.1 Primjena dubokog učenja

Neki od čestih primjena dubokog učenja su:

- Računalni vid – analiza slika, prepoznavanje lice, detekcija oblika, autonomna vožnja
- Obrada prirodnog jezika – razumijevanje i generiranje jezika, strojni prijevod, komunikacija čovjeka i računala (eng. *chatbot*)
- Medicina – analiza medicinskih slika, dijagnoza bolesti, otkrivanje i testiranje lijekova, planovi liječenja
- Autonomni sustavi – dronovi, industrijska automatizacija



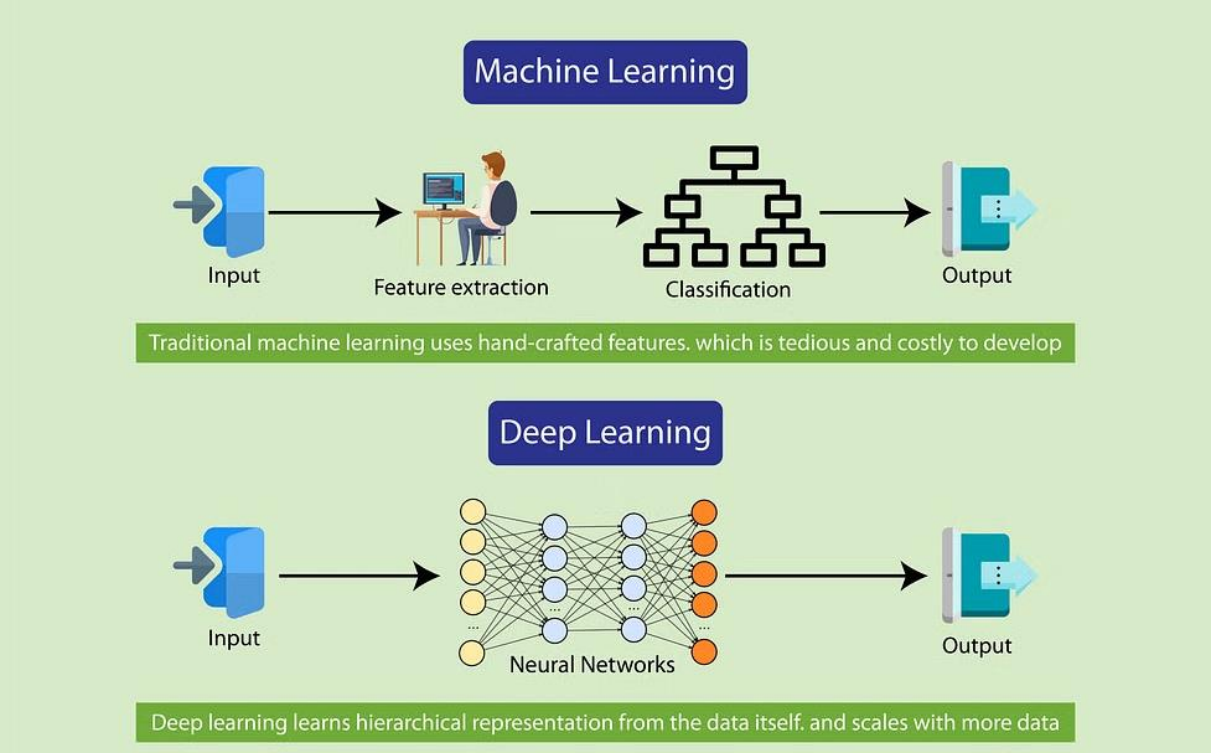
Slika 7. Primjene dubokog učenja [9]

2.3.2 Usporedba umjetne inteligencije, strojnog i dubokog učenja

Umjetna inteligencija je opći pojam kojem je cilj stvaranje inteligentnih strojeva. Strojno učenje je podskup umjetne inteligencije koji se bavi razvojem algoritama koji uče iz podataka. Duboko učenje je podskup strojnog učenja koji se fokusira na duboke neuronske mreže i njihovu sposobnost učenja složenih obrazaca. Iako su međusobno povezani i često se koriste zajedno, razlikuju se po svojem fokusu i metodologijama. Iz definicija i primjena, možemo zaključiti da je duboko učenje napredniji oblik strojnog učenja, te njemu podređen pojam. Glavna prednost, i ono što razlikuje strojno i duboko učenje je odsutnost ljudskog faktora. Kao što je ranije rečeno, duboko učenje koristi neuronske mreže, koje imaju mogućnost rada uz odsutnost dijela podataka. Stoga je glavna prednost dubokog učenja mogućnost rada bez ljudskog faktora.

Zaključno, strojno učenje koristi algoritme i uči samostalno, ali potrebna mu je ljudska intervencija za ispravak grešaka. Duboko učenje koristi složenije algoritme, vlastitu

neuronsku mrežu kako bi se prilagodilo i ispravilo grešku bez, ili uz minimalnu ljudsku intervenciju. O tome koje ćemo izabrati, ovisi o količini podataka i kompleksnosti problema. Najbolji primjer za razumijevanje kompleksnosti dubokog učenja je autonomna vožnja. Takav algoritam koristi mnoge slojeve neuronskih mreža kako bi izbjegao prepreke, objekte, prepoznao svjetla semafora i prema svim faktorima prilagodio brzinu. [11]
Usporedba strojnog, te dubokog učenja prikazana je na slici. [Slika 8.]



Slika 8. Usporedba strojnog i dubokog učenja [10]

2.4 Neuronske mreže

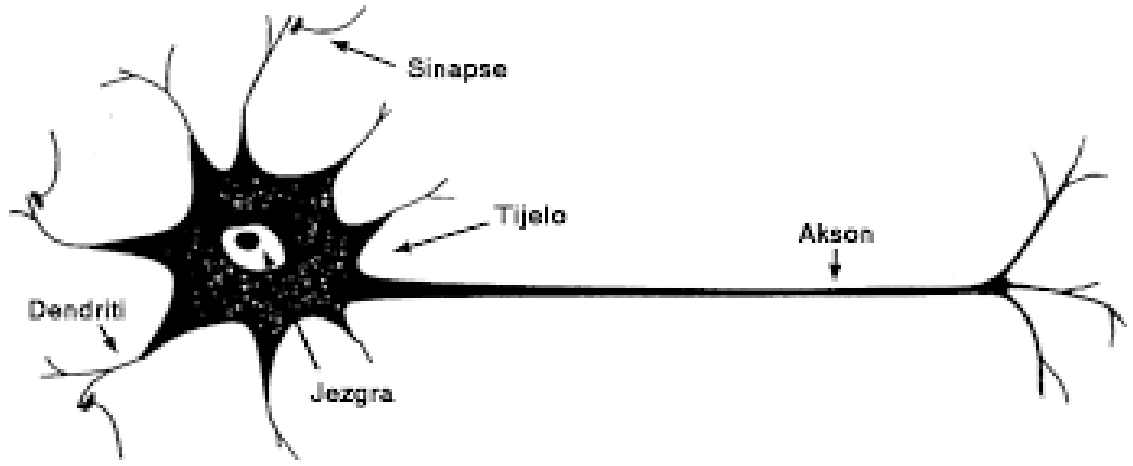
Umjetne neuronske mreže stvorene su u svrhu simulacije i oponašanja neurona u ljudskom mozgu. Takva ideja nastojala se uvesti u računala i strojeve, kako bi isti imali mogućnost razumijevanja i donošenja odluka poput ljudi. Dizajnirane su programiranjem i ponašaju se, te grade kao spojene moždane stanice. Naš se mozak ponaša kao paralelni procesor koji ima mogućnost povlačenja informacija iz različitih dijelova memorije, a to je i ideja umjetnih neuronskih mreža. Moć i važnost neuronskih mreža leži u sposobnosti učenja složenih veza iz datih podataka, čime one postaju ključni alat u području umjetne inteligencije. [32]

Neke od svojstva, a ujedno i prednosti umjetnih neuronskih mreža: [13]

- svestranost (mogu se primijeniti u širokom spektru, uključujući ekonomiju, medicinu, umjetnost, prepoznavanje govora, prevođenje, elektrotehniku, vojsku…)
- nelinearnost (modeli različite kompleksnosti mogu uspostaviti nelinearnu vezu između ulaznih i izlaznih podataka što je vrlo korisno kad se ne radi o jednostavnim linearnim uzorcima)
- neprestana poboljšanja
- redundantnost (neuronske mreže ne prestaju raditi čak i kad im se uništi ili izbriše neki dio)
- mogućnost rada bez obzira na odsutnost dijela podataka
- mogućnost stvaranja odnosa između podataka koji nisu eksplicitno zadani
- prilagodljivost (podešavanjem parametra, arhitekture mreže i podatcima za trening omogućuje se prilagodba i primjena na različite probleme i zadatke)

2.4.1 Građa i struktura biološkog neurona

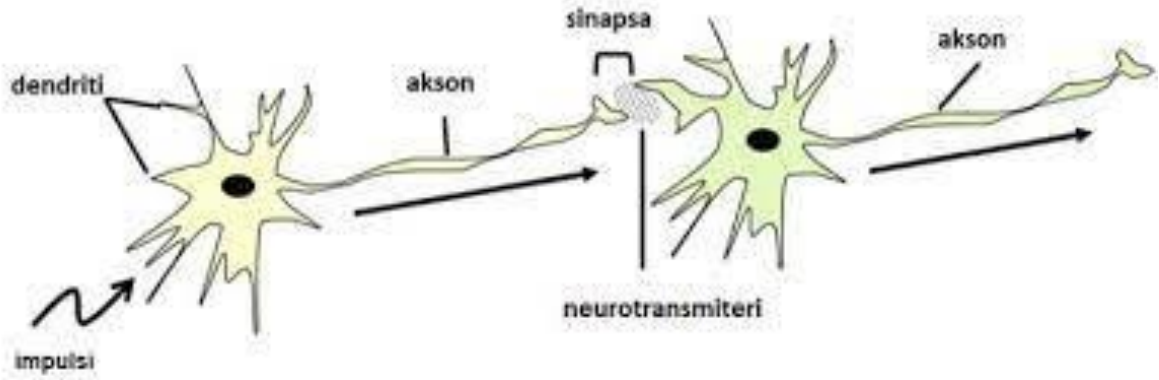
Biološki neuron sastoji se od dijelova prikazanih na slici. [Slika 9.] To su: tijelo, akson, sinapse i dendriti.



Slika 9. Struktura biološkog neurona [11]

U tijelu stanice nalazi se jezgra koja sadrži informacije o značajkama. Oko stanice tijela nalaze se dendriti koji prenose signale i primaju informacije drugih neurona, a aksoni prenose signale, odnosno informacije na sljedeći neuron. Prijenos se događa zbog elektrokemijske reakcije, uzrokovane oslobađanjem materijala potrebnog stanici za prijenos signala, koji se naziva transmiter. Takav prijenos omogućuju sinapse koje se nalaze između završetka aksona

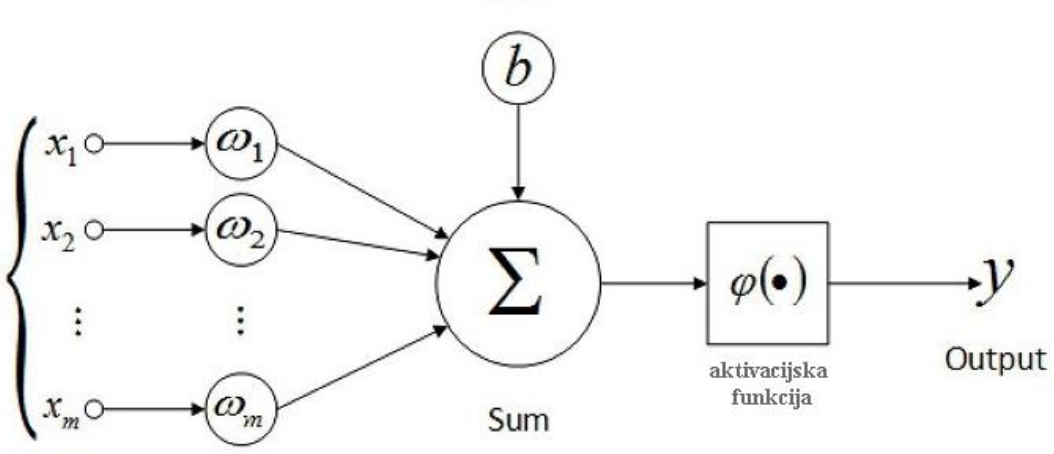
prethodnog neurona i dendrita sljedećeg neurona. Spoj dvaju neurona prikazan je na slici. [Slika 10.]



Slika 10. Spoj dvaju neurona [12]

2.4.2 Model umjetnog neurona

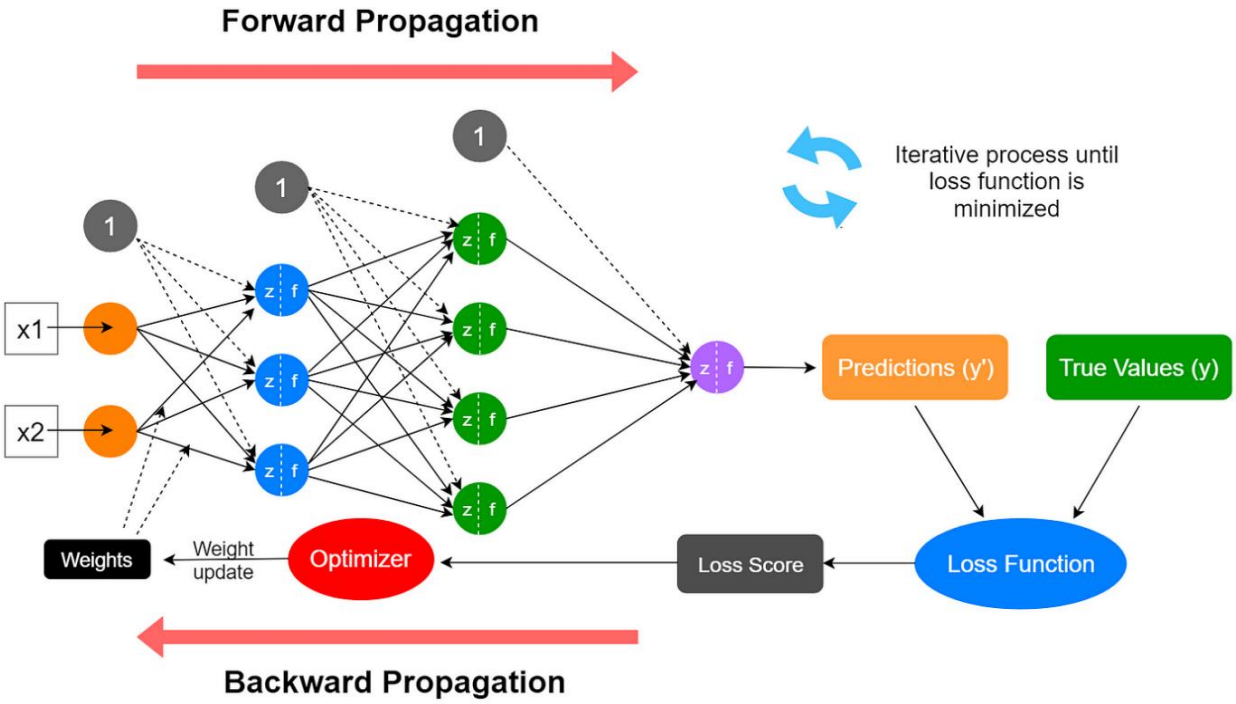
Umjetni neuroni kreirani su po principu biološkog, što je objašnjeno u prethodnom odlomku. Svakom signalu (x_1, x_2, x_m) dodaje se numerička vrijednost. Na ulazu u neuron signali se množe težinskim faktorom (ω_1, ω_2, ω_m). Težinski faktori analogni su dendritima biološkog neurona, odnosno govore o jakosti sinapse, spoju dva neurona. Umnožak signala i težinskih faktora potrebno je sumirati. Ako je zbroj signala veći od praga osjetljivosti neurona, nelinearna aktivacijska funkcija generira izlazni signal neurona iznosa. Prijenos signala prikazan je na slici. [Slika 11.] [14]



Slika 11. Model umjetnog neurona [12]

2.4.3 Učenje neuronske mreže

Pod pojmom učenje neuronske mreže (*training, learning*), podrazumijevamo promjenu, odnosno optimizaciju težinskih faktora kako bi projektirana mreža dala bolje rezultate i imala što manje pogreške. Težinski faktori određuju se prema pravilima učenja koji su ključni u optimizaciji neuronske mreže. To su perceptron trening (eng. *perceptron training*), linearno programiranje, pravilo širenja unatrag, delta pravilo i algoritam unazadne propagacije izlazne pogreške (eng. *backward propagation*), te prolaz unaprijed (eng. *forward propagation*). Razlika prolaza unaprijed i unazad je vremenska faza u kojoj se odvijaju. Faza u kojoj se odvija prolaz unaprijed nazivamo faza predviđanja. To je faza u kojoj mreži dajemo ulaz, mreža ga obradi, te generira izlaz. Prolaz unazad se koristi tijekom treninga, pri ažuriranju težinskih faktora. Prolaz unaprijed i unazad prikazani su na slici. [Slika 12.]



Slika 12. Usporedba prolaza unaprijed i prolaza unatrag [Pogreška! Izvor reference nije p ronađen.**]**

Informacija prolazi kroz neuronsku mrežu i tvori vrijednost predviđanja koju je potrebno usporediti sa stvarnom vrijednošću. Težinski faktori u mreži se tada prilagođavaju na temelju razlike, tj. usporedbe stvarne (eng. *True values*) i predviđene vrijednosti (eng. *Predictions*). Prilagodba težinskih faktora je proces u kojem mreža uči bolje predviđati stvarne vrijednosti, a samim time i smanjivati razlike stvarne i dobivene vrijednosti.

2.4.4 Koraci razvoja neuronske mreže

Razvoj neuronske mreže uključuje niz koraka i iteracija kako bi se stvorio model koji može rješavati određeni problem. To su: [17]

1. Definicija problema – potrebno je jasno definirati problem kojeg želimo riješiti korištenjem neuronskih mreža, promisliti o načinu rješavanja istog (npr. klasifikacija, regresija…), te naći izvor podataka koji su potrebni za rješavanje istog .
2. Sakupljanje i priprema podataka – kako bi trening mreže mogao započeti, potrebno je unijeti ulazne podatke. U ovom koraku razvoja treba provesti pripremu podataka, njihovo čišćenje, skaliranje i podjelu (skupovi za učenje, provjeru i testiranje).
3. Stvaranje arhitekture mreže – arhitektura mreže podrazumijeva broj neurona u pojedinim slojevima mreže, a koji ovise o broju ulaznih i izlaznih signala. U ovom koraku odabiremo i aktivacijsku funkciju koja djeluje na sumu težinskih faktora i ulaznih signala, te funkciju gubitka koja mjeri razliku stvarnih i predviđenih vrijednosti.
4. Učenje neuronske mreže – s obzirom da je ovaj korak najvažniji za postizanje točnosti izrađene mreže, pobliže je opisan u odlomku ispred.
5. Provjera i testiranje mreže – kako bi konačno provjerili izrađen model, koristimo se spomenutim skupom podataka za testiranje. Ako skup ne daje zadovoljavajuće izlaze, potrebno je strukturu modela izmijeniti. To možemo napraviti promjenom sloja mreža, zamjenom broj izlaza i ulaza, te promjenom aktivacijske funkcije.
   Skup koraka prikazan je na slici. [Slika 13.]

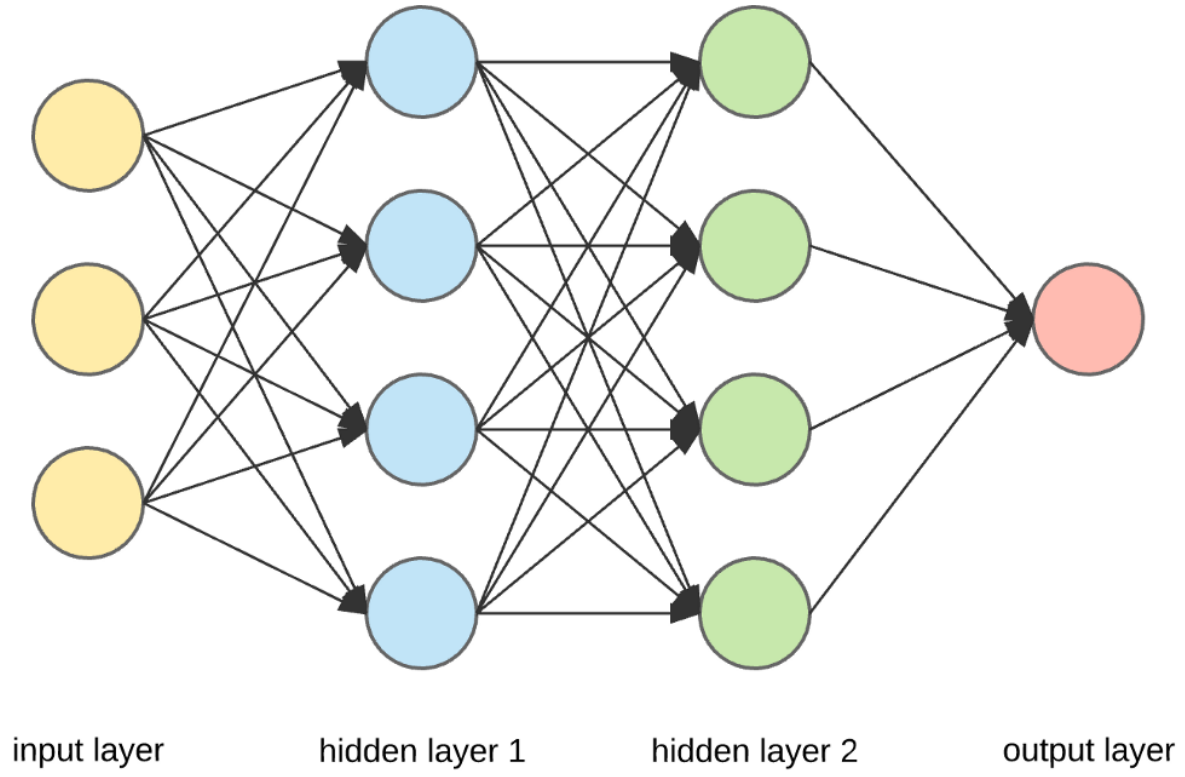
Slika 13. Proces razvoja neuronske mreže [Pogreška! Izvor reference nije pronađen.**]**

2.4.5 Vrste neuronskih mreža i njihovi slojevi

Ključni slojevi u neuronskoj mreži uključuju već spomenuti ulazni i izlazni sloj (eng. *Input and Output Layer*), te skrivene slojeve (eng. Hidden Layers). Skriveni slojevi nalaze se između ulaza i izlaza, te služe za ekstrakciju značajki iz ulaznih podataka. Sastoji se od niza neurona koji obrađuju podatke i prenose ih dalje. Neuronske mreže s više sakrivenih slojeva nazivaju se dubokim neuronskim mrežama (eng. *Deep neural netoworks*). Arhitektura i broj slojeva u neuronskoj mreži zavise o specifičnom zadatku i problemu koji se rješava.
Postoji više vrsta neuronskih mreža, a svaka je kreirana za specifičan problem i primjenu. [1735] Najčešće korištene su:

- Konvolucijske neuronske mreže (eng. *Convolutional Neural Network*) – koriste se za analizu slika i videa, primjena konvolucijskih slojeva za automatsko učenje karakteristike iz podataka
- Neuronske mreže s progresijom unaprijed (eng. *Feedforward Neural Network*) – najjednostavniji tip neuronske mreže gdje informacije teku u jednom smjeru, iz ulaznog do izlaznog sloja

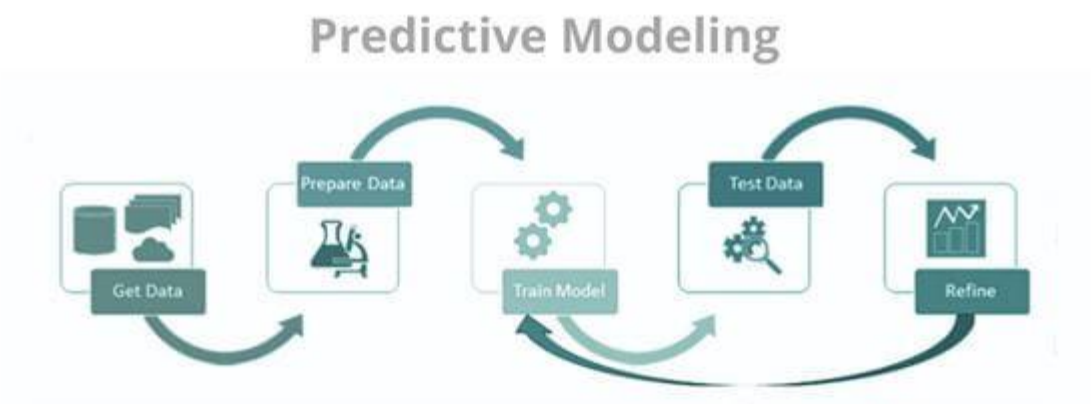
- Rekurentna neuronska mreža (eng. Recurrent Neural Network) – koriste se za obradu prirodnog jezika, prepoznavanje govora, strojni prijevod i generiranje teksta. Pogodna je za nizove podataka kao što su tekst i govor.



Slika 14. Arhitektura neuronske mreže [19]

2.5 Modeli predviđanja

Modeli predviđanja (eng. *Prediction models*) su matematički algoritmi ili modeli napravljeni da bi radili predviđanja o budućim događajima, ishodima ili vrijednostima, a temelje se na povijesnim ili ulaznim podatcima. Koriste se za prognoze i aproksimacije. Opće smjernice za izradu modela predviđanja uključuju definiciju problema, sakupljanje podataka, obradu podataka, arhitekturu značajki, podjelu podataka po skupovima, odabir modela, trening modela, evaluaciju i interpretaciju rezultata modela, te na kraju dokumentaciju. Stvaranje dobrog modela predviđanja često podrazumijeva iterativni proces. Proces je prikazan na slici. [Slika 15.] Primjena modela za predviđanje u poslovnom svijetu donosi brojne koristi, uključujući poboljšanu efikasnost i smanjenje rizika. Koriste se u financijskoj analizi, upravljanju rizicima, smanjenju troškova i povećanju efikasnosti proizvodnje, optimizaciji procesa, predviđanju bolesti na temelju prijašnjih stanja… [31]



Slika 15. Proces izrade modela predviđanja [20]

3. IZVEDBA ZADATKA

3.1 Web aplikacija

Web aplikacije postaju ključni dio modernizacije, odnosno digitalizacije u sektoru zdravstva. Njihova primjena ima pozitivan utjecaj za cjelokupni zdravstveni sustav, ne samo pacijente već i zdravstvene radnike. Razvoj istih došao je kao odgovor na izazove koji uključuju efikasnost, pristupačnost i kvalitetu pružanja usluga. Web aplikacije zdravstvu donose brojne prednosti, pa su tako medicinske informacije nikad dostupnije.

3.2 Programski jezik Python

Programski jezik Python napisao je Guido van Rossum, a prvi put je objavljen 1991. godine. Python je objektno orijentirano programski jezik dizajniran s fokusom na laku i intuitivnu čitljivost koda, te jednostavnoj sintaksi. To ga čini prikladnim za početnike, ali i kvalitetnim i moćnim alatom za iskusnije programere. Python je jezik otvorenog koda, što znači da je besplatan za korištenje i distribuciju. Takva otvorenost omogućila je brz razvoj tog programskog jezika, pružajući podršku kroz različite biblioteke i module. Python je stekao popularnost zbog svoje svestranosti i široke mogućnosti primjene. Koristi se u web razvoju, analizi podataka, umjetnoj inteligenciji, automatizaciji, ekonomiji, te mnogim drugim područjima. U ovom radu korišten je Python 3.12 jer sadrži biblioteke potrebne za razvoj modela predviđanja, a koji je cilj ovog rada. Kao što je ranije spomenuto, biblioteke potrebne za složeniju analizu i razvoj, a koje nisu implementirane u 3.12 verziji, instalirane su koristeći terminal (eng. *Command Prompt*) u Windowsu. Korištene biblioteke i moduli navedeni su u nastavku. [19]

Slika 16. Logo programskog jezika Python [19]

3.2.1 NumPy

NumPy je jedna od najpopularnijih biblioteka za programski jezik Python. Jedinstvena je jer pruža podršku za rad s višedimenzionalnim poljima (eng. *arrays*) i matricama, zajedno s kolekcijom matematičkih funkcija koje olakšavaju matematičke operacije na strukturama podataka. To uključuje osnovne operacije (zbrajanje, oduzimanje, množenje, dijeljenje), trigonometrijske funkcije, logaritme, eksponencijalne funkcije, Fourierove transformacije… NumPy ima ključnu ulogu pri programiranju u Pythonu i omogućava kvalitetno i točno rukovanje numeričkim podacima. [20]

Slika 17. Logo biblioteke NumPy [20]

3.2.2 Pandas

Pandas je biblioteka programskog jezika Python koja pruža efikasne strukture podataka za rad s tabličnim podacima, vremenskim serijama i drugim vrstama podataka. Ova biblioteka često se koristi u analizi podataka, manipulaciji podacima i pripremi podataka za strojno učenje. Jedna od najvažniji karakteristika i mogućnosti je struktura podataka poznata kao Podatkovni okvir (eng. *dataframe)*. *Dataframe* predstavlja dvodimenzijsku tablicu (tablice i stupci), te omogućava jednostavan i efikasan rad s podatcima. Osim tablične podjele, važno je spomenuti strukturu podataka s nazivom Serije (eng. *Series)*, a koja predstavlja jednodimenzijski niz podataka, te je osnovna jedinica za rad s podacima. [21]

Slika 18. Logo biblioteke Pandas [21]

3.2.3 Scikit – learn

Scikit-learn je popularna biblioteka za strojno učenje u programskom jeziku *Python*. Često se naziva *sklearn*. Ova biblioteka predstavlja jednostavan i efikasan alat za analizu i modeliranje podataka. Uključuje algortime strojnog učenja za klasifikaciju (npr. *Support Vector Machines*), regresiju (npr. *Linear Regression*), klasteriranje (eng. *Clustering*), te smanjenje dimenzionalnosti. [23]

Slika 19. Logo biblioteke Scikit-learn [23]

3.2.4 Pickle

Modul *pickle* implementira binarne protokole za serijsko (eng. serializing) i de-serijsko (eng. *deserializing*) pretvaranje Python strukture objekta. „*Pickling*" je postupak kojim se Python hijerarhija objekata pretvara u niz bajtova, a "*unpickling*" je obrnuta operacija. Pri obrnutoj operaciji se niz bajtova ponovno pretvara u hijerarhiju objekata.
Pickle radi s binarnim podacima, što znači da može serijalizirati i de-serijalizirati bilokoji Python objekt, uključujući kompleksne strukture podataka, klase i funkcije.

3.2.5 Streamlit

Streamlit je alat koji omogućava brzu i jednostavnu izradu web aplikacija. To je Python biblioteka koja ima mogućnost transformacije analiziranih podataka i modela strojnog učenja u interaktivnu web aplikaciju. Ova biblioteka se ističe jednostavnošću i minimalizmom, uz postizanje maksimalne funkcionalnosti. Jedna od ključnih karakteristika je automatsko osvježavanje (eng. *refresh*). Ta je karakteristika vrlo praktična za razvoj jer ubrzava iterativne operacije u procesu programiranja. Što se tiče umetanja interaktivnih elemenata, jasno su definirane kratke i jednostavne sintakse što omogućava korisniku istraživanje podataka i prilagodbu parametra u realnom vremenu. [24]

Slika 20. Logo Streamlit biblioteke [24]

3.2.6 Keras

Keras, kao otvorena biblioteka, pruža korisničko sučelje za implementaciju umjetnih neuronskih mreža. Njezina filozofija naglašava brzo ostvarivanje ciljeva i smanjenje vremena od ideje do postizanja rezultata. Popularna je zbog pristupačnosti i upotrebljivosti, omogućujući rad i na mobilnim uređajima i najmoćnijim grafičkim procesorima. Keras se ističe po svojoj jednostavnosti, fleksibilnosti i moći u rješavanju različitih problema. [25]

3.2.7 TensorFlow

TensorFlow, otvorena biblioteka, doprinosi istraživanju i razvoju u području strojnog učenja. Ova biblioteka ističe se obuhvatnim i prilagodljivim alatima i knjižnicama namijenjenim strojnom učenju. Posebno je pogodna za izradu višeslojnih neuronskih mreža te se često koristi u zadacima poput klasifikacije, razumijevanja, percepcije i predviđanja. TensorFlow podržava korištenje u različitim programskim jezicima, pružajući fleksibilnost koja omogućuje razvoj i primjenu u različitim sektorima i industrijama. [26]

3.2.8 Spyder

Spyder je integrirano razvojno okruženje (IDE) otvorenog koda za rad s programskim jezikom Python. Posebno je prilagođeno za rad u područjima gdje je ključna analiza podataka. Kod za ovaj rad pisan je koristeći *Spyder*. [27]

3.3 Priprema baze podataka

Bolesti za koje je u ovom radu izrađen model predviđanja su dijabetes, Parkinsonova bolest, te bolest srca. Prije izrade modela predviđanja, potrebno je napraviti bazu podataka za svaku od njih koju ćemo koristi kao skup za učenje, provjeru i testiranje.

3.3.1 Baza podataka za dijabetes

Za izradu baze podataka za dijabetes korištena je .csv datoteka. Pri izradi su istraženi trendovi i korelacije kod dijagnosticiranja dijabetesa. U zadnjih nekoliko godina, sve više se nagađa o povezanosti dijabetesa s brojem trudnoća. Dokazano je da su žene s većim brojem trudnoća sklonije takvom tipu bolesti. Sljedeći važan podatak za dijagnosticiranje dijabetesa je razina glukoze u krvi, gdje se normalnom razinom smatra između 70 i 100 mg/dL, te donji dijastolički tlak koji predstavlja tlak u arterijama dok srce odmara između dva otkucaja. Zdravim, i normalnim dijastoličkim tlakom smatraju se vrijednosti između 80 i 90 mmHg. Šećerna bolest utječe i na neke funkcionalne karakteristike epiderme, te je tako dokazano stanjivanje epiderme. Iz tog razloga, za ovaj model, uzet je u obzir i debljina kože. Naravno, važno je obratiti pozornost i na razinu inzulina u krvi. Inzulin je hormon koji luči žlijezda gušterača (pankreas), a služi za regulaciju šećera u krvi. U njegovom nedostatku, regulacija je otežana, te se može smatrati okidačem za šećernu bolest. U bazu podataka uključen je i BMI, tj. indeks tjelesne mase (eng. *Body mass index*), s obzirom na često spominjanje korelacije debljine i dijabetesa. Ako je BMI visok, postoji mogućnost povećanog rizika od razvoja dijabetesa tipa 2. Kao popularna karakteristika u dijagnozi, koristi se i etimologija, te se tako izračunava i „*Diabetes Pedigree Function*“ (DPF). DPF izračunava vjerojatnost pojave dijabetesa ovisno o dobi subjekta i obiteljskoj povijesti dijabetesa. Za kraj, potrebno je osigurati da naša baza podataka ima varijablu, odnosno „*target*“ koji govori ima li osoba dijabetes. Vrijednosti koje dobiva ta varijabla su nula ili jedinica. Nula za osobu koja nema rizik od dijabetes, a suprotno za jedinicu. Za varijable trudnoća, razina glukoze, razina inzulina i ostalo navedeno istražene su brojčane vrijednosti koje se koriste u medicinskoj dijagnostici. Uvid u dio baze podataka dan je slikom. [Slika 21.] [28]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
..	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1
..	...	...	...
763	0.171	63	0
764	0.340	27	0
765	0.245	30	0
766	0.349	47	1
767	0.315	23	0

[768 rows x 9 columns]

Slika 21. Baza podataka za dijabetes

3.3.2 Baza podataka za srčane bolesti

Za izradu baze podataka za srčane bolesti također je korištena .csv datoteka. Kako bi model bio što kvalitetniji, istraženi su najvažniji parametri kod dijagnostike bolesti srca. Općenito, rizik od bolesti srca raste s godinama. S godinama ljudi mogu akumulirati izloženost raznim čimbenicima rizika poput visokog krvnog tlaka, visokog kolesterola i dijabetesa, što dodatno može povećati rizik od bolesti srca. Tako je jedan od uvjeta kod ovog modela baš dob. Sljedeća važna karakteristika je spol. Povijesni gledano, bolesti srca češće su povezane s muškarcima. Muškarci često doživljavaju bolesti srca u mlađoj dobi u usporedbi s ženama. Pri dijagnostici srčanih bolesti, veliki rizik su i razina šećera, te krvni tlak. Više razine šećera u krvi često oštećuju krvne žile i srce. Zbog toga osobe s visokom razinom šećera u krvi imaju veći rizik od razvoja bolesti srca. Što se tiče tlaka, hipertenzija, tj. visoki krvni tlak, dobro je poznati faktor rizika za bolest srca. Visok krvni tlak dovodi do zadebljanja stijenki arterija, što otežava protok krvi. Povećano opterećenje srca pridonosi srčanim udarima, ali i moždanim udarima. Također, kolesterol igra važnu ulogu u razvoju bolesti srca.

Važnu ulogu u dijagnostici igra i ultrazvučna pretraga koja bojama prikazuje i analizira protok, brzina, te smjer. Takva pretraga daje dobar uvid u krvožilni sustav. Isto tako važna pretraga koja je uzeta u obzir pri kreiranju baze podataka jest elektrokardiogram (EKG). Rezultati EKG-a pružaju informacije o ritmu srca, veličini srčanih komora, provođenju električnih impulsa kroz srce te mogućim oštećenjima srčanog mišića. Zadnja vrijednost koja je uzeta u obzir je maksimalan broj otkucaja srca. Broj otkucaja srca govori nam o aritmiji srca koja uvelike može utjecati na bolesti srca. Uvid u dio baze podataka dan je slikom. [Slika 22.] [29]

	age	sex	trestbps	chol	fbs	restecg	thalach	ca	target
0	63	1	145	233	1	0	150	0	1
1	37	1	130	250	0	1	187	0	1
2	41	0	130	204	0	0	172	0	1
3	56	1	120	236	0	1	178	0	1
4	57	0	120	354	0	1	163	0	1
..	...	...	...	...	...	...	...	..	...
298	57	0	140	241	0	1	123	0	0
299	45	1	110	264	0	1	132	0	0
300	68	1	144	193	1	1	141	2	0
301	57	1	130	131	0	1	115	1	0
302	57	0	130	236	0	0	174	1	0

[303 rows x 9 columns]

Slika 22. Baza podataka za srčane bolesti

3.3.3 Baza podataka za Parkinsonovu bolest

Baza podataka za Parkinsonovu bolest preuzeta je, a temelji se na istraživanju koje je provelo američko sveučilište na temelju promjena u frekvenciji glasa. Dijagnoza Parkinsonove bolesti mogla bi se sve češće postavljati na temelju simptoma karakterističnih za promjene glasa. Neke od motoričkih promjena kod pacijenata s Parkinsonovom bolesti su akintetički tremor, rigidnost, bradikineza i posturalna nestabilnost. Akintetički tremor najčešće se pojavljuje u frekvenciji 4 do 6 Hz, a s vremenom se pojavljuje u duljim razdobljima. Tremor podrazumijeva drhtanje. Rigidnost se definira kao povećan otpor pri izvođenju pasivnih kretnji. Najčešće se javlja u području vrata i zdjelice, ali i u udovima. Bradikineza podrazumijeva teškoće u pokretu. Pokreti koji su smatrani automatiziranim, primjerice hod i gestikulacije postaju otežani. Glas je smanjenog volumena, što se naziva hipofonija. [30]

3.4 Učitavanje biblioteka

Za svaku od bolesti, potrebno je prvo napraviti model predviđanja. Za izradu modela korištene su ranije spomenute biblioteke koje koristi Python. One omogućuju rad s podacima, standardizaciju, izgradnje, te evaluaciju modela strojnog učenja. Dio koda kojim su uvezene potrebne biblioteke i alati, prikazan je na slici. [Slika 23.]

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Slika 23. Učitavanje biblioteka

3.5 Analiza baze podataka

Analiza baze podataka je sastavni dio procesa izrade modela predviđanja. Ona daje temelj za donošenje odluka u početnoj fazi razvoja modela, što vodi do preciznije modela. Dio koda gdje se odrađuje analiza podataka, dan je slikom. [Slika 24.]

```
# getting more information about the dataset
diabetes_dataset.info()
```

Slika 24. Dohvaćanje informacija o bazi podataka

Odziv na kod prikazan je slikom. Ovim dijelom koda dobivene su informacije o tipovima podataka koji su obuhvaćeni (npr. *integer* i *float*), te broju stupca i redaka, odnosno dimenziji skupa podataka. Ova informacija je korisna jer govori koje vrste podataka očekivati u svakom stupcu, što je važno prilikom analize i obrade podataka te izrade modela strojnog učenja.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Slika 25. Informacije o bazi podataka

3.6 Treniranje modela

Treniranje modela znači pronaći optimalne težine za ulazne značajke kako bi se postigla što bolja predikcija. Potrebno je stvoriti model koji predstavlja logističku regresiju.. U *scikit-learnu*, ovaj objekt sadrži parametre i metode potrebne za treniranje i predviđanje s logističkom regresijom. Dio koda prikazan je slikom. [Slika 26.]

```
model = LogisticRegression()
```

Slika 26. Kod modela za logističku regresiju

U logističkoj regresiji, linearna kombinacija ulaznih značajki transformira se pomoću logističke funkcije kako bi se dobila vjerojatnost pripadnosti jednoj od dvije klase (uobičajeno označenih kao 0 i 1). Metoda fit u logističkoj regresiji koristi se za treniranje modela. Tijekom treniranja, model pokušava prilagoditi težine kako bi minimizirao razliku između stvarnih klasifikacija u trening skupu i onih koje model predviđa. Taj dio koda prikazan je na slici.

```
#training the  model with Training Data
model.fit(X_train, Y_train)
```

Slika 27. Prikaz funkcije za treniranje

X_train predstavlja ulazne podatke koje model koristi za učenje, a *Y_train* su odgovarajuće oznake koje model pokušava naučiti predviđati. Tijekom ovog procesa, model prilagođava svoje parametre kako bi što bolje odgovarao podacima.
Nakon završetka ove linije koda, model je treniran i spreman za predviđanje na novim, tj. neviđenim podacima.

3.7 Točnost

Točnost (eng. *accuracy*) je važna jer pruža jednostavan način za procjenu općenite učinkovitosti napravljenog modela. Jednostavno izražava omjer točnih predikcija prema ukupnom broju predikcija. Točnost pruža globalni pregled performansi modela na cijelom skupu podataka. Dio koda koji analizira točnost prikazan je na slici. [Slika 28.]

```
# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

Slika 28. Kod za utvrđivanje točnosti modela

Za ovaj skup podataka postignuta je točnost od 78.5%. To znači da izrađeni model ispravno klasificira oko 78.5% instanci u skupu za treniranje, što je prikazano na slici. [Slika 29.]

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  0.7850162866449512
```

Slika 29. Postignuta točnost

3.8 Izrada modela predviđanja

Konačno, potrebno je unijeti dio koda koji služi treniranom modelu za predviđanje temeljeno na ulaznim podacima. U ovom dijelu koda definirani su ulazni podatci koji predstavljaju unaprijed definirane ulazne značajke (broj trudnoća, razina glukoze, BMI…), nakon čega su pretvarane u niz. Na kraju na ekranu ispisan je rezultat koji daje odluku; je li osoba dijabetičar ili nije. Taj dio koda prikazan je na slici. [Slika 30.]

```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)


prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

Slika 30. Kod za izradu modela predviđanja

3.9 Spremanje modela

Za kraj, korištena je biblioteka *pickle* spomenuta ranije, kako bi se model spremio u datoteku. Taj se proces naziva „pickling“. Ta tehnika omogućuje spremanje treniranog modela kako bi se mogao ponovno koristiti u budućnosti, čime se izbjegava potrebu za ponovnim treniranjem svaki put kada se želi koristiti model. U ovom je radu model korišten za web aplikaciju, te je iz tog razloga spremljen. Kod za spremanje prikazan je na slici. [Slika 31.]

```
filename = 'diabetes_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

```
loaded_model= pickle.load(open('diabetes_model.sav', 'rb'))
```

Slika 31. Kod za spremanje modela

Nakon što je model spremljen, prelazimo na izradu modela za predviđanje srčane bolesti, te Parkinsonove bolesti.

3.10 Model predikacije korištenjem neuronske mreže

Isti problem, tj. model predikcije napravljen je i korištenjem neuronske mreže. Logistička regresija je linearni model, dok je neuronska mreža složeniji, nelinearni model koji dobro radi sa složenim uzorcima u podatcima. Sljedeća razlika je razlika u odabiru biblioteka. Naime, logistička regresija koristi *skicit-learn*, dok neuronske mreže koriste *TensorFlow* i *Keras*.

Postupak i postignuta točnost dobivena arhitekturom neuronske mreže prikazani su u nastavku.

3.10.1 Kreiranje modela neuronske mreže

Sekvencijalni model koji je kreiran je osnovni *Keras* model s nazivom *Sequential*. Takav model podrazumijeva jednostavnu linearnu raspodjelu sloj po sloj. Svaki sloj ima ulaze iz prethodnog sloja, te izlaze u sljedeći. Ulazni sloj dodavan je naredbom *Dense*. Broj neurona u tom sloju je 12, a naredbom *input_dim=X.shape* određen je broj ulaza u sloj. Aktivacijska funkcija kojom je dodana nelinearnost modelu jest *reLU*. Drugi, skriveni sloj, ima 8 neurona i *reLU* aktivacijsku funkciju. Ovaj dio modela pomaže učenje složenijih značajki iz podataka. Treći *Dense* sloj predstavlja izlazni sloj, te je za njega korištena aktivacijska funkcija *sigmoid*. Taj dio koda prikazan je na slici.

```
# Creating a neural network model
model = Sequential()
model.add(Dense(12, input_dim=X.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Slika 32. Kod modela za neuronsku mrežu

Model je potrebno trenirati. Kao i kod logističke regresije, korištena je funkcija *fit*. Određeno je 50 epoha. Epohe određuju broj prolaza kroz skup podataka, a *batch size* označava broj uzoraka koji se koristi za ažuriranje težina u koracima optimizacije.
Model je potrebno evaluirati korištenjem metode *evaluate*, a dobivena točnost se pohranjuje u varijablu *training_data_accuracy*. Prolaz po epohama prikazan je slikom.

```
Epoch 1/50
62/62 [==============================] - 1s 1ms/step - loss: 0.6929 - accuracy: 0.5293
Epoch 2/50
62/62 [==============================] - 0s 1ms/step - loss: 0.6361 - accuracy: 0.7085
Epoch 3/50
62/62 [==============================] - 0s 1ms/step - loss: 0.5946 - accuracy: 0.7378
Epoch 4/50
62/62 [==============================] - 0s 1ms/step - loss: 0.5559 - accuracy: 0.7410
Epoch 5/50
62/62 [==============================] - 0s 1ms/step - loss: 0.5262 - accuracy: 0.7492
Epoch 6/50
62/62 [==============================] - 0s 1ms/step - loss: 0.5042 - accuracy: 0.7573
Epoch 7/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4877 - accuracy: 0.7671
Epoch 8/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4746 - accuracy: 0.7687
Epoch 9/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4656 - accuracy: 0.7769
Epoch 10/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4578 - accuracy: 0.7720
Epoch 11/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4524 - accuracy: 0.7818
Epoch 12/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4474 - accuracy: 0.7818
Epoch 13/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4444 - accuracy: 0.7801
Epoch 14/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4422 - accuracy: 0.7850
Epoch 15/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4384 - accuracy: 0.7801
Epoch 16/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4368 - accuracy: 0.7866
Epoch 17/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4341 - accuracy: 0.7899
Epoch 18/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4333 - accuracy: 0.7818
Epoch 19/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4316 - accuracy: 0.7866
Epoch 20/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4288 - accuracy: 0.7964
Epoch 21/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4274 - accuracy: 0.7866
Epoch 22/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4266 - accuracy: 0.7899
Epoch 23/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4253 - accuracy: 0.7883
Epoch 24/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4239 - accuracy: 0.7899
Epoch 25/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4216 - accuracy: 0.7932
Epoch 26/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4208 - accuracy: 0.7899
Epoch 27/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4205 - accuracy: 0.7964
Epoch 28/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4176 - accuracy: 0.7948
Epoch 29/50
62/62 [==============================] - 0s 1ms/step - loss: 0.4176 - accuracy: 0.8046
```

Slika 33. Prikaz epoha

Nadalje je potrebno napraviti polje (eng. *numpy array*) koji predstavlja nove ulazne podatke na temelju kojih želimo napraviti predviđanje. Korištenjem *scaler* naredbe, podaci se skaliraju

kako bi se uskladili s onima na kojima je model treniran. To je važno jer model očekuje ulazne podatke u istom obliku kao i tijekom treniranja. Metodom *predict* dobiva se predviđanje na skaliranim ulaznim podacima. Rezultat je vjerojatnost da ulazni podatak pripada pozitivnom razredu. Vjerojatnost koju model daje uspoređena je s vrijednošću 0.5. Ako je predviđena vrijednost veća od 0.5, smatra se da model predviđa pozitivan slučaj (1), a u suprotnom slučaju negativan slučaj (0). Kad je *binary_prediction* jednako 0, ispisuje se "The person is not diabetic", a u suprotnom se ispisuje "The person is diabetic". [Slika 34.]

```
69      # creating a button for Prediction
70
71      if st.button('Diabetes Test Result'):
72
73      # Convert the specific input values to integers if they are meant to be whole numbers
74          Pregnancies = int(Pregnancies)
75          BloodPressure = int(BloodPressure)
76          Age = int(Age)
77          Glucose = int(Glucose)
78          SkinThickness = int(SkinThickness)
79          Insulin = int(Insulin)
80
81          diab_prediction = diabetes_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,
82
83          if (diab_prediction[0] == 1):
84             diab_diagnosis = 'The person is diabetic'
85          else:
86             diab_diagnosis = 'The person is not diabetic'
87      st.success(diab_diagnosis)
88
```

Slika 34. Model predviđanja dijabetesa

3.11 Kreiranje web aplikacije

Potrebno je učitati model neuronske mreže napravljen u prethodnom koraku i učitati biblioteke potrebne za stvaranje aplikacije. Taj dio koda prikazan je na slici. [Slika 35.]

```
8   import pickle
9   import streamlit as st
10  from streamlit_option_menu import option_menu
11
12  #loading done models
13  diabetes_model = pickle.load(open('C:/Users/Lucija/Desktop/diplomski rad/models/diabetes_model.sav', 'rb'))
14  parkinsons_model = pickle.load(open('C:/Users/Lucija/Desktop/diplomski rad/models/parkinsons_disease_model.sav', 'rb'))
15  heart_disease_model = pickle.load(open('C:/Users/Lucija/Desktop/diplomski rad/models/heart_disease_model.sav', 'rb'))
```

Slika 35. Učitavanje modela

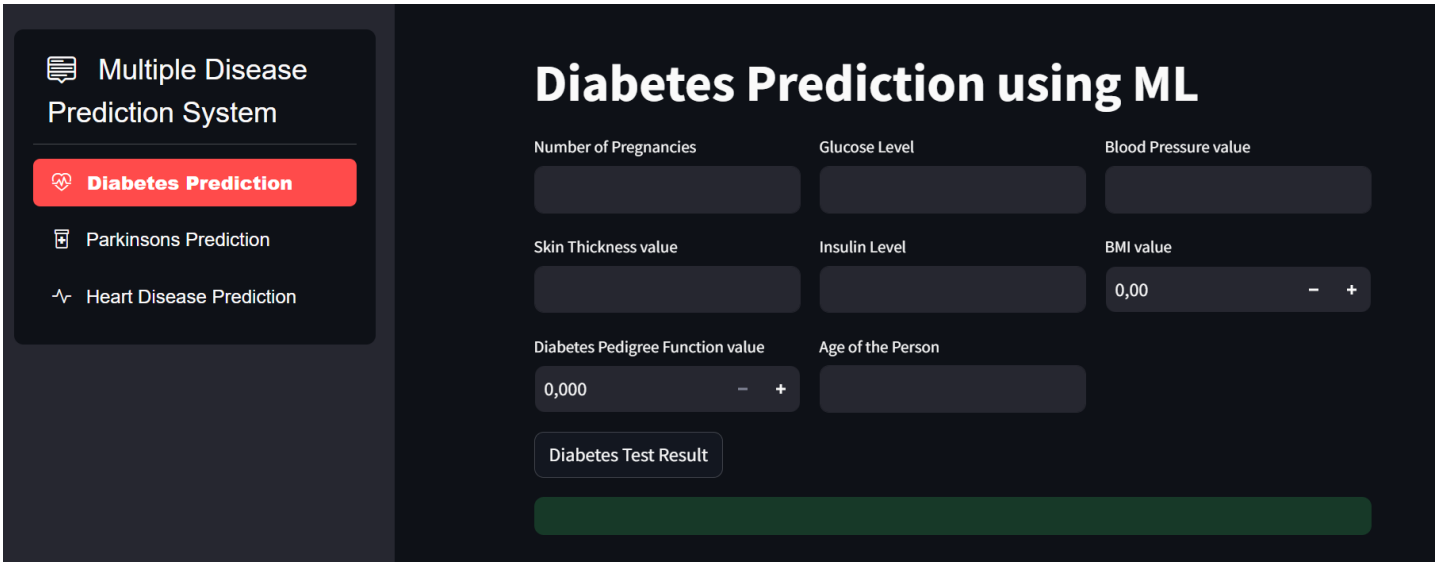
Nakon toga, potrebno je odrediti vizualni aspekt aplikacije. Moguće je uređivanje teksta, ikone, navigatora.

```
with st.sidebar:

    selected = option_menu('Multiple Disease Prediction System',
                           ['Diabetes Prediction','Parkinsons Prediction',
                            'Heart Disease Prediction'],

                           icons = ['heart-pulse','prescription2', 'activity'],

                           default_index = 0)
```

Slika 36. Vizualni aspekt aplikacije

Nakon što je kreiran vizualni dio, izrađeno je na korisničko sučelje. U sučelju je napravljena navigacija gdje korisnik može birati između tri opcije. To su dijabetes, bolesti srca i Parkinsonova bolest. Klikom na svaku od njih otvara se novi prozor u kojeg korisnik upisuje svoje simptome i informacije o stanju. Kreiran je i gumb kojem signaliziramo modelu da napravi predviđanje.

Konačan izgled web aplikacije prikazan je na slici. [Slika 37. Izgled web aplikacije



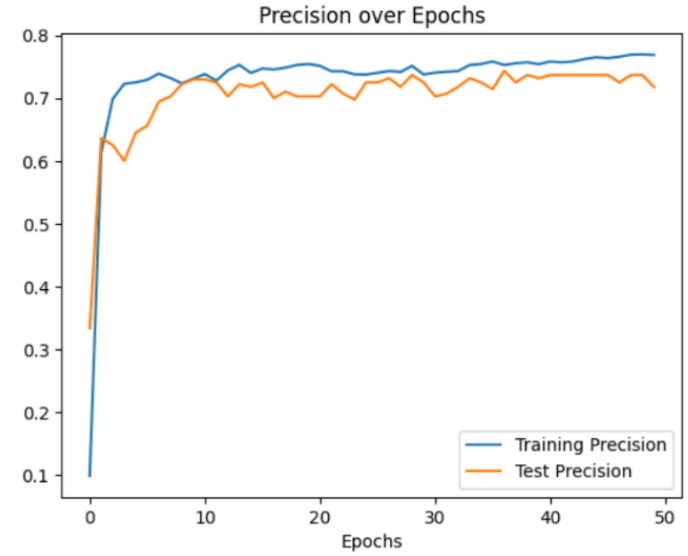
Slika 37. Izgled web aplikacije

3.12 Evaluacija modela

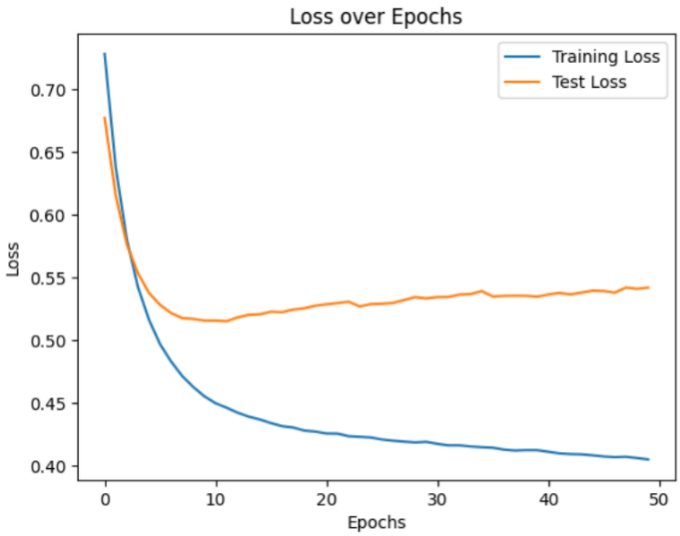
Ključna metrika koja se koristi za procjenu performansi modela su preciznost, te funkcija gubitka. Preciznošću se mjeri točnost pozitivnih predviđanja i zbroja stvarno pozitivnih, te lažno pozitivnih predviđanja. Funkcija gubitka daje predodžbu o razlici između predviđenih vrijednosti, te stvarnih vrijednosti. Cilj modela je da tu razliku minimizira.

3.12.1 Evaluacija modela za dijabetes

Slike prikazuju grafove točnosti i funkcije gubitka za model. Plavom je linijom prikazan skup podataka za trening, a narančastom skup za testiranje. Iz grafa preciznosti vidljivo je da je preciznost slična za skup podataka za treniranje i testnog skupa podataka. To znači da model prilično dobro predviđa za nove, još neviđene podatke. Iz grafa gubitka vidimo da oba skupa padaju po epohama, što je očekivano i dobro. Vidimo da krivulja podataka za testiranje kreće u lagani rast, što bi ukazivalo na pretreniranost mreže. Pretreniranost mreže može se izbjeći daljnjim poboljšanjima.



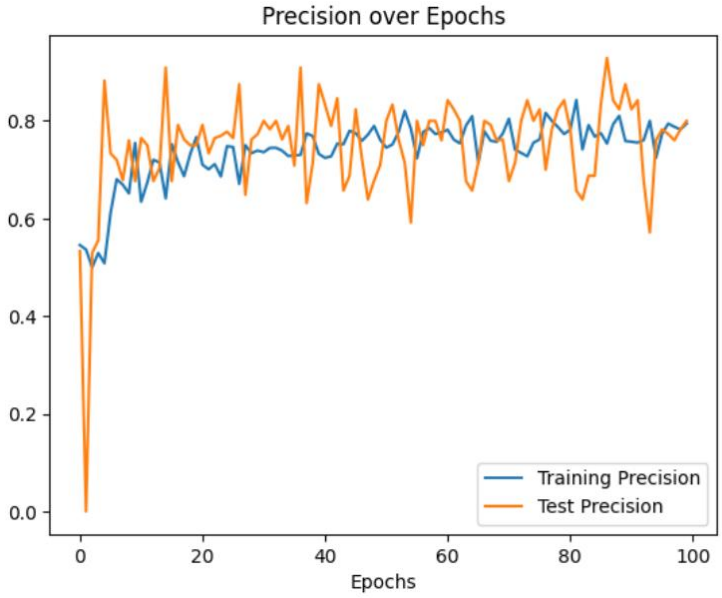
Slika 38. Model preciznosti



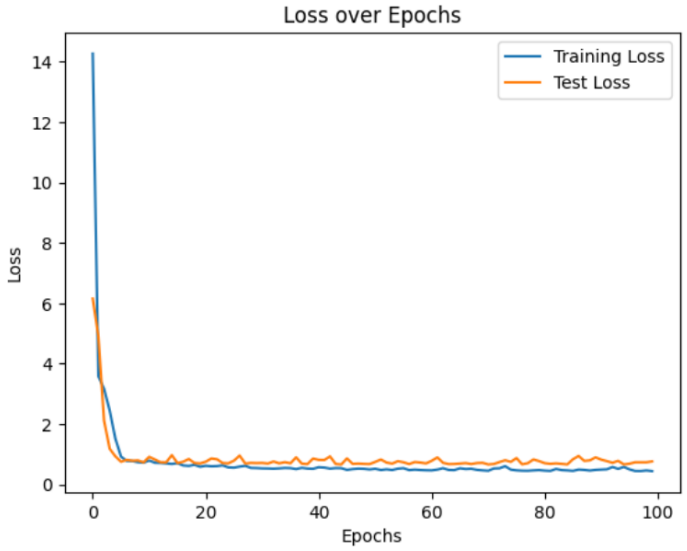
Slika 39. Funkcija gubitka

3.12.2 Evaluacija modela za srčane bolesti

Ponovno su napravljeni grafovi za funkciju gubitka, te preciznost. Narančasta linija predstavlja skup podataka za testiranje, a plava podatke za trening. Zanimljivo je vidjeti da kod funkcije preciznosti postoji veća amplituda, ali se na kraju skup podataka za treniranje i testni skup podudaraju. To nam pokazuje da i ovaj model dobro reagira na nove podatke, što je i cilj takvog modela. Funkcija gubitka je uobičajena, pada po epohama za oba skupa podataka, te se ne povećava. To znači da je razlika između predviđenih i stvarnih vrijednosti mala. Grafovi su prikazani na slikama. [Slika 40., Slika 41.]



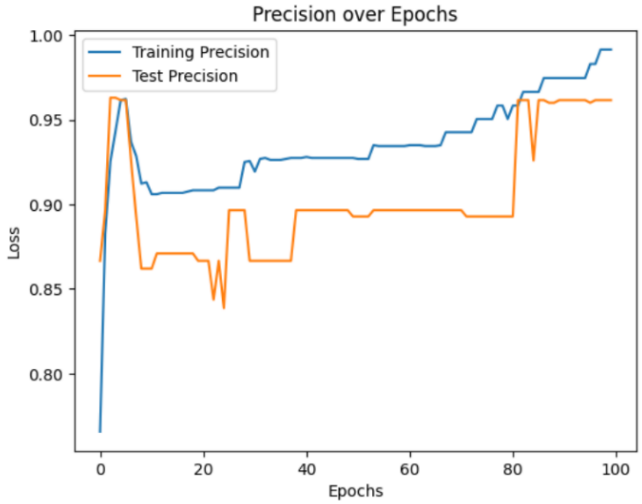
Slika 40. Model preciznosti



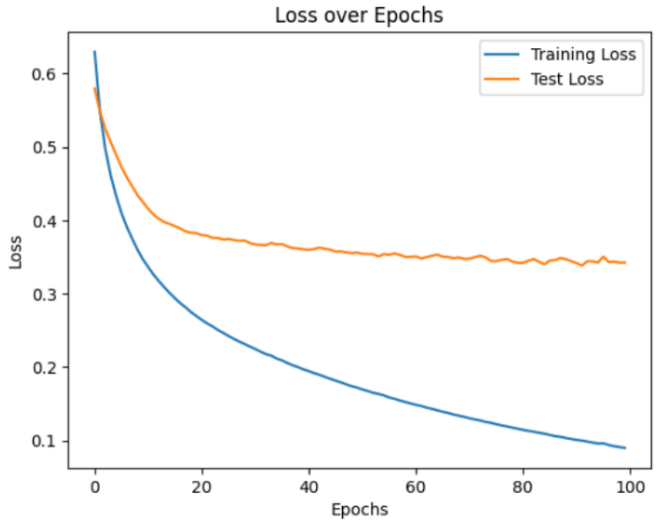
Slika 41. Funkcija gubitka

3.12.3 Evaluacija modela za Parkinsonovu bolest

Ponovno su napravljeni grafovi za funkciju gubitka, te preciznost. Narančasta linija predstavlja skup podataka za testiranje, a plava podatke za trening. Iz grafova vidimo da je preciznost viša za skup podataka za trening, što je uobičajeno. Dobro je da preciznost skupa podataka za testiranje raste po epohama, što dokazuje da je model uspješan. Što se tiče funkcije gubitka, ponovno krivulje prikazuju pad po epohama, te nema naknadnog rasta gubitka. Iz grafova možemo zaključiti da model vrlo dobro radi s novim podatcima, što je i bio cilj ovog rada. Grafovi su prikazani slikama. [Slika 42., Slika 43.]



Slika 42. Model preciznosti



Slika 43. Funkcija gubitka

3.13 Moguća poboljšanja i kritički osvrt

Iako modeli daju zadovoljavajuće rezultate, moguća su poboljšanja. Proces poboljšanja modela podrazumijeva iterativni postupak. Moguće je isprobavanje različitih tehnika, te pažljiva evaluacija performansi modela. Za početak, poboljšanje se može primijeniti na skup podataka koji koristimo za trening, pa je tako potrebno osigurati kvalitetne i dovoljno velike skupove podataka za treniranje. Ukoliko dolazi do promjena, model je potrebno ažurirati kako bi osigurali dobro reagiranje i ponašanje s promjenama u skupu podataka. Često se događa da model postane pretreniran (eng. *overfitted*). Pretreniranost je pojava u strojnom učenju gdje model postiže visoku točnost na skupu za učenje, ali loše se generalizira na neviđene podatke, odnosno testni skup podataka. To se dobro vidi na modelu za dijabetes, te na slici. [Slika 39.] Vidljivo je da gubitak lagano raste na krivulji podataka za testiranje. To bi se moglo spriječiti prikupljanjem dodatnih podataka, odnosno obogaćivanjem skupa za učenjem. Time se smanjuje mogućnost pretreniranja. Pri toj pojavi moguće je koristiti i tehniku regularizacije, kako bi model bio robusniji. Regularizacija u strojnom učenju je postupak podešavanja parametara koji ograničavaju, reguliraju ili smanjuju procjene koeficijenata. Drugim riječima, ova tehnika usmjerava model da ne nauči previše kompleksne ili fleksibilne obrasce, izbjegavajući time rizik od pretreniranja. Sljedeći korak koji je moguće poduzeti jest podešavanje hiperparametara, te vremena treniranja. Parametre je moguće podesiti koristeći skup za provjeru. Tako bi se optimizirale performanse za skup podataka za neviđene, odnosno testne podatke. Kad model i dalje ne bi davao zadovoljavajuće rezultate, trebali bi upotrijebiti različiti model i pristupiti problemu na kompleksniji način.

4. ZAKLJUČAK

U radu je prikazan razvoj web aplikacije za predviđanje triju različitih bolesti. U početku rada prikazana je, te detaljno pojašnjena teorijska osnova rada. Rad je izrađen korištenjem programskog jezika *Python*, te uz pomoć biblioteke *Streamlit* pomoću koje je rađena web aplikacija. Evaluacija modela je pokazala da su modeli vrlo dobro, te prate i dobro predviđaju unos novih podataka. Integracija modela u web aplikaciju omogućuje pristupačnost, te praktičnost korisnicima. Unatoč postignućima, prepoznati su izazovi u prikupljanju i korištenju medicinskih podataka. Naravno, za upotrebu u praksi potrebno je napraviti poboljšanja, ali odlično može poslužiti kao prototip. Budući rad trebao bi se usmjeriti na prikupljanje većih i raznovrsnijih skupova podataka, suradnju sa stručnjacima iz područja medicine, te osiguranje sigurnosti i daljnje unapređenje modela.

LITERATURA

1. History of Data Science. “Dartmouth Summer Research Project: The Birth of Artificial Intelligence.” History of Data Science, 2021.
2. www.historyofdatascience.com/dartmouth-summer-research-project-the-birth-of-artificial-intelligence/ preuzeto dana 15.10.2023.
3. https://www.infotech.com/research/ss/build-your-generative-ai-roadmap preuzeto dana 16.10.2023.
4. https://www.springboard.com/blog/data-science/regression-vs-classification/ preuzeto dana 16.10.2023.
5. https://hbr.org/topic/subject/ai-and-machine-learning preuzeto dana 17.10.2023.
6. https://medium.com/@stevenK8/logistic-regression-145b268a465a preuzeto dana 16.10.2023.
7. João Gama. Machine Learning : ECML 2005 : 16th European Conference on Machine Learning : Porto, Portugal, October 3-7 2005 : Proceedings. 2023. New York, Ny ; Great Britain, Springer, 2005.
8. https://www.mcg.ai/post/reinforcement-learning preuzeto dana 17.10.2023.
9. https://course.elementsofai.com/hr/1/1 preuzeto dana 17.10.2023.
10. https://www.researchgate.net/figure/The-difference-between-machine-learning-and-deep-learning-8_fig2_355050755 preuzeto dana 18.10.2023.
11. https://techvidvan.com/tutorials/unsupervised-learning/ preuzeto dana 16.10.2023.
12. https://www.google.com/url?sa=i&url=https%3A%2F%2Fhrcak.srce.hr%2Ffile%2F322233&psig=AOvVaw1kxTi4aP_CCEoiAopPQg2F&ust=1700200324239000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCKjx3fTpx4IDFQAAAAAdAAAAABAE preuzeto dana 17.10.2023.
13. https://www.google.com/url?sa=i&url=https%3A%2F%2Frepozitorij.pmf.unizg.hr%2Fislandora%2Fobject%2Fpmf%253A6644%2Fdatastream%2FPDF%2Fview&psig=AOvVaw1u9gHe76f-JPfoF1kaylI2&ust=1700200429923000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCIiPhKfqx4IDFQAAAAAdAAAAABAE preuzeto dana 17.10.2023.
14. Jun Wang, and Et Al. Advances in Neural Networks -- ISNN 2006 : Third International Symposium on Neural Networks, Chengdu, China, May 28-June 1, 2006 : Proceedings, Part III. Berlin, Springer, 2006., 70–120.
15. https://www.researchgate.net/figure/The-forward-propagation-of-a-neural-network-a-the-operating-process-of-neural-networks_fig1_355876971
16. https://www.newscientist.com/article-topic/machine-learning/ preuzeto dana 17.10.2023.
17. Sreeraj M, and Jestin Joy. An Introduction to Machine Learning. Sreeraj M, 1 Dec. 2019, pp. 113–150.
18. https://datascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation.com preuzeto dana 18.10.2023.
19. https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a preuzeto dana 18.10.2023.
20. https://en.wikipedia.org/wiki/Python_(programming_language) preuzeto dana 18.10.2023.
21. https://en.wikipedia.org/wiki/NumPy preuzeto dana 18.10.2023.
22. https://en.wikipedia.org/wiki/Pandas_(software) preuzeto dana 18.10.2023.
23. https://en.wikipedia.org/wiki/Scikit-learn preuzeto dana 18.10.2023.
24. https://streamlit.io/ preuzeto dana 21.10.2023.

25. https://keras.io/ preuzeto dana 21.10.2023.
26. https://www.tensorflow.org/ preuzeto dana 21.10.2023.
27. https://www.spyder-ide.org/ preuzeto dana 21.10.2023.
28. Sokolov, Vladimir. Dijabetes. 2009.
29. Haney, Johannah. Heart Disease. Benchmark Books, 2005., 105-121
30. https://www.michaeljfox.org/data-sets preuzeto dana 21.10.2023.
31. Tomislav Volarić, and Boris Crnokić. Umjetna Inteligencija U Obrazovanju I Robotici. 2022.
32. Charniak, Eugene, et al. *Artificial Intelligence Programming*. Psychology Press, 21 Jan. 2014. , 70–120
33. Sreeraj M, and Jestin Joy. An Introduction to Machine Learning. Sreeraj M, 1 Dec. 2019, pp. 113–150.
34. Berndt Müller, et al. Neural Networks. 2023. Springer Science & Business Media, 6 Dec. 2012., 20-87
35. Raschka, Sebastian, and Vahid Mirjalili. Python Machine Learning : Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow. 2023. Birmingham (Uk), Packt Publishing, 2017., 27-96

PRILOZI

I. Model predviđanja za dijabetes (logistička regresija)

II. Model predviđanja za dijabetes (neuronska mreža)

III. Model predviđanja za srčane bolesti

IV. Model predviđanja za bolest Parkinson

V. Učitavanje modela i kreiranje web aplikacije

I. Model predviđanja za dijabetes (logistička regresija)

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.metrics import accuracy_score

"""Data Collection and Analysis"""

# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')

# printing the first 5 rows of the dataframe
diabetes_dataset.head()

# number of rows and columns in the dataframe
diabetes_dataset.shape

# getting more information about the dataset
diabetes_dataset.info()

print(diabetes_dataset)

# checking for missing values in each column
diabetes_dataset.isnull().sum()

# getting some statistical measures about the data
diabetes_dataset.describe()

# number of rows and Columns in this dataset
diabetes_dataset.shape

# getting the statistical measures of the data
diabetes_dataset.describe()

diabetes_dataset['Outcome'].value_counts()

diabetes_dataset.groupby('Outcome').mean()

# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']

print (X)

print(Y)

"""Train Test Split"""

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

print(X.shape, X_train.shape, X_test.shape)

"""Training the Model"""

model = LogisticRegression()

#training the  model with Training Data
model.fit(X_train, Y_train)

```

```
"""Model Evaluation

Accuracy Score
"""

# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy score of the training data : ', training_data_accuracy)

"""Building a Predictive System"""

input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)


prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')

import pickle

filename = 'diabetes_model.sav'
```

```
pickle.dump(model, open(filename, 'wb'))

loaded_model= pickle.load(open('diabetes_model.sav', 'rb'))
```

II. Model predviđanja za dijabetes (neuronska mreža)

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
# Loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
```

```
# Separating the data and labels
X = diabetes_dataset.drop(columns='Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

```
# Standardizing the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Splitting the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
# Creating a neural network model
model = Sequential()
model.add(Dense(12, input_dim=X.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
# Compiling the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
# Training the model with Training Data
model.fit(X_train, Y_train, epochs=50, batch_size=10, verbose=1)
```

```
# Evaluating the model on training data
_, training_data_accuracy = model.evaluate(X_train, Y_train)
print('Accuracy score on training data: {:.2f}%'.format(training_data_accuracy * 100))
```

```
# Making predictions on a new input data
input_data = np.array([[5, 166, 72, 19, 175, 25.8, 0.587, 51]])
input_data_scaled = scaler.transform(input_data)
prediction = model.predict(input_data_scaled)
```

```
# Converting the prediction to binary (0 or 1)
binary_prediction = (prediction > 0.5).astype(int)[0, 0]
```

```
if binary_prediction == 0:
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

III. Model predviđanja za srčane bolesti

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
#loading the csv data to a Pandas Dataframe
heart_data = pd.read_csv('heart.csv')
```

```
#print first 5 rows of the dataset
heart_data.head()
```

```
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
model = LogisticRegression()
```

```
#training the model with Training Data
model.fit(X_train, Y_train)
```

```
#accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy on training Data: ', training_data_accuracy)
```

```
#accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy on test Data: ', test_data_accuracy)
```

```
input_data = (41,0,130,204,0,0,172,0)

#change the input data to numpy array
input_data_as_numpy_array= np.asarray(input_data)


# reshape numpy array as we are predicting for only one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print ('This person does not have a heart disease.')
else:
  print ('This person has a heart disease.')

import pickle
```

```
filename = 'heart_disease_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

```
loaded_model= pickle.load(open('heart_disease_model.sav', 'rb'))
```

IV. Model za predviđanje bolesti Parkinson

IV. Model za predviđanje bolesti Parkinson

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score

"""Data Analysis

"""

# loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('/content/parkinsons.csv')

# printing the first 5 rows of the dataframe
parkinsons_data.head()

# number of rows and columns in the dataframe
parkinsons_data.shape

# getting more information about the dataset
parkinsons_data.info()

# checking for missing values in each column
parkinsons_data.isnull().sum()

# getting some statistical measures about the data
parkinsons_data.describe()

# distribution of target Variable
parkinsons_data['status'].value_counts()

# grouping the data based on the target variable
parkinsons_data.groupby('status').mean()

"""Data Pre-Processing -->
Separating the features & Target
"""

X = parkinsons_data.drop(columns=['name','status'], axis=1)
Y = parkinsons_data['status']

print(X)

print(Y)

"""Splitting the data to training data & Test data"""

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

print(X.shape, X_train.shape, X_test.shape)

scaler = StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

print(X_train)
```

```
model = svm.SVC(kernel='linear')

# training the SVM model with training data
model.fit(X_train, Y_train)

"""Model Evaluation"""

# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)

print('Accuracy score of training data : ', training_data_accuracy)

# accuracy score on training data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

print('Accuracy score of test data : ', test_data_accuracy)

"""Building a Predictive System"""

input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098

# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the data
std_data = scaler.transform(input_data_reshaped)
```

```

if (prediction[0] == 0):
  print("The Person does not have Parkinsons Disease")

else:
  print("The Person has Parkinsons Disease")

import pickle

filename = 'parkinsons_disease_model.sav'
pickle.dump(model, open(filename, 'wb'))

loaded_model= pickle.load(open('parkinsons_disease_model.sav', 'rb'))

for column in X.columns:
  print(column)
```

V. Kod za učitavanje modela i kreiranje web aplikacije

```
import pickle
import streamlit as st
from streamlit_option_menu import option_menu

#loading done models
diabetes_model = pickle.load(open('C:/Users/Lucija/Desktop/diplomski rad/models/diabetes_model.sav', 'rb'))
parkinsons_model = pickle.load(open('C:/Users/Lucija/Desktop/diplomski rad/models/parkinsons_disease_model.sav', 'rb'))
heart_disease_model = pickle.load(open('C:/Users/Lucija/Desktop/diplomski rad/models/heart_disease_model.sav', 'rb'))

#sidebar

with st.sidebar:

    selected = option_menu('Multiple Disease Prediction System',
                           ['Diabetes Prediction','Parkinsons Prediction',
                            'Heart Disease Prediction'],

                           icons = ['heart-pulse','prescription2', 'activity'],

                           default_index = 0)

    #Diabetes Prediction Page

if (selected== 'Diabetes Prediction'):

    #page title
    st.title('Diabetes Prediction using ML')


    # getting the input data from the user
    col1, col2, col3 = st.columns(3)

     with col1:
         Pregnancies = st.text_input('Number of Pregnancies')

     with col2:
         Glucose = st.text_input('Glucose Level')

     with col3:
         BloodPressure = st.text_input('Blood Pressure value')

     with col1:
         SkinThickness = st.text_input('Skin Thickness value')

     with col2:
        Insulin = st.text_input('Insulin Level')

     with col3:
        BMI = st.number_input('BMI value')

     with col1:
        DiabetesPedigreeFunction = st.number_input('Diabetes Pedigree Function value', min_value=0.0,
     max_value=5.0,
     step=1e-6, format="%.3f")

     with col2:
        Age = st.text_input('Age of the Person')

     # code for Prediction
     diab diagnosis = ''
```

```
    # creating a button for Prediction

    if st.button('Diabetes Test Result'):

    # Convert the specific input values to integers if they are meant to be whole numbers
        Pregnancies = int(Pregnancies)
        BloodPressure = int(BloodPressure)
        Age = int(Age)
        Glucose = int(Glucose)
        SkinThickness = int(SkinThickness)
        Insulin = int(Insulin)

        diab_prediction = diabetes_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,

        if (diab_prediction[0] == 1):
           diab_diagnosis = 'The person is diabetic'
        else:
           diab_diagnosis = 'The person is not diabetic'
    st.success(diab_diagnosis)



# Heart Disease Prediction Page
if (selected == 'Heart Disease Prediction'):

    # page title
    st.title('Heart Disease Prediction using ML')

    col1, col2, col3 = st.columns(3)
    with col1:
        age = st.text_input('Age')

    with col2:
        sex = st.text_input('Sex (Female - 1, Male - 2)')


    with col1:
        trestbps = st.text_input('Resting Blood Pressure')

    with col2:
        chol = st.text_input('Serum Cholestoral in mg/dl')

    with col3:
        fbs = st.text_input('Fasting Blood Sugar > 120 mg/dl')

    with col1:
        restecg = st.text_input('Resting Electrocardiographic results')

    with col2:
        thalach = st.text_input('Maximum Heart Rate achieved')


    with col3:
        ca = st.text_input('Major vessels colored by flourosopy')
```

```
    # code for Prediction
    heart_diagnosis = ''

 # creating a button for Prediction
    if st.button('Heart Disease Test Result'):
       # Convert input values to numeric format
       age = float(age)
       sex = float(sex)
       trestbps = float(trestbps)
       chol = float(chol)
       fbs = float(fbs)
       restecg = float(restecg)
       thalach = float(thalach)
       ca = float(ca)

        # Make prediction
       heart_prediction = heart_disease_model.predict([[age, sex, trestbps, chol, fbs, restecg, thalach, ca]])

       if (heart_prediction[0] == 1):
          heart_diagnosis = 'The person is having heart disease'
       else:
          heart_diagnosis = 'The person does not have any heart disease'

    st.success(heart_diagnosis)

    # Parkinson's Prediction Page
if (selected == "Parkinsons Prediction"):

    # page title
    st.title("Parkinson's Disease Prediction using ML")

    col1, col2, col3, col4, col5 = st.columns(5)

    with col1:
        fo = st.text_input('MDVP:Fo(Hz)')

    with col2:
        fhi = st.text_input('MDVP:Fhi(Hz)')

    with col3:
        flo = st.text_input('MDVP:Flo(Hz)')

    with col4:
        Jitter_percent = st.text_input('MDVP:Jitter(%)')

    with col5:
        Jitter_Abs = st.text_input('MDVP:Jitter(Abs)')

    with col1:
        RAP = st.text_input('MDVP:RAP')

    with col2:
        PPQ = st.text_input('MDVP:PPQ')

    with col3:
        DDP = st.text_input('Jitter:DDP')

    with col4:
        Shimmer = st.text_input('MDVP:Shimmer')

    with col5:
        Shimmer_dB = st.text_input('MDVP:Shimmer(dB)')
```

```
192
193     with col1:
194         APQ3 = st.text_input('Shimmer:APQ3')
195
196     with col2:
197         APQ5 = st.text_input('Shimmer:APQ5')
198
199     with col3:
200         APQ = st.text_input('MDVP:APQ')
201
202     with col4:
203         DDA = st.text_input('Shimmer:DDA')
204
205     with col5:
206         NHR = st.text_input('NHR')
207
208     with col1:
209         HNR = st.text_input('HNR')
210
211     with col2:
212         RPDE = st.text_input('RPDE')
213
214     with col3:
215         DFA = st.text_input('DFA')
216
217     with col4:
218         spread1 = st.text_input('spread1')
219
220     with col5:
221         spread2 = st.text_input('spread2')
222
226     with col2:
227         PPE = st.text_input('PPE')
228
229
230
231     # code for Prediction
232     parkinsons_diagnosis = ''
233
234     # creating a button for Prediction
235     if st.button("Parkinson's Test Result"):
236         parkinsons_prediction = parkinsons_model.predict([[fo, fhi, flo, Jitter_percent, Jitter_Abs, RAP, PPQ,DDP,Shimme
237
238         if (parkinsons_prediction[0] == 1):
239           parkinsons_diagnosis = "The person is very likely to have Parkinson's disease. Please consult with your doctor
240         else:
241           parkinsons_diagnosis = "The person does not have Parkinson's disease."
242
243     st.success(parkinsons_diagnosis)
```