Web aplikacija za procjenu rizika obolijevanja i praćenje tijeka kardiovaskularnih bolesti zasnovana na postupcima nadziranog strojnog učenja

Šabanović, Dina

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

WEB APLIKACIJA ZA PROCJENU RIZIKA OBOLIJEVANJA I PRAĆENJA TIJEKA KARDIOVASKULARNIH BOLESTI ZASNOVANA NA POSTUPCIMA NADZIRANOG STROJNOG UČENJA

Diplomski rad

Dina Šabanović

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Dina Šabanović
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1327R, 07.10.2022.
JMBAG:	0165082526
Mentor:	prof. dr. sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Mirko Köhler
Član Povjerenstva 1:	prof. dr. sc. Goran Martinović
Član Povjerenstva 2:	izv. prof. dr. sc. Ivica Lukić
Naslov diplomskog rada:	Web aplikacija za procjenu rizika obolijevanja i praćenje tijeka kardiovaskularnih bolesti zasnovana na postupcima nadziranog strojnog učenja
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu diplomskog rada potrebno je analizirati aktualne probleme, izazove i postupke utvrđivanja rizika obolijevanja i praćenja tijeka kardiovaskularnih bolesti. Uzimajući u obzir stanje u području i postojeća slična rješenja, treba odrediti značajke modela i prikladne postupke nadziranog strojnog učenja prema vrsti problema, pripremiti odgovarajuće skupove podataka za treniranje i testiranje modela, trenirati i ispitati model strojnog učenja i pripremiti ga za web implementaciju. Također, potrebno je predložiti postupak stvaranja preporuka u ovisnosti o stanju
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	26.09.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	26.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 26.09.2024.

Ime i prezime Pristupnika:	Dina Šabanović
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1327R, 07.10.2022.
Turnitin podudaranje [%]:	13

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za procjenu rizika obolijevanja i praćenje tijeka kardiovaskularnih bolesti zasnovana na postupcima nadziranog strojnog učenja**

izrađen pod vodstvom mentora prof. dr. sc. Goran Martinović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Kardiovaskularne bolesti predstavljaju jedan od najznačajnijih globalnih zdravstvenih izazova, s milijunima smrtnih slučajeva svake godine diljem svijeta. Ove bolesti značajno utječu na kvalitetu života i dugovječnost pojedinaca. Prepoznavanje rizičnih čimbenika i pravovremene preventivne intervencije ključne su za smanjenje učestalosti i ozbiljnosti kardiovaskularnih bolesti. Kontinuirani nadzor i rano otkrivanje potencijalnih problema omogućuju pravovremeno djelovanje što može značajno poboljšati zdravstvene ishode i kvalitetu života pacijenata. Pravilan nadzor pomaže u prilagodbi životnih navika i terapijskih pristupa, smanjujući rizik od ozbiljnih komplikacija poput srčanih ili moždanih udara. Integracija suvremenih tehnologija može dodatno unaprijediti ovaj proces pružanjem detaljnih uvida u pacijentovo stanje.

Cilj ovog diplomskog rada je razviti web aplikaciju koja će omogućiti korisnicima da pravovremeno prepoznaju rizik od kardiovaskularnih bolesti. Aplikacija na temelju unesenih parametara, analizira podatke i izračunava postoji li kod osobe rizik za razvoj kardiovaskularnih bolesti. Uz to, aplikacija pohranjuje sve relevantne podatke, što omogućuje praćenje zdravstvenog stanja pacijenata tijekom vremena i pruža podršku liječnicima u stvaranju preporuka za pacijente. Time se omogućuje dugoročno praćenje i prilagodba liječenja.

U drugom poglavlju rada opisana je građa i funkcija srca i krvnih žila, najčešće kardiovaskularne bolesti te čimbenici rizika i načini prevencije. U drugom poglavlju su, također, predstavljeni postojeći slični sustavi i rješenja koji su poslužili kao osnova za usporedbu razvijene aplikacije. Treće poglavlje posvećeno je zahtjevima aplikacije, uključujući detaljnu analizu arhitekture aplikacije, korištene baze podataka te primjene nadziranog strojnog učenja za razvoj modela. Četvrto poglavlje objašnjava korištene tehnologije, alate i programske jezike, kao i programsko rješenje aplikacije. Na kraju, peto poglavlje rada daje upute i objašnjenje načina korištenja aplikacije te kroz tri ispitna slučaja provjerava daje li aplikacija očekivane rezultate.

2. IZAZOVI U LIJEČENJU KARDIOVASKULARNIH BOLESTI, POTPORA LIJEČENJU I STANJE U PODRUČJU

Kardiovaskularne bolesti najčešći su uzrok smrti kod muškaraca i žena u Europi, a predstavljaju bitan javnozdravstveni problem u cijelom svijetu [1].

U ovome poglavlju bit će opisan rad srca i krvožilnog sustava, kardiovaskularne bolesti i faktori rizika za nastanak kardiovaskularnih bolesti te će biti predstavljena neka već postojeća rješenja za procjenu rizika razvoja kardiovaskularnih bolesti.

2.1. Uloga i građa krvožilnog sustava

Srce i krvne žile tvore zatvoreni sustav za transport krvi – krvožilni sustav. U tom transportnom sustavu srce ima ulogu crpke koja omogućuje stalni protok krvi kroz žilni sustav. Kruženje krvi nužno je za opskrbu stanica tijela kisikom i hranjivim tvarima te za odvođenje nusproizvoda metabolizma i ugljikova dioksida. Krvožilni sustav, putem hormona i čimbenika zgrušavanja koje krv prenosi, sudjeluje u hormonskoj regulaciji tjelesnih funkcija i zgrušavanju krvi. Krvotok, također, služi i za termoregulaciju.
Anatomske tvorbe krvožilnog sustava su:

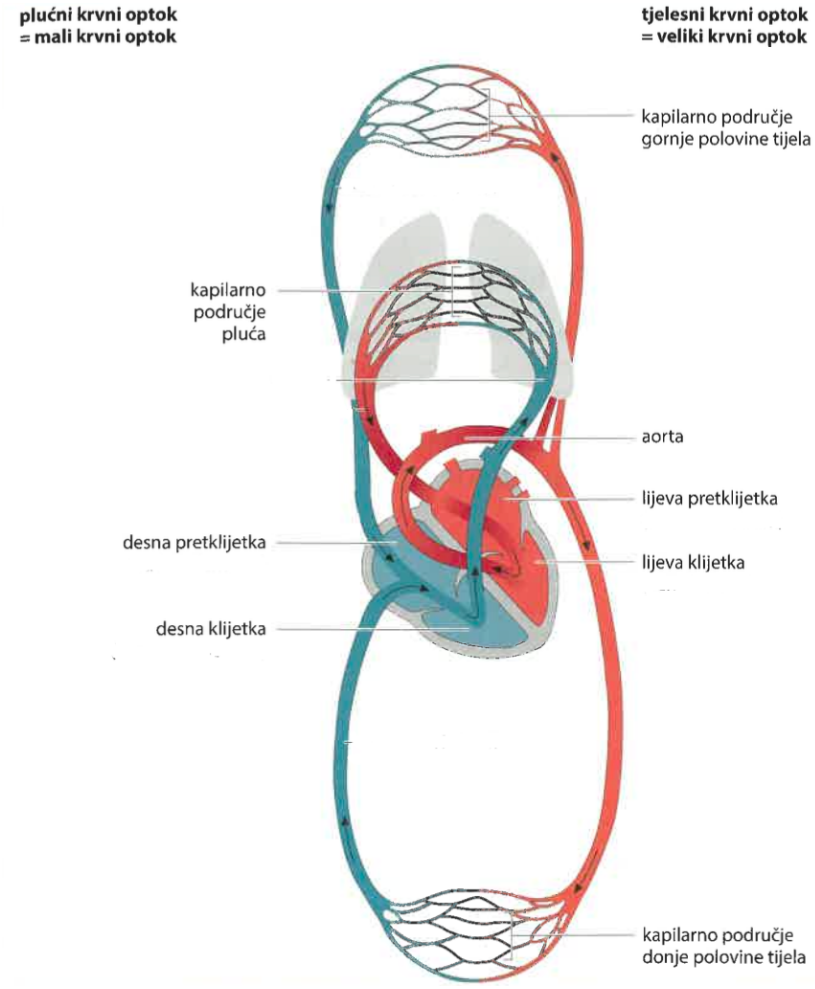
- **srce** (crpka)
- **krvne žile** (transportni sustav)

Srce kao crpnu stanicu možemo podijeliti na dvije crpne jedinice: „desno srce“ i „lijevo srce“, koje su odvojene stijenkom. Svaka crpna jedinica sastoji se od dviju šupljina, **pretklijetke** i **klijetke**.
Podjela krvnih žila na arterijske i venske krvne žile ovisi o smjeru protoka krvi u odnosu na srce. Krvne žile koje odvode krv od srca nazivamo **arterijama**. Najveća arterija koja izlazi iz lijeve klijetke, naziva se aortom. Arterije se u svojemu toku granaju, pri čemu im se promjer stalno smanjuje. Odjeljak arterije s najmanjim promjerom naziva se **arteriolom.** Krv iz arteriola kratkim **metaarteriolama** odlazi u **kapilare**, dio krvožilnoga sustava vidljiv samo mikroskopski, u kojima dolazi do izmjene tvari i plinova između krvi i okolnih tkiva. **Vene** su krvne žile koje krv, nakon prolaska kroz kapilare, dovode ponovno u srce. Za razliku od arterija, ovdje se krv prikuplja prvo u najmanjim odjeljcima, **venulama**, a potom u krvnim žilama sve većega promjera.
Krvotok funkcionalno možemo podijeliti na:

- mali i veliki krvni optok
- visokotlačni i niskotlačni sustav.

Mali krvni optok odvodi krv u pluća te, nakon izmjene plinova, ponovno u srce. Desno srce putem plućne arterije tjera krv siromašnu kisikom prema plućnim alveolama, gdje dolazi do izmjene plinova. Krv bogata kisikom potom, putem plućnih vena, odlazi prema lijevoj pretklijetci i zatim dalje u lijevu klijetku.

Veliki krvni optok opskrbljuje sve organe tijela. Krv bogata kisikom iz lijevog srca iz lijevog srca do organa dolazi putem aorte i njezinih ogranaka, a putem vena vraća se natrag u srce [2]. Slika 2.1 prikazuje veliki i mali krvni optok.



Slika 2.1: Shemtski prikaz velikog i malog krvnog optoka [2]

2.2. Kardiovaskularne bolesti

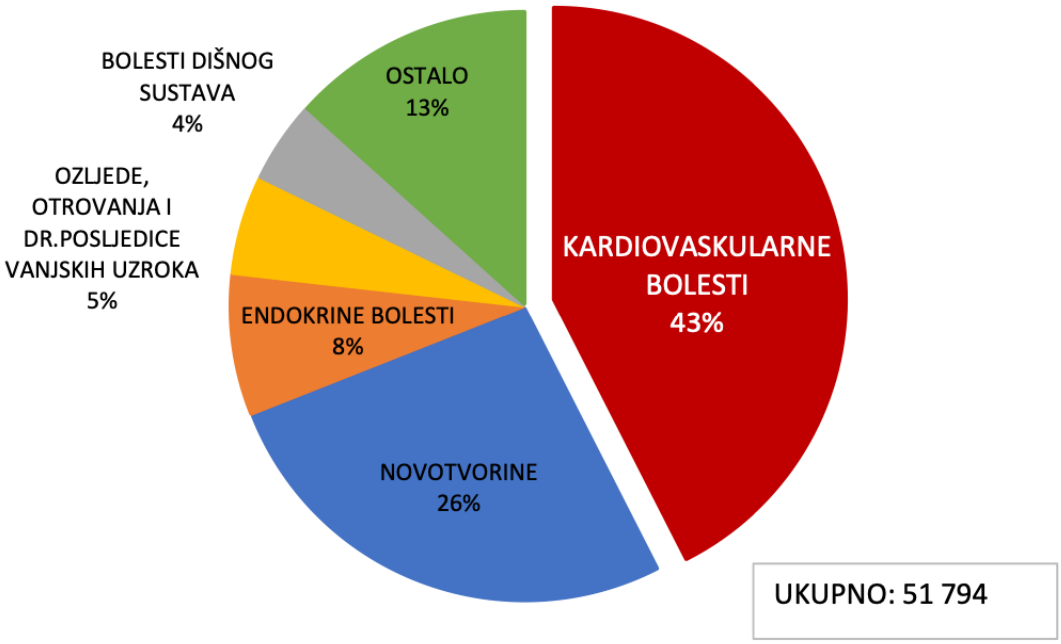
Kako im i samo ime kaže, kardiovaskularne bolesti (KVB) [3] su bolesti srca i krvnih žila. U većini slučajeva kardiovaskularne bolesti nastaju nakupljanjem masnih naslaga u stijenkama krvnih žila u procesu koji se naziva ateroskleroza. U kardiovaskularne bolesti spadaju sve bolesti srca, vaskularne bolesti mozga i bolesti krvnih žila mozga.

- **Koronarna (ishemijska) bolest srca** srčana je bolest uzrokovana suženjem ili začepljenjem koronarnih arterija, što dovodi do nedovoljne opskrbe srčanoga mišića kisikom. Ovo je najčešća kardiovaskularna bolest, a ponekad, prvi simptom bolesti bude srčani udar.
- **Srčani udar ili infarkt miokarda** nastaje kada je dotok krvi i kisika u jedno ili više područja srčanog mišića ozbiljno smanjen ili u potpunosti blokiran što može uzrokovati trajno oštećenje srca i smrt.
- **Moždani udar** označava naglo nastali neurološki poremećaj uzrokovan poremećajem moždane cirkulacije što dovodi do nedovoljne opskrbe dijelova mozga kisikom i hranjivim tvarima.

Ostala stanja uključuju:

- **Aritmija** – poremećaj srčanog ritma
- **Bolest aorte** – bolest koja uzrokuje širenje ili pucanje aorte
- **Kardiomiopatije** – nasljedne i specifične stečene bolesti srčanog mišića
- **Kongenitalna bolest srca** – urođene mane koje utječu na normalan razvoj i funkcioniranje srca
- **Duboka venska tromboza**– krvni ugrušci u venama nogu
- **Plućna embolija** – začepljenje plućne arterije raznim materijalima, najčešće krvnim ugruškom, koji su doneseni krvotokom
- **Zatajenje srca**
- **Bolest srčanih zalistaka**
- **Bolest perikarda** – upala tanke tkivne ovojnice koja okružuje srce
- **Vaskularne bolesti** – svako stanje koje utječe na krvožilni sustav.

Postupak dijagnosticiranja kardiovaskularnih bolesti ovisi o simptomima i vrsti stanja za koje liječnik sumnja da pacijent ima. Liječnik će ispitati medicinsku i obiteljsku anamnezu, čimbenike rizika te obaviti pregled. Uz to, obavit će se niz laboratorijskih testova i slikovnih pretraga [4]. Prema podacima Svjetske zdravstvene organizacije, 2019. godine su kardiovaskularne bolesti bile uzrok smrti 17,9 milijuna ljudi u svijetu, odnosno odgovorne su za 32% sveukupne smrtnosti. Najveći broj smrti od kardiovaskularnih bolesti, čak 85%, rezultat je srčanog ili moždanog udara. KVB su odgovorne i za 38% prijevremenih smrti, odnosno smrti osoba mlađih od 70 godina [5].



Slika 2.2: Uzroci smrti u Hrvatskoj 2019. godine [5]

2.3. Čimbenici rizika obolijevanja od kardiovaskularnih bolesti

Postoji niz čimbenika rizika koji povećavaju vjerojatnost obolijevanja od kardiovaskularnih bolesti. Na određene faktore poput spola, dobi i genetskog nasljeđa nije moguće utjecati, no znatno je više onih koje je moguće kontrolirati svojim svakodnevnim navikama te tako smanjiti rizik za razvoj bolesti.

2.3.1. Prevencija razvoja kardiovaskularnih bolesti

Preventivne aktivnosti uključuju primarnu prevenciju (ograničenje broja novih bolesnika), sekundarnu prevenciju (smanjenje rizika u asimptomatskih bolesnika, što uključuje pravodobno otkrivanje) i tercijarnu prevenciju (rehabilitacija i smanjenje invalidnosti uz očuvanje kakvoće života u bolesnika s razvijenom bolešću). [6]. Usvajanjem zdravijih navika i promjenom načina života moguće je unaprijediti zdravlje, a time i smanjiti rizik za razvoj kardiovaskularnih bolesti.

1. **Povećanje fizičke aktivnosti**

   Istraživanja pokazuju da je čak 60 % svjetske populacije nedovoljno fizički aktivno. Smatra se da 150 minuta umjerene fizičke aktivnosti tjedno ili 60 minuta pojačane aktivnosti smanjuju

rizik od kardiovaskularnih bolesti za 30%. Fizička aktivnost snižava vrijednosti krvnog tlaka, regulira vrijednosti šećera u krvi, snižava masnoće u krvi, povoljno djeluje na faktore zgrušavanja krvi te smanjuje razinu stresa koji također ima ulogu u razvoju bolesti. Posljedice fizičke neaktivnosti su ogromne. Primjerice, žena srednjih godina koja je fizički aktivna manje od jednog sata tjedno ima dva puta veći rizik od kardiovaskularnih bolesti sa smrtnim ishodom od fizički aktivne žene.

2. **Prestanak pušenja**

   Pušenje oštećuje stijenke krvnih žila, povećava odlaganje masnih naslaga i ubrzava stvaranje krvnih ugrušaka. Nikotin ubrzava srčanu frekvenciju i podiže vrijednosti krvnog tlaka. Smatra se da pušenje povećava rizik od moždanog udara i koronarne srčane bolesti za 100%.

3. **Uravnotežena i zdrava prehrana**

   Prehrana bi trebala biti uravnotežena, a bitno je smanjiti ukupni unos masti kao i unos zasićenih masnih kiselina koje nalazimo u mesu i nekim mliječnim proizvodima te transmasnih kiselina, koje nalazimo u obrađenoj hrani kao što su kolači i brza hrana. Dobro je povećati unos nezasićenih masnih kiselina iz ribe, sjemenki, maslinovog ulja ili avokada. Poželjno je povećati unos voća i povrća kao i integralnih žitarica, koje su bogate vlaknima, folnom kiselinom i B vitaminima. Potrebno je smanjiti unos soli i alkohola. Cilj je pravilnom prehranom održavati optimalnu tjelesnu težinu, vrijednosti krvnog tlaka, šećera i masnoća u krvi.

4. **Smanjenje prekomjerne tjelesne težine**

   Pretilost je jedan od najvećih zdravstvenih problema današnjice, a sve češće se viđa već u predškolskoj dobi. Danas je u svijetu 17,6 milijuna djece pretilo. Pretilost povećava vrijednosti krvnog tlaka te povećava rizik od nastanka ateroskleroze i pojave šećerne bolesti.

5. **Regulacija krvnog tlaka**

   Najmanje 970 milijuna ljudi pati od povišenog krvnog tlaka, odnosno arterijske hipertenzije, glavnog neovisnog čimbenik za razvoj kardiovaskularnih bolesti i jedan od osnovnih razloga preuranjene smrtnosti. Hipertenzija može biti primarna, kod koje ne nalazimo točan uzrok, a najčešće je posljedica genetskih faktora i/ili stila života i sekundarna, koja nastaje u određenim oboljenjima. Hipertenzija pospješuje aterosklerozu, zbog čega se krvne žile sužavaju i smanjuje se dotok kisika organima. Vrijednosti krvnog tlaka ovise o količini krvi koju srce pumpa te stanju krvnih žila – što su krvne žile uže i tvrđe, tlak je viši. Vrijednosti krvnog tlaka iznad 140/90 mmHg trebaju se liječiti neovisno o životnoj dobi pojedinca.

6. **Regulacija razine šećera u krvi**

   Normalne vrijednosti glukoze u krvi, izmjerene ujutro, trebaju biti ispod 5,6 mmol/L. Vrijednosti iznad 7 mmol/L u dva neovisna mjerenja ili iznad 11,1 mmol/L neovisno o obroku, ukazuju na šećernu bolest. Količine glukoze u tijelu regulirane su različitim hormonima, a glavni regulatorni hormon je inzulin kojeg proizvode stanice gušterače. Nedostatak inzulina povisit će razine glukoze u krvi. Kod dijabetesa tipa 1 autoimuni sustav uništava stanice koje proizvode inzulin, dok kod tipa 2 gušterača ne proizvodi dovoljno inzulina ili tijelo ne reagira na njega. Vježbanje i prehrana mogu poboljšati razinu šećera kod tipa 2, ali tip 1 zahtijeva unos inzulina putem injekcija [7].

2.4. Postojeća slična rješenja

Primjena umjetne inteligencije, osobito strojnog učenja, sve se više koristi u dijagnostici. Prednost strojnog učenja je u tome što je moguće dijagnosticirati bolesti, poput kardiovaskularnih bolesti, uz niske troškove i razumnu točnost, bez potrebe za brojnim invazivnim kliničkim ispitivanjima. Skup informacija i značajki iz podataka pacijenata može značajno doprinijeti točnosti dijagnostike. U [8] i [9] opisana su istraživanja kojima je cilj analizirati različite tehnike strojnog učenja za predviđanje hoće li osoba, s obzirom na određene osobne karakteristike i simptome, razviti srčanu bolest. U jednom istraživanju korišteni su algoritmi metoda potpornih vektora, slučajna šuma, stablo odluke i naivni Bayes. Svi algoritmi su pokazali dobre rezultate, ali slučajna šuma je imala najveću točnost od 99%, dok je najlošiji rezultat imalo stablo odluke, čija je točnost iznosila 85%. U drugom istraživanju korištena je metoda potpornih vektora, stablo odluke, linearna regresija i algoritam k-najbližih susjeda. Rezultati su ovdje bili nešto lošiji nego u prethodnom istraživanju. Najveću točnost, koja je iznosila 87%, imao je algoritam k-najbližih susjeda, a linearna regresija pokazala se kao najlošija metoda s točnošću od 78%. Unatoč napretku tehnologije, liječnici igraju ključnu ulogu u potvrdi rezultata koje modeli strojnog učenja generiraju, osiguravajući da su dijagnoze ispravne i da je liječenje primjereno pacijentovom stanju. S daljnjim razvojem algoritama i povećanjem dostupnosti podataka, strojno učenje ima potencijal još više unaprijediti dijagnostiku bolesti i olakšati rad liječnicima.

Na internetu je moguće pronaći velik broj aplikacija pomoću kojih je moguće provjeriti koliki rizik osoba ima za razvoj kardiovaskularnih bolesti. Sve aplikacije rade na sličnom principu. Potrebno je ispuniti upitnik o dobi, spolu, nacionalnosti, životnim navikama te aplikacija na temelju danih odgovora izračunava koliki je rizik da osoba razvije kardiovaskularne bolesti u nekom periodu.

- **ESC CVD Risk Calculation App**

  Aplikacija podržana od strane Europskog kardiološkog društva (ESC), namijenjena je zdravstvenim radnicima. Moguće je procijeniti individualni kardiovaskularni rizik do 10 godina ili tijekom čitavog života za različite populacije pacijenata.

- **American College of Cardiology (ACC) Atherosclerotic Cardiovascular Disease (ASCVD) Risk Calculator**

  Ovaj alat procjenjuje vjerojatnost da osoba u dobi od 40 do 79 godina razvije neku kardiovaskularnu bolest u sljedećih 10 godina. Kalkulator također predlaže mjere koje pacijent može poduzeti kako bi smanjio svoj rizik.

3. MODEL I GRAĐA WEB APLIKACIJE I ANALIZA PODATAKA

U ovome poglavlju će biti opisani zahtjevi koje aplikacija treba ispuniti te model i građa aplikacije. Također, bit će opisani korišteni postupci strojnog učenja te podaci korišteni za razvoj modela.

3.1. Funkcionalni i nefunkcionalni zahtjevi na aplikaciju

Analiza zahtjeva je kritičan proces koji omogućava procjenu uspjeha softverskog projekta. Zahtjevi na softver se općenito dijele na funkcionalne i nefunkcionalne zahtjeve. Funkcionalni zahtjevi su osnovne funkcionalnosti koje sustav treba ponuditi krajnjem korisniku. Ovi su zahtjevi predstavljeni u obliku ulaza koji se daje sustavu, operacija koje se izvode i očekivanog izlaza, a korisnik ih može izravno vidjeti u konačnom proizvodu. Nefunkcionalni su zahtjevi kvalitativni zahtjevi koje sustav mora zadovoljiti [10].
Zahtjevi koje ova aplikacija treba ispuniti su:

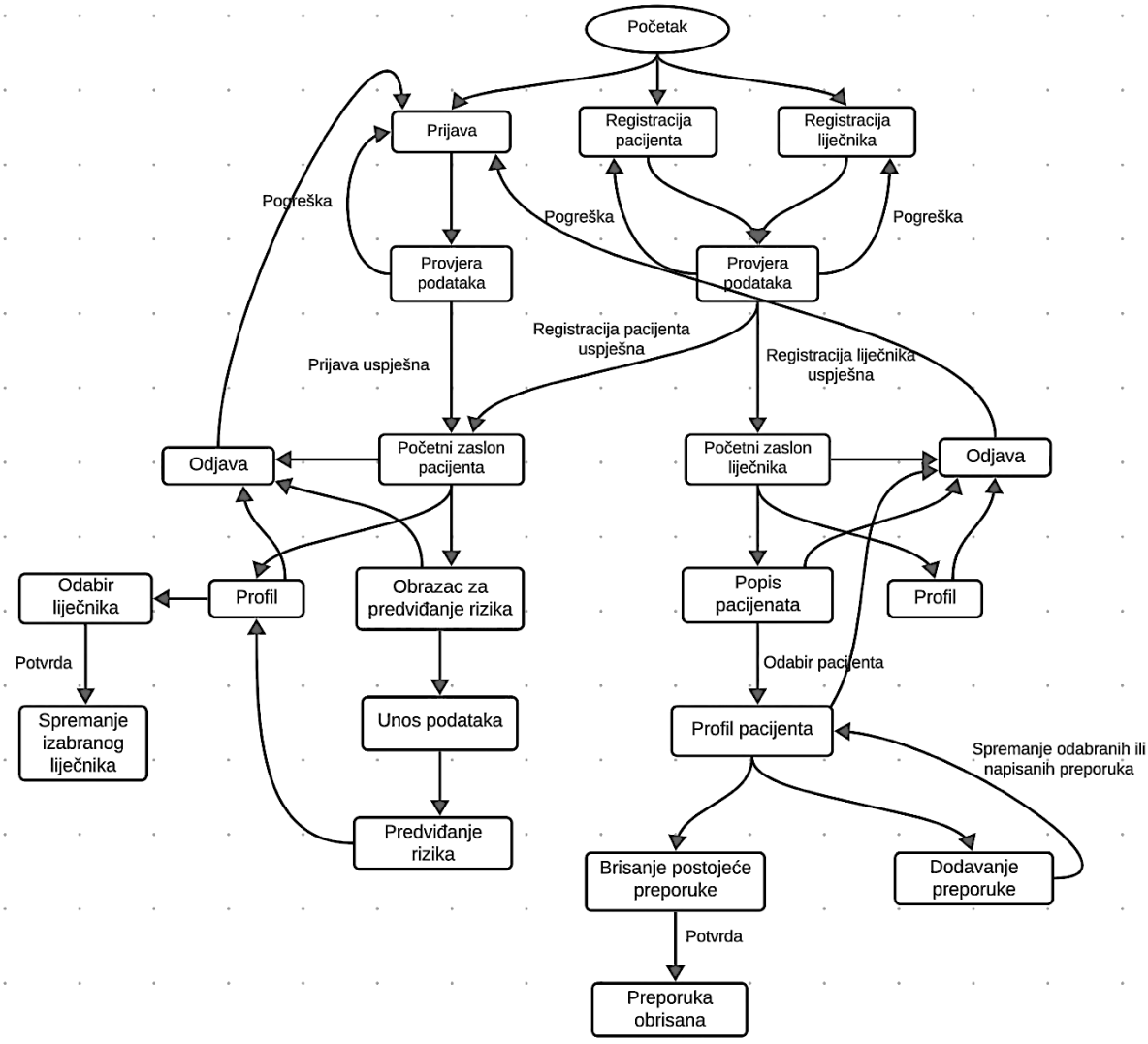
- Registracija korisnika kao pacijenta
- Registracija korisnika kao liječnika
- Prijava liječnika i pacijenta u aplikaciju
- Korisnik u ulozi pacijenta može ispuniti upitnik za izračun rizika za nastanak kardiovaskularnih bolesti
- Korisnik u ulozi pacijenta ima mogućnost odabira liječnika
- Korisnik u ulozi pacijenta ima mogućnost pregleda unesenih podataka i rezultata svih predviđanja koje je napravio
- Korisnik u ulozi liječnika ima mogućnost pregleda unesenih podataka i rezultata svih predviđanja za sve svoje pacijente
- Korisnik u ulozi liječnika može odabrati preporuke sustava na temelju zdravstvenog stanja pacijenta, urediti ih ili napisati nove
- Korisnik u ulozi liječnika može obrisati preporuke
- Korisnik u ulozi pacijenta može vidjeti liječničke preporuke
- Odjava korisnika iz aplikacije

Osim navedenih funkcionalnih zahtjeva, aplikacija bi trebala biti i sigurna, pouzdana te jednostavna za korištenje.

3.2. Građa web aplikacije

Idejno rješenje korištenja web aplikacije nalazi se na slici 3.1. Nakon prijave, korisnici vide različite opcije ovisno o ulozi, pacijent može pristupiti svom profilu, napraviti predviđanje rizika

ili se odjaviti, dok liječnik može posjetiti svoj profil, pregledati popis pacijenata ili se odjaviti. Profil pacijenta prikazuje osobne podatke, predviđanja, liječničke preporuke i mogućnost promjene liječnika. Nakon što pacijent ispuni obrazac za predviđanje rizika, rezultat se prikazuje na profilu. Liječnik može pregledati profile pacijenata, vidjeti njihove podatke i predviđanja te dodavati ili brisati preporuke.



Slika 3.1: Dijagram toka korištenja web aplikacije

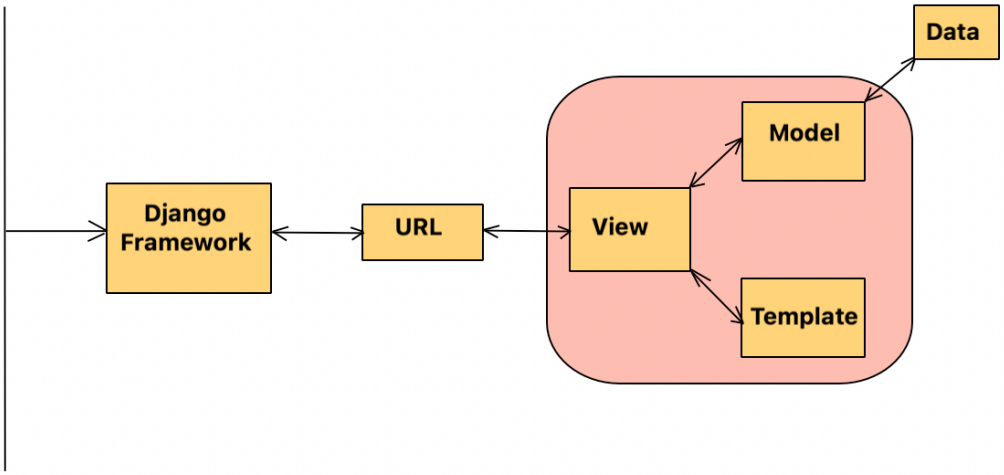
3.2.1. Obrazac programske arhitekture

Za potrebe razvoja web rješenja korišteni su programski jezik Python i razvojni okvir Django. Aplikacije razvijene koristeći Django prate arhitekturni obrazac poznat kao ***Model-View-Template*** (MVT) [11].

Model radi izravno s bazom podataka, definira podatke i ponašanje podataka. Modeli pružaju sredstva za strukturiranje i manipulaciju podacima u web aplikaciji. Zaključno, sloj modela djeluje kao most, omogućavajući interakciju s bazom podataka pomoću Python objekata.

View (pogled) komponenta može biti Python klasa ili funkcija, a obično se pohranjuju u views.py datoteci. Pogled sadrži logiku koja obrađuje web zahtjeve i generira odgovore. U suštini, sloj *view* dohvaća podatke iz modela, prikazuje ih u predlošku, ali, također, i prima zahtjeve iz predloška koji mogu modificirati podatke u modelu ili bazi podataka.

Template (predložak) je prezentacijski sloj web aplikacije, odnosno, sloj s kojim korisnici komuniciraju. Predlošci se koriste za prikaz podataka dobivenih iz modela putem *view-a*, stvarajući dinamičke web stranice koje su jednostavne za održavanje i proširenje. Predložak se sastoji od statičkih datoteka koje se dinamički generiraju koristeći ***Django Template Language*** (DTL). Umjesto ručnog mijenjanja HTML-a svaki put kada se podaci promijene, DTL omogućuje korištenje varijabli i logičkih izraza unutar predložaka. Varijable, označene sintaksom {{varijabla}}, omogućuju umetanje podataka iz modela, poput naslova ili korisničkih informacija, izravno u HTML. Prilikom prikazivanja stranice, Django zamjenjuje te varijable stvarnim vrijednostima iz baze podataka, čime se generira dinamičan sadržaj. Osim umetanja varijabli, DTL podržava osnovne logičke operacije, kao što su petlje i uvjetni izrazi, koji omogućuju kontrolu toka unutar predloška. Na primjer, tagovi poput {% if %} i {% for %} omogućuju prikaz različitih elemenata ovisno o uvjetima ili iteriranje kroz kolekcije podataka. Time se omogućuje prilagodba prikaza bez potrebe za direktnim pisanjem poslovne logike unutar predloška. DTL time osigurava jasnu separaciju između poslovne logike i prezentacijskog sloja. Ovaj pristup olakšava održavanje i proširenje web stranica, jer se promjene podataka automatski reflektiraju u predlošku bez potrebe za ručnim prilagodbama.



Slika 3.2: Arhitekturni obrazac Model-View-Template

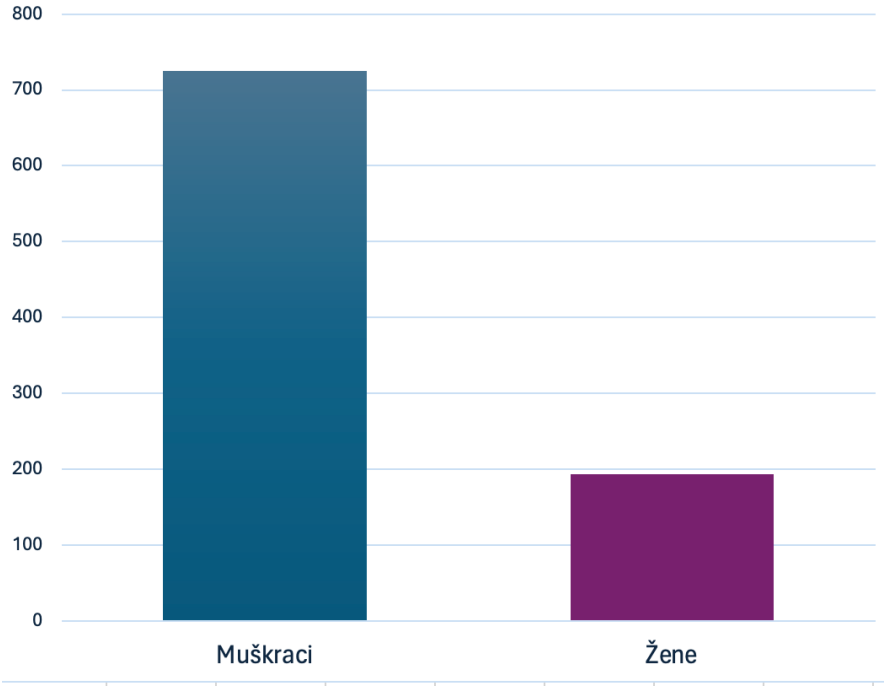
3.3. Skup podataka korišten za razvoj modela

Skup podataka [12] korišten za razvoj modela strojnog učenja preuzet je s platforme *Kaggle* u .csv formatu, a nastao je spajanjem više različitih skupova podataka. Sastoji se od 918 redova i 12 stupaca. Na temelju 11 značajki model klasificira pacijente u dvije kategorije. Tablica 3.1 prikazuje korištene značajke.

Tablica 3.1: Značajke u skupu podataka

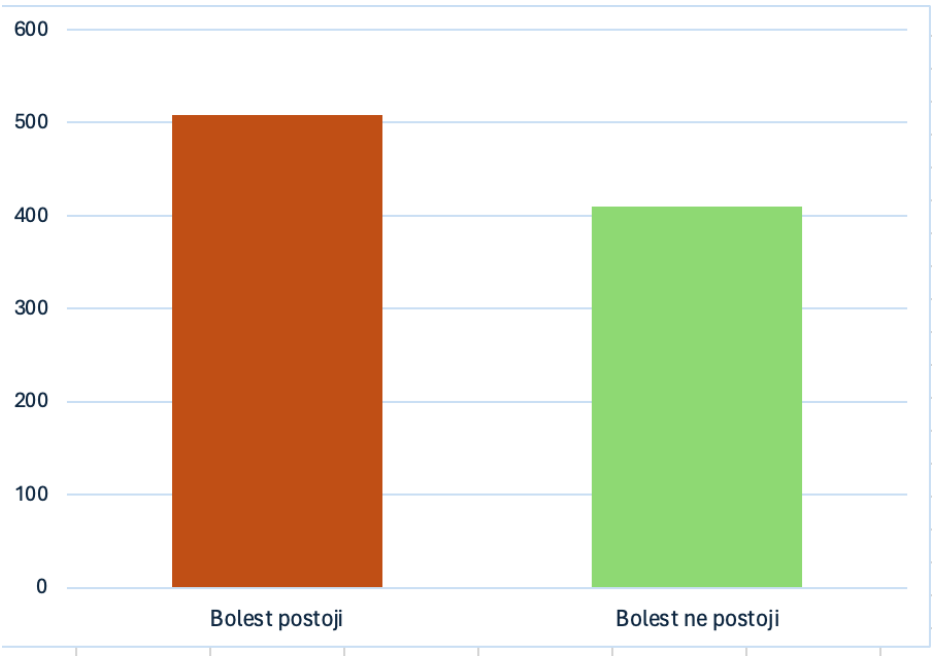
Značajka	**Vrsta**	**Opis**
Age	Numerička	Dob pacijenta izražena u godinama
Sex	Kategorička	M: Muški, F: Ženski
ChestPainType	Kategorička	Vrsta boli u prsima TA: Tipična angina, ATA: Atipična angina, NAP: Neanginalna bol, ASY: Asimptomatski
RestingBP	Numerička	Krvni tlak izražen u mmHg
Cholesterol	Numerička	Kolesterol izražen u mg/dl
FastingBS	Kategorička	Šećer u krvi natašte 1: Ako je >120 mg/dl, 0: Ako je ≤ 120 mg/dl
RestingECG	Kategorička	Rezultati elektrokardiograma u mirovanju Normal: Normalni nalazi, ST: Abnormalnosti ST-T vala, LVH: Hipertrofija lijeve klijetke
MaxHR	Numerička	Maksimalan broj otkucaja srca u minuti
ExerciseAngina	Kategorička	Angina uzrokovana tjelesnom aktivnošću Y: Da, N: Ne
Oldpeak	Numerička	Depresija ST-segmenta pri opterećenju Decimalne vrijednosti (npr. 1.0, 0.5, 2.5)
ST_Slope	Kategorička	Nagib ST-segmenta Up: Uspon, Flat: Ravan, Down: Silazno
HeartDisease	Kategorička	Izlazna klasa, 1: Prisutna srčana bolest, 0: Nema srčane bolesti

Pacijenti su u dobi od 28 do 77 godina, ali najviše je onih u 50-im i 60-im godinama života. Broj muškaraca, njih 725, veći je od broja žena kojih je tek 193. Odnos muških i ženskih pacijenata prikazan je na slici 3.3.



Slika 3.3: Vizualizacija broja muških i ženskih pacijenata

Također, broj pacijenata s kardiovaskularnim bolestima premašuje broj pacijenata bez bolesti, pri čemu 508 pacijenata ima srčanu bolest, dok ih 410 nema. Odnos broja pacijenata sa i bez bolesti prikazan je na slici 3.4.



Slika 3.4: Vizualizacija broja pacijenata s bolesti i bez bolesti

3.4. Prikladni postupci analize podataka i mjerila vrednovanja

Prije odabira postupka s najboljim performansama testirani su različiti algoritmi strojnog učenja koji su odabrani po uzoru na [8] i [9], a njihova učinkovitost je procijenjena primjenom različitih metoda vrednovanja. Korištene metode vrednovanja objašnjene su u [13]. Za potrebe testiranja i vrednovanja modela korišten je *Jupyter Notebook*.

Matrica zabune je tablica koja prikazuje i uspoređuje stvarne vrijednosti s predviđenim vrijednostima modela. Matrica zabune omogućava izračun točnosti, preciznosti, odziva i F1 mjere koji će biti korišteni za vrednovanje modela. Tablica 3.2. prikazuje izgled matrice zabune.

Tablica 3.2: Matrica zabune

Stvarno	**Predviđeno**	
	Pozitivno	Negativno
Pozitivno	TP	FN
Negativno	FP	TN

TP predstavlja broj pozitivnih uzoraka koji su ispravno klasificirani kao pozitivni, a TN broj negativnih uzoraka koji su ispravno klasificirani kao negativni. FP je broj negativnih uzoraka koji su pogrešno klasificirani kao pozitivni, dok je FN broj pozitivnih uzoraka koji su pogrešno klasificirani kao negativni.

Točnost (engl. *Accuracy*) je udio ispravno klasificiranih primjera u skupu svih primjera (izraz 3-1).

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad , \quad (3\text{-}1)$$

Preciznost (engl. *Precision*) je omjer ispravno klasificiranih pozitivnih uzoraka i ukupnog broja uzoraka koje je klasifikator označio kao pozitivne (izraz 3-2).

$$P = \frac{TP}{TP+FP} \quad (3\text{-}2)$$

Odziv (engl. *Recall*) je omjer ispravno klasificiranih pozitivnih uzoraka u odnosu na ukupni broj stvarnih pozitivnih uzoraka, uključujući i one koje je model pogrešno klasificirao kao negativne (izraz 3-3).

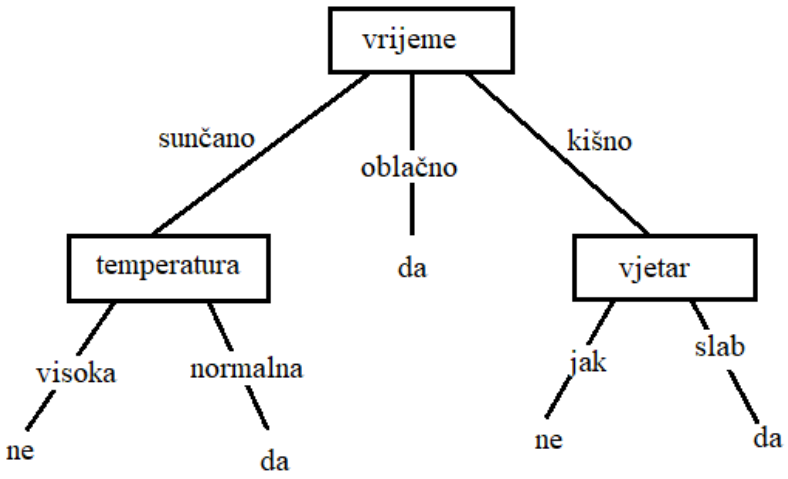
$$R = \frac{TP}{TP+FN} \quad (3\text{-}3)$$

Mjera F1 (engl. *F1 score*) je harmonijska sredina preciznosti i odziva. Balansira između točnosti pozitivnih predviđanja i sposobnosti pronalaženja svih pozitivnih instanci (izraz 3-4).

$$F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R} \tag{3-4}$$

3.4.1. Stablo odluke

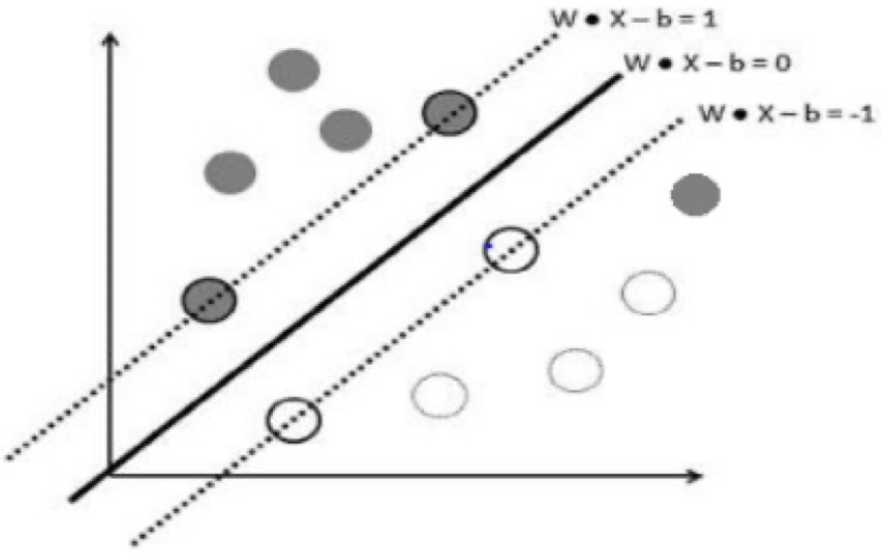
Stabla odluke [14] predstavljaju jednu od najpopularnijih metoda u području strojnog učenja, posebno za klasifikaciju. Osnovne komponente stabla odluke su čvorovi, grane i listovi. Čvorovi predstavljaju atribute koji se koriste za donošenje odluka. Na svakom čvoru odabire se atribut koji je najinformativniji za razdvajanje podataka. Grane povezuju čvorove i označene su vrijednostima atributa iz čvora iz kojeg proizlaze. Listovi predstavljaju krajnje odluke ili klase kojima podaci pripadaju. Stabla odlučivanja funkcioniraju na principu "podijeli i vladaj", gdje se podaci razdvajaju na manje skupove sve dok se ne postigne potpuna homogenost, odnosno, dok svi podaci u skupu ne vode do iste odluke. Proces izrade stabla odluke započinje s učenjem, a zatim se koristi za predviđanje klasa novih, nepoznatih podataka. Algoritmi stabla odluke obično rade odozgo prema dolje, počevši od korijena stabla i završavajući na listovima. Nakon što je stablo izgrađeno može se primijeniti proces podrezivanja kako bi se uklonile beskorisne grane što čini stablo jednostavnijim za interpretaciju i često poboljšava njegove performanse u klasifikaciji. Stabla odluke su popularna jer omogućuju jednostavnu i intuitivnu vizualizaciju procesa donošenja odluka, čime su razumljiva i korisnicima koji nemaju tehničko znanje. Jednostavno stablo odluke prikazano je na slici 3.2.



Slika 3.4: Jednostavno stablo odluke

3.4.2. Metoda potpornih vektora

Metoda strojeva potpornih vektora (*engl. Support Vector Machine*) [15] algoritam je strojnog učenja koji analizira podatke i prepoznaje obrasce ili granice odluke unutar skupa podataka, a uglavnom se koristi za klasifikaciju i regresijsku analizu. Primarni cilj metode potpornih vektora jest identificirati hiper-ravninu koja jasno odvaja podatkovne točke različitih klasa. Model se sastoji od tri linije. Jedna od njih je $w \cdot x - b = 0$, što predstavlja marginalnu liniju ili marginu. Linije $w \cdot x - b = 1$, i $w \cdot x - b = -1$ predstavljaju pozicije najbližih točaka podataka iz obje klase. Krugovi koji leže na hiper-ravnini nazivaju se potporni vektori. Ispunjeni krug u drugoj klasi naziva se izvanredna vrijednost (*outlier*). Ona se ignorira kako bi se izbjeglo prekomjerno prilagođavanje modela i postigla što preciznija klasifikacija. Cilj SVM-a je maksimizirati okomitu udaljenost između dvaju rubova hiper-ravnine kako bi se smanjila mogućnost pojave generalizacijske pogreške. Kako hiper-ravnina ovisi o broju potpornih vektora, kapacitet generalizacije raste s manjim brojem potpornih vektora.

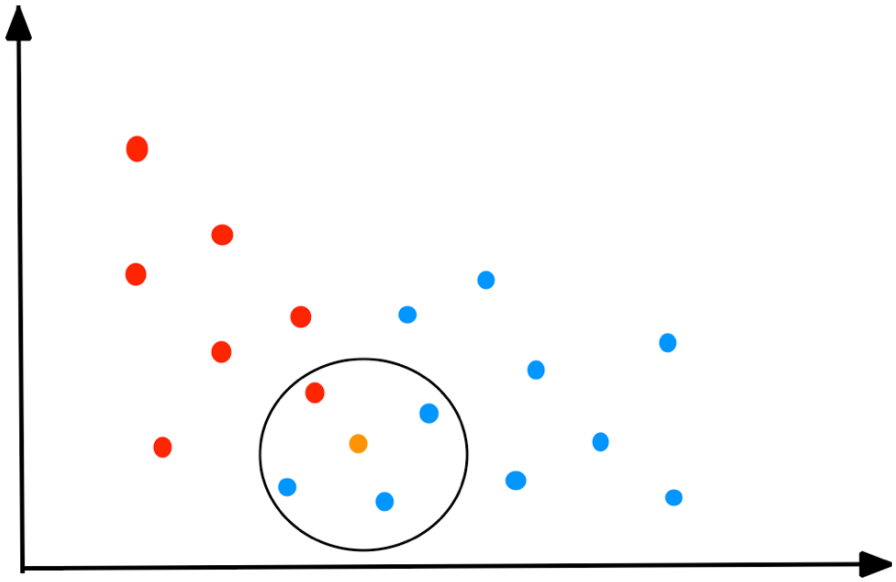


Slika 3.5: Metoda strojeva potpornih vektora [11]

3.4.3. Algoritam k najbližih susjeda

Algoritam k najbližih susjeda (*engl. k-Nearest-Neighbors (k-NN)*) [16] neparametarski je klasifikacijski algoritam, što znači da ne postavlja nikakve pretpostavke o osnovnom skupu podataka. Poznat je po svojoj jednostavnosti i učinkovitosti. Riječ je o nadziranom učenju, gdje se koristi označeni skup podataka za treniranje u kojem su podaci kategorizirani u različite klase, kako bi se mogla predvidjeti klasa neoznačenih podataka. K-NN se uglavnom koristi kao klasifikator. Koristi se za klasifikaciju podataka na temelju najbližih ili susjednih primjera iz skupa za treniranje u određenom području. Algoritam k najbližih susjeda izračunava udaljenost novog

podatka od svakog njegovog susjeda prema vrijednosti k. Nakon što odredi k najbližih susjeda, algoritam predviđa da podatak pripada onoj klasi kojoj pripada i najveći broj njegovih najbližih susjeda. Unutarnja varijabla k određuje broj susjeda koji se moraju uzeti u obzir. Vrijednost k utječe na algoritam jer pomoću nje možemo izgraditi granice svake klase. K-NN je posebno značajan u slučajevima gdje nema prethodnog znanja o podacima koji se koriste.



Slika 3.6: Algoritam k najbližih susjeda

3.4.4. Slučajna šuma

Slučajne šume [17] spadaju u ansambl metode strojnog učenja. Ansambl se sastoji od pojedinačno treniranih klasifikatora čija se predviđanja kombiniraju. Istraživanja su pokazala da je ansambl često precizniji od bilo kojeg pojedinačnog klasifikatora unutar njega. Slučajna šuma klasifikacijski je postupak koji se sastoji od skupa stabala odluka. Svako stablo daje svoj glas, a konačna klasifikacija se određuje zbrajanjem glasova svih stabala, pri čemu se najpopularnija klasa odabire na temelju većine glasova. Svako stablo u slučajnoj šumi gradi se na temelju nasumično odabranog uzorka zapisa iz trening skupa s ponavljanjem, poznatog kao *bootstrap* uzorak, dok oko trećine zapisa ostaje izvan uzorka i koriste se kao OOB (*out-of-bag*) podaci. OOB podaci koriste se za dobivanje nepristrane procjene pogreške klasifikacije, odnosno, za testiranje.

3.5. Predobradba podataka i stvaranje modela

Prije treniranja modela bilo je potrebno izvršiti predobradbu podataka. Sve nule u stupcu *Cholesterol* zamijenjene su medijanom. Kategoričke varijable pretvaraju se u numeričke vrijednosti koristeći alat *LabelEncoder*. Tijekom treniranja, za svaku kategoričku varijablu kreiran

je poseban *LabelEncoder* koji se koristi za transformaciju podataka, a zatim se pohranjuje za kasniju upotrebu. Kada se novi podaci trebaju obraditi, koristi se već pohranjeni *LabelEncoder* kako bi se novi podaci transformirali na isti način kao i podaci korišteni tijekom treniranja. Nakon obrade, podaci su podijeljeni na značajke (X) i ciljnu varijablu (y). Nakon toga, generiraju se polinomske značajke koje omogućuju modelu da bolje uhvati složene odnose među značajkama, stvarajući nove kombinacije postojećih značajki. Ove transformirane značajke se zatim skaliraju, što znači da se sve značajke prilagođavaju kako bi imale sličan raspon vrijednosti. Standardizacija sprječava da značajke s većim vrijednostima neproporcionalno utječu na model i poboljšava učinkovitost algoritama strojnog učenja, osobito onih osjetljivih na različite skale značajki. Konačno, podaci se dijele na skupove za treniranje i testiranje, pri čemu se 20% podataka rezervira za testiranje modela, dok se 80% podataka koristi za treniranje. Slika 3.6 prikazuje programski kod predobradbe podataka.

```
median_cholesterol = df['Cholesterol'].median()
df['Cholesterol'] = df['Cholesterol'].replace(0, median_cholesterol)

# Preprocessing function
def preprocessing(df, is_training=True, label_encoders=None):
    categorical_columns = ['Sex', 'ChestPainType',
                           'RestingECG', 'ExerciseAngina', 'ST_Slope']

    if is_training:
        label_encoders = {}
        for col in categorical_columns:
            le = LabelEncoder()
            df[col] = le.fit_transform(df[col])
            label_encoders[col] = le

        X = df.drop(columns=['HeartDisease'])
        y = df['HeartDisease']
        return X, y, label_encoders
    else:
        for col in categorical_columns:
            if col in label_encoders:
                df[col] = label_encoders[col].transform(df[col])
        return df

X, y, label_encoders = preprocessing(df, is_training=True)

poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
X_poly = poly.fit_transform(X)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_poly)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
                                                    random_state=42)
```

Slika 3.7: Programski kod obrade podataka

Nakon predobradbe podataka, trenirani su modeli koristeći algoritme objašnjene u poglavlju 3.4. Kako bi modeli imali što bolje performanse, potrebno je pronaći najbolje moguće postavke za svaki model. Svaki model ima svoje specifične parametre koji utječu na njegovu učinkovitost. Različite kombinacije parametara mogu značajno promijeniti performanse modela. Koristeći alat *GridSearchCV* pretražuju se različite kombinacije parametara za svaki model kako bi se pronašla ona s kojom modela postiže najbolje rezultate. Za slučajnu šumu kombinira se različiti broj stabala koja model koristi, maksimalnu dubinu tih stabala i minimalan broj uzoraka potrebnih za podjelu čvora. Kod metode potpornih vektora parametar C određuje koliko model može tolerirati pogreške, što utječe na to koliko će strogo razdvajati klase, kernel definira oblik granice koja razdvaja različite klase podataka, a gamma igra ulogu u oblikovanju te granice i utječe na osjetljivost modela na pojedinačne podatke. Za algoritam k-najbližih susjeda, pretražuje se broj najbližih

susjeda koje model uzima u obzir prilikom donošenja odluka, kao i način na koji se ti susjedi vrednuju. Za stablo odlučivanja ponovno je bitna maksimalna dubina stabla i minimalan broj uzoraka potreban za podjelu čvora. Slika 3.7 prikazuje programski kod parametara koji će se kombinirati za svaki model.

```
param_grids = {
    RandomForestClassifier(random_state=42): {
        'n_estimators': [100, 200, 300],
        'max_depth': [10, 15, 20],
        'min_samples_split': [2, 5, 10]
    },
    SVC(random_state=42, probability=True): {
        'C': [0.1, 1, 10],
        'kernel': ['linear', 'rbf'],
        'gamma': ['scale', 'auto']
    },
    KNeighborsClassifier(): {
        'n_neighbors': [3, 5, 7, 10],
        'weights': ['uniform', 'distance']
    },
    DecisionTreeClassifier(random_state=42): {
        'max_depth': [10, 15, 20],
        'min_samples_split': [2, 5, 10]
    },
}
```

Slika 3.8: Programski kod parametara za optimizaciju modela strojnog učenja

Modeli su zatim evaluirani kako bi se pronašao model s najboljim performansama, a rezultati evaluacije prikazani su u tablici 3.3.

Tablica 3.3: Evaluacija modela strojnog učenja

Model	Točnost	Mjera F1	Preciznost	Odziv
Slučajna šuma	88.04%	89.42%	92.08%	86.92%
SVM	84.78%	86.67%	88.35%	85.05%
K najbližih susjeda	83.15%	85.02%	88%	83.24%
Stablo odlučivanja	82.07%	84.06%	87%	81.31%

Kako se može vidjeti iz tablice 3.3, svi modeli pokazali su zadovoljavajuće rezultate. Algoritam slučajne šume je, ipak, pokazao najbolje rezultate te će se koristiti za predviđanje rizika za razvoj kardiovaskularnih bolesti u web rješenju.

4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE

U ovome poglavlju bit će opisan način na koji je ostvareno programsko rješenje na strani korisnika i poslužitelja te tehnologije, alati i jezici koji su korišteni za razvoj projekta.

4.1. Korištene programske tehnologije, alati i jezici

Kako bi se uspješno ispunili zadani zahtjevi aplikacije, važno je odabrati prikladne tehnologije za razvoj web rješenja.

4.1.1. Programski jezik Python

Python [18] je moćan, objektno orijentirani programski jezik visoke razine poznat po svojoj čitljivosti i jednostavnosti. Python je programski jezik opće namjene, što znači da se može koristiti za stvaranje širokog spektra aplikacija, od web razvoja i razvoja igara do analize podataka i umjetne inteligencije. Jednostavna i čista sintaksa čini Python idealnim jezikom za početnike. Naredbe se temelje na engleskom jeziku, a jednostavno oblikovanje pomaže novim programerima da lako razumiju kod. Python također ima i veliki broj biblioteka za različite namjene i aktivnu zajednicu programera koji doprinose njegovom stalnom unapređenju. U posljednjim godinama, Python je doživio porast popularnosti i postao jedan od najčešće korištenih programskih jezika diljem svijeta. Python je u srcu mnogih tehnologija i aplikacija koje koristimo svakodnevno.

4.1.2. Django

Django [19] je besplatan razvojni okvir otvorenog koda koji može značajno ubrzati razvoj web aplikacija izrađenih u programskom jeziku Python. Prvi put je javno objavljen 2005. godine. Koristi se već dugi niz godina te je temeljito testiran i unaprijeđen od strane vrlo aktivne zajednice. Postoji i neprofitna organizacija *Django Software Foundation* koja promiče, podržava i unapređuje Django. Najveća snaga ovog razvojnog okvira je veliki skup značajki, Django sadrži više od 10000 paketa i time pokriva gotovo sve što je potrebno za razvoj web aplikacije. Paketi uključuju API-je, sustave za upravljanje sadržajem, autentifikaciju korisnika, validaciju obrazaca i CAPTCHA zaštitu.

4.1.3. HTML i CSS

HTML (engl. *HyperText Markup Language*) [20] osnovni je jezik za izradu web stranica, a stvorio ga je Tim Berners-Lee 1991. godine. HTML koristi oznake i atribute za opisivanje strukture i formatiranje sadržaja na internetskim stranicama. Web stranice sastoje se od različitih elemenata poput naslova, popisa, slika i poveznica, a HTML govori pretraživačima kako prikazati sadržaj

stranice. Prednosti HTML-a uključuju njegovu široku podršku u svim preglednicima, jednostavnost implementacije te mogućnost integracije s drugim jezicima poput CSS-a i JavaScript-a. CSS (engl. *Cascading Style Sheets)* se koristi za stiliziranje dokumenata napisanih u HTML-u. Dizajniran je kako bi omogućio razdvajanje sadržaja i prikaza što pruža veću fleksibilnost jer se sadržaj može napisati bez razmišljanja o njegovom prikazu.

4.1.4. Bootstrap

Bootstrap [21] je besplatan *front-end* razvojni okvir otvorenoga koda. Bootstrap omogućava stvaranje responzivnog web dizajna, odnosno, omogućava automatsko prilagođavanje ekrana posjetitelja web stranice ili aplikacije u ovisnosti o veličini i orijentaciji ekrana. Bootstrap, također, pruža definicije stilova za sve HTML elemente, a rezultat je ujednačen izgled teksta, tablica i obrazaca bez obzira koji se internetski preglednik koristi. Svaka Bootstrap komponenta sastoji se od HTML strukture, CSS deklaracija i, u nekim slučajevima, pratećeg JavaScript koda.

4.1.5. SQLite baza podataka

SQLite [22] je biblioteka napisana u C programskom jeziku koja implementira malu, brzu, samostalnu, visoko pouzdanu i potpuno opremljenu SQL bazu podataka. SQLite je najkorišteniji sustav baze podataka na svijetu koji je ugrađen u sve mobilne telefone i većinu računala te dolazi u paketu s bezbroj drugih aplikacija koje ljudi koriste svakodnevno. Izbor baze podataka zavisi od potreba projekta, a Django podržava više različitih baza podataka kao što su PostgreSQL, MySQL ili Oracle. Pri izradi novog Django projekta, prema zadanim postavkama koristi se SQLite baza podataka koja ne treba dodatnu konfiguraciju, a bila je dovoljna za potrebe izrade ovog diplomskog rada.

4.2. Virtualno okruženje

Pri radu na projektima koristeći Django preporuča se korištenje virtualnog okruženja. Virtualno okruženje [23] funkcionira kao samostalni kontejner ili izolirano okruženje u kojem su instalirani svi paketi i potrebne verzije specifične za određeni projekt što omogućava rad na više projekata s različitim ovisnostima na istom sustavu.

Virtualno okruženje kreirano je naredbom *python3 -m venv myenv.* Naredba kreira direktorij naziva *myenv* koji sadrži sve potrebne datoteke za rad u izoliranom okruženju. Virtualno okruženje aktivira se naredbom *source myenv/bin/activate.* Kada je virtualno okruženje aktivno, mogu se instalirati paketi koji su potrebni za razvoj ovog projekta, a koji neće utjecati na druge projekte.

Projekt je kreiran naredbom *django-admin startproject heart,* a zatim je unutar projekta heart kreirana aplikacija *predictor* koristeći naredbu *python manage.py startapp predictor.*

4.3. Stvaranje baze podataka

Za interakciju s bazom podataka koristi se **Django ORM** (*Object-Relational Mapping*) koji omogućuje interakciju s bazom podataka koristeći Python kod umjesto pisanja SQL upita, čime se olakšava rad s podacima. Svaki model je jedna Python klasa i predstavlja jednu tablicu u bazi podataka, a svaki atribut klase predstavlja stupac te tablice. Kada se stvori instanca modela, ona predstavlja jedan određeni zapis u toj tablici. Ovaj pristup omogućuje korisnicima da lako dodaju, ažuriraju, brišu i dohvate podatke putem intuitivnih Python metoda, a kada se koriste metode ORM-a, Django automatski generira odgovarajući SQL kod.

Svi modeli se nalaze u datoteci models.py. Potrebni modeli za razvoj web rješenja su *CustomUser* koji nasljeđuje *AbstractUser*, ali dodaje dodatna polja *user_type* za razlikovanje korisnika koji su pacijenti od korisnika koji su liječnici, te *email* koji mora biti jedinstven. Klase *Patient* i *Doctor* koriste *OneToOneField* za povezivanje s modelom *CustomUser.* Klasa *Patient* dodaje polja za datum rođenja i referencu na doktora koristeći *ForeignKey*, dok klasa *Doctor* dodaje polje za ID liječnika koji treba biti dužine 10 znamenki. Klasa *Prediction* pohranjuje sve podatke koji se koriste za predviđanje rizika za razvoj kardiovaskularnih bolesti, rezultat i datum predviđanja te ima referencu na pacijenta. Posljednja klasa je klasa *Recommendation* koja ima referencu na doktora koji daje preporuku i pacijenta kojem je preporuka namijenjena, a pohranjuje sadržaj preporuke i vrijeme kreiranja. Programski kodovi svih modela u *models.py* datoteci prikazani su na slikama 4.1, 4.2 i 4.3.

```python
class CustomUser(AbstractUser):
    USER_TYPE_CHOICES = (
        ('patient', 'Patient'),
        ('doctor', 'Doctor'))
    user_type = models.CharField(max_length=10, choices=USER_TYPE_CHOICES)
    email = models.EmailField(unique=True)

class Patient(models.Model):
    user = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
    date_of_birth = models.DateField()
    doctor = models.ForeignKey('Doctor', on_delete=models.SET_NULL, null=True, blank=True)

class Doctor(models.Model):
    user = models.OneToOneField(CustomUser, on_delete=models.CASCADE, related_name='doctor')
    doctor_id = models.CharField(max_length=10, unique=True)

    def clean(self):
        from django.core.exceptions import ValidationError
        if len(self.doctor_id) != 10:
            raise ValidationError("Doctor ID must be exactly 10 characters long.")
```

Slika 4.1: Programski kod modela za stvaranje korisnika

```python
class Prediction(models.Model):
    patient = models.ForeignKey(Patient, on_delete=models.CASCADE, related_name='predictions')
    age = models.IntegerField()
    gender = models.CharField(max_length=1, choices=[('M', 'Male'), ('F', 'Female')])
    chest_pain_type = models.CharField(max_length=3,
                                       choices=[('TA', 'Typical Angina'),
                                                ('ATA', 'Atypical Angina'),
                                                ('NAP', 'Non-Anginal Pain'),
                                                ('ASY', 'Asymptomatic')])
    restingbp = models.IntegerField()
    cholesterol = models.IntegerField()
    fastingbs = models.IntegerField()
    restingecg = models.CharField(max_length=6, choices=[('Normal', 'Normal'), ('ST', 'ST'),
                                                          ('LVH', 'LVH')])
    maxhr = models.IntegerField()
    exerciseangina = models.CharField(max_length=1, choices=[('Y', 'Yes'), ('N', 'No')])
    oldpeak = models.FloatField()
    st_slope = models.CharField(max_length=4, choices=[('Up', 'Upsloping'), ('Flat', 'Flat'),
                                                        ('Down', 'Downsloping')])
    heart_disease_risk = models.CharField(max_length=20)
    prediction_date = models.DateTimeField(auto_now_add=True)
```

Slika 4.2: Programski kod modela predviđanja

```python
class Recommendation(models.Model):
    content = models.TextField()
    patient = models.ForeignKey('Patient', on_delete=models.CASCADE)
    doctor = models.ForeignKey('Doctor', on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    recommendation = models.CharField(max_length=255, blank=True, null=True)

    class Meta:
        ordering = ['-created_at']
```

Slika 4.3: Programski kod modela preporuke

Nakon što su modeli definirani potrebno je pokrenuti naredbu *python manage.py makemigrations* koja generira odgovarajuće migracijske datoteke, a zatim naredbom *python manage.py migrate* primijeniti migracije na bazu podataka.

4.4. Registracija korisnika

Korisnici se mogu registrirati i koristiti aplikaciju u ulozi pacijenta ili liječnika. Proces registracije će biti objašnjen na primjeru registracije pacijenta. Registracija je implementirana kroz niz povezanih komponenti koje osiguravaju učinkovito i sigurno prikupljanje i pohranu korisničkih podataka.

URL za registraciju pacijenta definira se u datoteci *urls.py* kao *register/patient/*. Ova putanja povezuje korisnički zahtjev za registraciju pacijenta s funkcijom *register_patient* u datoteci

views.py. Kada korisnik posjeti ovu adresu, Django prepoznaje da se radi o registraciji pacijenta i automatski poziva funkciju *register_patient*, koja obrađuje sve zahtjeve vezane uz registraciju. Programski kod datoteke *urls.py* prikazan je na slici 4.4.

```
urlpatterns = [
    path('', views.home, name='home'),
    path('login/', views.login, name='login'),
    path('logout/', views.logout, name='logout'),
    path('register/doctor/', views.register_doctor, name='register_doctor'),
    path('register/patient/', views.register_patient, name='register_patient'),
    path('profile/', views.profile, name='profile'),
    path('heart/', views.heart, name='heart'),
    path('doctor/patients/', views.doctor_patients, name='doctor_patients'),
    path('patient/<int:id>/', views.patient_detail, name='patient_detail'),
    path('patient/<int:patient_id>/add_recommendation/',
         views.add_recommendation_to_patient, name='add_recommendation'),
    path('recommendation/delete/<int:recommendation_id>/',
         views.delete_recommendation, name='delete_recommendation'),
]
```

Slika 4.4: Programski kod urls.py datoteke

Funkcija *register_patient*, koja se nalazi u datoteci *views.py*, inicijalizira prazan obrazac *PatientRegistrationForm* kada korisnik prvi put otvara stranicu za registraciju. Kada korisnik ispuni obrazac i pošalje ga putem HTTP POST metode, funkcija stvara instancu obrasca koristeći unesene podatke. Ako su podaci u obrascu ispravni, funkcija poziva *form.save()* za kreiranje novog korisnika. Nakon uspješne registracije, korisnik se automatski prijavljuje pomoću *auth_login*` i preusmjerava na stranicu profila. Programski kod opisane funkcije prikazan je na slici 4.5.

```
def register_patient(request):
    if request.method == 'POST':
        form = PatientRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save()
            auth_login(request, user)
            return redirect('profile')
    else:
        form = PatientRegistrationForm()
    return render(request, 'register_patient.html', {'form': form})
```

Slika 4.5: Programski kod funkcije *register_patient*

Obrazac za registraciju *PatientRegistrationForm* nalazi se u datoteci *forms.py*, a proširuje ugrađeni obrazac *UserCreationForm*. *UserCreationForm* sadrži polja za korisničko ime *(username),* lozinku *(password1 i password2)* i omogućava kreiranje novog korisničkog računa uz automatsku

validaciju lozinki.*PatientRegistrationForm* dodaje polja *first_name*, *last_name*, *email* i *date_of_birth*. Ovaj obrazac sadrži metodu *clean_email* koja provjerava postoji li već korisnik s istim *email-om* u bazi podataka te *save* metodu koja postavlja email kao korisničko ime i *user_type* na *patient*. Nakon spremanja korisnika, kreira se *Patient* objekt i povezuje s korisnikom. Slika 4.6 prikazuje programski kod obrasca *PatientRegistrationForm.*

```
class PatientRegistrationForm(UserCreationForm):
    first_name = forms.CharField(max_length=30, label='First Name')
    last_name = forms.CharField(max_length=30, label='Last Name')
    email = forms.EmailField(required=True, label='Email')
    date_of_birth = forms.DateField(widget=forms.SelectDateWidget(years=range(1900, 2025)),
                                    label='Date of Birth')
    usable_password = None

    class Meta:
        model = CustomUser
        fields = ('first_name', 'last_name', 'email', 'password1', 'password2')

    def clean_email(self):
        email = self.cleaned_data.get('email')
        if CustomUser.objects.filter(email=email).exists():
            raise forms.ValidationError('A user with this email already exists.')
        return email

    def save(self, commit=True):
        user = super().save(commit=False)
        user.email = self.cleaned_data['email']
        user.username = self.cleaned_data['email']
        user.user_type = 'patient'
        if commit:
            user.save()
        patient = Patient(user=user, date_of_birth=self.cleaned_data['date_of_birth'])
        if commit:
            patient.save()
        return user
```

Slika 4.6: Programski kod obrasca *PatientRegistrationForm*

Na strani korisnika, *register_patient.html* prikazuje obrazac za registraciju unutar osnovnog dizajna stranice. *Template* koristi {% csrf_token %} što generira skriveno polje u obrascu s CSRF tokenom. Kada se obrazac pošalje na poslužitelj, Django provjerava CSRF token koji je poslan sa zahtjevom. Ako token ne odgovara onome koji je očekivan, zahtjev se odbacuje kao potencijalno maliciozan. Praktična metoda za generiranje jednostavnog i funkcionalnog HTML-a za obrasce je {{form.as_p}}. Ova metoda automatski upravlja prikazom svakog polja obrasca unutar <p> tagova, uključujući *label* i poruke o greškama, što omogućava brzu i učinkovitu izgradnju obrazaca. *Template register_patient.html* prikazan je na slici 4.7.

```
{% extends 'base.html' %}

{% block content %}
<h2>Register as Patient</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" class="btn btn-primary">Register</button>
</form>
{% endblock %}
```

Slika 4.5: Programski kod register_patient.html obrasca

Postupak registracije korisnika u ulozi liječnika isti je kao kod korisnika u ulozi pacijenta, jedina je razlika u poljima obrasca kojega ispunjavaju pri registraciji.

4.5. Prijava korisnika

Funkcija *login* koja se nalazi u *views.py* datoteci upravlja procesom prijave korisnika. Kada korisnik pošalje obrazac za prijavu putem HTTP POST metode, funkcija stvara instancu obrasca *EmailLoginForm* koristeći podatke koje je korisnik unio u obrazac. Ako su uneseni podaci valjani, funkcija poziva metodu *form.get_user()* za dohvaćanje korisničkog objekta i koristi *auth_login* za autentifikaciju i prijavu korisnika. Nakon što je korisnik uspješno prijavljen, preusmjerava se na početnu stranicu. Ako zahtjev nije POST, funkcija inicijalizira prazan obrazac i prikazuje ga putem predloška *login.html*. Slika 4.8 prikazuje programski kod funkcije *login*.

```
def login(request):
    if request.method == 'POST':
        form = EmailLoginForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            auth_login(request, user)
            return redirect('home')
    else:
        form = EmailLoginForm()
    return render(request, 'login.html', {'form': form})
```

Slika 4.6: Programski kod funkcije *login*

Klasa *EmailLoginForm* nalazi se u *forms.py* i proširuje *AuthenticationForm* te prilagođava obrazac kako bi se za prijavu koristila adresa elektroničke pošte umjesto korisničkog imena. Opisano je prikazano na slici 4.9.

```
class EmailLoginForm(AuthenticationForm):
    username = forms.EmailField(label='Email')
```

Slika 4.7: Programski kod *EmailLoginForm* obrasca

Programski kod login.html dokumenta gotovo je jednak programskom kodu register_patient.html dokumenta prikazanog na slici 4.6.

4.6. Implementacija modela strojnog učenja

Naredbom *python predictor/train_model.py* u terminalu, pokreće se skripta *train_model.py* u kojoj se nalazi sav kod potreban za predobradbu podataka i treniranje modela. Kada je model uspješno obučen, skripta sprema model i alate za obradu podataka kako bi ih bilo moguće dohvatiti za kasniju upotrebu. Programski kod dohvaćanja modela i alata iz spremljenih datoteka prikazan je na slici 4.10. Na slici 4.10 može se vidjeti da se za učitavanje unaprijed treniranog modela i alata za predobradbu podataka koristi *joblib* biblioteka.

```
model = joblib.load(os.path.join(models_path, 'best_rf_model.pkl'))
scaler = joblib.load(os.path.join(models_path, 'scaler.pkl'))
poly = joblib.load(os.path.join(models_path, 'poly.pkl'))
label_encoders = joblib.load(os.path.join(models_path, 'label_encoders.pkl'))
```

Slika 4.8: Dohvaćanje modela i alata za obradu iz spremljenih datoteka

Kada korisnik ispravno ispuni obrazac za procjenu rizika, podaci se obrađuju kako bi odgovarali formatu koji model očekuje, a zatim se predviđa prisutnost rizika za razvoj kardiovaskularnih bolesti na temelju podataka koje je korisnik unio u obrazac. Točnost modela se, također, šalje u predložak kako bi bila prikazana korisniku. Na slici 4.12 prikazan je programski kod koji obrađuje POST zahtjev za predviđanje rizika od kardiovaskularnih bolesti na temelju unosa korisnika. Podaci uneseni u obrazac se validiraju pomoću *form.is_valid()*. Ako je obrazac valjan, podaci iz njega se izdvajaju i spremaju u rječnik *user_data* gdje svako polje odgovara specifičnom unosu iz obrasca. Podaci se zatim pretvaraju u *pandas DataFrame* oblik kako bi bili kompatibilni s alatima za obradu podataka i modelom strojnog učenja koji očekuju ulaz u tabličnom obliku.

```
if request.method == 'POST':
    form = HeartDiseaseForm(request.POST)
    if form.is_valid():
        user_data = {
            'Age': form.cleaned_data['age'],
            'Gender': form.cleaned_data['gender'],
            'ChestPainType': form.cleaned_data['cp'],
            'RestingBP': form.cleaned_data['trestbps'],
            'Cholesterol': form.cleaned_data['chol'],
            'FastingBS': form.cleaned_data['fbs'],
            'RestingECG': form.cleaned_data['restecg'],
            'MaxHR': form.cleaned_data['maxhr'],
            'ExerciseAngina': form.cleaned_data['exang'],
            'Oldpeak': form.cleaned_data['oldpeak'],
            'ST_Slope': form.cleaned_data['slope']
        }
        user_data_df = pd.DataFrame([user_data])

        # Preprocess the user data
        user_data_encoded = preprocessing(user_data_df, is_training=False,
                                          label_encoders=label_encoders)
        user_data_poly = poly.transform(user_data_encoded)
        user_data_scaled = scaler.transform(user_data_poly)

        # Make prediction
        prediction = model.predict(user_data_scaled)[0]
        result = 'have' if prediction == 1 else "don't have"
```

Slika 4.9: Programski kod obrade korisničkih podataka i izvođenja predviđanja

4.7. Praćenje tijeka bolesti

Kako bi bilo moguće pratiti kako se mijenja zdravstveno stanje pacijenta, uneseni podaci, rezultat, datum i vrijeme predviđanja pohranjuju se u bazu podataka. Korisnika se zatim preusmjerava na njegov profil. Slika 4.12 prikazuje isječak programskog koda za spremanje unesenih podataka i rezultata predviđanje u bazu podataka. Funkcijom *Prediction.objects.create()* stvara se nova instanca modela *Prediction* koja se odmah pohranjuje u bazu podataka, a sadrži podatke preuzete iz obrasca.

```
Prediction.objects.create(
    patient=patient_profile,
    age=form.cleaned_data['age'],
    gender=form.cleaned_data['gender'],
    chest_pain_type=form.cleaned_data['cp'],
    restingbp=form.cleaned_data['trestbps'],
    cholesterol=form.cleaned_data['chol'],
    fastingbs=form.cleaned_data['fbs'],
    restingecg=form.cleaned_data['restecg'],
    maxhr=form.cleaned_data['maxhr'],
    exerciseangina=form.cleaned_data['exang'],
    oldpeak=form.cleaned_data['oldpeak'],
    st_slope=form.cleaned_data['slope'],
    heart_disease_risk=result,
    prediction_date=timezone.now().date()
)
return redirect('profile')
```

Slika 4.10: Isječak programskog koda za spremanje predviđanja u bazu podataka

Uneseni podaci i rezultat predviđanja zatim se dostupni pacijentu i liječniku na profilu pacijenta. Programski kod za prikaz predviđanja na profilu pacijenta prikazan je na slici 4.13.

```
{% if latest_prediction %}
    <!-- Display Latest Prediction -->
    <p><strong>Latest Prediction:</strong> You {{ latest_prediction.heart_disease_risk }}
        risk of heart disease.</p>
    <p><strong>Prediction Date:</strong>
        {{ latest_prediction.prediction_date|date:"F j, Y, g:i a" }}</p>
    {% if latest_prediction.heart_disease_risk == 'have' %}
        <p class="text-danger"><strong>Please consult your assigned doctor.</strong></p>
    {% endif %}
    <!-- Display Latest Input Data -->
    <h4>Latest Input Data:</h4>
    <ul class="list-group mb-3">
        <li class="list-group-item"><strong>Age:</strong>
            {{ latest_prediction.age }}</li>
        <li class="list-group-item"><strong>Gender:</strong>
            {{ latest_prediction.gender}}</li>
        <li class="list-group-item"><strong>Chest Pain Type:</strong>
            {{ latest_prediction.chest_pain_type }}</li>
        <li class="list-group-item"><strong>Resting Blood Pressure:</strong>
            {{ latest_prediction.restingbp }}</li>
        <li class="list-group-item"><strong>Cholesterol:</strong>
            {{ latest_prediction.cholesterol }}</li>
        <li class="list-group-item"><strong>Fasting Blood Sugar:</strong>
            {{ latest_prediction.fastingbs }}</li>
        <li class="list-group-item"><strong>Resting ECG:</strong>
            {{ latest_prediction.restingecg }}</li>
        <li class="list-group-item"><strong>Max Heart Rate:</strong>
            {{ latest_prediction.maxhr }}</li>
        <li class="list-group-item"><strong>Exercise Induced Angina:</strong>
            {{ latest_prediction.exerciseangina }}</li>
        <li class="list-group-item"><strong>Oldpeak:</strong>
            {{ latest_prediction.oldpeak }}</li>
        <li class="list-group-item"><strong>Slope:</strong>
            {{ latest_prediction.st_slope }}</li>
    </ul>
```

Slika 4.11: Isječak programskog koda za prikaz predviđanja na profilu pacijenta

Također, sustav korisniku u ulozi liječnika predlaže preporuke s obzirom na zdravstveno stanje pacijenta. Zdravstveno stanje procjenjuje se na temelju posljednjeg predviđanja kojega je pacijent napravio. Ukoliko rizik postoji, preporuke se temelje na unesenim parametrima. Recimo, ako je vrijednost tlaka koju je pacijent unio visoka, preporuka je da se redovno prati krvni tlak i da se uvedu promjene u način života. Liječnik može odabrati jednu ili više ponuđenih preporuka, urediti ih ili napisati novu. Ako kod pacijenta rizik za nastanak bolesti ne postoji, sustav govori da je s pacijentom sve u redu. Programski kod funkcije za stvaranje preporuka prikazan je na slici 4.14.

```python
def get_recommendations(latest_prediction):
    recommendations = []

    if latest_prediction.heart_disease_risk == 'have':
        if latest_prediction.cholesterol > 200:
            recommendations.append("Heart-healthy diet and regular moderate-intensity exercise"
                                   "recommended.")
        if latest_prediction.restingbp > 130:
            recommendations.append("Regularly monitor your blood pressure and adopt lifestyle"
                                   "changes.")
        if latest_prediction.fastingbs == 1:
            recommendations.append("Balanced diet, regular exercise and further medical"
                                   "evaluation for diabetes recommended.")
        if latest_prediction.exerciseangina == 'Y':
            recommendations.append("Avoid strenuous activities until further evaluation"
                                   "is completed.")
        if latest_prediction.chest_pain_type != 'ASY' or \
            latest_prediction.restingecg != 'Normal' or \
           latest_prediction.oldpeak > 1 or \
            latest_prediction.st_slope != 'Flat':
            recommendations.append("Further medical evaluation needed.")
    else:
        recommendations.append("The patient is okay with no significant risks detected.")

    return recommendations
```

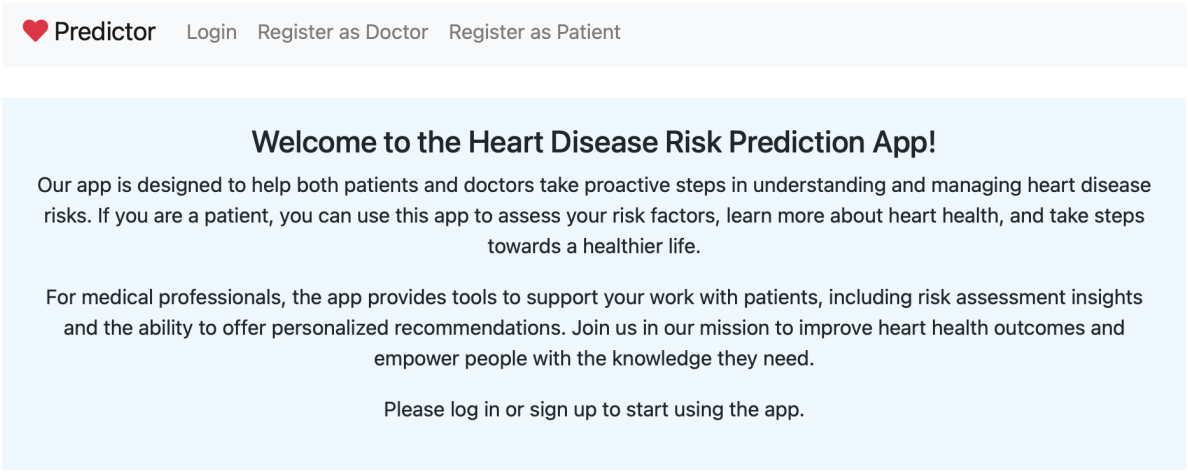
Slika 4.12: Programski kod metode za stvaranje preporuka

5. NAČIN KORIŠTENJA I ANALIZA RADA APLIKACIJE

U ovome poglavlju bit će objašnjen način korištenja aplikacije te će se ispitati radi li aplikacija kako je očekivano.

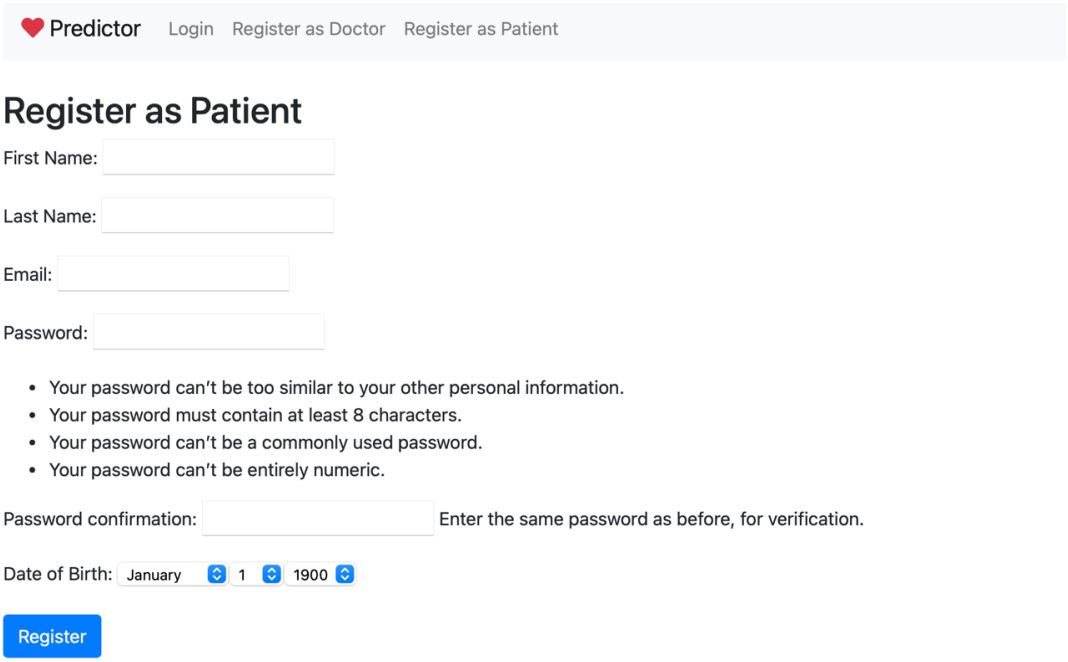
5.1. Početni zaslon, prijava i registracija korisnika

Nakon pokretanja aplikacije, otvara se početni zaslon s kratkim opisom aplikacije. Pomoću navigacijske trake moguće je otvoriti sučelje za prijavu, registraciju pacijenta ili registraciju liječnika.

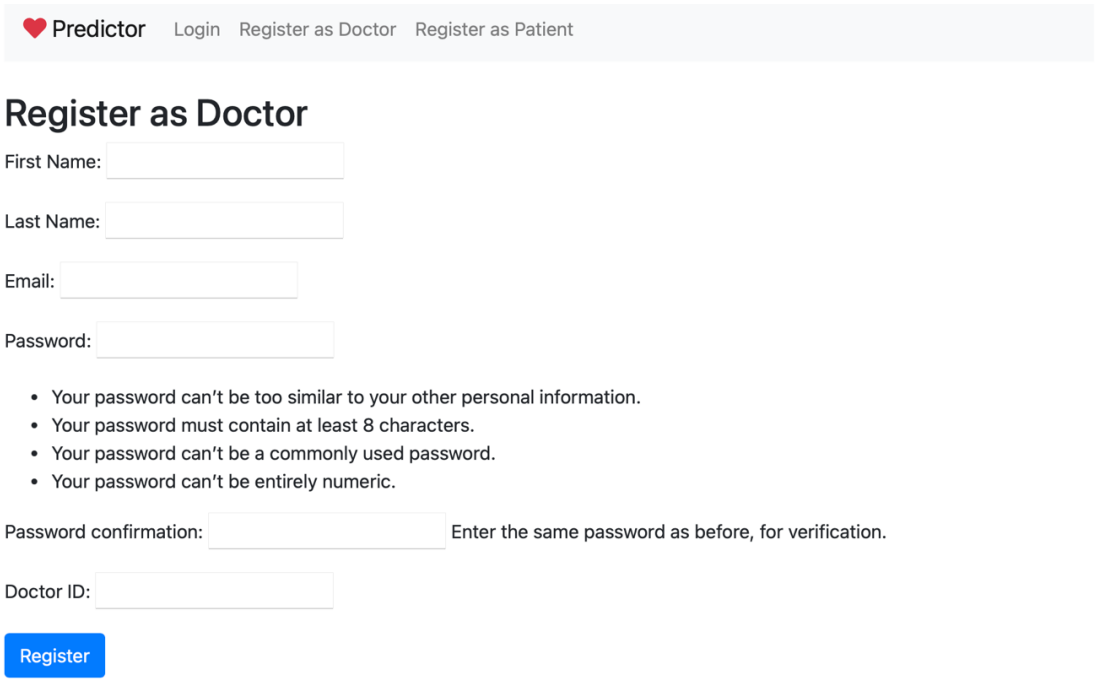


Slika 5.1: Početni zaslon neregistriranog korisnika

Sučelja za registraciju pacijenta i liječnika su jako slična, jedina je razlika u tome što pacijent pri registraciji unosi datum rođenja, a liječnik ne, ali liječnik unosi svoj identifikacijski broj liječnika.

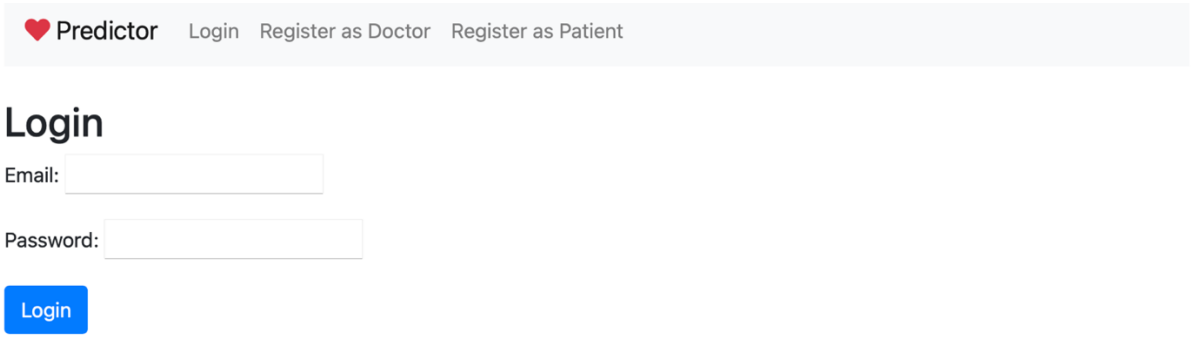


Slika 5.2: Sučelje za registraciju pacijenta



Slika 5.3: Sučelje za registraciju liječnika

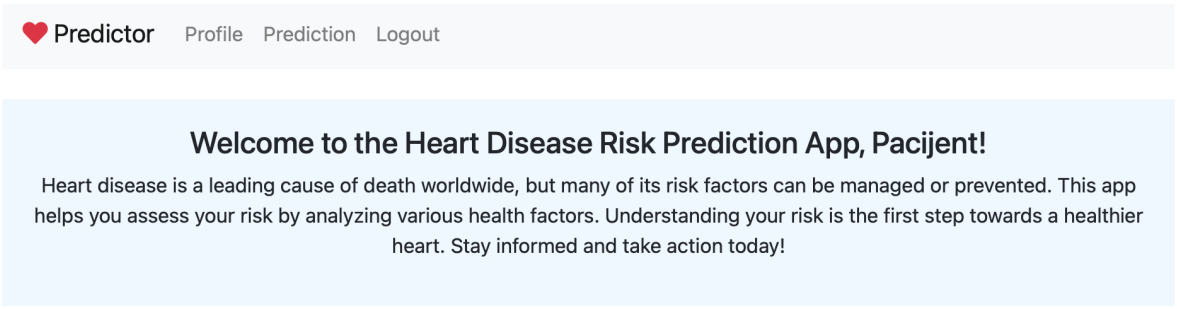
Ako je korisnik već registriran, u aplikaciju se prijavljuje koristeći email adresu i lozinku.



Slika 5.4: Sučelje za prijavu korisnika

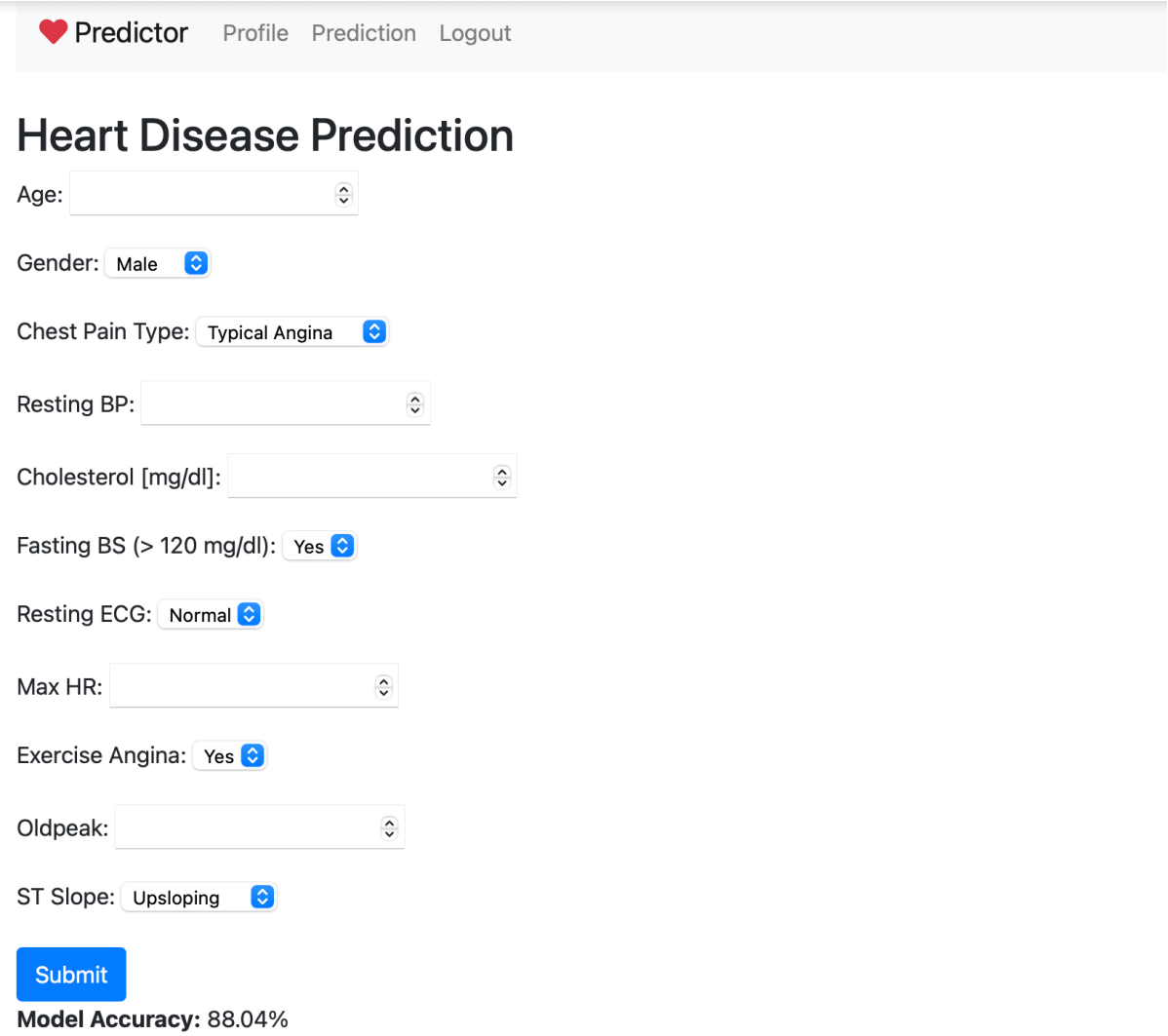
5.2. Korištenje aplikacije u ulozi pacijenta

Nakon uspješne prijave, pacijent ponovno vidi poruku sličnu poruci koja se pojavljuje pri pokretanju aplikacije neregistriranim korisnicima. Poruka na početnom zaslonu drugačija je za neregistrirane korisnike, pacijente i liječnike.



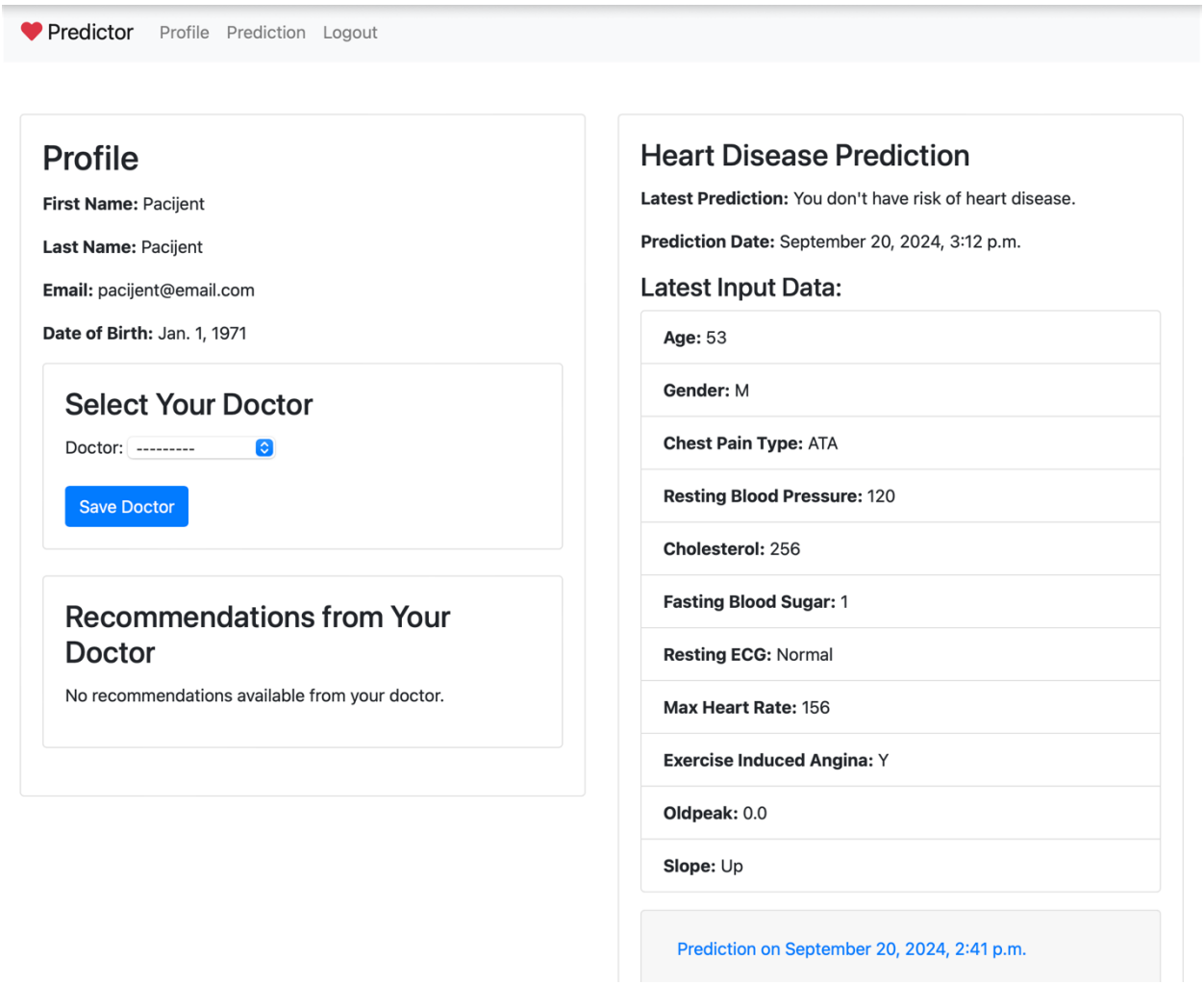
Slika 5.5: Poruka na početnom zaslonu korisniku u ulozi pacijenta

Sučelje za izračun rizika dostupno je samo korisnicima koji su pacijenti, a do njega se dolazi pomoću poveznice *Prediction* u navigacijskoj traci aplikacije. Kako bi se izračunao rizik od razvoja kardiovaskularne bolesti potrebno je ispuniti obrazac koji se otvara dolaskom na sučelje. Korisniku je prikazano i da točnost modela iznosi 88.04% što se može vidjeti na slici 5.6.



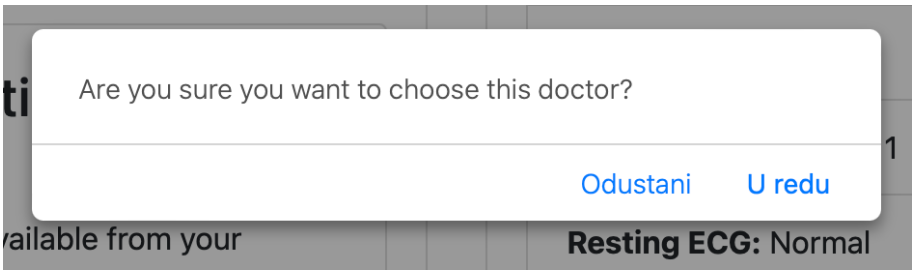
Slika 5.6: Sučelje za izračun rizika

Kada je izračun rizika na temelju podataka koje je korisnik unio završen, na profilu pacijenta prikazuju se rezultat i uneseni podaci za posljednje predviđanje. Ispod toga je moguće vidjeti datume prijašnjih predviđanja, a klikom na njih otvaraju se rezultati, kao i uneseni parametri.



Slika 5.7: Vlastiti profil pacijenta

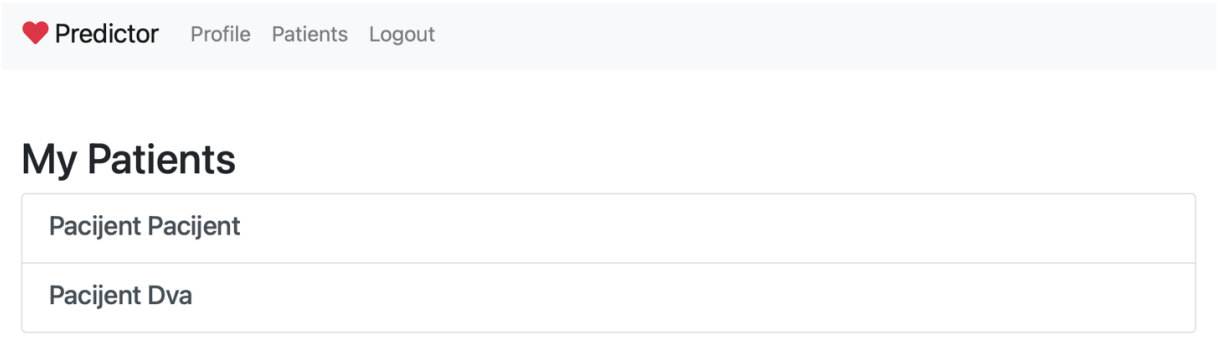
Na slici 5.7 može se vidjeti da ovaj pacijent još nema izabranog liječnika. Pacijentu su u padajućem izborniku ponuđeni svi liječnici koji su se registrirali u aplikaciju, a on može odabrati jednog od njih. Nakon pritiska tipke *Save Doctor* korisniku se otvara prozor u kojem treba još jednom potvrditi odabir liječnika. Na slici 5.8 prikazan je prozor za potvrdu odabira liječnika.



Slika 5.8: Prozor za potvrdu odabira liječnika

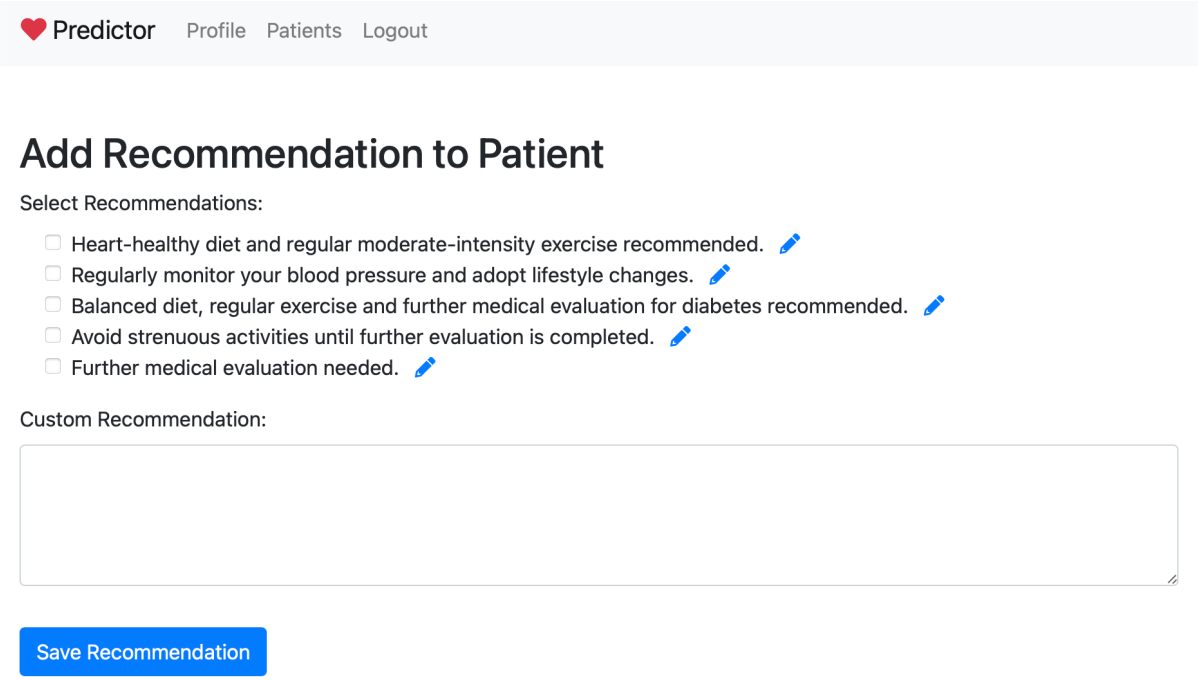
5.3. Korištenje aplikacije u ulozi liječnika

Nakon uspješne prijave, liječnik, također, na početnom zaslonu vidi poruku sličnu onoj koju vide i pacijent i neregistrirani korisnik. Liječniku je dostupno sučelje na kojemu je popis svih njegovih pacijenata, a do njega dolazi pomoću poveznice *Patients* u navigacijskoj traci aplikacije. Slika 5.9 prikazuje sučelje za prikaz popisa pacijenata.

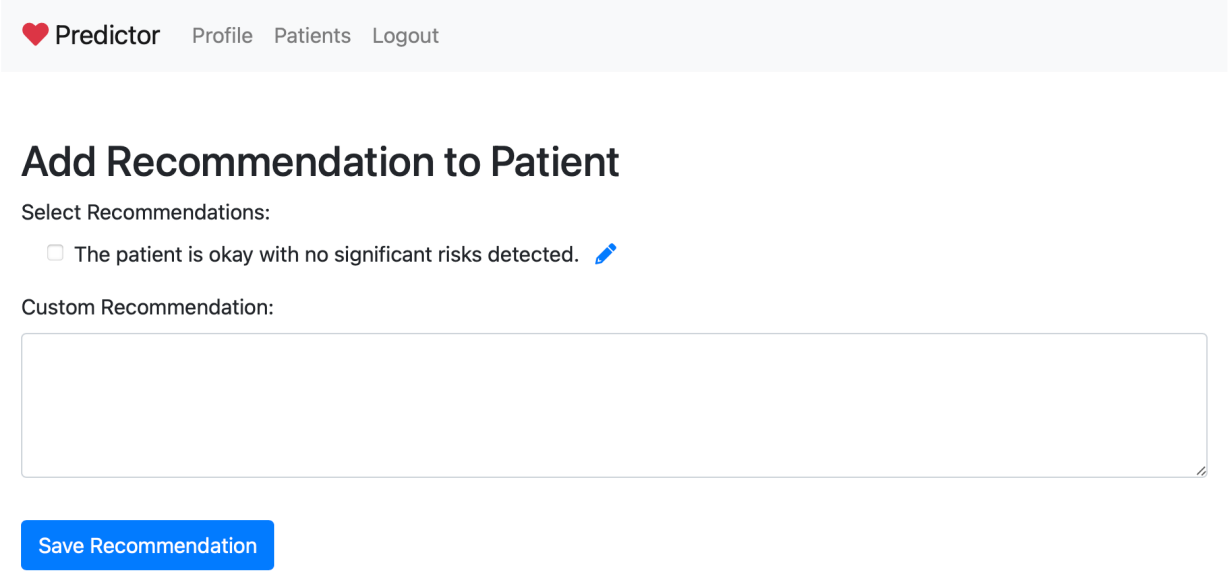


Slika 5.9: Sučelje za prikaz svih pacijenata

Klikom na ime i prezime pacijenta iz popisa, otvara se njegov profil. Na profilu pacijenta nalazi se tipka *Add a recommendation for this patient,* a pritiskom na nju otvara se sučelje za stvaranje preporuka. Liječnik može odabrati jednu ili više preporuka ponuđenih od strane sustava, klikom na ikonu olovke pored preporuke može urediti tu preporuku, a može u tekstualni okvir upisati i potpuno novu, vlastitu preporuku. Slika 5.10 prikazuje sučelje za preporuke kod pacijenta koji ima rizik za razvoj kardiovaskularnih bolesti, a slika 5.11 sučelje za preporuke kod pacijenta koji nema rizik za razvoj bolesti..

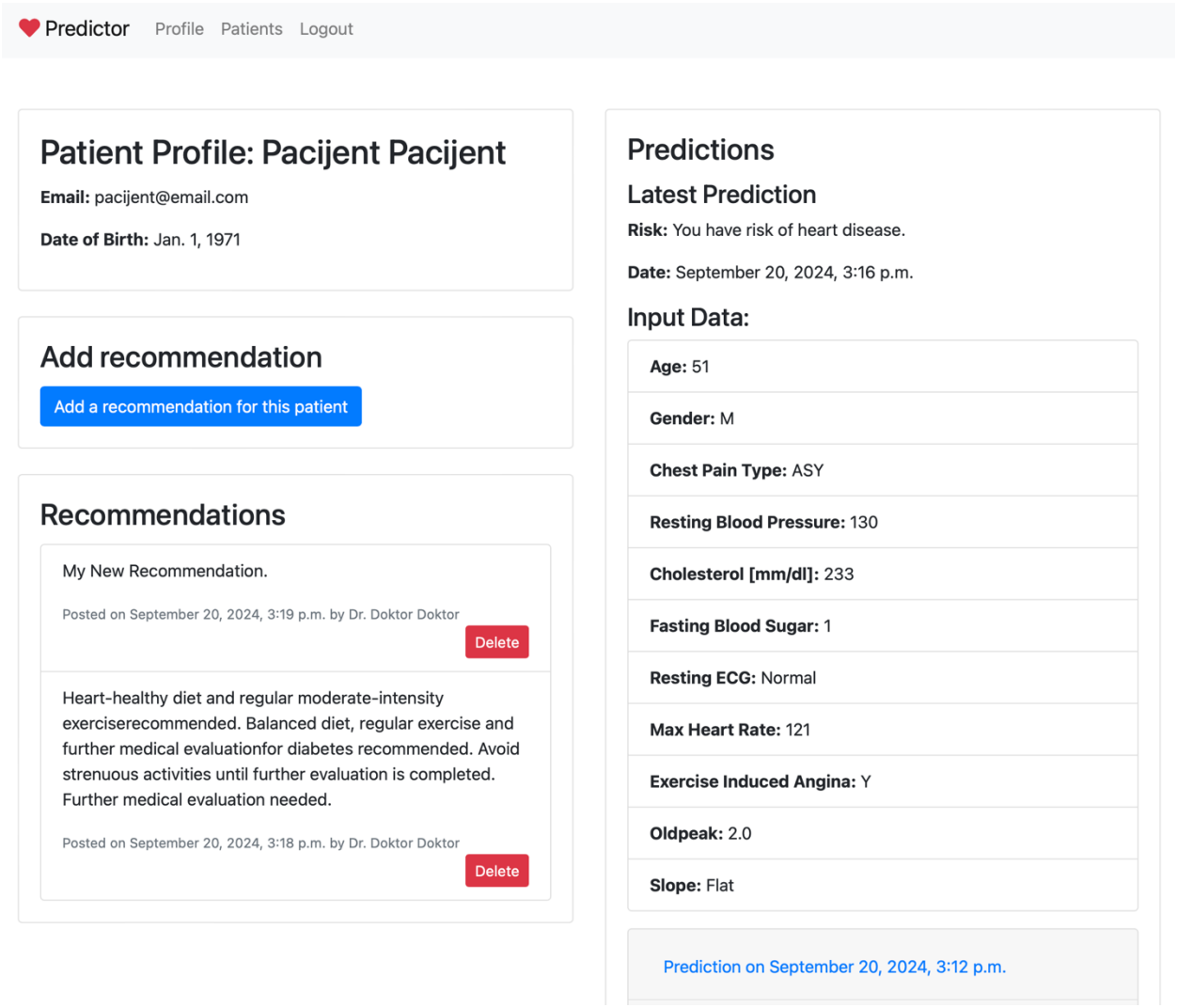


Slika 5.10: Sučelje za preporuke za pacijenta koji ima rizik za nastanak bolesti



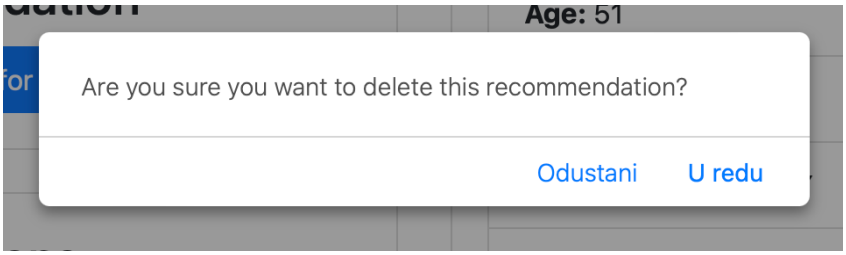
Slika 5.11: Sučelje za preporuke za pacijenta koji nema rizik za nastanak bolesti

Ako je liječnik dodao preporuke, na profile će ih vidjeti i liječnik i pacijent, ali liječnik preporuke može brisati, dok pacijent ne može.



Slika 5.12: Profil pacijenta iz perspektive liječnika

Kada liječnik želi obrisati preporuku, nakon pritiska tipke *Delete* pored preporuke, pojavljuje se prozor u kojem je potrebno još jednom potvrditi brisanje preporuke.



Slika 5.13: Prozor za potvrdu brisanja preporuke

5.4. Analiza rada aplikacije

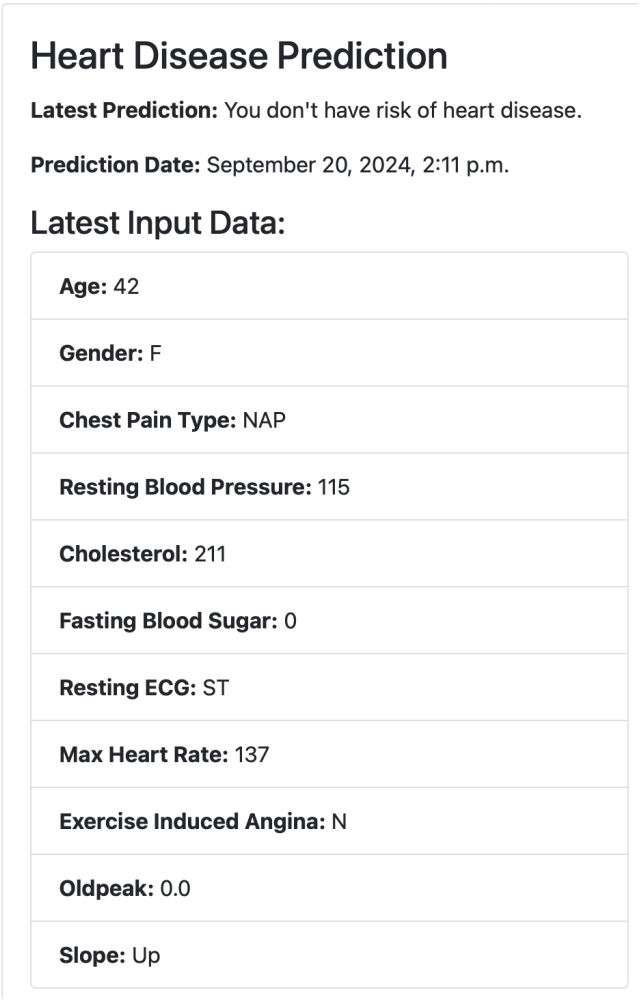
Kako bi se provjerilo daje li aplikacija zaista točne rezultate, predviđanje rizika za razvoj kardiovaskularnih bolesti bit će napravljeno koristeći podatke iz skupa podataka te će se usporediti dobiveni i očekivani rezultat.

5.4.1. Testni slučaj 1

Tablica 5.1 prikazuje podatke iz skupa podataka za prvi testni slučaj, a slika 5.14 unesene podatke i rezultat dobiven aplikacijom.

Tablica 5.1: Podaci iz skupa podataka za testni slučaj 1

Age	42
Gender	F
Chest Pain Type	NAP
Resting Blood Pressure	115
Cholesterol	211
Fasting Blood Sugar	0
Resting ECG	ST
Max Heart Rate	137
Exercise Induced Angina	N
Oldpeak	0.0
Slope	Up
HeartDisease	**0**



Slika 5.14: Uneseni podaci i dobiveni rezultat za testni slučaj 1

Uspoređujući tablicu 5.1 i sliku 5.14 može se vidjeti da za ovaj slučaj rezultat odgovara očekivanom rezultatu, odnosno da pacijent nema rizik za razvoj kardiovaskularnih bolesti.

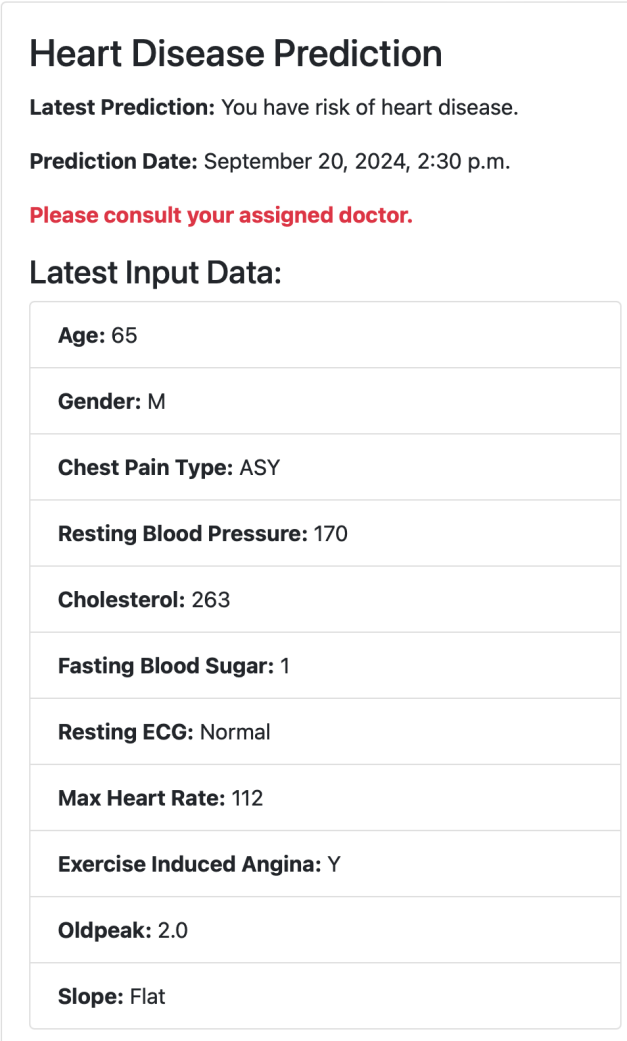
5.4.2. Testni slučaj 2

Tablica 5.2 prikazuje podatke iz skupa podataka za drugi testni slučaj, a slika 5.15 unesene podatke i rezultat dobiven aplikacijom.

Tablica 5.2: Podaci iz skupa podataka za testni slučaj 2

Age	65
Gender	M
Chest Pain Type	ASY
Resting Blood Pressure	170
Cholesterol	263
Fasting Blood Sugar	1
Resting ECG	Normal

Max Heart Rate	112
Exercise Induced Angina	Y
Oldpeak	2
Slope	Flat
HeartDisease	**1**



Slika 5.15: Uneseni podaci i dobiveni rezultat za testni slučaj 2

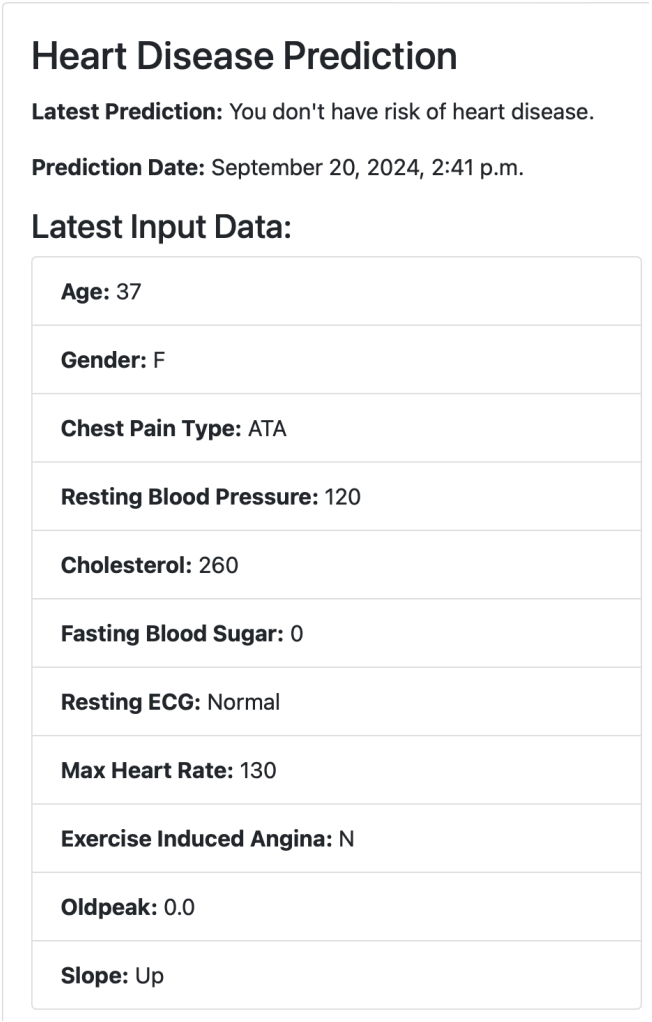
Na slici se može vidjeti da je dobiveni rezultat jednak očekivanom, kod pacijenta postoji rizik za razvoj kardiovaskularnih bolesti.

5.4.3. Testni slučaj 3

Tablica 5.3 prikazuje podatke iz skupa podataka za posljednji testni slučaj, a slika 5.16 unesene podatke i rezultat dobiven aplikacijom

Tablica 5.3: Podaci iz skupa podataka za testni slučaj 3

Age	37
Gender	F
Chest Pain Type	ATA
Resting Blood Pressure	120
Cholesterol	260
Fasting Blood Sugar	0
Resting ECG	Normal
Max Heart Rate	130
Exercise Induced Angina	N
Oldpeak	0
Slope	Up
HeartDisease	**0**



Slika 5.16: Uneseni podaci i dobiveni rezultat za testni slučaj 3

Rezultat i u testnom slučaju 3 odgovara očekivanom rezultatu za unesene podatke, odnosno kod pacijenta ne postoji rizik za razvoj kardiovaskularne bolesti.
Za sva tri testna slučaja rezultat je bio jednak očekivanom rezultatu iz čega se može zaključiti da aplikacija radi ispravno.

6. ZAKLJUČAK

U ovom diplomskom radu opisane su kardiovaskularne bolesti te je razvijena web aplikacija za procjenu rizika i praćenje tijeka razvoja kardiovaskularnih bolesti s ciljem poboljšanja kvalitete zdravstvene skrbi. Aplikacija je dizajnirana da bude korisna za pacijente, ali i za zdravstvene radnike. Zdravstveni radnici mogu koristiti ovu aplikaciju kao alat za podršku dijagnozi, što im omogućuje da preciznije i učinkovitije procjene rizike kod svojih pacijenata. S druge strane, pacijenti mogu koristiti aplikaciju kako bi pratili svoje kardiovaskularno zdravlje, što je posebno korisno za one koji su već identificirani kao osobe s čimbenicima rizika ili za one koji žele unaprijed otkriti moguće rizike. Ključna komponenta aplikacije je model strojnog učenja zasnovan na algoritmu slučajne šume. Ovaj model omogućuje korisnicima aplikacije da procijene svoj rizik od kardiovaskularnih bolesti i pomaže liječnicima u donošenju preporuka na osnovu zdravstvenog stanja pacijenata. Aplikacija ne omogućuje samo procjenu rizika, nego i stalno praćenje tijeka bolesti, jer su sva predviđanja i uneseni podaci dostupni u bazi podataka.

Ispitivanje ispravnosti rada aplikacije provedeno je za tri slučaja iz skupa podataka te su uspoređeni očekivani rezultati i rezultati koje je dala aplikacija. Rezultati ispitivanja pokazuju da aplikacija ispravno procjenjuje postoji li kod osobe rizik za razvoj kardiovaskularnih bolesti ili ne. Aplikaciju bi se moglo proširiti integracijom s vanjskim sustavima za dodatnu provjeru korisničkih podataka. Također, dodavanje naprednih funkcionalnosti za administraciju korisničkih računa i sadržaja bi moglo poboljšati korisničko iskustvo i sigurnost. Integracija s dodatnim izvorima podataka i vanjskim sustavima mogla bi omogućiti još preciznije prognoze, što bi dodatno unaprijedilo mogućnosti aplikacije u prevenciji kardiovaskularnih bolesti.

LITERATURA

[1] B. Maćešić, B. Špehar, Prevencija kardiovaskularnih bolesti u primarnoj zdravstvenoj zaštiti, Sestrinski glasnik, br. 1, sv. 19, str. 30-41, 2014.

[2] Duale Reihe, Anatomija (urednici hrvatskog izdanja: V. Katavić, Z. Petanjek, I. Vinter), Medicinska naklada, Zagreb, 2018.

[3] Hrvatski zavod za javno zdravstvo, Kardiovaskularne bolesti, dostupno na: https://www.hzjz.hr/aktualnosti/kardiovaskularne-bolesti/ [15.06.2024.]

[4] Hrvatski zavod za javno zdravstvo, Što su kardiovaskularne bolesti?, dostupno na: https://www.hzjz.hr/sluzba-epidemiologija-prevencija-nezaraznih-bolesti/sto-su-kardiovaskularne-bolesti/ [18.06.2024.]

[5] Hrvatski zavod za javno zdravstvo, Publikacija: Kardiovaskularne bolesti u Republici Hrvatskoj, dostupno na: https://www.hzjz.hr/periodicne-publikacije/publikacija-kardiovaskularne-bolesti-u-republici-hrvatskoj/ [16.09.2024.]

[6] D. Vrdoljak, Nove smjernice kardiovaskularne prevencije u kliničkoj praksi (europsko kardiološko društvo, verzija 2012.) - kratki pregled za liječnike obiteljske medicine, Medix, sv. 101/102, str. 200.-207., 2012.

[7] N. Mandić, Rizik za razvoj bolesti srca, Pliva Zdravlje, dostupno na: https://www.plivazdravlje.hr/aktualno/clanak/33570/Rizik-za-razvoj-bolesti-srca.html [18.06.2024.]

[8] V. Sharma, S. Yadav and M. Gupta, Heart Disease Prediction using Machine Learning Techniques, 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, str.177.-181., 2020.

[9] A. Singh and R. Kumar, Heart Disease Prediction Using Machine Learning Algorithms, 2020 International Conference on Electrical and Electronics Engineering, str. 452.-457., 2020.

[10] H. Atha, *Understanding Functional and Non.Functional Requirements in App Development*, moveoapps.com, 2021., dostupno na: https://www.moveoapps.com/blog/functional-and-non-functional-requirements-in-app-development/ [22.06.2024.]

[11] AnythinPython, Exploring Django's Model-View-Template(MVT) Architecture, dostupno na: https://www.anythingpython.com/2024/02/06/django_architecture/ [26.08.2024.]

[12] Korištena baza podataka, dostupno na: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/data [03.08.2024.]

[13] Y. Lu, T. Ye and J. Zheng, Decision Tree Algorithm in Machine Learning, 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications, Dalian, str. 1014.-1017., 2022.

[14] N. W. S. Wardhani, M. Y. Rochayani, A. Iriany, A. D. Sulistyono and P. Lestantyo, Cross-validation Metrics for Evaluating Classification Performance on Imbalanced Data, International Conference on Computer, Control, Informatics and its Applications, str. 14.-18., 2019.

[15] S. Ghosh, A. Dasgupta and A. Swetapadma, A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification, International Conference on Intelligent Sustainable Systems, str. 24.-28., 2019.

[16] K. Taunk, S. De, S. Verma, A. Swetapadma, A Brief Review of Nearest Neighbor Algorithm for Learning and Classification, International Conference on Intelligent Computing and Control Systems, str. 1255.-1260., 2019.
[17] V. Y. Kulkarni and P. K. Sinha, Pruning of Random Forest classifiers: A survey and future directions, International Conference on Data Science & Engineering, str. 64.-68., 2012.
[18] Datacamp, What is Python? Everything You Need to Know to Get Started, dostupno na: https://www.datacamp.com/blog/all-about-python-the-most-versatile-programming-language [03.08.2024.]
[19] IBM, What is Django?, dostupno na: https://www.ibm.com/topics/django [04.08.2024.]
[20] GeeksForGeeks, HTML Introduction, dostupno na: https://www.geeksforgeeks.org/html-introduction/ [26.08.2024.]
[21] TechTarget, Bootstrap, dostupno na: https://www.techtarget.com/whatis/definition/bootstrap [26.08.2024.]
[22] SQLite, What Is SQLite?, dostupno na: https://www.sqlite.org [27.08.2024.]
[23] A. Chandiramaniand P. Singh, Management of Django Web Development in Python, Journal of Management and Service Science, sv. 1, br. 2, str.1.-17., 2021.

SAŽETAK

WEB APLIKACIJA ZA PROCJENU RIZIKA OBOLIJEVANJA I PRAĆENJA TIJEKA KARDIOVASKULARNIH BOLESTI ZASNOVANA NA POSTUPCIMA NADZIRANOG STROJNOG UČENJA

Glavni cilj ovog diplomskog rada bio je razvoj web aplikacije koja procjenjuje rizik od obolijevanja od kardiovaskularnih bolesti koristeći metode nadziranog strojnog učenja. Aplikacija je implementirana uz pomoću programskih jezika i tehnologija kao što su Python, Django i HTML. U središtu aplikacije nalazi se model temeljen na algoritmu slučajne šume, koji je nakon usporedbe sa stablom odlučivanja, metodom strojeva potpornih vektora i algoritmom k najbližih susjeda pokazao najbolje rezultate. Osim što omogućuje pacijentima procjenu svog rizika obolijevanja od kardiovaskularnih bolesti, aplikacija također pruža liječnicima mogućnost stvaranja preporuka temeljenih na zdravstvenom stanju pacijenata. Praćenje tijeka bolesti i praćenje zdravstvenih podataka omogućeno je zahvaljujući sustavu za pohranu. Uz to što pomaže u ranom prepoznavanju rizika, aplikacija omogućuje kontinuirani nadzor i prilagodbu liječenja prema promjenama u zdravstvenom stanju korisnika. Provjera ispravnosti aplikacije obavljena je kroz tri testna slučaja i pokazuje da aplikacija radi ispravno.

Ključne riječi: algoritam slučajne šume, kardiovaskularne bolesti, nadzirano strojno učenje, procjena rizika obolijevanja, web aplikacija.

ABSTRACT

A WEB APPLICATION FOR CARDIOVASCULAR DISEASE RISK ASSESSMENT AND DISEASE COURSE MONITORING BASED ON SUPERVISED MACHINE LEARNING METHODS

The main goal of this thesis was to develop a web application that assesses the risk of cardiovascular disease using supervised machine learning methods. The application was implemented using programming languages and technologies Python, Django, and HTML. At the core of the application is a machine learning model based on the random forest algorithm, which, after comparison with decision trees, support vector machines, and the k-nearest neighbors algorithm, had the best results. In addition to allowing patients to assess their risk of developing cardiovascular disease, the application also provides doctors with the ability to create recommendations based on the health status of patients. Disease course monitoring and health data tracking are enabled by a storage system. Besides helping in the early detection of risk, the application allows for continuous monitoring and adjustment of treatment according to changes in the user's health status. The application's validation was carried out through three test cases and shows that the application functions correctly.

Keywords: Random Forest algorithm, cardiovascular diseases, supervised machine learning, risk assessment, web application.

PRILOZI

Prilog 1. Diplomski rad u datoteci docx

Prilog 2. Diplomski rad u datoteci pdf

Prilog 3. Programski projekt web aplikacije

Prilog 4. Model strojnog učenja