Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nal Lukšič

Sistemska arhitektura za podporo zdravniku pri odločanju

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: dr. Andrej Brodnik
Somentor: dr. Janez Žibert

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Sistemi za podporo pri odločanju zahtevajo nekoliko specifično arhitekturo, ki vključuje poseben gradnik. Le-ta ima običajno dodatno zalogo podatkov, ki mu omogočajo, da ob upoštevanju ostalih zdravstvenih podatkov pripravi zdravniku statistične podatke o stanju pacienta.

V nalogi načrtajte takšno arhitekturo, ki bo črpala podatke iz FHIR sistema. Podatke naj gradnik lokalno obdela in hrani. Ob zdravnikovem pregledu stanja konkretnega pacienta naj sistem omogoča ovrednotenje le-teh na podlagi shranjenih podatkov. Začrtano arhitekturo tudi implementirajte in implementacijo ovrednotite.

Na začetku bi se rad zahvalil svojim staršem, ki so me skozi celotno študijsko pot podpirali. Prav tako gre zahvala Fakulteti za računalništvo in informatiko in vsem njenim profesorjem, ki so mi skozi tri leta podali veliko znanja. Za izdelavo diplomskega dela pa se gre zahvalit mojemu mentorju Andreju Brodniku in somentorju Janezu Žibertu, ki sta mi pomagala pri diplomski nalogi. Zahvala gre tudi Nenadu Žinkoviču in Gašperju Andrejcu iz podjetja Parsek, ki sta mi vedno bila na voljo za kakršnokoli pomoč. Še kot zadnjim pa bi se zahvalil svojim kolegom študentom, ki so mi celoten študij olajšali z druženji.

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
AI	artificial intelligence	umetna inteligenca
GUI	general user interface	uporabniški vmesnik
REST	representational state transfer	metoda prenosa podatkov
API	application programming interface	vmesnik uporabniškega programa
IDE	integrated development environment	integrirano razvojno okolje
FHIR	fast healthcare interoperability resources	protokol za hitro uporabo virov v e-zdravstvu
AUC	area under curve	območje pod krivuljo
ROC	receiver operating characteristic	delovna karakteristika sprejemnika
HTML	hypertext markup language	označevalni jezik
JS	javascript	objektni skriptni jezik
PHP	hypertext preprocessor	programski jezik
CSS	cascadian style sheets	slogovni jezik

Povzetek

Naslov: Sistemska arhitektura za podporo zdravniku pri odločanju

Avtor: Nal Lukšič

Diplomsko delo opisuje arhitekturo sistema, ki pomaga zdravstvenim uslužbencem bolje razumeti različne bolezni in vpliv meritev nanje. Sistem deluje na podlagi znanj, ki jih črpa iz velikih količin kliničnih podatkov. V delu se uporabljajo podatki, pridobljeni iz FHIR strežnika in so umetno generirani. Glavno jedro arhitekture je spletna aplikacija, do katere lahko dostopa uporabnik. Njen osrednji namen je zagotovitev primernega okolja, kjer se lahko izvaja statistična analiza podatkov. Analiza se izvede z uporabo statističnih testov za testiranje domnev, kjer se ugotavlja statistična različnost posameznih meritev glede na določeno bolezen. Meritve so razvrščene glede na izračunane p-vrednosti. Dodatno se pri skalarnih spremenljivkah izvede še ROC analiza, s katero ugotavljamo napovedano ustreznost skalarnih meritev. Sistem je osnovan za potrebe zdravstvenih uslužbencev, ki skrbijo za diagnosticiranje bolnikov in si želijo boljši vpogled v delovanje bolezni. Arhitektura je sestavljena iz dveh ključnih procesov, in sicer vnos novih podatkov ter pridobitve rezultatov analize.

Ključne besede: analiza, analitika, statistika, medicina, veliki podatki, zdravstvo, diagnoza, bolezni, kritični atributi.

Abstract

Title: Diploma thesis

Author: Nal Lukšič

The thesis describes the architecture of the system, which helps health professionals better understand the various conditions and an impact of observations on them. The system works on the basis of the knowledge from large amounts of clinical data. In this thesis, the data is being derived from FHIR servers and it is program generated. In the core of the architecture is a web application that can be accessed by the user. Its purpose is to provide an appropriate environment where statistical analysis of data can be performed. The analysis is performed by an algorithm that calculates the importance factors for individual measurements of patients and returns them according to their value. With the help of this system, the user has a better insight into various conditions. The system is designed for the needs of health care professionals who are responsible for diagnosing patients and appreciate a better overview of conditions. The architecture consists of two key processes, namely the input of new data and the computing of the analysis results.

Keywords: analysis, analytic, statistics, big data, medicine, diagnose, conditions, critical attributes.

Poglavje 1

Uvod

Raziskave iz leta 2013 kažejo, da v ZDA vsako leto zaradi zdravniških napak umre od 210.000 do 400.000 bolnikov [8]. Najpomembnejši razlog za tovrstne napake je narava dela. Stres, ki ga zdravniki občutijo dnevno, pripelje do več napak [7]. Tovrstne napake je mogoče zmanjšati, če imajo zdravstveni delavci ustrezno računalniško podporo. Raziskave so pokazale, da lahko informacijska tehnologija v medicini izboljša kakovost storitev za 20 odstotkov [3].

Kot je opisano v poglavju 2 obstaja mnogo sistemov, za pomoč zdravnikom pri diagnosticiranju. Ti so sicer zelo učinkoviti, vendar so mnogokrat prezapleteni in zahtevajo že obstoječo bazo podatkov.

Cilj diplomskega dela je predstavitev arhitekture sistema, ki uporabniku nudi medicinsko analitiko, na osnovi že zbranih podatkov. Sistem uporabniku nudi znanja o povezavah med boleznimi in meritvami.

Sistem mora nuditi storitev, ki je preprosta za uporabo, pregledna, odzivna in povečljiva v primeru, da se število podatkov znatno poveča.

1.1 Struktura naloge

V uvodnem poglavju je opisana problematika, ki jo obravnava diplomsko delo. Opisana sta trenutni položaj v zdravstvu in težava, ki jo s to diplomo

rešujemo. V drugem poglavju se osredotočamo na podjetja, ki že nudijo podobne izdelke in rešitve. V tretjem poglavju so predstavljene in ovrednotene tehnologije in metodologije, uporabljene pri razvoju sistema. Celotna rešitev in potek dela sta opisana v četrtem delu. V tem delu so po podpoglavjih razdeljeni ključni osrednji deli sistema. Temu v petem delu sledijo rezultati in ugotovitve. Podan je tudi sklep, kaj smo v diplomskem delu novega izvedeli in kakšne so možnosti nadaljnjega razvoja.

Poglavje 2

Obstoječe rešitve

Številne zdravstvene institucije že uporabljajo sisteme, ki temeljijo na podobnih arhitekturah, vendar so te največkrat skrite očem javnosti. Sistemi se razvijajo projektno po naročilu. Obstaja mnogo podjetij, ki se ukvarjajo z implementacijo sistemov s tovrstno arhitekturo, vendar v večini le za naročnika in so posledično javnosti skriti. V tem poglavju je predstavljenih nekaj podjetij, katerih izdelki so zdravstveni sistemi podobni našemu in imajo javno objavljeno arhitekturo.

2.1 HealthCatalyst

HealthCatalyst je relativno majhno podjetje z okoli 400 zaposlenimi in se nahaja v ZDA. Podjetje so ustanovili izkušeni zdravniki in strokovnjaki na področju analize podatkov. Podjetje je kot eno prvih tovrstnih na trgu ponujalo rešitve zdravstvenim ustanovam, ki so bile razvite na osnovi obdelave velikih količin podatkov. Danes ponujajo ogromen nabor rešitev v zdravstvu, od plošč za nadziranje bolnikov, do sistemov za pomoč diagnosticiranja [11]. Podjetje ponuja 3 glavne produkte, in sicer:

1. Healthcare Analytics Platform je platforma, ki ponuja dostop do več kot 120 različnih virov podatkov. S tem zelo olajša delo zdravstvenim ustanovam, ker ni potrebe po integraciji drugih podatkov. Brez tovr-

stne platforme so algoritmi za delo s podatki neuporabni, zato je to eden ključnih delov ponujene rešitve.

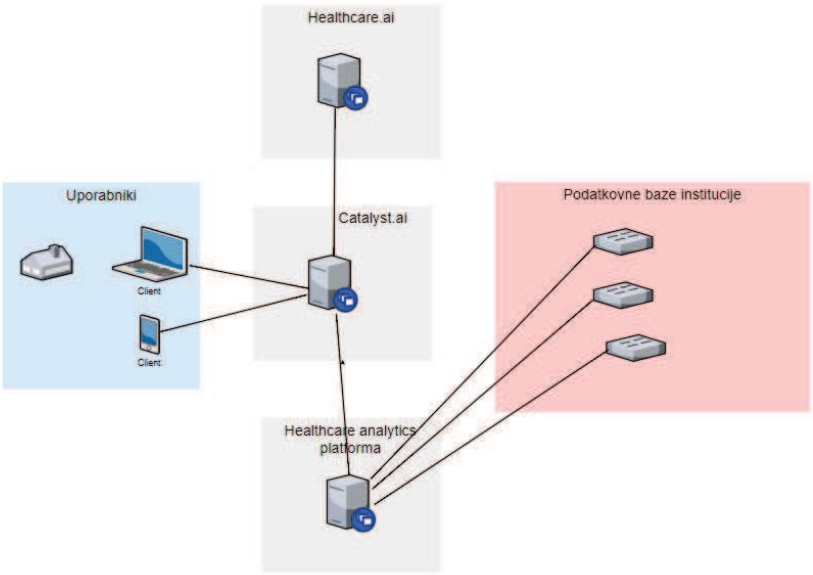
2. Healthcare.ai je javno dostopna programska oprema, ki vključuje različne algoritme strojnega učenja. Izvorna koda je javno dostopna na njihovem Github repozitoriju. Cilj teh algoritmov je zagotoviti dostopnost do tovrstnih tehnologij vsem zdravstvenim ustanovam ne glede na finančna sredstva.

3. Catalyst.ai je celoten sistem, ki povezuje različne module. Catalyst.ai združuje Healthcare Analytics platformo in healthcare.ai. Arhitektura sistema je prikazana na sliki 2.1 Od naše arhitekture se razlikuje po razčlenjenosti, kajti sistem je sestavljen iz treh delov, ki so med seboj povezani. Drugačni so tudi podatki, ki jih uporablja. V našem primeru se osredotočamo na podatke pridobljene iz FHIR strežnikov, Catalyst.ai pa uporablja vmesnike za dostop do različnih podatkov shranjenih v podatkovnih bazah naročnika.

2.2 Infermedica

Infermedica nudi rešitev pomoči pri diagnosticiranju bolnika. Njihov sistem deluje na principu vprašanj, ki sprašujejo o stanju bolnika. S pomočjo odgovorov lahko umetna inteligenca ustvari mapo z osnovnimi podatki. Celoten sistem temelji na arhitekturi, ki je prikazana na sliki 2.2. Glavna razlika med našo arhitekturo in arhitekturo Infermedice je, da za ustvarjanje znanja uporabljajo strokovnjake. Ti s pomočjo že znanih podatkov učijo sistem, ki nato s pomočjo umetne inteligence vrne napoved.

Strokovnjaki črpajo znanja iz različnih virov. Razlika med našim sistemom in Infermedico je tudi dostop do sistema. Infermedica kot podjetje prodaja dostope do svojega sistema, do katerega je mogoče dostopati preko HTTP zahtevkov. Spletni vmesnik pa ne vključuje algoritmov obdelave zahtevkov.



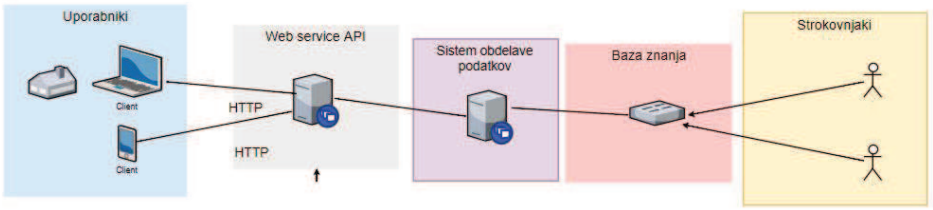
Slika 2.1: Arhitektura sistema Catalyst.ai.

2.3 CloudMedX health

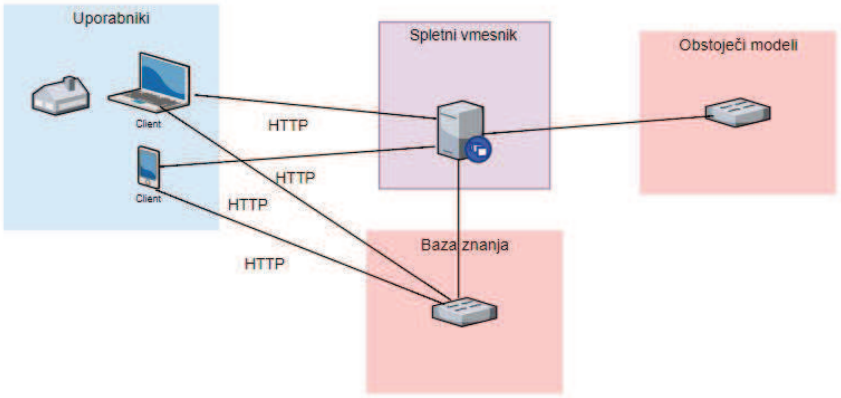
CloudMedX health je eno večjih podjetij, ki se ukvarja z razvojem kliničnih platform, ki vključujejo tehnologije umetne inteligence. Njihov cilj je izboljšati in skrajšati bolnikov čas v bolnici, s pomočjo strojnega učenja [2].

Arhitektura sistema CloudMedX temelji na dveh različnih virih podatkov. Ob začetku uporabe sistema mora uporabnik v sistem preko HTTP zahtev vnesti veliko količino podatkov. Te podatke sistem združi s podatkovnimi modeli, ki predstavljajo pomemben del sistema. Uporabnik lahko nato do funkcionalnosti sistema dostopa preko spletnega vmesnika. Arhitektura je predstavljena na sliki 2.3

Njihov sistem deluje po naslednjem principu: oseba v sistem vpiše opis bolnika, kjer so podatki lahko strukturirani ali pa ne. Nato umetna inteligenca s pomočjo že zgrajenih modelov iz besedila izlušči najpomembnejše podatke. Iz teh je sistem zmožen napovedati, katere bolezni ima bolnik. Re-



Slika 2.2: Arhitektura sistema Infermedice.

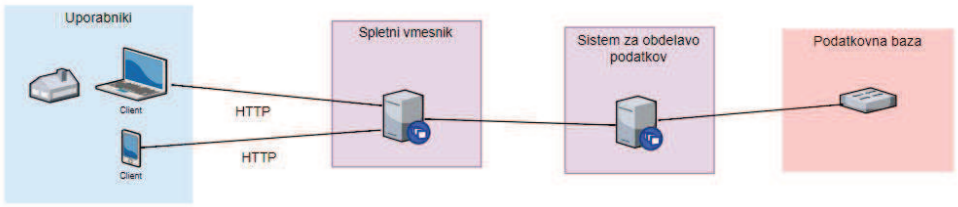


Slika 2.3: Arhitektura sistema CloudmedX.

zultati so prikazani v pregledni obliki, ki pomaga zdravnikom pri diagnozi bolezni. Razlika med sistemom CloudMedX in našim je, katere podatke uporablja in kako jih obdeluje.

2.4 Lumiata

Lumiata je podjetje specializirano za razvoj sistemov, ki zmanjšujejo stroške in tveganja. Glavni izdelek je platforma, ki temelji na analitični obdelavi medicinskih podatkov [1].



Slika 2.4: Arhitektura sistema Lumiata.

Lumiata za razliko od prejšnjih ponudnikov uporablja podatke v FHIR JSON formatu. Sistem je zgrajen iz spletnega vmesnika, kjer uporabnik vnaša nestrukturirane podatke. Ti so nato poslani v sistem, ki jih strukturira in shrani v podatkovno bazo. Sistem tako lahko črpa podatke za nadaljnjo obdelavo. Od naše rešitve se Lumiatin izdelek razlikuje po tipu podatkovne baze, saj mi uporabljamo FHIR strežnik, Lumiatini podatki pa so shranjeni v različnih podatkovnih bazah. Glavni del sistema, ki se uporablja za obdelavo podatkov, se pri Lumiati nahaja ločeno od spletnega vmesnika, medtem ko je v našem primeru združeno.

Poglavje 3

Uporabljene tehnologije in metodologije

V tem poglavju so opisane metode in orodja, ki so ključna za razvoj sistema. Glede na to, da gre v diplomskem delu za obdelavo velikih količin podatkov, so temu primerna tudi orodja.

3.1 Canvas.js

Canvas.js je javno dostopno orodje, ki omogoča izris grafov na spletnih straneh. Orodje omogoča izris podatkov v 2d koordinatnem sistemu. V našem primeru smo uporabili tip izrisa *stepLine*, ki omogoča povezovanje točk v krivuljo. Orodje smo uporabili za izris ROC krivulje.

3.2 Synthia

Synthia je javno dostopni projekt, katerega namen je generiranje umetnih medicinskih podatkov. Projekt vsebuje različne procese, ki generirajo JSON objekte z vključevanjem naključnih podatkov, ki so vključeni v projekt. Synthia za delovanje potrebuje Javo 1.8 ali novejšo različico [10].

V nalogi smo Synthio uporabili za generiranje medicinskih podatkov, ka-

tere smo prenesli v FHIR strežnik za kasnejšo obdelavo. Generirani podatki so v JSON formatu in strukturirani po pravilih, katere določa protokol FHIR. Primer takšnega snopa je prikazanih na sliki 3.1.

Slika 3.1: Struktura FHIR JSON snopa.

3.3 Numpy knjižnica

Numpy združuje orodja za znanstveno računanje. Numpy smo v diplomskem delu uporabili za shranjevanje podatkov v tabelo (*numpy- .array()*) in njihovo obdelovanje. Numpy tabela zelo olajša obdelovanje velikega nabora podat-

kov. Prednost knjižnice je shranjevanje velikih tabel, saj zavzame štirikrat manj prostora kot navadna Python tabela.

Knjižnica vsebuje različne algoritme, za obdelovanje podatkov. V diplomskem delu smo uporabili osnovne funkcionalnosti knjižnice, kot na primer iskanje povprečja (*numpy.mean()*), računanje standardnega odklona (*numpy.std()*), združitev dveh tabel (*numpy.concatenate()*) in druge.

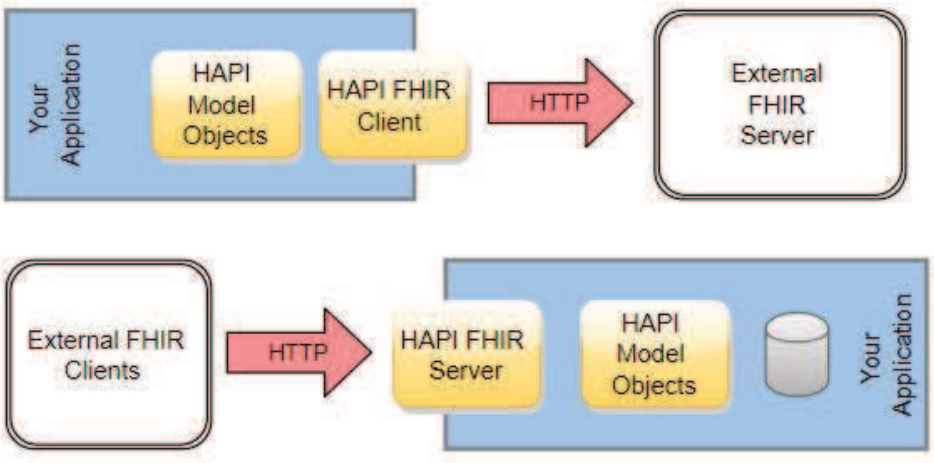
3.4 FHIR strežnik

FHIR je zbirka standardov, ki med drugim narekujejo, kako hraniti medicinske podatke in je najnovejši standard HL7 z mnogimi podobnostmi prejšnjih različic [5]. Deluje na osnovi REST arhitekture, kjer se podatki pošiljajo v XML ali JSON formatu. Njegova prednost je, da lahko do njega dostopamo preko RESTFUL API in je zato univerzalen ne glede na operacijski sistem odjemalca.

FHIR strežnik je implementacija celotne storitve, kamor lahko pošiljamo in odvzemamo podatke. V tem diplomskem delu je bil uporabljen FHIR HAPI strežnik. HAPI vključuje tudi spletni vmesnik, preko katerega lahko izvršimo RESTFUL klice. Razvila se je tudi skupnost, kjer lahko javno dostopne FHIR HAPI strežnike uporabljajo vsi za testiranje izdelkov. Primera uporabe, ki sta uporabljena v tem diplomskem delu, sta predstavljena na sliki 3.2.

HAPI FHIR je javno dostopen projekt, ki vključuje različne implementacije FHIR strežnikov. V diplomskem delu smo uporabili podprojekt *jpaserver-example*, ki implementira lokalen FHIR strežnik, ki je preprost za uporabo. Strežnik vključuje osnovne operacije, kot je na primer pošiljanje podatkov v strežnik in dostopanje do podatkov preko preprostih klicev. Strežnik ne omogoča zahtevnejših iskalnih nizov, kjer bi podatke vračal na osnovi matematičnih operacij.

Za vzpostavitev strežnika potrebujemo program Maven 3.5.4, ki skrbi za vzpostavitev okolja in vzpostavljeno Javo različice 1.6 ali novejšo. Strežnik se nato vzpostavi s pomočjo Jetty, ki je vključen v projekt.



Slika 3.2: Uporaba FHIR storitev [5].

Lokalni FHIR strežnik smo uporabili za lažje testiranje delovanja sistema. Podatki v javnih strežnikih so pomanjkljivi in ne vsebujejo atributov, ki so potrebni za delovanje našega sistema. Primeri takšnih podatkov so vrednosti meritev, sklici bolezni na bolnika in podobno.

3.5 Postopki statistične analize podatkov

V diplomskem delu se uporablja veliko statističnih testov, opisanih v nadaljevanju. Testi so lahko parametrični ali neparametrični odvisno od porazdelitve vzorca. Statistični testi se uporabljajo za testiranje domnev na neki populaciji. V primeru, ko so vzorci oziroma populacije normalno porazdeljeni, se uporabljajo parametrični testi, v primeru nenormalno porazdeljenih vzorcev pa neparametrični testi. Kot mero statistične različnosti med meritvami pri bolnih in zdravih preiskovancih uporabljamo p-vrednost. Če je p-vrednost manjša od 0.05, obstaja statistična razlika med meritvami glede na analizirano bolezen [4].

3.5.1 Test normalnosti

Testi normalnosti preverjajo, ali meritve nekega vzorca pripadajo populaciji normalno porazdeljenih meritev. Obstaja več različnih testov normalnosti. V našem primeru smo uporabili test Kolmogorova in Smirnova (KS test), ki testira domnevo ali je funkcija porazdelitve verjetnosti danih meritev normalna [9]. Hipotezi, ki ju testiramo, sta:

H_0: `Podatki so normalno porazdeljeni.`
H_1: `Podatki niso normalno porazdeljeni.`

V primeru, da je p-vrednost manjša od 0.05, lahko ničelno hipotezo zavrnemo, v nasprotnem primeru pa je ne moremo zavrniti.

V diplomski nalogi smo uporabili funkcijo kstest(a, b) iz knjižnice spicy.stats. Vhod funkcije sta porazdelitev vzorca testne populacije (a) in porazdelitev, s katero jo primerjamo (b).

3.5.2 T test dveh vzorcev

T test dveh vzorcev uporabljamo za testiranje povprečja dveh populacij. S t testom preverjamo naslednji hipotezi:

H_0: $\mu_1 = \mu_2$
H_1: $\mu_1 \neq \mu_2$,

kjer μ_1 predstavlja povprečje prve populacije, μ_2 pa povprečje druge populacije. Testna statistika t testa ima naslednjo enačbo [9]

$$t = \frac{\bar{m_1} - \bar{m_2}}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \ , \tag{3.1}$$

kjer je $\bar{m_1}$ povprečje prvega vzorca, $\bar{m_2}$ povprečje drugega vzorca, N_1 velikost prvega vzorca, N_2 velikost drugega vzorca, s_1 standardni odklon vzorca prve populacije in s_2 standardni odklon vzorca druge populacije.

Če je p-vrednost t statistike manjša od 0.05, potem ničelne hipoteze ne sprejmemo in velja, da sta populacijski povprečji statistično različni.

Predpostavke t testa so, da so meritve normalno porazdeljene in v našem primeru, da imamo različne variance v populacijah.

V nalogi je kot t test uporabljena funkcija `ttest_ind(a, b)` iz knjižnice spicy.stats, ki kot vhodne spremenljivke sprejme vzorca zdravih (a) in bolnih (b) in vrne p-vrednost.

3.5.3 Mann Whitney U test

Mann Whitney U test uporabljamo v primeru testiranja različnosti median dveh vzorcev, ko so meritve nenormalno porazdeljene. Hipotezi testa sta:

H_0: $M_1 = M_2$
H_1: $M_1 \neq M_2$,

kjer je M_1 mediana prve populacije in M_2 mediana druge populacije. Test deluje tako, da šteje vrstni red meritev prvega vzorca in drugega vzorca. Če je vsota vrstnega reda meritev prvega vzorca različna od vsote vrstnega reda meritev drugega vzorca, potem je bolj verjetna hipoteza H_1, sicer pa hipoteza H_0. V primeru ko je p-vrednost manjša od 0.05, lahko rečemo, da sta mediani obeh populacij statistično značilno različni [9].

Ta test smo uporabljali v primerih, ko smo s KS testom dobili p-vrednosti manjše od 0.05. V nalogi je uporabljena funkcija `mannwhitneyu(a, b)` iz knjižnice spicy.stats, ki tako kot pri t testu, kot vhodni spremenljivki vzame vrednost zdravih (a) in bolnih (b) ter vrne p-vrednost.

3.5.4 χ^2 test

χ^2 test uporabljamo v primeru kategorijskih spremenljivk. To so spremenljivke, ki lahko zavzamejo samo določeno število vrednosti. Primeri takšnih spremenljivk so spol in bolezen tipa ena, dva ali tri. V takšnih primerih za ugotavljanje povezave med kategorijskimi spremenljivkami uporabljamo χ^2 test. Z njim testiramo, ali obstaja povezava med kategorijskimi spremenljivkami (ali sta kategorijski spremenljivki odvisni). To izvedemo tako, da izračunamo dejanske frekvence kategorijskih vrednosti in jih primerjamo s

pričakovanimi frekvencami kategorijskih vrednosti v primeru predpostavke, da sta kategorijski spremenljivki neodvisni. Večje kot je odstopanje, večjo vrednost doseže χ^2 test. To pomeni, da lahko ničelno hipotezo o neodvisnosti kategorijski spremenljivki zavrnemo, če je vrednost dovolj visoka. Mejna vrednost χ^2 je dosežena ob p-vrednost = 0.05. Če je p-vrednost manjša pomeni, da sta kategorijski spremenljivki med seboj različni [9].

V nalogi je uporabljena funkcija `chi2_contingency(a)` iz knjižnice spicy.stats, ki kot vhod vzame matriko frekvenc vrednosti (a) in vrne p-vrednost kategorije.

Fisherjev natančni test

V primeru, ko je v kontingenčni tabeli, ki jo uporabljamo pri χ^2 testu frekvenca v katerikoli celici tabele manjša od pet, lahko namesto χ^2 testa uporabimo Fisherjev natančni test. Ta test deluje podobno kot χ^2 test, le da na drugačen način (bolj natančno) izračuna p-vrednost [9].

V nalogi uporabimo metodo `fisher_exact(a)` iz knjižnice spicy.stats, ki kot vhod vzame matriko frekvenc vrednosti (a) in vrne p-vrednost.

3.5.5 ROC krivulja

Analizo ROC krivulje uporabimo v primeru, ko bi radi pokazali, ali je neka skalarna spremenljivka primerna za napovedovanje nekega dogodka [6]. V našem primeru je to bolezen. Z ROC analizo poskušamo določiti optimalni prag, pri katerem lahko dobro napovemo bolezen in hkrati ne naredimo prevelike napake pri napovedovanju zdravih. S postavitvijo praga pri skalarni spremenljivki določimo, kateri preiskovanci so zdravi in kateri so bolni. Pri tem lahko dobimo naslednje kombinacije izidov:

- Pravilno določimo preiskovanca, da je bolan (TP).
- Napačno določimo preiskovanca, da je bolan, čeprav je zdrav (FP).
- Pravilno določimo preiskovanca, da je zdrav (TN).

- Napačno določimo preiskovanca, da je zdrav, čeprav je bolan (FN).

Mere za ustreznost praga v medicini sta specifičnost in senzitivnost:

$$\texttt{Senzitivnost} = TP/(TP + FN) \tag{3.2}$$

$$\texttt{Specifičnost} = TN/(TN + FP) \tag{3.3}$$

Z izbiro različnih pragov v skalarni spremenljivki lahko izračunamo različne vrednosti specifičnosti in senzitivnosti. Če to naredimo za vse možne pragove pri dani skalarni spremenljivki, lahko narišemo ROC krivuljo, kjer abscisa predstavlja 1-specifičnost, ordinata pa senzitivnost. V takem grafu lahko izračunamo ploščino pod ROC krivuljo, ki jo označimo z AUC (area under curve). Večji, kot je AUC, večja je napovedna vrednost spremenljivke.

V nalogi je uporabljena funkcija `.roc_curve()` iz knjižnice metrics, ki kot vhod vzame vektor vrednosti in vektor istoležnih elementov, ki vsebuje 0, če je opazovanec zdrav in 1, če je bolan in vrne p-vrednost kategorije.

V našem primeru smo izračunali tudi optimalni prag z uporabo Youdenove metode, ki jo izračunamo po naslednji formuli.

$$J = \texttt{prag pri max}(\texttt{senzitivnost} + \texttt{specifičnost}) \tag{3.4}$$

Prag, pri katerem je vsota senzitivnosti in specifičnosti največji, bodo vrnili najbolj točne rezultate.

V našem primeru smo za vsako skalarno spremenljivko izračunali AUC s funkcijo auc(a, b) iz knjižnice metrics, ki kot vhod vzame specifičnost (a) in senzitivnost (b) in vrne ploščino območja pod krivuljo.

Poglavje 4

Razvoj sistema

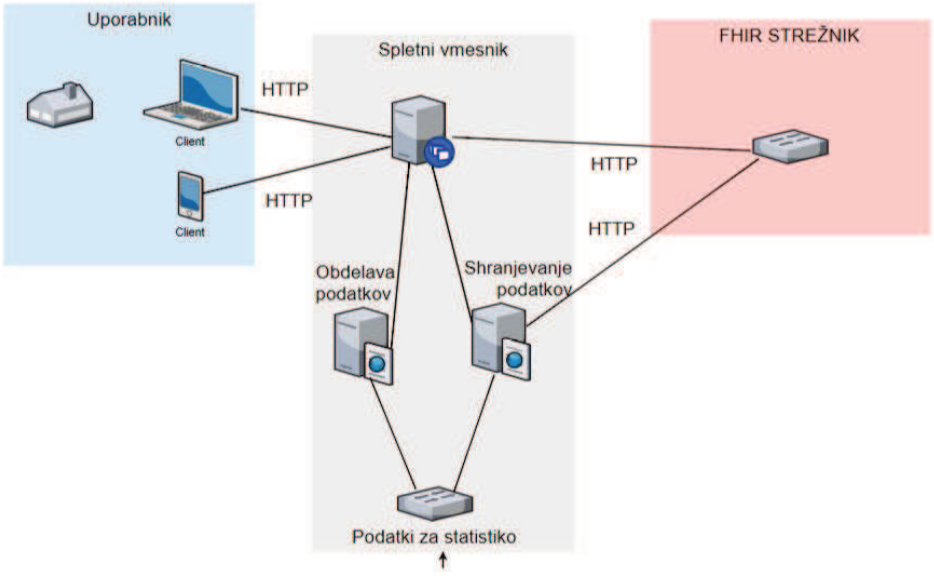
Za razvoj sistema smo uporabili računalnik Lenovo Thinkpad T540p, z 10GB RAMa, i5-4200M Intel procesorjem in Windows 10 operacijskim sistemom. Za razvoj Python skript smo uporabili Python 3.6.5. Za izgradnjo spletnega vmesnika pa smo uporabili HTML 5, PHP 7.2.8 in Javascript ECMAScript 6. Za vzpostavitev strežnika smo uporabili Javo 1.8 in Maven 3.5.6.

4.1 Arhitektura

Osrednji poudarek diplomskega dela je arhitektura sistema. Ta sestoji iz treh glavnih komponent, kot kaže slika 4.1. Prva komponenta je spletni vmesnik, preko katerega lahko uporabnik uporablja funkcionalnosti sistema.

Vmesnik teče na spletnem naslovu, ki je dostopen vsem uporabnikom preko spleta. Vsi glavni procesi sistema tečejo preko spletnega vmesnika. Dve najpomembnejši funkcionalnosti sistema sta shranjevanje in obdelava podatkov, ki uporabljata podatke, pripravljene za statistiko, kot je prikazano na sliki 4.1.

Spletni vmesnik komunicira tudi z drugo komponento imenovano FHIR strežnik preko HTTP zahtevkov. Obstaja mnogo javno dostopnih FHIR strežnikov, do katerih lahko dostopamo. V našem primeru smo uporabili lokalno vzpostavljen FHIR HAPI strežnik opisan v poglavju 3.4. Njegova prednost



Slika 4.1: Arhitektura sistema diplomskega dela.

pred javno dostopnimi je, da zahteva, pravilnost podatkov vnešenih v strežnik. Prav tako so javno dostopni strežniki po določenem času izpraznjeni. To je velika težava, saj je glavna funkcionalnost FHIR strežnika, dostop do velike količine medicinskih podatkov. Če teh podatkov ni, sistem ne more delovati.

Zadnja komponenta je uporabnik, katerega naloga je vnašanje novih podatkov v FHIR strežnik preko spletnega vmesnika in uporaba funkcionalnosti spletnega vmesnika. Uporabnik z njim komunicira preko HTTP zahtevkov.

4.2 Prenos podatkov

Največji izziv diplomske naloge predstavlja prenos podatkov v in iz FHIR strežnika, ki je opisan v poglavju 3.4. Podatki se namreč prenašajo v snopih po 10 do 20 objektov. Čas vrnitve snopa je seveda odvisna od hitrosti strežnika, ki v vsakem primeru zahteva veliko časa.

4.2.1 Prenos podatkov na strežnik

Težavo počasnega pošiljanja podatkov v sistem smo rešili tako, da na začetku podatke zložimo v transakcijske snope, katere pošljemo v sistem. Pred pošiljanjem je podatke treba prestrukturirati v pravilni format. Podatki morajo ustrezati FHIR zahtevam opisanim v poglavju 3.2, da jih lahko nato pošljemo v sistem. Na ta način zgradimo osnovni nabor podatkov.

Ko v nadaljevanju želi uporabnik vnesti nove podatke na strežnik, lahko to stori preko spletnega vmesnika. Ta omogoča takojšno prestrukturiranje podatkov, ki so nato poslani na strežnik preko POST zahtevkov.

4.2.2 Dostop do podatkov

Težavo počasnega dostopa do podatkov sistem rešuje tako, da vse podatke iz FHIR strežnika prenesemo le enkrat, nato pa jih v primerni obliki shranimo v bazo podatkov pripravljenih za statistično obdelavo. Podatke prenesemo iz strežnika s pomočjo GET zahtevkov, ki vračajo podatke o bolnikih, boleznih in meritvah. Podatke med prenosom shranjujemo v Python slovar, katerega ključi so imena bolezni in meritev. Ta funkcionalnost je na sliki 4.1 prikazana kot komponenta `Shranjevanje podatkov`.

Po prenosu vseh podatkov, shranimo slovar v lokalne csv datoteke. Te omogočajo zelo hiter dostop do velike količine podatkov, ki so razvrščeni glede tip bolezni v različne datoteke. Znotraj posazamezne datoteke so po vrsticah razvrščene vrednosti posameznih meritev. Podatki vseh meritev ne glede na tip bolezni so shranjeni tudi v datoteki `all.csv`. Podatki iz te datoteke uporabljamo kot referenčne vrednosti zdravih preiskovancev.

Podatki so v primeru poizvedb na voljo takoj, treba jih je le prenesti iz željene datoteke in po vrsticah vnesti v Python tabelo ali slovar.

Datoteke omogočajo tudi dodajanje novih vrednosti meritev. Sistem je zmožen dodati nove vrednosti zelo preprosto, na konec vrstice.

4.3 Obdelave

Pomemben del diplomske naloge je obdelava podatkov. Njen cilj je prikazati, kako lahko zgrajeno arhitekturo v zdravstvu uporabimo. Osredotoča se na pomoč uporabnikom k boljšemu razumevanju razvoja različnih bolezni. Ta funkcionalnost je na sliki 4.1 prikazana, kot komponenta `Obdelava podatkov` in je sestavljena iz dveh modulov.

4.3.1 Statistična obdelava

Prvi modul se osredotoča na izračun p-vrednost in na podlagi te pove, kateri atributi so bolj izraziti pri določeni bolezni. P-vrednost je v odstotkih izražena odvisnost bolezni od meritve. Do te vrednosti pridemo po postopku opisanem s psevdokodo Algorithm 1.

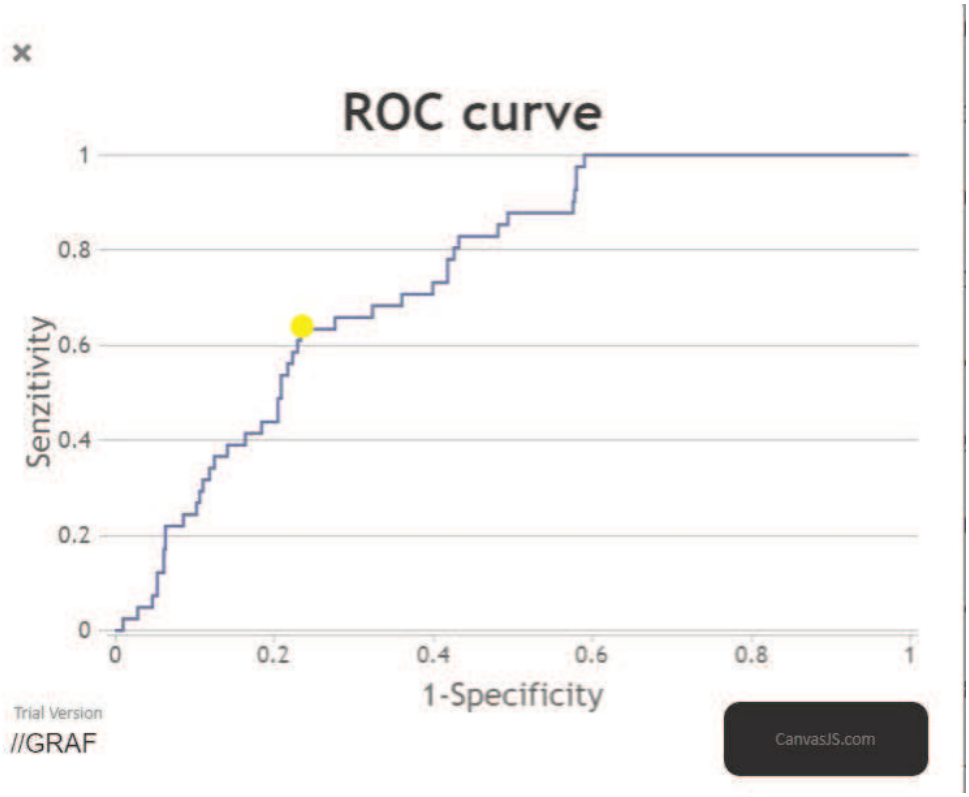
Rezultat vsakega testa je p-vrednost. Če so p-vrednosti manjše od 0.05, potem je ta meritev statistično različna med bolnimi in zdravimi za analizirano bolezen.

```
Function StatisticalSsignificance(vrednostiMeritev)
    x ← vrednostMeritev
    //Ali je meritev skalar ali kategorija?
    if tipMeritev(x) == Skalar then
        //Ali je porazdelitev vrednosti meritev normalna?
        if porazdelitev(x) == Normalna then
            pVrednost ← tTest(x)
        else
            pVrednost ← mannWhitneyUTest(x)
        end
    else
        //Ali ima vsaka vrednost kategorije vsaj pet meritev?
        if frekvence(x) > 5 then
            pVrednost ←χ²Test(x)
        else
            pVrednost ← fisherExactTest(x)
        end
    end
    return pVrednost
end
```

Algorithm 1: Psevdokoda poteka izračuna p-vrednosti.

4.3.2 Analiza ROC krivulje

Drugi modul se osredotoča na analizo ROC krivulje. Za vsak atribut izračunamo točke ROC krivulje po postopku opisanem v poglavju 3.7.6. S pomočjo teh lahko izračunamo še AUC, ki nam pove, kako pomemben je atribut za napovedovanje bolezni. S krivuljo lahko izračunamo tudi optimalni prag, katerega dobimo z Youdnovo metodo, opisano v poglavju 3.7.6. Na sliki 4.2 lahko vidimo, kako izgleda ROC krivulja. Ploščina celotnega območja, ki se nahaja pod modro črto, se imenuje AUC. Rumena pika na sliki označuje točko optimalnega praga, ki ga izračunamo po Youdenovi metodi.



Slika 4.2: Primer ROC krivulje.

4.4 Ovrednotenje

V tem poglavju se osredotočamo na kriterije ocenjevanja arhitekture sistema, ki so bili določeni v uvodu. Ocene posameznih kriterijev so utemeljene na osnovi pojasnil iz prejšnjih poglavij.

4.4.1 Enostavost uporabe

Zelo pomemben cilj diplomske naloge je enostavnost uporabe. Vsak uporabnik bi moral znati uporabljati sistem, ne glede na tehnična predznanja.

Iz slik, ki so vključene v dodatku, so prikazane možnosti uporabe sistema. Spletni vmesnik lahko upravljamo s pomočjo gumbov, s katerimi izbiramo, kaj želimo izvesti. Ostalo delo sistem opravi sam v ozadju, zato sistem od

uporabnika zahteva le vnos podatkov. S tem je zagotovljena enostavna uporaba sistema.

4.4.2 Preglednost

Preglednost definiramo kot stopnjo težavnosti razbiranja rezultatov statističnih analiz. Uporabnik ne sme imeti težav pri razbiranju informacij, ki jih vrne sistem.

Kot je prikazano na slikah v dodatku A.2 in A.3, so podatki zelo berljivi. Rezultati različnih statističnih analiz so prikazani na različnih straneh za boljšo preglednost. Sistem po analizi podatke razvrsti od najbolj do najmanj relevantnega. Podatki, ki so statistično relevantni (imajo p-vrednost nižjo od 0.05 ali AUC vrednost višjo od 0.75) so obarvani črno. Stran z rezultati vsebuje tudi legendo, ki pojasnjuje pomen različnih znakov. Pri ROC analizi je mogoče prikazati podatke tudi v grafičnem prikazu, kot je razvidno na sliki 4.2.

4.4.3 Odzivnost

Odzivnost sistema definiramo kot odzivni čas, ki ga sistem potrebuje za izvršitev ukaza. V našem primeru mora biti čas med klikom gumba in prikazom rezultatov statističnih analiz zelo kratek.

Največjo težavo odzivnosti sistema je predstavljal prenos podatkov iz FHIR strežnika. To težavo smo rešili tako, da se podatki prenesejo le enkrat in shranijo lokalno, kot je opisano v poglavju 4.2. V sistemu so uporabljena orodja, ki obdelavo velikih podatkov pohitrijo, kot na primer funkcije knjižnice Numpy.

4.4.4 Povečljivost

V primeru, da število podatkov znatno naraste, mora biti sistem sposoben normalno delovati.

Za izgradnjo sistema so bila uporabljena orodja, ki podpirajo delo z veliko količino podatkov. Algoritmi statistične analize so bili razviti v Pythonu, ker njegove knjižnice omogočajo učinkovito delo z veliko količino podatkov. Uporabljene so bile knjižnice Numpy, Scipy in Sklearn. Tudi FHIR strežnik, v katerem so shranjeni podatki, je namenjen delu z velikimi količinami podatkov.

Poglavje 5

Sklepne ugotovitve

Diplomska naloga se osredotoča na oblikovanje arhitekture sistema, ki bi omogočala pregledno analitiko zdravnikom. S pomočjo sistema lahko zdravniki hitro in preprosto dostopajo do informacij o povezavah med boleznimi in meritvami.

Pri gradnji sistema smo se osredotočali na ključna merila, predstavljena v uvodu. Sistem, opisan v diplomskem delu, je preprost za uporabo, pregleden, odziven in povečljiv.

V smislu arhitekture je sistem mogoče nadgraditi na različne načine. Podatke bi lahko črpali iz različnih virov, ne le iz FHIR strežnika, s čimer bi lahko pridobili večji nabor podatkov.

Sistem je mogoče nadgraditi tudi v smislu statistične analize. Diplomsko delo predstavlja delujoč prototip, ki prikazuje delovanje zgrajene arhitekture. Uporabili smo osnovne statistične teste, ki so uporabni in nadgradljivi. V sistem bi lahko umestili bolj zahtevne statistične analize, ki bi uporabniku vrnile bolj poglobljena znanja, pridobljena iz podatkov.

Izboljšali bi lahko tudi odzivnost, kajti FHIR strežniki za obdelavo zahtevka porabijo veliko časa. Z uporabo strežnika, ki bi hranil podatke v drugačnem formatu, bi lahko hitrost vnosa podatkov bistveno izboljšali.

Literatura

[1] Rohan Bhardwaj, Ankita R Nambiar, and Debojyoti Dutta. A study of machine learning in healthcare. In *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*, volume 2, pages 236–241. IEEE, 2017.

[2] Bill Siwicki. Sutter Physician Services, CloudMedx to apply artificial intelligence to patient care. *Healthcare IT news*, 2016.

[3] Basit Chaudhry, Jerome Wang, Shinyi Wu, Margaret Maglione, Walter Mojica, Elizabeth Roth, Sally C Morton, and Paul G Shekelle. Systematic review: impact of health information technology on quality, efficiency and costs of medical care. *Annals of internal medicine*, 144(10):742–752, 2006.

[4] Morten W. Fagerland. T-tests, non-parametric tests, and large studies. *BMC Medical Research Methodology*, 12(1):78, Jun 2012.

[5] Steve G. Langer Hussain Mohannad A. and Marc Kohli. Learning HL7 FHIR Using the HAPI FHIR Server and Its Use in Medical Imaging with the SIIM Dataset. *Journal of Digital Imaging*, 31(3):334–340, 2018.

[6] Hajian-Tilaki Karimollah. Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian Journal of Internal Medicine*, 4(2):627–635, 2013.

[7] Leape LL. Error in medicine. *JAMA*, 272(23):1851–1857, 1994.

[8] Allen Marshall. How many die from medical mistakes in US hospitals. *Scientific American*, 9:20, 2013.

[9] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John Wiley & Sons, 2010.

[10] Maximillian Johnson Paul-Andre Henegar. Synthia. `https://github.com/synthia-synth/synthia`, 2016. [Dostopano: 16. 8. 2018].

[11] Dale Sanders, David A Burton, and Denis Protti. The healthcare analytics adoption model: A framework and roadmap. *Health Catalyst*, 2013.

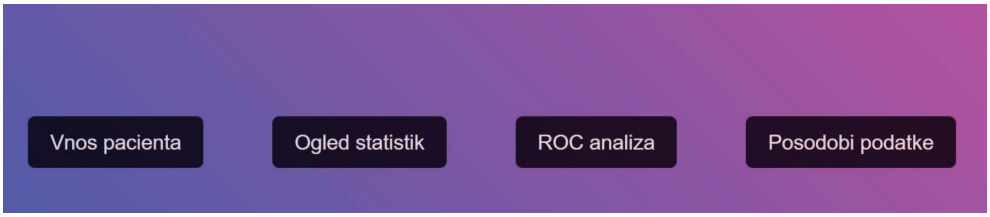
Dodatek A

Delovanje spletnega vmesnika

Spletni vmesnik je sestavljen iz 4 gumbov, ki kažejo na naslednje strani. Pregledna plošča je predstavljena na sliki A.1

Če uporabnik pritisne gumb *Vnos pacienta*, se mu bo odprl obrazec, kjer lahko vnese podatke o pacientu. Ta se nahaja v datoteki `form.php`. Vsebuje HTML obrazec, kamor vnesemo podatke o bolniku, bolezni in opažanju. Po pritisku gumba POŠLJI, bo javascript skripta poskrbela, da se bodo vsi trije podatki pretvorili v svoje objekte. Objekte nato vnesemo v transakcijski snop, ki se samodejno pošlje v FHIR strežnik. Če so se podatki vnesli uspešno, se bo prikazalo okno s sporočilom *"Pacient je bil uspešno dodan"*, v nasprotnem primeru pa okno s sporočilom *"Prišlo je do napake, prosim poskusite ponovno".*

Če uporabnik pritisne gumb *Ogled statistik*, se bo odprla stran, ki zahteva vnos imena bolezni. Ta stran se nahaja v datoteki `mid1.php` in služi vnosu bolezni. Po pritisku gumba SUBMIT, se odpre datoteka `rezults.php`. Ta na začetku zažene skripto `Analiza.py`, ki vzame podatke iz datoteke, katere ime je enaka bolezni. Podatke nato pretvori v slovar in analizira, kakor je opisano v poglavju 4.2. Po končani analizi, se podatki shranijo v datoteko `rezults1.csv`. Iz te se podatki prenesejo na stran in prikažejo v pregledni tabeli. Ta je vidna v sliki A.2. Tabela vsebuje podatke o opažanjih, razvrščene po p-vrednostih na desni. Vsi podatki, ki imajo p-vrednost manjšo



Slika A.1: Pregledna začetna plošča.

od 0.05, so relevantni in zato potemnjeni. Tabela vsebuje še podatke o tipu opažanja. Prvi tip opažanj so skalarji, ki vsebujejo podatke o povprečni vrednosti in standardnem odklonu bolnih in zdravih, ter število obeh. Drugi tip opažanj so kategorije, ki vsebujejo podatke o vseh različnih vrednostih, število zdravih in bolnih, ter vrednosti obeh. P-vrednostim so dodane tudi oznake, ki povedo, kateri algoritem je bil uporabljen za izračun vrednosti. Če je prikazan znak (a) pomeni, da je bil uporabljen T test, (b) Mann Whitneyev test, (c) χ^2 test in (d) Fisherjen natančni test. V tabelah so prikazani zgolj rezultati analize, če je imelo posamezno opažanje vsaj 5 primerkov. Ta tabela služi zdravnikom, ki jih zanima katera opažanja so pri določeni bolezni najbolj izrazita.

Če uporabnik pritisne gumb *ROC analiza*, se mu tako kot pri *Ogled statistike* najprej odpre okno za vnos pacienta. Kljub temu da je okno videti enako kot pri *Ogled statistike*, je ta implementiran v datoteki `mid2.php`. Po vnosu bolezni se tako kot prejšnji strani odpre nova stran in sicer `roc.php`. Tu se na začetku zažene `Analiza.py`, ki vrne `rezultati2.csv`. Ti se na strani prikažejo v pregledni obliki. Kot je prikazano na sliki A.3, so prikazani različni podatki za vsako opažanje in sicer AUC, specifičnosti, senzitivnost in optimalni prag. Podatki so urejeni po AUC vrednosti in so v primeru, če so nad 0.75 potemnjeni. Če uporabnik pritisne na željen atribut se mu izriše tudi graf, ki vsebuje ROC krivuljo, ki je razvidna iz slike 4.2

Če uporabnik pritisne gumb *Posodobi podatke*, se zažene Python skripta `generate.py`. Ta se uporablja, ko uporabnik želi prenesti vse podatke iz FHIR strežnika na računalnik. Delovanje skripte je opisano v poglavju 4.3.

Legenda:
a.....T test
b.....Mann Whitney U test
c.....Chi square test
d.....Fisher exact test

Rezultati, ki so temnejši imajo vrednosti nad 0.05

Atribut	Tip	Zdravi	Št. zdravih	Bolanih	Št. bolanih	P vrednost
Creatinine	skalar	1.63±1.08	684	1.04±0.38	41	0.000b
Microalbumin Creatinine Ratio	skalar	118.94±97.40	304	7.68±5.33	41	0.000b
Estimated Glomerular Filtration Rate	skalar	74.83±47.95	535	117.13±29.92	41	0.000b
Hemoglobin A1c/Hemoglobin.total in Blood	skalar	4.92±2.03	697	6.62±1.39	41	0.000b
Body Height	skalar	148.40±38.99	1862	170.53±5.75	41	0.001b
High Density Lipoprotein Cholesterol	skalar	61.34±13.02	617	51.74±21.19	41	0.008b

Slika A.2: Tabela z rezultati statistične analize.

Atribut	Youden cutoff	Specificity	Sensitivity	AUC
Creatinine	1.593	1.000	1.000	1.000
Glucose	129.989	1.000	0.875	0.984
Triglycerides	500.321	1.000	0.875	0.960
Low Density Lipoprotein Cholesterol	141.967	1.000	0.875	0.938
Total Cholesterol	261.430	0.933	0.875	0.896
Body Height	177.209	0.845	1.000	0.874

Slika A.3: Tabela z rezultati ROC analize.