Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andrej Zupanc

Napovedovanje parkinsonove bolezni z analizo govora s pametnim telefonom

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Z analizo govora je možno avtomatsko razpoznati znake parkinsonove bolezni pri govorcu. Kandidat naj v diplomskem delu izdela pregled po literaturi takega pristopa in razvije mobilno aplikacijo, ki bo omogočala napovedovanje parkinsonove bolezni na osnovi predlaganega diagnostičnega testa. Ustrezni klasifikator naj kandidat razvije z orodji strojnega učenja iz klasificiranih posnetkov govora. Po potrebi naj razvije tudi programski vmesnik, ki bo omogočal delovanje klasifikatorja in shranjevanje diagnostičnih podatkov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Andrej Zupanc, z vpisno številko **63060318**, sem avtor diplomskega dela z naslovom:

Napovedovanje parkinsonove bolezni z analizo govora s pametnim telefonom

(angl. Predicting Parkinson's Disease with Voice Analysis on a Smartphone)

IZJAVLJAM

1. da sem diplomsko delo izdelal samostojno pod mentorstvom akad. prof. dr. Ivana Bratka;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 23. januarja 2017 Podpis avtorja:

Zahvalil bi se mentorju akad. prof. dr. Ivanu Bratku za njegovo pomoč in nasvete pri izdelavi diplomskega dela. Zahvalil bi se tudi družini za podporo in zaupanje.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
SVM	support vector machine	metoda podpornih vektorjev
API	application programming interface	aplikacijski programski vmesnik
CNN	condensed nearest neighbor	zgoščeni najbližji sosed
ENN	edited nearest neighbor	popravljeno pravilo najbližjih sosedov
SMOTE	synthetic minority oversampling technique	tehnika za sintetično nadvzorčenje manjšinskega razreda
ADASYN	adaptive synthetic sampling	prilagoditveno sintetično vzorčenje
k**NN**	k-nearest neighbors	k-najbližjih sosedov
HNR	harmonics-to-noise ratio	razmerje med harmoničnostjo in šumom
NHR	noise-to-harmonics ratio	razmerje med šumom in harmoničnostjo
PB	Parkinson's disease	parkinsonova bolezen

Povzetek

Zgodnja diagnoza bolezni lahko precej vpliva na njen nadaljnji potek, pravočasno zdravljenje in kvaliteto življenja. Vendar vse bolezni niso ozdravljive, zdravniki lahko pomagajo le lajšati simptome. Takšna je parkinsonova bolezen, ki je nevrodegenerativna bolezen, katere glavni simptomi so tremor v mirovanju, počasno začenjanje gibov, mišična rigidnost in tudi težave z govorom. Zato smo se v tej diplomski nalogi odločili razviti sistem za zgodnje odkrivanje parkinsonove bolezni, ki bi znal prepoznati znake parkinsonove bolezni v govoru. V ta namen smo razvili mobilno aplikacijo, vmesnik API in klasifikator. Vmesnik API shrani zvočne posnetke, posnete z mobilno aplikacijo, jih analizira in klasificira s klasifikatorjem, ki je bil prav tako razvit v sklopu diplomske naloge. Po končani klasifikaciji vmesnik API vrne rezultat mobilni aplikaciji, ki uporabnika obvesti o rezultatu analize njegovega glasu. Aplikacija je bila razvita za operacijski sistem Android. Vmesnik API je razvit s pomočjo knjižnice Flask. Različice klasifikatorjev so bile razvite s knjižnico Scikit learn in Keras, med katerimi smo izbrali najboljšega in ga implementirali v vmesnik API. Predstavljen je tudi primer uporabe tega izdelka.

Ključne besede: strojno učenje, parkinsonova bolezen, mobilna aplikacija, neuravnoteženi podatki.

Abstract

Title: Predicting Parkinson's Disease with Voice Analysis on a Smartphone

Author: Andrej Zupanc

Early diagnosis can have significant effect on disease progression, its treatment and patient's quality of life. However, some diseases are incurable, and doctors can only help relieve symptoms. One of such is Parkinson's disease, a neurodegenerative disease marked by tremor, slowness of movement, muscular rigidity and difficulty with speaking. The aim of this paper was to develop a system for early diagnosis of Parkinson's disease which could recognize signs of Parkinson's disease in a person's voice. For this purpose, a mobile application, an API interface and a classifier were developed. The API interface saves voice recordings made by the mobile application, then analyses and classifies them with the classifier. After the classification is done, the API interface sends the result back to the mobile application which informs its user about the outcome of their voice analysis. The application was developed for Android operating system. The API interface is based on the Flask library. Different classifiers using libraries Scikit-learn and Keras were developed. Then, the most appropriate classifier was chosen and implemented into the API interface. An example of how the application can be used is also described.

Keywords: machine learning, Parkinson's disease, mobile application, imbalanced data.

Poglavje 1

Uvod

Zgodnja in natančna diagnoza je pomembna pri vseh boleznih, kajti tako lahko zdravniki hitro ukrepajo in pričnejo z zdravljenjem bolnika. Vendar se izkaže, da je postaviti zanesljivo diagnozo v zgodnjem stadiju bolezni zelo težko, še posebej se to izkaže za bolezni, povezane z možgani in živčevjem. Takšna je tudi parkinsonova bolezen.

Parkinsonova bolezen je ena izmed najpogostejših nevrodegenerativnih bolezni, ki je trenutno še neozdravljiva. Ocenjeno število bolnikov s parkinsonovo boleznijo v Sloveniji je okoli sedem tisoč [25]. Parkinsonova bolezen je počasi napredujoča bolezen, katere glavni simptomi so tremor v mirovanju, počasno začenjanje gibov, mišična rigidnost in težave z ravnotežjem [1]. Te simptome, ki so najbolj opazni, lahko spremljajo še drugi, kot so depresija, kognitivne težave, težave s požiranjem in težave z govorom. Težave oseb s parkinsonovo boleznijo se pri govoru pokažejo kot monoton govor, premori med govorom, neenakomeren ritem govora in težave z začenjanjem govora [22].

Cilj te diplomske naloge je razviti mobilno aplikacijo skupaj s programskim vmesnikom in klasifikatorjem, ki bo na podlagi izgovorjenega trajajočega a-glasu napovedovala prisotnost oziroma odsotnost parkinsonove bolezni pri uporabniku.

V poglavju 3 so predstavljeni podatki, ki smo jih uporabili pri izdelavi

klasifikatorja, in uporaba metod, s katerimi smo umetno uravnotežili podatke. Podatki, ki smo jih pridobili na spletu in uporabili za učenje, so bili zelo neuravnoteženi glede na zastopanost posameznih razredov. Zato smo jih morali najprej ustrezno urediti. Uporabili smo štiri metode: metodi za nadvzorčenje, SMOTE in ADASYN, ki dodata sintetične podatke v učno množico, ter kombinaciji metode SMOTE s podvzorčenjem, ki iz učne množice zbrišeta napačno klasificirane podatke.

Po uporabi teh metod smo generirali učne množice, ki so služile kot vhodni podatki učnim algoritmom. Uporabili smo učne algoritme za klasifikacijo podatkov, predvsem smo uporabili metode SVM, logistično regresijo, naključne gozdove in nevronske mreže. Opis in primerjava med različnimi učnimi algoritmi in podatkovnimi množicami sta opisana v poglavju 4. Pri testiranju učnih algoritmov smo uporabili desetkratno prečno preverjanje. Tako smo dobili dokaj zanesljiv podatek o točnosti modelov in lažje izbrali najboljšega izmed vseh testiranih modelov. Ta model smo nato implementirali na lastnem strežniku, kjer smo morali dodati programski paket Octave, s katerim računamo atribute zvočnih posnetkov uporabnikovega govora. Bolj podrobno sta strežnik in vmesnik API opisana v poglavju 5.2.

Da bi lahko uporabnik koristil že implementirana klasifikator in vmesnik API, smo razvili tudi mobilno aplikacijo, opisano v poglavju 5.1. Aplikacija omogoča snemanje uporabnikovega glasu, vodenje zgodovine posnetkov in rezultatov klasifikacije posnetka, ter pošiljanje posnetka na spletni strežnik, kjer se posnetek analizira in vrne rezultat klasifikacije. V poglavju 6 smo predstavili primer uporabe našega sistema. Opravljeno delo in rezultate smo povzeli v poglavju 7, kjer so omenjene tudi možnosti za izboljšave in predlogi za nadaljnji razvoj tega sistema.

Rezultat te diplomske naloge je uspešna implementacija mobilne aplikacije, vmesnika API in klasifikatorja za določanje prisotnosti PB, ki kot celota omogočajo enostavno uporabo in zanesljivo klasifikacijo uporabnika na podlagi njegovega glasu.

Poglavje 2

Pregled obstoječih raziskav

Analizo trajajočega glasu črke a pri bolnikih s PB so leta 1997 prvi uporabili Javier Gamboa in drugi iz Univerzitetne bolnišnice Príncipe de Asturias v Madridu[12]. Pri tem so iz zvočnega zapisa izluščili atribute, kot so osnovna frekvenca, razlike v periodah, razlike v amplitudah in HNR (angl. harmonic-to-noise ratio, razmerje med signalom in šumom), ki jih uporabljamo v tej diplomski nalogi. Ti atributi naj bi bili nespremenjeni ne glede na stopnjo PB.

Kot je pokazal Max Little [17], so lahko za klasifikacijo na podlagi glasu boljše nelinearne metode za računanje atributov iz zvočnih posnetkov. S tem je tudi pokazal, da se lahko te metode uporabljajo v zdravstvenih napravah na uporabnikovem domu in da lahko s podatki iz teh naprav uspešno napovedujemo uporabnikovo stopnjo UPDRS (angl. unified Parkinson's disease rating scale, enotna ocenjevalna lestvica za bolnike s parkinsonovo boleznijo) [26]. V nadaljevanju so Athanasios Tsanas in drugi [27, 28] implementirali in testirali še več nelinearnih metod, ki se lahko uporabljajo tudi pri razpoznavi glasu. Z namenom spodbujanja raziskav na področju napovedovanja PB s pomočjo govora sta Max Little in Athanasios Tsanas ustanovila skupino Parkinson's Voice Initiative[1].

S porastom pametnih mobilnih telefonov so se odprle možnosti za upo-

[1] `http://www.parkinsonsvoice.org/index.php`

rabo le-teh pri določanju bolezenskih znakov. Ker imamo telefon vedno s sabo, lahko telefon dlje časa spremlja morebitne bolezenske znake. V ta namen je bilo razvitih več aplikacij. Aplikacija HopkinsPD [31] in aplikacija FoxInsight[2] sta le dve izmed mnogih[3], ki jih lahko uporabimo za spremljanje bolezenskih znakov PB. Aplikacijo ParkinsonCheck[4], s katero lahko preverimo tip tremorja med risanjem spiral na mobilnem telefonu, so razvili raziskovalci Fakultete za računalništvo in informatiko [21].

Vendar sta aplikaciji HopkinsPD in FoxInsight namenjeni bolj natančnemu spremljanju simptomov PB in ne nudita klasifikacije uporabnika glede na njegov zvočni zapis. To vrzel skušamo zapolniti z našo aplikacijo.

[2] `https://foxinsight.michaeljfox.org`

[3] `http://parkinsonslife.eu/top-apps-for-the-parkinsons-community/`

[4] `http://www.parkinsoncheck.net`

Poglavje 3

Podatki

Pridobivanje podatkov iz zdravstvenih raziskav je lahko dolgotrajno, zahtevno in porabi veliko resursov. Sami smo to izkusili, ko se je ponudila možnost posneti nekaj bolnikov s PB. Zaradi premajhnega števila lastnih govornih posnetkov smo za našo aplikacijo poleg svojih podatkov uporabili še dve podatkovni bazi, ki sta dostopni na repozitoriju za strojno učenje UCI (angl. UCI machine learning repository) [15].

3.1 Uporabljene podatkovne zbirke

Pri izbiri podatkovnih zbirk smo morali biti pazljivi, da so bili podatki med sabo kompatibilni. Podatkovni bazi, ki smo ju uporabili v tej nalogi, so objavili:

- Athanasios Tsanas in Max Little [26], ki sta s pomočjo desetih zdravstvenih ustanov in podjetjem Intel s pomočjo naprave za nadzor na daljavo (angl. telemonitoring) spremljala 42 oseb v obdobju šestih mesecev, ki so bile v zgodnji fazi PB. Vsaka oseba je posnela približno 200 zvočnih zapisov. Vendar v tej zbirki ni bilo nobene kontrolne skupine, saj teh podatkov niso uporabili za klasifikacijo pacientov, ampak so poskušali napovedati stopnjo bolezni po UPDRS lestvici. V nadaljevanju to podatkovno množico imenujemo Parkinson Telemonitoring.

- Max Little [17], ki je skupaj z Nacionalnim centrom za glas in govor (angl. National Centre for Voice and Speech) iz Denverja v Koloradu naredil raziskavo z 31 osebami, med katerimi jih je 23 imelo PB, drugih osem je predstavljalo kontrolno skupino. Vsaka oseba je posnela približno šest zvočnih zapisov. V nadaljevanju to podatkovno množico imenujemo Parkinson Speech.

3.1.1 Lastni vzorec

Med izdelavo te diplomske naloge se je ponudila možnost, da bi lahko sami posneli nekaj zvočnih zapisov bolnikov s PB. Pri tem nam je pomagal prof. dr. Zvezdan Pirtošek z Nevrološke klinike Univerzitetnega kliničnega centra v Ljubljani. Zbrali smo zvočne zapise štirih oseb, med katerimi sta bili dve s PB, ena te bolezni ni imela, za eno osebo pa nismo vedeli. Glas oseb se je posnel z naglavnimi slušalkami, ki so imele mikrofon. Zvok se je shranil na osebni računalnik s pomočjo programa PRAAT [5].

Podatkovna zbirka	Število vzorcev	Število oseb	Število atributov
Parkinson Telemonitoring	5875	42 (42PB + 0 zdravih)	26
Parkinson Speech	195	31 (23PB + 8 zdravih)	23
Lastna podatkovna množica	8	4 (2PB + 1 brez PB + 1 za katerega ne vemo)	10

Tabela 3.1: Opis uporabljenih podatkovnih množic.

Kot smo že omenili, smo lahko ti dve podatkovni množici združili, ker sta uporabljali enake algoritme. Ker smo želeli v tej diplomski nalogi analizirati tudi lastne posnetke, smo morali paziti, da smo lahko sami z istimi algoritmi izračunali atribute, ki so v teh dveh podatkovnih zbirkah. Vseh atributov sami nismo mogli izračunati, ker nekateri algoritmi niso prosto dostopni.

Ostale atribute, za katere smo imeli dostopne algoritme, smo izračunali s pomočjo knjižnice, katero so Tsanas in drugi [27, 28, 29, 16, 18] razvili v programu Matlab. Knjižnica se imenuje Orodje za analizo zvoka (angl. voice analysis toolbox) in je dostopna na spletni strani dr. Athanasiosa Tsanasa[1]. Knjižnica uporablja iste algoritme kot program PRAAT, s katerim so bili izračunani nekateri atributi v uporabljenih podatkovnih množicah, vendar smo to knjižnico lažje vključili v naš vmesnik. Za njeno uporabo smo v tej diplomski nalogi uporabili programski paket Octave [9].

3.2 Atributi

Kot prikazuje tabela 3.1, ima podatkovna množica Parkinsons Telemonitoring 26 atributov, množica Parkinson Speech pa 23, vendar večina med sabo ni združljivih. Tiste atribute, ki so samo v eni podatkovni množici in ne v drugi, ali obratno, smo zavrgli, ker predstavljajo drugačne lastnosti zvočnega zapisa. Prav tako smo zavrgli vse atribute, katerih vrednosti nismo mogli sami izračunati s knjižnico Orodje za analizo zvoka. Ko smo zavrgli vse atribute, ki jih nismo mogli uporabiti, nam jih je ostalo še 10. V nadaljevanju smo zaradi nedvoumnosti uporabili kar originalna imena atributov v angleščini. Ti atributi so sledeči:

- Atributi, ki merijo tresenje zvočnega signala (razlike v periodah) in razlike v amplitudah zvočnega signala [11]:
  - Jitter (ddp): absolutna razlika med razliko zaporednih period. Nato pa še to delimo s povprečno periodo signala.

$$ddp = \frac{N * \sum_{i=2}^{N-1} |(T_{i+1} - T_i) - (T_i - T_{i-1})|}{(N-2) * \sum_{i=1}^{N} T_i} \tag{3.1}$$

    Kjer je spremenljivka N število intervalov signala, spremenljivka T_i pa predstavlja dolžino i-te periode.

[1] `http://people.maths.ox.ac.uk/~tsanas/software.html`

– Shimmer (apq3): absolutno povprečje razlike med periodo amplitude in povprečjem amplitud njenih dveh sosed. Nato še to delimo s povprečno amplitudo signala.

$$apq3 = \frac{N * \sum_{i=2}^{N-1} |T_i - (T_{i-1} + T_i + T_{i+1})/3|}{(N-2) * \sum_{i=1}^{N} T_i} \tag{3.2}$$

Kjer je spremenljivka N število intervalov signala, spremenljivka T_i pa predstavlja velikost i-te amplitude.

– Shimmer (apq5): enako kot Shimmer (apq3), le da se tu uporabijo amplitude štirih najbližjih sosed.

$$apq5 = \frac{N * \sum_{i=2}^{N-2} |T_i - (T_{i-2} + T_{i-1} + T_i + T_{i+1} + T_{i+2})/5|}{(N-4) * \sum_{i=1}^{N} T_i} \tag{3.3}$$

Kjer je spremenljivka N število intervalov signala, spremenljivka T_i pa predstavlja velikost i-te amplitude.

– Shimmer (ddp): absolutna razlika med razliko zaporednih amplitud. Nato pa še to delimo s povprečno amplitudo signala. Vrednost tega atributa je točno trikratnik atributa Shimmer (apq3). Izračuna se s podobno enačbo, kot je enačba 3.1, le da namesto period uporabimo amplitude signala.

- HNR (angl. harmonics-to-noise ratio): razmerje med signalom in šumom.
- NHR (angl. noise-to-harmonics ratio): razmerje med šumom in signalom.
- RPDE (angl. recurrence probability density entropy, entropija gostote verjetnosti ponovitve): predstavlja vrednost, s katero ovrednotimo pravilnost delovanja glasilk [18]. Višja vrednost tega atributa nakazuje na govorne težave, ki so lahko povezane s PB. Pri osebah s PB se nepravilno delovanje glasilk opazi v monotonem in šibkejšem glasu [22].

- DFA (angl. detrended fluctuation analysis, analiza odstranjevanja trendov nihanja): predstavlja vrednost, s katero ovrednotimo zasoplost v zvočnem zapisu [16]. Višja vrednost tega atributa nakazuje na govorne težave, ki so lahko povezane s PB. Osebe s PB imajo slab nadzor nad dihanjem med fonacijo, kar privede do porabe določene količine zraka, še preden oseba prične s fonacijo [22].

- PPE (angl. pitch period entropy, entropija trajanja višine glasu): predstavlja vrednost, s katero ovrednotimo sposobnost ohranjanja iste višine tona skozi cel zvočni zapis posnetega glasu [17]. Višja vrednost tega atributa nakazuje na govorne težave, ki so lahko povezane s PB.

- Razred: definiran je razred, kamor spada vzorec. Če ima vrednost 1, pomeni, da je bil vzorec pridobljen od osebe, ki ima PB. Če ima vrednost 0, pomeni, da ta vzorec ne predstavlja PB.

Da nam je knjižnica, s katero smo računali zgornje atribute, izračunala samo 9 zgoraj naštetih atributov, je bilo potrebno popraviti izvorno kodo. Po popravku izvorne kode nam je knjižnica za vsak zvočni zapis vračala le teh 9 atributov. Tako bomo lahko podatkovno množico razširili s podatki, ki jih bo posnela naša aplikacija in poslala v analizo.

3.3 Neuravnoteženost podatkov

Neuravnoteženost predstavlja novo težavo. Če želimo, da ima klasifikator na voljo čim več podatkov, moramo obe podatkovni bazi obdržati v prvotnem stanju, kar pomeni, da imamo 6070 podatkovnih točk, ki opisujejo večinski razred. Če to primerjamo z manjšinskim razredom, kjer je teh točk samo 48, vidimo, da imamo zelo neuravnoteženo podatkovno množico. To se izkaže za težavo pri razvoju klasifikatorja, saj ta potrebuje več podatkov iz manjšinske množice za učenje pravil, s katerimi bo klasificiral nove podatke. Neuravnotežene podatkovne množice se rešujejo z naslednjimi pristopi:

- Uteževanje podatkov: podatkom iz manjšinske množice damo večjo težo, tako lahko učni algoritem kaznujemo za napačno klasifikacijo podatka.

- Podvzorčenje (angl. undersampling): podatki iz podatkovne množice se brišejo in s tem poskušajo uravnotežiti podatke. Poznamo več vrst teh algoritmov. Med njimi so naključno brisanje, selektivno brisanje s pomočjo Tomek povezav (angl. Tomek links) [24] ali ENN pravila [30], in drugi.

- Nadvzorčenje (angl. oversampling): podatki se umetno generirajo in dodajajo k podatkovni množici ter tako poskušajo uravnotežiti podatke. Umetno generirani podatki spadajo v isti razred kot podatki iz manjšinske množice, ker umetno generiramo podatke v okolici podatkov iz manjšinske množice. Tudi tukaj poznamo več vrst algoritmov. Med te algoritme spadajo naključno dodajanje, selektivno dodajanje z algoritmi SMOTE [7] in ADASYN [13].

Za naš primer smo uporabili algoritme z nadvzorčenjem in kombinacijo nadvzorčenja in podvzorčenja, implementirane v Python knjižnici Imbalanced-learn[2]. Ti algoritmi so: tehnika za sintetično nadvzorčenje manjšinskega razreda (angl. synthetic minority oversampling technique, SMOTE), prilagoditveno sintetično vzorčenje (angl. adaptive synthetic sampling, ADASYN), SMOTE s kombinacijo Tomek povezav, SMOTE s kombinacijo popravljenega pravila najbližjih sosedov (angl. edited nearest neighbor rule, ENN) in naključno dodajanje podatkov v manjšinsko množico. Vsi uporabljeni algoritmi so učne podatke uravnotežili v razmerju 50:50 in so na kratko opisani v nadaljevanju.

3.3.1 SMOTE

Namesto naključnega potvarjanja podatkov manjšinskega razreda se v tej metodi novi sintetični podatki generirajo s pomočjo metode kNN. To pomeni,

[2] `https://github.com/scikit-learn-contrib/imbalanced-learn`

da se za vsak vzorec manjšinskega razreda izračuna množica najbližjih sosedov istega razreda. Algoritem iz te množice naključno izbere nekaj vzorcev in izračuna razliko med njimi in prvotno izbranim vzorcem. Število naključno izbranih vzorcev je odvisno od stopnje nadvzorčenja in to število se v času izvajanja algoritma ne spreminja. To razliko množi s številom med nič in ena in ga doda podatkovni zbirki. Tako zmanjšamo pristranskost klasifikatorja manjšinskemu razredu v prid. S to tehniko tudi bolje generaliziramo podatke manjšinskega razreda [7].

3.3.2 ADASYN

Z razliko od metode SMOTE, kjer se količina umetno generiranih podatkov v okolici naključno izbranega vzorca manjšinskega razreda ne spreminja, je pri metodi ADASYN generiranje teh podatkov prilagodljivo. Zamisel te metode je, da med računanjem množice kNN vzorca manjšinskega razreda izračuna tudi, koliko je v tej množici vzorcev večinskega razreda, in to število deli z velikostjo kNN množice. To izračuna za vse prvotne vzorce manjšinske množice in s tem dobi razmerje med vzorci večinskega razreda in velikostjo kNN množice. Šele ko so izračunane kNN množice vseh vzorcev manjšinskega razreda in razmerja v teh množicah, prične ta metoda dodajati sintetične vzorce v manjšinski razred. Večje kot je razmerje pri posamezni množici, več podatkov ta metoda generira v njegovi okolici in obratno. S to metodo zmanjšamo pristranskost podatkov in premaknemo odločitveno mejo, ki loči podatke med različnimi razredi, bližje vzorcem, ki jih težje pravilno klasificiramo [13].

3.3.3 SMOTE s kombinacijo podvzorčenja

Ker se po uporabi metode nadvzorčenja prostor, ki ga je zasedal manjšinski razred, lahko razširi tudi na prostor, ki ga je zasedal večinski razred, se s tem še poslabša definicija razrednih gruč. Takrat lahko pride tudi do prevelikega prilagajanja (angl. overfitting). Zato smo uporabili metodo SMOTE s kom-

binacijo Tomek povezav [2] in ENN [3]. Glavne značilnosti teh dveh metod so:

- Tomek povezave - metoda je definirana kot povezava med dvema vzorcema iz različnih razredov, ki sta najbližje drug drugemu in med njima ni kakšnega drugega vzorca, ki bi bil bližje kateremu koli od teh dveh vzorcev [24]. Ko tvorita dva vzorca različnih razredov med sabo Tomek povezavo, sta oba izbrisana iz podatkovne zbirke. Tomek povezave so lahko uporabljene kot metode podvzorčenja ali čiščenja podatkov. V našem primeru so uporabljene za čiščenje podatkov.

- ENN - ta metoda izbriše še več vzorcev iz podatkovne zbirke kot Tomek povezave. Izbriše namreč vsak vzorec, ki je napačno klasificiran s svojimi tremi najbližjimi sosedi.

Ti dve metodi s kombinacijo SMOTE algoritma dajeta v praksi na splošno zelo dobre rezultate, še posebej ko je manjšinski razred zelo redek v primerjavi z večinskim [3].

Poglavje 4

Razvoj klasifikatorja

Razvoj klasifikatorja se lahko prične šele, ko dobimo dovolj veliko količino kategoriziranih podatkov, s katerimi se lahko klasifikator uči. Podatki morajo biti kategorizirani zato, ker v tej diplomski nalogi uporabljamo algoritme nadzorovanega strojnega učenja. Manj kot je podatkov, slabše se lahko klasifikator izkaže na dolgi rok. Čeprav je v testnih verzijah veliko obetal in imel majhno klasifikacijsko napako, je lahko na dolgi rok klasifikator zelo nezanesljiv, saj parametre, s pomočjo katerih klasificira nove podatke, izračuna na majhni množici podatkov.

Če si predstavljamo vsak podatek v učni množici kot posamezno točko v prostoru, so lahko pri majhni učni množici točke istega razreda čisto skupaj in so zato preveč specifične za svoj razred, kar se izkaže za zelo slabo. Ko bo klasifikator na voljo širši javnosti, veliko novih podatkov ne bo pripadalo tej specifični množici, pa vendar bodo v istem razredu in učni model jih bo klasificiral napačno. Zato želimo imeti čim večjo učno množico, ker lahko tako zajamemo večji prostor in bolj posplošimo parametre učnega modela.

4.1 Uporabljene metode

V naših podatkih uporabljamo samo dva diskretna razreda oziroma dve kategoriji (PB je ali ni), zato uporabljamo metode za klasifikacijo podatkov. Predvsem smo se osredotočili na metodo podpornih vektorjev (SVM), logistično regresijo, naključne gozdove in nevronske mreže.

Logistična regresija, naključni gozdovi in SVM so implementirani v knjižnici Scikit Learn [19], ki je ena izmed priljubljenih knjižnic za strojno učenje v okolju Python.

Nevronske mreže so implementirane v visoko nivojski knjižnici Keras [8], ki se uporablja kot dodatek knjižnicama TensorFlow ali Theano [23]. S knjižnico Keras lahko hitro in enostavno definiramo arhitekturo nevronske mreže v knjižnici TensorFlow ali Theano, kar nam omogoča lažje testiranje in lažji razvoj klasifikatorja. V tej diplomski nalogi smo uporabili knjižnico Keras skupaj s knjižnico Theano. Obe sta prav tako implementirani v okolju Python.

4.1.1 Metoda podpornih vektorjev

Metoda SVM je trenutno ena izmed najbolj priljubljenih metod nadzorovanega strojnega učenja. Uporablja se tako za klasifikacijske kot regresijske probleme. Tako kot nekatere metode strojnega učenja, na primer nevronske mreže in k-najbližjih sosedov, tudi metoda SVM uporabi vse razpoložljive atribute in s kombinacijo le-teh napove razred vhodnega primera. Metoda SVM želi v danem prostoru atributov optimalno ločiti podatke med sabo s tako imenovano hiper-ravnino. Hiper-ravnina mora biti enako oddaljena od podpornih vektorjev v vsej svoji dolžini. Podporni vektorji so tisti primeri, ki so najbližje hiper-ravnini. Razdalji med hiper-ravnino ter podpornimi vektorji pravimo razlika (angl. margin). Optimalna rešitev je tista, ki ima največjo razliko med hiper-ravnino in podpornimi vektorji [14, 20].

Velikokrat vhodnih podatkov v prvotnem prostoru ne moremo ločiti z linearno hiper-ravnino. Vendar če prostor transformiramo, se izkaže, da je ta

transformirani prostor lahko ločljiv z linearno hiper-ravnino. Ta transformirani prostor ima več dimenzij kot prvoten prostor. Transformacijo omogoča jedrna funkcija, ki je lahko linearna ali nelinearna. Nekatere izmed bolj znanih jedrnih funkcij so linearna, polinomska, radialna in sigmoidna. V našem učnem modelu uporabljamo: radialno jedrno funkcijo, parameter C, ki algoritmu poda ceno narobe klasificiranega primera, in je nastavljen na 100, ter parameter γ, ki je v našem primeru nastavljen na 0.01, kar pomeni nižjo pristranskost in višjo varianco modela.

4.1.2 Logistična regresija

Logistična regresija je postala ena izmed bolj priljubljenih klasifikatorskih učnih metod za reševanje problemov v medicini, marketingu, kreditni oceni ipd., ker zelo hitro konvergira z zelo šumnimi vhodnimi podatki. Funkcija, ki jo uporablja logistična regresija, je podana spodaj in vrača rezultate kot verjetnost z vrednostjo med nič in ena. V spodnji enačbi spremenljivka x predstavlja vhodni vektor, spremenljivka w pa vektor uteži. Vrednost 0.5 je ločnica med dvema razredoma, če govorimo o dvorazredni klasifikaciji, kot jo imamo v našem primeru. Logistična regresija išče takšne uteži w, ki bodo minimizirale funkcijo napake na učnih podatkih [20]. V našem primeru bomo uporabili funkcijo napake L_2.

$$Logistic(w * x) = \frac{1}{1 + e^{-w*x}} \tag{4.1}$$

4.1.3 Naključni gozdovi

Pri naključnih gozdovih gre za ansambelsko metodo, ki je zgrajena na podlagi mnogo odločitvenih dreves. Ta odločitvena drevesa so ponavadi majhna in enostavna. V našem primeru uporabljamo naključne gozdove s 500 drevesi. To pomeni, da ko hočemo klasificirati podatek, se ta podatek klasificira na vseh 500 drevesih. Končni rezultat je tisti razred, v katerega je podatek klasificiralo največ dreves. V našem primeru ima vsako drevo globino tri in

lahko pri gradnji posameznega drevesa uporabi pri vsakem vozlišču najboljši atribut iz naključne podmnožice vseh atributov. V našem primeru smo omejili velikost podmnožice na največ tri atribute, ki so predstavljeni v poglavju 3.2.

4.1.4 Nevronske mreže

Nevronske mreže so sestavljene iz vozlišč, ki so med sabo povezana z usmerjenimi povezavami. Predstavljajo možganske nevrone in povezave med njimi. Nevron je sestavljen iz vhodne funkcije, aktivacijske funkcije in izhoda. Najprej izračuna uteženo vsoto vseh svojih vhodov. Uteži, s katerimi se izračuna utežena vsota, so dobljene iz notranjega stanja nevronske mreže, ki se skozi proces učenja stalno spreminja in prilagaja. Za aktivacijsko funkcijo smo uporabili funkcijo ReLU (angl. rectified linear unit) in sigmoidno funkcijo. Aktivacijska funkcija izračuna izhodno vrednost na podlagi utežene vsote. Funkcija ReLU je definirana z naslednjo enačbo:

$$f(x) = max(0, x) \tag{4.2}$$

V tej enačbi spremenljivka x predstavlja rezultat vhodne funkcije nevrona.

Sigmoidna funkcija je definirana kot:

$$f(t) = \frac{1}{1 + e^{-t}} \tag{4.3}$$

Kjer spremenljivka t predstavlja rezultat vhodne funkcije nevrona.

Na takšen način deluje vsak nevron v nevronski mreži. V tej diplomski nalogi smo uporabili večnivojske nevronske mreže z dvema skritima nivojema. Vsak skrit nivo ima po 20 nevronov in uporablja funkcijo ReLU kot aktivacijsko funkcijo. Izhodni nivo nevronske mreže za aktivacijsko funkcijo uporabi sigmoidno funkcijo. Večnivojske nevronske mreže delujejo podobno kot enonivojske nevronske mreže. Razlika je v številu korakov, ki jih potrebujejo za izračun, in je enaka številu skritih nivojev plus ena. Ta nevronska mreža bo uporabljala vzvratno učenje, kjer uporabimo posplošeno pravilo delta. Posplošeno pravilo delta lahko na kratko opišemo tako, da na izhodnem nivoju

izračunamo vrednost delta kot napako klasifikacije. Iz izhodnega nivoja tudi širimo vrednost delta en nivo nazaj proti vhodnemu nivoju in posodobimo uteži med tema dvema nivojema. Tako posodabljamo uteži vse do začetnega nivoja [20].

4.2 Ocenjevanje učnih modelov

Za vsak učni model smo izračunali matriko zmot (angl. confusion matrix), kot je prikazana v tabeli 4.1, s katero si bomo pomagali pri razlagi uspešnosti učnih metod. Uspešnost učnih modelov smo ocenjevali z naslednjimi metodami [14]:

- preciznost (angl. precision): predstavlja razmerje med pravilno pozitivno napovedanimi primeri in vsemi pozitivno napovedanimi primeri, in je definirana kot:

$$preciznost = \frac{TP}{TP + FP} \tag{4.4}$$

- priklic (angl. recall): predstavlja razmerje med pravilno klasificiranimi pozitivnimi primeri in vsemi pozitivnimi primeri, in je definiran kot:

$$priklic = \frac{TP}{TP + FN} \tag{4.5}$$

- mera F (angl. F-measure): predstavlja kompromis med preciznostjo in priklicem, in je definirana kot:

$$meraF = \frac{(1 + \beta^2) * priklic * preciznost}{\beta^2 * priklic + preciznost} \tag{4.6}$$

  kjer β pomeni relativno pomembnost razmerja med preciznostjo in priklicem. Po navadi je β enaka ena.

- klasifikacijska točnost: predstavlja odstotek vseh pravilno klasificiranih primerov, in je definirana kot:

$$točnost = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.7}$$

- krivulja ROC: je klasična metoda za iskanje najboljšega učnega modela. Predstavljena je kot krivulja na grafu, kot je prikazano na sliki 4.1, in ponazarja razmerje med senzitivnostjo (angl. sensitivity) in specifičnostjo (angl. specificity). Definicija senzitivnosti je enaka kot pri priklicu, specifičnost pa predstavlja odstotek med pravilno klasificiranimi negativnimi primeri in vsemi negativnimi primeri, torej nasprotno kot senzitivnost. Na grafu je to prikazano kot senzitivnost na osi Y in na osi X je vrednost ena minus specifičnost ($1 - specifičnost$). Senzitivnost in specifičnost imata zalogo vrednosti med nič in ena. Boljši kot je učni model, bolj se bo krivulja približala levemu zgornjemu delu grafa. Če sedaj izračunamo ploščino pod krivuljo, vidimo, da večja kot je ploščina, boljši je učni model. Tej ploščini rečemo ploščina pod krivuljo ROC (angl. area under ROC curve, AUC).

	Negativno napovedan primer	Pozitivno napovedan primer
Negativen primer	Število pravilno napovedanih negativnih primerov (TN)	Število nepravilno napovedanih negativnih primerov (FP)
Pozitiven primer	Število nepravilno napovedanih pozitivnih primerov (FN)	Število pravilno napovedanih pozitivnih primerov (TP)

Tabela 4.1: Matrika zmot.

Metode smo testirali s k-kratnim prečnim preverjanjem, kjer smo k nastavili na deset. K-kratno prečno preverjanje je metoda, ki razdeli vhodne podatke na k različnih enako velikih delov, izmed katerih je en označen kot validacijska množica, preostali podatki pa kot množica, s katero se algoritem uči. Pravilnost klasifikacije preverimo z validacijsko množico in s tem izračunamo točnost napovedi učnega modela. To se nato ponovi še (k-1)-krat. Tako dobimo k približkov klasifikacijske točnosti. Končna ocena napake pa

je povprečje vseh k približkov.

Celotni postopek smo v našem primeru ponovili še desetkrat, da bi dobili čim boljšo oceno pravilnosti vsakega učnega modela. Tako smo na koncu za vsak model dobili 100 vrednosti klasifikacijske točnosti in vrednosti AUC. Algoritem k-kratno prečno preverjanje je v vsaki množici ohranil prvotno distribucijo podatkov glede na razred (v našem primeru približno 125:1 v prid podatkom s PB).

Pri uporabi k-kratnega prečnega preverjanja smo morali biti zelo pazljivi. Če se podatki pred zagonom k-kratnega prečnega preverjanja umetno generirajo, namreč lahko dobimo preveč optimistične vrednosti za klasifikacijsko točnost in AUC. V primeru, da najprej uravnotežimo podatke z metodami za nadvzorčenje in šele nato testiramo model s k-kratnim prečnim preverjanjem, povečamo verjetnost, da se med novimi podatki pojavi podoben ali celo enak podatek, ki je v prvotni podatkovni množici. Tak način vodi do lažje klasifikacije podatkov in preveč optimističnih vrednosti klasifikacijske točnosti [4]. Pri izdelavi naših učnih modelov in ocenjevanju modela smo generirali nove učne podatke na vsakem koraku k-kratnega prečnega preverjanja. To pomeni, da smo stokrat uravnotežili učno množico, ki jo je nato učni algoritem uporabil za učenje. Validacijske množice, ki jo generira algoritem za prečno preverjanje, nismo nikoli uravnotežili, s tem se je ohranila prvotna distribucija podatkov glede na razred.

4.3 Izbira najboljšega učnega modela

Za lažjo izbiro najboljšega učnega modela smo desetkratno prečno preverjanje ponovili desetkrat, tako smo za vsak model dobili po 100 vrednosti CA ter AUC. Vse v poglavju 4.1 naštete učne algoritme smo učili na podatkovnih zbirkah, ki smo jih generirali z metodami SMOTE, ADASYN, naključnega dodajanja, SMOTE s Tomek povezavami in SMOTE z metodo ENN. Te metode generiranja umetnih podatkov manjšinskega razreda smo natančneje opisali v poglavju 3.3. S tem smo dobili rezultate CA in AUC

dvajsetih različnih učnih modelov. Izračunali smo povprečne vrednosti in rezultate predstavili v spodnjih tabelah. Vsaka tabela predstavlja rezultate učnih algoritmov, ki so za uravnoteženje učne množice uporabili isto metodo. Tako lahko na primer v tabeli 4.2 vidimo rezultate učnih algoritmov, ki so se učili na podatkih, generiranih z metodo SMOTE v kombinaciji z metodo ENN.

	SVM	Logistična regresija	Naključni gozdovi	Nevronske mreže
CA	0.916555	0.856670	0.888598	0.906606
AUC	0.871892	0.873705	0.856311	0.896943

Tabela 4.2: Točnost napovedi, kjer so podatki generirani s pomočjo metode SMOTE in ENN.

	SVM	Logistična regresija	Naključni gozdovi	Nevronske mreže
CA	0.919522	0.863112	0.891201	0.979409
AUC	0.871404	0.878185	0.856139	0.873854

Tabela 4.3: Točnost napovedi, kjer so podatki generirani s pomočjo metode SMOTE in Tomek povezav.

	SVM	Logistična regresija	Naključni gozdovi	Nevronske mreže
CA	0.917066	0.849869	0.913806	0.981894
AUC	0.891484	0.866309	0.856373	0.875049

Tabela 4.4: Točnost napovedi, kjer so podatki generirani s pomočjo naključnega nadvzorčenja.

	SVM	Logistična regresija	Naključni gozdovi	Nevronske mreže
CA	0.914167	0.842734	0.892536	0.991895
AUC	0.902173	0.860472	0.880855	0.946440

Tabela 4.5: Točnost napovedi, kjer so podatki generirani s pomočjo metode ADASYN.

	SVM	Logistična regresija	Naključni gozdovi	Nevronske mreže
CA	0.919191	0.863063	0.893062	0.966872
AUC	0.864038	0.876918	0.850128	0.946310

Tabela 4.6: Točnost napovedi, kjer so podatki generirani s pomočjo metode SMOTE.

Če učne modele razvrstimo po vrednostih AUC, se najbolje odrežejo nevronske mreže, kjer se podatki uravnotežijo z metodo ADASYN. Takoj za njimi se uvrsti učni model nevronske mreže, kjer se podatki uravnotežijo z metodo SMOTE. Tretji najboljši učni model, če jih primerjamo samo po AUC vrednostih, je SVM, kjer se podatki uravnotežijo z metodo ADASYN.

Iz vrednosti CA in AUC je možno sklepati, da se učni modeli najbolje učijo na podatkih, generiranih z metodo SMOTE in ADASYN. V nadaljevanju se bomo posvetili primerjavi obeh nevronskih mrež, ki sta se najbolje odrezali glede na vrednosti AUC in CA. Oba učna modela imata skoraj identično vrednost AUC (v našem primeru 0.946440 z metodo ADASYN in 0.946310 z metodo SMOTE).

V tabeli 4.8, kjer so predstavljene nevronske mreže z metodo SMOTE, in tabeli 4.7, kjer so predstavljene nevronske mreže z metodo ADASYN, so prikazane vrednosti za preciznost, priklic in mero F posameznega učnega modela.

Pri učnemu modelu nevronske mreže z metodo ADASYN najbolj izstopata dva priklica. Priklic za razred, ki predstavlja podatke brez PB, ima vrednost 0.25, kar lahko pomeni, da zelo slabo klasificira podatke brez PB. Majhno število podatkov brez PB namreč pod vprašaj daje izračunane vrednosti, saj se lahko po daljši uporabi tega modela izkaže, da so podatki brez PB klasificirani bolje od trenutno prikazanih vrednosti. Priklic za razred, ki predstavlja podatke s PB, pa ima vrednost 0.99, kar lahko nakazuje na preveliko prilagajanje podatkom tega razreda. Ker je število podatkov s PB že večje, lahko po našem mnenju ti podatki kažejo na večjo verjetnost, da je ta model preveč specifičen za razred s PB. Glede na prikazane trenutne vrednosti so v temu modelu napake tipa 1 zelo redke, napake tipa 2 pa bolj pogoste.

Razred	Preciznost	Priklic	Mera F	Št. vzorcev v razredu
Ni PB	0.17	0.25	0.20	4
Je PB	0.99	0.99	0.99	602
Povprečje	0.99	0.99	0.99	Skupaj 606

Tabela 4.7: Matrika z ocenami učnega modela nevronskih mrež s podatki, uravnoteženimi z metodo ADASYN.

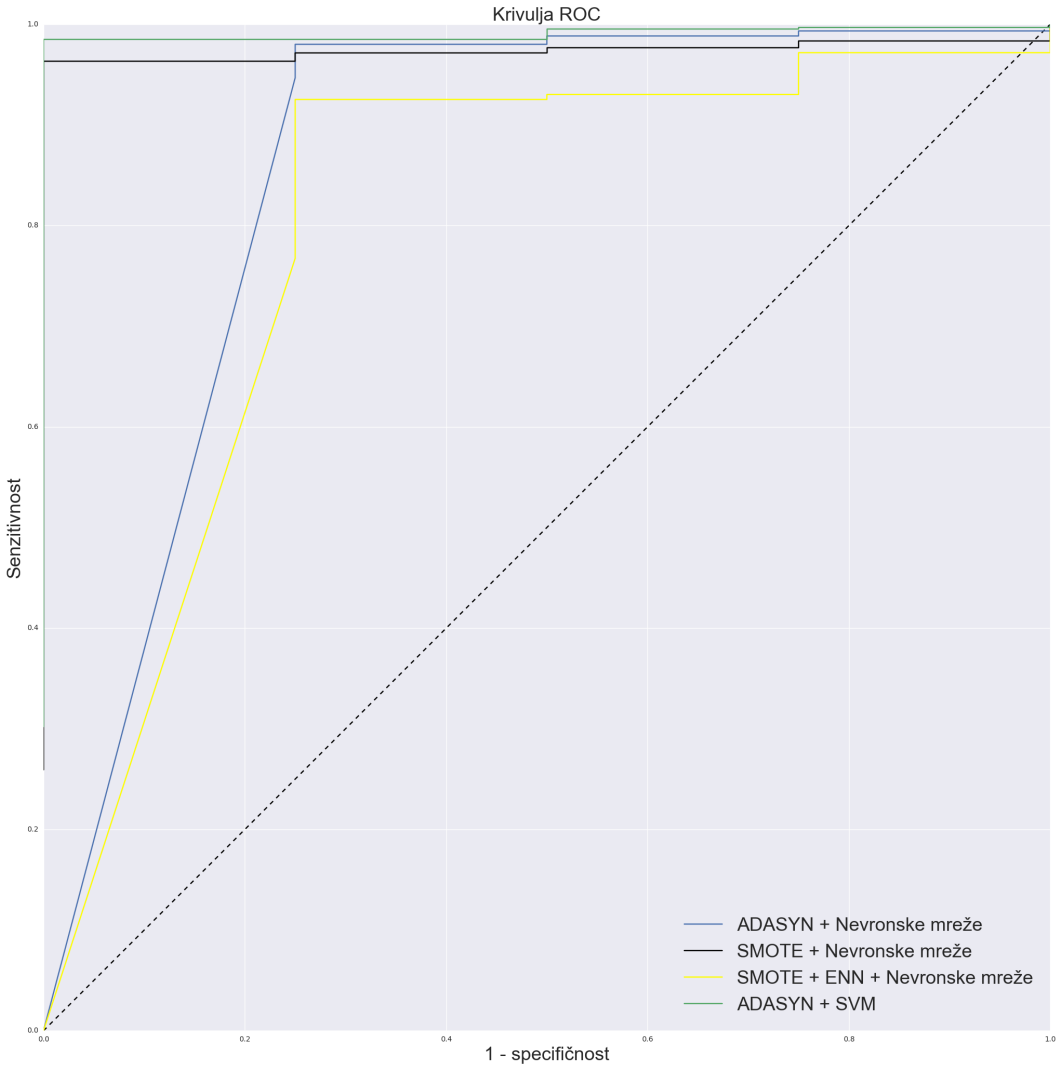
Pri učnemu modelu nevronske mreže z metodo SMOTE opazimo, da bolje klasificira podatke brez PB, na kar nakazuje vrednost priklica, ki je 1. Enako kot pri nevronski mreži z metodo ADASYN imamo tudi v tem primeru premalo podatkov v razredu brez PB, da bi lahko sklepali o razliki med tema dvema učnima modeloma. Je pa v temu učnemu modelu priklic za razred s PB manjši, kar pomeni, da se pri klasifikaciji podatkov s PB zmoti večkrat kot nevronska mreža z metodo ADASYN. To lahko pomeni, da ni preveč specifičen za razred PB in je lahko boljši kot nevronske mreže z metodo ADASYN. Glede na prikazane vrednosti so v temu modelu napake tipa 1 bolj pogoste, napake tipa 2 pa zelo redke.

Razred	Preciznost	Priklic	Mera F	Št. vzorcev v razredu
Ni PB	0.13	1.00	0.23	4
Je PB	1.00	0.96	0.98	602
Povprečje	0.99	0.96	0.97	Skupaj 606

Tabela 4.8: Matrika z ocenami učnega modela nevronskih mrež s podatki, uravnoteženimi z metodo SMOTE.

Zaradi neuravnoteženosti podatkov ne moremo nepremišljeno zaupati točnosti klasifikatorja [6], zato smo različne klasifikatorje primerjali še s krivuljo ROC, kot je prikazano na sliki 4.1. Kot smo že omenili v poglavju 4.2, je najboljši klasifikator tisti, ki ima ROC krivuljo najbližje zgornjemu levemu robu grafa in ima najbolj strmo krivuljo. V našem primeru sta takšna dva klasifikatorja.

Prvi je učni model nevronske mreže z metodo SMOTE, ki smo ga izbrali, ker ima bolj strmo krivuljo in je bližje zgornjemu levemu robu grafa kot učni model nevronske mreže z metodo ADASYN. Drugi najboljši klasifikator glede na ROC krivuljo je učni model SVM z metodo ADASYN. Na grafu lahko sicer vidimo, da je učni model SVM z metodo ADASYN boljši od učnega modela nevronske mreže z metodo SMOTE, vendar izračunana vrednost AUC za metode potrdi naš začetni izbor. Izračun je namreč pokazal, da je vrednost AUC za učni model nevronske mreže z metodo SMOTE za 0.04414 višja kot AUC za učni model SVM z metodo ADASYN. Po analizi teh najbolj obetavnih metod smo za klasifikator našega sistema izbrali učni model nevronske mreže z metodo SMOTE.



Slika 4.1: Krivulje ROC najboljših štirih algoritmov.

Poglavje 5

Razvoj aplikacije in vmesnika API

5.1 Android Aplikacija

Aplikacija je bila razvita za mobilne naprave, ki jih poganja operacijski sistem Android. Tako je na voljo širokemu spektru uporabnikov, saj je imelo 86 % vseh prodanih pametnih mobilnih telefonov v drugem četrtletju tega leta[1] nameščen operacijski sistem Android. Mobilni telefoni se uporabljajo za različne namene, vendar jih je veliko ljudi pričelo uporabljati tudi za iskanje informacij o bolezenskih znakih, ki se jim pojavijo. Raziskava je pokazala, da je leta 2015 v ta namen kar 62 % ljudi v Ameriki uporabilo svoj mobilni telefon[2]. Za razvoj aplikacije smo uporabili razvojno okolje Android Studio[3], ker omogoča hiter in zanesljiv razvoj aplikacij. Poleg tega vsebuje tudi veliko že pripravljenih komponent, ki jih lahko z manjšimi spremembami uporabimo v svoji aplikaciji.

[1] http://www.gartner.com/newsroom/id/3415117

[2] http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/

[3] https://developer.android.com/studio/index.html

5.1.1 Uporabniški vmesnik

Uporabniški vmesnik sestoji iz štirih aktivnosti[4]:

- začetna stran: tu imamo možnost, da glas, ki je bil nazadnje posnet, pošljemo v analizo. Iz te aktivnosti se lahko prestavimo v preostale aktivnosti.
- posnemi glas: aktivnost vsebuje navodila, kako pravilno posneti glas, če želimo, da strežnik glas ustrezno analizira.
- prikaz zgodovine: tu so navedeni vsi posnetki, ki smo jih opravili v tej aplikaciji, skupaj z rezultati analiz in informacijo o poslanih posnetkih. Če še kateri posnetek ni bil poslan, obstaja v tej aktivnosti možnost za naknadno pošiljanje.
- informacije o aplikaciji: tu so navedene informacije o lastniku, čemu je aplikacija namenjena in za kaj se lahko uporablja, ter opozorilo, da rezultati analize in klasifikacije niso absolutna resnica, zato naj se uporabnik obrne na specialista, ki mu lahko ta rezultat pomaga pri dokončni diagnozi.

Da aplikacija ne bi bila omejena samo na slovensko govoreče uporabnike, smo ji dodali možnost spremembe jezika. Aplikacija tako podpira slovenski in angleški jezik. Android hrani tekst, ki je uporabljen v aplikaciji (na primer napis na gumbu, navodila za uporabo in druge) v xml datoteki, imenovani strings.xml, ki je shranjena v mapi values. Za vsak jezik posebej smo ustvarili datoteke string.xml, kot je prikazano na spodnji sliki 5.1. V našem primeru mapa values-sl hrani datoteko s slovenskim besedilom in mapa values-en datoteko z angleškim besedilom. Če želimo dodati še kakšen jezik, ustvarimo novo mapo values in v njej datoteko string.xml z besedilom v želenem jeziku in v kodi dodamo možnost izbire novega jezika.

[4] `http://android.fri.uni-lj.si/index.php/Android_Aplikacije#Aktivnost`

```
res/values/strings.xml
res/values-en/strings.xml
res/values-sl/strings.xml
```

Slika 5.1: Prikaz gnezdenja jezikovnih datotek.

Za lažjo navigacijo v aplikaciji smo uporabili gradnik za orodno vrstico (angl. toolbar), ki se nahaja na zgornjem delu ekrana in je prisotna v vseh aktivnostih. V orodni vrstici vsake aktivnosti so nam na voljo nekatere izmed teh bližnjic, ki se nahajajo desno zgoraj na ekranu. Če ikono posamezne bližnjice nekaj trenutkov držimo, se ob njej pojavi kvadratek s kratkim opisom njene funkcije. Spodaj so na kratko opisane funkcije posameznih ikon.

-  : nas vrne na začetno stran aplikacije.
-  : odpre aktivnost Zgodovina.
-  : zamenja jezik aplikacije iz slovenščine v angleščino ali obratno, odvisno od prvotnega stanja.
-  : odpre aktivnost Informacije o aplikaciji.

5.1.2 Podatkovna baza

Če želimo, da ima uporabnik pregled nad zgodovino svojih posnetkov glasu, si mora aplikacija zapomniti posnetke in spreminjati podatke o tem zapisu. V ta namen smo uporabili podatkovno bazo SQLite[5], ki je na voljo v Android programskem vmesniku[6]. Namesto SQLite podatkovne baze bi lahko za naš namen uporabili tudi tekstovno datoteko, vendar je rokovanje s podatkovno

[5] `https://sqlite.org/`

[6] `https://developer.android.com/reference/android/database/sqlite/package-summary.html`

bazo lažje, varnejše in hitrejše. V podatkovni bazi je samo ena tabela, ki ima atribute ID, Record_file, Response, Date, Status, ustvarjena pa je z ukazom na sliki 5.2.

Za boljše razumevanje smo na kratko opisali uporabljene atribute v podatkovni bazi:

- ID: unikaten identifikator vrstice.
- Record_file: unikatno ime datoteke, kamor se je shranil posnetek.
- Response: odgovor, ki ga vrne strežnik po opravljeni analizi. Če je odgovor 1, to pomeni, da oseba kaže znake bolezni, odgovor 0 pa pomeni, da znakov bolezni ne kaže.
- Date: datum in čas, ko je bil posnetek ustvarjen.
- Status: ali je bil posnetek že poslan v analizo ali ne.

```
CREATE_DB = "CREATE TABLE recordings (" + FeedEntry._ID +
" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, " +
FeedEntry.RECORD_FILE + " TEXT NOT NULL, " +
FeedEntry.RESPONSE + " REAL, " +
FeedEntry.DATE + " TEXT, " +
FeedEntry.STATUS + " TEXT);"
```

Slika 5.2: Prikaz ukaza, s katerim smo ustvarili podatkovno bazo.

5.1.3 Zajem glasu osebe

Zvok uporabnikov snemamo z vgrajenim mikrofonom v mobilnem telefonu, do katerega lahko dostopamo preko programskega vmesnika v Android operacijskem sistemu. Ker želimo imeti posnete datoteke v formatu WAV (angl. waveform audio file format), je bilo potrebno dodati metode, ki nam to omogočajo. V ta namen smo v našo aplikacijo dodali modul[7], ki nam omogoča

[7] https://github.com/mobro/android

snemanje posnetkov v želenem formatu. WAV format določa shranjevanje toka bitov zvočnega posnetka na osebni računalnik. WAV datoteke so shranjene v binarnem formatu in so sestavljene iz več skupkov (angl. chunk). Vsak skupek predstavlja določene metapodatke ali (shranjene) podatke v datoteki (podrobna specifikacija je dostopna na spletni strani univerze McGill[8] ali na spletni strani Dan Watersa[9]). Za ta format smo se odločili, ker knjižnica VoiceBox deluje samo s tem formatom datotek. Parametri datoteke, kot so definirani v aplikaciji, so prikazani na sliki 5.3. Pomen spremenljivk pa je naslednji:

- izvor zvoka (audioSource): z njo določimo izvor zvoka. V našem primeru je to vgrajen mikrofon v mobilnem telefonu.

- frekvenca vzorčenja (sampleRateInHz): določa, na koliko vzorcev se med pretvorbo analognega v digitalni signal razdeli 1 sekunda zvočnega zapisa. Višja kot je frekvenca vzorčenja, manj podatkov izgubimo med pretvorbo v digitalni signal, saj je časovni razmik med posameznimi vzorci analognega signala manjši.

- zvokovna reprodukcija (channelConfig): določa število zvočnih kanalov. Za naš namen je dovolj samo en zvočni kanal.

- kodiranje zvoka (audioFormat): določa kodiranje in bitno globino, ki je uporabljena za zvočni zapis.

Posnetki se najprej shranijo v pomnilnik mobilnega telefona. Nato jih lahko uporabnik ob poljubnem času pošlje v analizo na spletni strežnik. Ker WAV format ne uporablja kompresije pri shranjevanju zvoka v datoteke, so lahko te precej večje kot pri drugih formatih. Zato je priporočljivo pošiljanje datotek preko domačega brezžičnega omrežja.

[8] `http://www-mmsp.ece.mcgill.ca/documents/audioformats/wave/wave.html`

[9] `https://blogs.msdn.microsoft.com/dawate/2009/06/23/intro-to-audio-programming-part-2-demystifying-the-wav-format/`

```
int audioSource = MediaRecorder.AudioSource.MIC;
int sampleRateInHz = 44100;
int channelConfig = AudioFormat.CHANNEL_IN_MONO;
int audioFormat = AudioFormat.ENCODING_PCM_16BIT;
```

Slika 5.3: Glavni parametri datoteke v WAV formatu.

5.1.4 Komunikacija s strežnikom

Za povezavo med strežnikom in odjemalcem ter ohranjanje seje skrbi protokol HTTP[10]. Povezavo aktiviramo preko vmesnika API, tako da pošljemo zahtevo za analizo posnetka, ki vključuje avdio datoteko. Strežnik nam nato vrne rezultat analize. HTTP protokol pozna več tipov zahtev: GET, POST, PUT, DELETE in druge. V tej aplikaciji bomo uporabili zahtevo tipa POST s tipom vsebine (angl. content type) multipart/form-data, saj s tem omogočimo prenos večjih datotek preko HTTP protokola. Specifikacija tipa vsebine multipart/form-data je na voljo na spletni strani skupnosti IETF (Internet Engineering Task Force)[11]. Za uspešno pošiljanje zahtev strežniku smo morali zahtevo generirati ročno, saj Android API ne pozna metode, ki bi samodejno generirala POST zahtevo s tipom vsebine multipart/form-data. Na sliki 5.4 je prikazana zahteva, ki jo generira naša aplikacija, in katero sprejme implementirani vmesnik.

5.2 Vmesnik API

Aplikacija sama po sebi ne omogoča analize zvoka in nima implementiranega klasifikatorja, zato je takšna za končnega uporabnika še neuporabna. Da bi to spremenili, smo morali razviti še lasten vmesnik API, ki je implementiran s pomočjo Python knjižnice Flask[12]. Flask je mikro ogrodje, s katerim lahko

[10] https://www.w3.org/Protocols/

[11] https://www.ietf.org/rfc/rfc2388.txt

[12] http://flask.pocoo.org

```
Content-Length: 158931
Accept-Encoding: gzip
Connection: Keep-Alive
Content-Type: multipart/form-data;boundary=*****
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2;
GT-I9195 Build/KOT49H)
Host: 192.168.1.254:5000
Cache-Control: no-cache

ImmutableMultiDict([('file', <FileStorage:
'1472128153129.yes.wav' ('audio/wav')>)])"
```

Slika 5.4: Prikaz POST zahteve, ki jo pošljemo vmesniku.

na zelo enostaven način implementiramo svoj vmesnik. Temelji na procesorju predlog Jinja2[13] ter knjižnici za delo z enostavnim vmesnikom med spletnim strežnikom in spletno aplikacijo Werkzeug[14]. Vmesnik smo implementirali tako, da odgovarja samo na POST zahteve, ki vsebujejo pravilno generirano zahtevo, kot je prikazano na sliki 5.4. Glavne specifikacije zahteve so:

- URL vmesnika: celoten naslov, kjer se bo vmesnik odzval na zahtevo. V našem primeru je to: http://<spletni naslov>:5000/uploadFile.
- tip zahteve: zahteva mora biti tipa POST s tipom vsebine multipart/data-form, ki omogoča pošiljanje datotek prek HTTP protokola.
- vrsta datoteke: vmesnik sprejme samo datoteke v formatu WAV.
- ime datoteke: ime datoteke mora biti zapisano v obliki: <poljubno ime datoteke>.<stanje bolnika>.wav (na primer testposnetek.no.wav). Poljubno ime datoteke je lahko poljubna kombinacija črk in številk in ne vpliva na nadaljnjo analizo. Pri stanju bolnika je lahko v uporabi ena

[13] http://jinja.pocoo.org/docs/dev/
[14] http://werkzeug.pocoo.org

izmed treh možnosti. Vmesnik sprejme možnosti yes, no in unknown (v slovenskem jeziku so ta stanja: ja, ne in ne vem). Ker vmesnik zahteva poimenovanje bolezenskih stanj v angleškem jeziku, bomo v nadaljevanju uporabili angleške različice. Stanje yes pomeni, da uporabnik že ve, da ima PB. Stanje no pomeni, da uporabnik nima PB. Stanje unknown pa pomeni, da uporabnik ne ve in bi rad s pomočjo analize izvedel, če že kaže znake bolezni.

Vse datoteke, ki so poslane na strežnik preko API zahteve, se shranijo in se po zaključku analize ne brišejo. Za ta korak smo se odločili, ker lahko naknadno izboljšamo klasifikator. V primeru, da se večje število bolnih in zdravih uporabnikov odloči uporabiti to aplikacijo, lahko te podatke uporabimo za izdelavo novega klasifikatorja, ki bo zajel več podatkov iz različnih virov. Za izboljšanje klasifikatorja se lahko uporabijo le datoteke, kjer je stanje bolnika yes ali no, ker pri teh vemo, v katero kategorijo spadajo. Datoteke s stanjem bolnika unknown se pri izdelavi novega klasifikatorja ne bi uporabile, saj je nejasno, v katero kategorijo bi jih lahko uvrstili. Vseeno pa jih lahko analiziramo, in sicer s pomočjo knjižnice za analizo zvoka. Po končani analizi posamezne datoteke nam knjižnica za analizo zvoka vrne atribute, ki jih sporočimo klasifikatorju. Ta nam lahko vrne vrednost 1 (datoteka vsebuje znake PB) ali pa vrednost 0 (datoteka ne vsebuje znakov PB). Dobljena vrednost se preko aplikacije kot rezultat pošlje uporabniku.

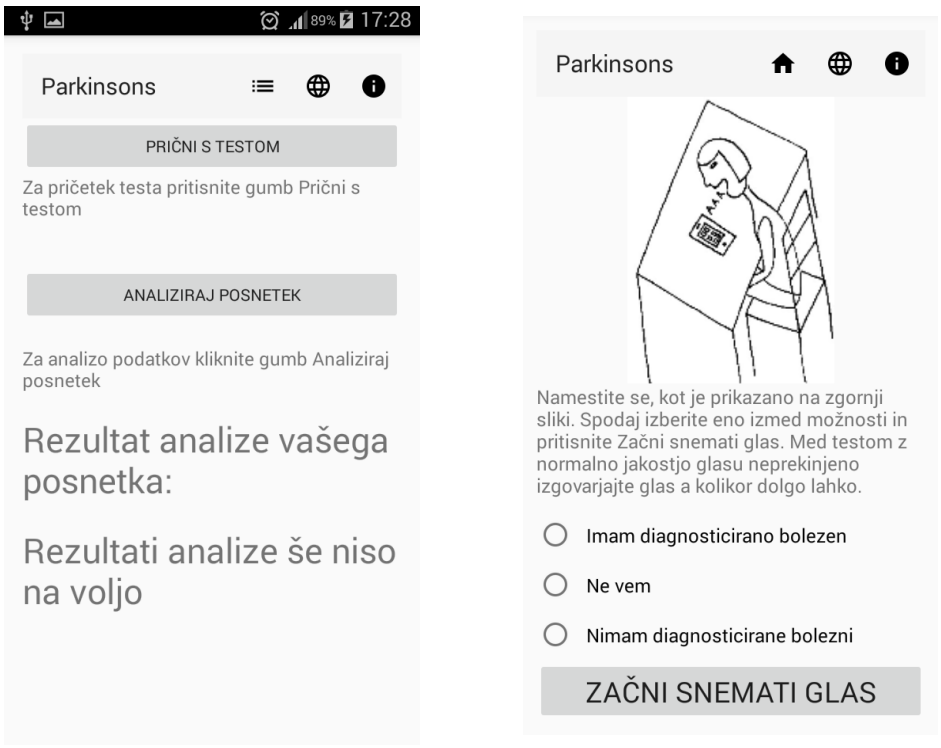
Za uporabo knjižnice za analizo zvoka je potrebno na strežnik namestiti tudi programski paket Octave, ki knjižnico poganja. Delovanje te knjižnice je podrobneje opisano v poglavju 3.2.

Poglavje 6

Primer uporabe

Mediji večkrat poročajo o pogostosti PB in njenih simptomih. Včasih se nam zato lahko zazdi, da imamo tudi sami katerega izmed naštetih bolezenskih znakov, ki spremljajo to zahrbtno bolezen. Zaradi bojazni pred PB in podatkom o pomembnosti hitre diagnoze se odločimo, da obiščemo osebnega zdravnika, ki bo potrdil ali ovrgel našo domnevo o prisotnih tipičnih bolezenskih znakih. Predvidevamo, da nas bo po potrebi napotil še k nevrologu na dodatna testiranja. Če osebni zdravnik teh simptomov ne bo opazil, nas bo vsaj potolažil in pojasnil, da se nam je vse skupaj le zdelo. Pred obiskom zdravnika se želimo na spletu vseeno še bolj natančno seznaniti z boleznijo, tam pa naletimo na aplikacijo, ki je bila narejena v tej diplomski nalogi. Aplikacija nam omogoči, da lahko še pred obiskom osebnega zdravnika preverimo, ali simptomi v resnici obstajajo ali pa so samo namišljeni.

Aplikacijo naložimo na svoj mobilni telefon in odpre se nam začetno okno, kot je prikazano na sliki 6.1a. Najprej se z aplikacijo malo spoznamo in pregledamo možnosti, ki so v orodni vrstici na vrhu zaslona. V primeru, da ne govorimo slovensko, lahko s klikom na ikono spremenimo jezik aplikacije v angleški. Ker znamo slovenski jezik, tega ne storimo. S klikom na ikono se nam prikažejo podatki o aplikaciji in kaj naj bi ta aplikacija omogočala, skupaj z opozorilom o zanesljivosti rezultata. Preverimo tudi zgodovino, ki je v našem primeru še prazna, saj smo aplikacijo šele namestili in še nismo posneli

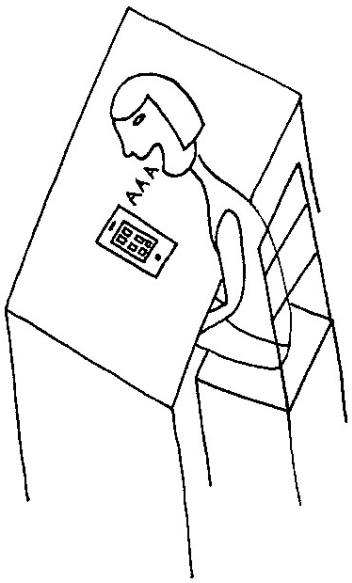


(a) (b)

Slika 6.1: (a) Začetna aktivnost. (b) Aktivnost, kjer posnamemo svoj glas.

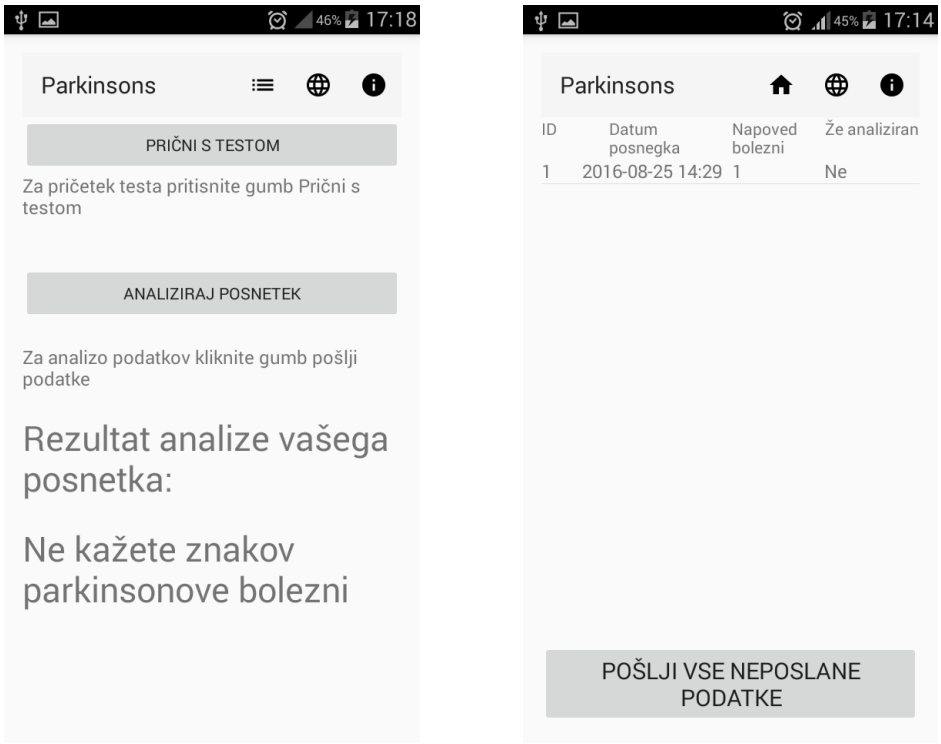
svojega glasu. Po vrnitvi v glavno aktivnost pričnemo s testom, ki nas najprej prestavi v aktivnost, v kateri bo mogoče posneti svoj glas. Ta aktivnost je prikazana na sliki 6.1b, kjer vidimo navodila za pravilno uporabo aplikacije. Z nepravilno uporabo namreč tvegamo tudi napačen odgovor strežnika. Pravilna uporaba je skicirana na sliki 6.2. Držimo se navodil in postavimo telefon na mizo pred sebe. Telefon naj bo z ekranom obrnjen proti stropu, mikrofon pa usmerimo proti sebi. Na zaslonu opazimo, da imamo na izbiro več možnosti, izbrati pa moramo eno izmed njih. Ker nismo prepričani v samodiagnozo, prav tako nam diagnoze za PB še ni postavil nevrolog, označimo opcijo ne vem in želim preveriti. Globoko zajamemo sapo in pritisnemo tipko Začni snemati glas. Z glasnostjo, kot jo imamo med pogovorom z drugimi osebami, izgovorimo črko a in ta ton držimo kolikor dolgo lahko. Ko nam zmanjka sape, stisnemo tipko Končaj snemanje, ki nas vrne v začetno ak-

tivnost. Tam lahko nato uporabimo možnost Analiziraj posnetek, ki pošlje nazadnje ustvarjeno datoteko na strežnik, kjer se analizira. V primeru, da na mobilnem telefonu nimamo aktivirane brezžične povezave, nas aplikacija o tem tudi obvesti.



Slika 6.2: Pravilna uporaba (Avtor: Petra Zupanc).

Po analizi in klasifikaciji glasu, ki lahko traja tudi do 2 minuti, se nam v začetni aktivnosti, kot je prikazano na sliki 6.3a, prikaže rezultat klasifikacije. Če sedaj kliknemo na ikono Zgodovina ☰, vidimo, da je ta posnetek zabeležen. Če nimamo na voljo brezžične povezave in se v tem času večkrat posnamemo, se posnetki shranijo v zgodovino, kot je prikazano na sliki 6.3b. Ko imamo omogočen dostop do spleta, lahko vse shranjene in neposlane posnetke pošljemo v analizo. Po končani analizi se rezultati posodobijo in jih lahko vidimo v meniju Zgodovina. Sedaj ko smo dobili rezultat in je bil naš glas klasificiran kot primer, ki ne kaže znakov PB, smo se pomirili in simptomi niso več prisotni.



(a) (b)

Slika 6.3: (a) Začetna aktivnost, ko dobi aplikacija rezultat analize. (b) Zgodovina posnetkov.

Poglavje 7

Zaključek

V diplomski nalogi smo uspešno implementirali mobilno aplikacijo, ki uporabniku omogoča snemanje glasu in pošiljanje glasu na strežnik, kjer se glas analizira in klasificira. Klasifikacija se izvaja s klasifikatorjem, ki smo ga tudi sami implementirali. Ker še ne obstaja strežnik, ki bi omogočal analizo, shranjevanje in klasifikacijo, smo morali implementirati tudi lasten strežnik, ki izračuna atribute glasu s pomočjo knjižnice za analizo zvoka in klasificira te atribute z našim klasifikatorjem. S podatki, ki so nam bili na voljo, smo naredili klasifikacijski model, ki pravilno klasificira 96.7 % učnih podatkov in ima vrednost AUC enako 94.6 %. Med izdelavo klasifikatorja smo se srečali s pomanjkanjem podatkov oziroma z močno neuravnoteženimi podatki glede na razred. Težavo smo rešili z uporabo nadvzorčenja skupaj s podvzorčenjem. Tako smo dobili uravnoteženo učno množico s podatki, ki smo jih obdelali z več algoritmi, implementiranimi v programskih knjižnicah Scikit learn, Keras in Theano. Knjižnici Keras in Theano smo potrebovali zaradi uporabe nevronskih mrež, ki so se na koncu izkazale za najboljši klasifikator. Pri uporabi metod za generiranje sintetičnih podatkov moramo biti zaradi uravnoteževanja pazljivi in natančno spremljati delovanje klasifikatorja na podatkih, poslanih z mobilnih naprav. Treba je priznati, da je naš klasifikator za splošno praktično uporabo verjetno premalo zanesljiv, ker je bila za učenje uporabljena podatkovna množica, v kateri je bilo bistveno premalo

podatkov brez PB. Kljub sintetičnemu algoritmičnemu uravnoteževanju učne množice je klasificiranje novih primerov brez PB premalo zanesljivo.

7.1 Izboljšave

Ker smo trenutno pri izdelavi učnega modela omejeni na podatke, ki smo jih našli na spletu, bi ob pogosti uporabi naše aplikacije lahko izdelali klasifikator izključno iz podatkov, ki so jih naložili uporabniki z uporabo aplikacije. Tako bi lahko uporabili druge metode za analiziranje zvočnega posnetka, ki izračunajo večji nabor atributov. Namesto trenutno uporabljene knjižnice bi lahko uporabili programski paket OpenSmile [10], ki bi ga lahko implementirali celo v aplikaciji na mobilnem telefonu. V tem primeru bi se računanje atributov iz posnetega glasu opravilo kar na mobilnem telefonu. Na strežnik bi se poslali samo relevantni atributi. S tem bi povečali število zahtev, ki jih lahko strežnik obdela in klasificira, uporabnik pa bi uporabil manjšo količino prenosa mobilnih podatkov, saj bi se velikost poslanih podatkov drastično zmanjšala. Prav tako bi lahko to aplikacijo vključili v že obstoječo aplikacijo ParkinsonCheck. Na ta način bi lahko še izboljšali zanesljivost klasifikacije uporabnika.

Literatura

[1] Dušan Andoljšek, Michael Reingold, Robert Berkow, Mark H. Beers, and Andrew J. Fletcher. *Veliki zdravstveni priročnik: za domačo uporabo: [najpopolnejši pregled medicinskih informacij]*. Ljubljana: Mladinska knjiga, 2005.

[2] Gustavo E.A.P.A. Batista, Ana L.C. Bazzan, and Maria Carolina Monard. Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18, 2003.

[3] Gustavo E.A.P.A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

[4] Rok Blagus and Lara Lusa. Joint use of over- and under-sampling techniques and cross-validation for the development and assessment of prediction models. *BMC bioinformatics*, 16:1–10, 2015.

[5] Paul Boersma and David Weenink. Praat: doing phonetics by computer, 2016. `http://www.praat.org/`. Uporabljena verzija 6.0.19.

[6] Nitesh V. Chawla. Data mining for imbalanced datasets: An overview. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 853–867. Springer US, Boston, MA, 2005.

[7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[9] John W. Eaton et al. GNU Octave, 2015. `http://www.octave.org`. Uporabljena verzija 4.0.2.

[10] Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 835–838, New York, NY, USA, 2013. ACM.

[11] Mireia Farrús, Javier Hernando, and Pascual Ejarque. Jitter and shimmer measurements for speaker recognition. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[12] Javier Gamboa, Félix Javier Jiménez-Jiménez, Alberto Nieto, Jose Montojo, Miguel Ortí-Pareja, José Antonio Molina, Esteban García-Albea, and Ignacio Cobeta. Acoustic voice analysis in patients with parkinson's disease treated with dopaminergic drugs. *Journal of Voice*, 11(3):314 – 320, 1997.

[13] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, June 2008.

[14] I. Kononenko and M.R. Šikonja. *Inteligentni sistemi*. Založba FE in FRI, 2010.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] M. Little, P. McSharry, I. Moroz, and S. Roberts. Nonlinear, biophysically-informed speech pathology detection. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 2, pages II–II, May 2006.

[17] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56(4):1015–1022, April 2009.

[18] Max A. Little, Patrick E. McSharry, Stephen J. Roberts, Declan A.E. Costello, and Irene M. Moroz. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1):23, 2007.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[21] Aleksander Sadikov, Vida Groznik, Jure Žabkar, Martin Možina, Dejan Georgiev, Zvezdan Pirtošek, and Ivan Bratko. Parkinsoncheck smart phone app. In *ECAI 2014*, volume 263, pages 1213–1214. IOS Press, 2014.

[22] Patricija Širca. Govor oseb s Parkinsonovo boleznijo: diplomsko delo, 2012. Univerza v Ljubljani, Pedagoška fakulteta.

[23] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[24] Ivan Tomek. Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772, Nov 1976.

[25] Klara Tostovršnik. Ko vas omreži Parkinson. *eSinapsa*, št. 7, 2014. Dostopno na `http://www.sinapsa.org/eSinapsa/stevilke/2014-7`.

[26] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig. Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests. *IEEE Transactions on Biomedical Engineering*, 57(4):884–893, April 2010.

[27] Athanasios Tsanas. *Accurate telemonitoring of Parkinson's disease symptom severity using nonlinear speech signal processing and statistical machine learning*. PhD thesis, University of Oxford, 2012.

[28] Athanasios Tsanas, Max A. Little, Patrick E. McSharry, and Lorraine O. Ramig. New nonlinear markers and insights into speech signal degradation for effective tracking of parkinson's disease symptom severity. In *International Symposium on Nonlinear Theory and its Applications (NOLTA), Krakow, Poland, 5-8 September*, pages 457–460, 2010.

[29] Athanasios Tsanas, Max A. Little, Patrick E. McSharry, and Lorraine O. Ramig. Nonlinear speech analysis algorithms mapped to a standard metric achieve clinically useful quantification of average parkinson's disease symptom severity. *Journal of The Royal Society Interface*, 2010.

[30] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3):408–421, July 1972.

[31] Andong Zhan, Max A. Little, Denzil A. Harris, Solomon O. Abiola, E. Dorsey, Suchi Saria, and Andreas Terzis. High frequency remote

monitoring of parkinson's disease via smartphone: Platform overview and medication response detection. *arXiv preprint arXiv:1601.00960*, 2016.