MODEL ZA PREPOZNAVANJE ČUSTVENIH STANJ NA OSNOVI MERITEV EEG

Interdisciplinarno

Raziskovalna naloga

Avtorja: Tadej Vobner
Luka Jeza

Mentor na šoli: doc. dr. Boris Zmazek, prof. kem
Zunanji mentor: doc. dr. Bernard Ženko, univ. dipl. inž. el

Ptuj, 2022

MODEL ZA PREPOZNAVANJE ČUSTVENIH STANJ NA OSNOVI MERITEV EEG

Interdisciplinarno

Raziskovalna naloga

Ptuj, 2022

ZAHVALA

Zahvaljujeva se doc. dr. Bernardu Ženku za pomoč in usmerjanje pri raziskovalnem delu.

KAZALO VSEBINE

KAZALO SLIK

KAZALO PREGLEDNIC

KAZALO ENAČB

ABSTRACT

In our research, we wanted to find out how people's emotions are related to EEG signals and to predict emotions based on these EEG signals. For this purpose, we used machine learning methods such as naive Bayes, kNN, decision trees, neural networks and deep neural networks. In our research, we found that models were most successful when taking into account the EEG of the individual (the accuracy of different models was between 29.3% and 54.5%). However, if we predict emotions of the individual whose training data was not included in the model learning, the resulting basic models are not significantly more accurate than random selection of the experienced emotion (the accuracy of different models was between 24% and 28.6%), the accuracy of the optimised model is 42.6%

POVZETEK

V svoji raziskovalni nalogi sva želela ugotoviti povezavo med čustvi ljudi in EEG signali ter na podlagi teh EEG signalov napovedovati čustva. V ta namen sva uporabila metode strojnega učenja, kot so naivni Bayes, k najbližjih sosedov, odločitvena drevesa, nevronske mreže in globoke nevronske mreže. V raziskovalni nalogi sva ugotovila, da je mogoče najuspešneje napovedati čustva za posameznike, z upoštevanjem EEG odtisa (točnost takšnih modelov je bila med 29.3 % in 54.5 %). Če pa smo napovedovali čustva posameznika, katerega učni podatki niso bili uporabljeni za učenje modela, se točnost osnovnih modelov bistveno ne razlikuje od naključnega izbora doživljanega čustva (točnost modelov je bila med 24 in 28.6 %), točnost optimiziranega modela pa je 42.6 %.

1 UVOD

Umetna inteligenca je področje računalništva in se ukvarja z razvojem metod in postopkov, ki so sposobni posnemati kognitivne funkcije, ki jih po navadi povezujemo s človeškim umom. Strojno učenje je najbolj znana veja umetne inteligence in se ukvarja s sistemi, ki se na podlagi podatkov učijo in tako izboljšajo svojo točnost pri napovedovanju izidov. Naučene modele lahko zapišemo z matematičnimi funkcijami različne kompleksnosti. Strojno učenje je najbolj uporabno pri modeliranju kompleksnih sistemov, pri katerih povezave med vhodi in izhodi niso poznane ali je njihova analiza matematično zamudna. Eden izmed takšnih kompleksnih sistemov so človekovi možgani in povezava čustev z elektromagnetnimi signali. Vemo, da čustva nastajajo v možganih in da kot posledica možganskih procesov nastajajo elektromagnetna valovanja, ki jih lahko, čeprav površno, izmerimo s pomočjo elektroencefalografije (EEG, elektrofiziološka metoda spremljanja).

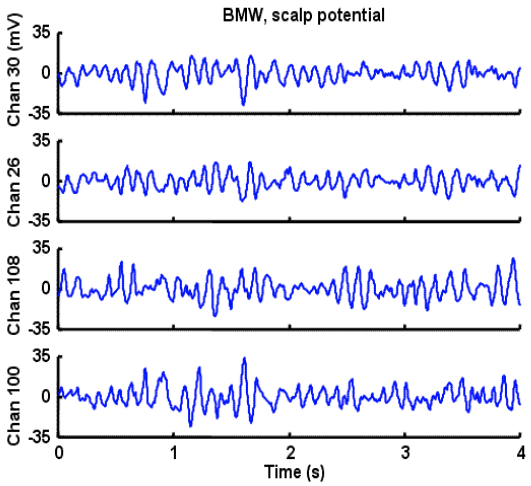
Za to tematiko raziskovalne naloge sva se odločila, ker verjameva, da je povezava med človekom in računalnikom na čustveni ravni eden izmed naslednjih pomembnih korakov pri komunikaciji med človekom in računalnikom in je nujno potrebna za tako imenovano potopitveno izkušnjo navidezne resničnosti.

Cilj naloge je pokazati, da obstaja povezava med EEG signali in čustvi, ki jih oseba doživlja, ter zgraditi in opisati modele, ki uspešno napovedujejo čustvena stanja iz podatkov EEG. Zraven tega želiva pokazati, da je takšen model uporaben za različne namene. Zastavila sva si raziskovalno vprašanje: “Ali lahko za osebo, ki ni bila vključena v nabor podatkov pri učenju modela, uspešno napovemo njeno čustveno stanje?” Za uspešno napovedovanje bova štela, če bo točnost napovednega modela večja od točnosti naključnega izbora, v našem primeru torej večja kot 20 % (1/5, ker imamo pet možnih čustvenih stanj). Najina hipoteza je, da bodo modeli, ki ne vključujejo osebe v učni množici podatkov, bistveno manj točni in da se elektromagnetni valovi pri istem čustvu pri različnih posameznikih nekoliko razlikujejo. Zato meniva, da bo potrebno model prilagoditi vsakemu posamezniku, da bi dobila zadovoljivo točnost.

2 TEORETIČNI DEL

2.1 EEG

Elektroencefalografija (kratica EEG) je proces merjenja možganske električne aktivnosti z elektrodami na površini glave skozi čas. EEG deluje kot okno v možgane, saj razkriva sinaptično delovanje, ki je zmerno do močno povezano s stanjem možganov. Na sliki 1 je prikazanih nekaj različnih meritev EEG pri budni in sproščeni osebi z zaprtimi očmi. Izmerjeni signali izvirajo iz zunanje plasti možganov (možganske skorje), ki je v veliki meri odgovorna za naše misli, čustva in vedenje. EEG in MEG (magnetoencefalografija) sta edini široko dostopni tehnologiji z zadostno časovno ločljivostjo za spremljanje teh hitrih dinamičnih sprememb. Njihova slabost pa je prostorska ločljivost, ki je v primerjavi s sodobnimi metodami strustrukturnega slikanja možganov, kot so računalniška tomografija (CT), pozitronska tomografija (PET) in slikanje z magnetno resonanco (MRI) slaba. [1]



Slika 1: Primer EEG posnetka (Brain Sciences Institute v Melbournu)

2.1.1 Unikatnost EEG signalov

Veliko raziskav kaže, da ima vsak posameznik svoj "EEG odtis", ter da lahko z visoko točnostjo identificiramo posameznika preko EEG signalov. Točnost takšnih pristopov je od 85 do 100 % [2]. Raziskave tudi kažejo, da se ti signali skozi srednja časovna obdobja ne spreminjajo. Poskusi s 15 udeleženci v več sejah, med njimi pa so bili narejeni časovni razmiki od nekaj dni do šestih mesecev, so pokazali, da lahko možganske dejavnosti posameznika ostanejo stabilne v relativno dolgem časovnem obdobju [3].

2.1.2 Povezava med EEG in nastankom čustev

Psihofiziološke raziskave kažejo, da obstaja neposredna povezava med količino delovanja v levem frontalnem režnju in desnem frontalnem režnju ter čustvi. Aktivnejšo levo frontalno področje kaže

na pozitiven odziv, bolj aktiven desni predel frontalnega režnja pa na negativni odziv [4]. To kaže na velik potencial za klasifikacijo čustev na podlagi EEG, vendar je mogoče razlikovati le med pozitivnimi in negativnimi čustvi. Pristop ne zagotavlja zadostne granularnosti, da bi bil uporaben za interakcijo med človekom in računalnikom. Zato je treba razviti tudi ustrezno metodo za predstavitev in razvrščanje čustev.[5]

2.1.3 Značilnosti EEG signalov

Signal EEG lahko zajema informacijo o delovanju možganov zato, ker ko se posamezni nevron sproži, pri tem pride do spremembe napetosti. Vhodni signal sproži prenos nekaterih natrijevih ionov v celico, kar povzroči dvig napetosti v celici (v primerjavi z zunanjostjo). Ko to povečanje napetosti doseže določen prag, se sproži akcijski potencial in hiter pritok natrijevih ter počasnejši odtok kalijevih ionov. Akcijski potencial je val električnega praznjenja, ki potuje po dendritih do sosednjih nevronov. Med tem dogodkom, ki traja le približno dve milisekundi, napetost preide iz mirovanja v napetost od -60 mV do +20 mV. Električna aktivnost, izmerjena s površinskimi elektrodami, predstavlja potenciale električnega polja, ki izhajajo iz kombinirane aktivnosti številnih nevronskih celic. Dejavnosti, ki so najjasneje vidne v EEG signalu, so aktivnosti, ki potekajo v nevronih možganske skorje (najbližje lobanji), ker je najbližje elektrodam. Globlje strukture, kot sta talamus in možgansko deblo, niso vidne neposredno. Uspešnost zaznave kortikalna aktivnost z EEG omejujejo tkiva in kosti med elektrodami na glavi in možgansko skorjo. Zaradi tega je amplituda EEG potencialov le nekaj mikrovoltov, čeprav to predstavlja seštevek aktivnosti številnih nevronov (s spremembami napetosti v milivoltih). Čeprav EEG ni točna meritev, pa še vedno omogoča pomemben vpogled v električno aktivnost možganske skorje. Frekvenca in amplituda sta značilnosti posnetih vzorcev EEG. Frekvenčno območje je običajno od 1 do 80 Hz (razdeljeno na pasove alfa, beta, itd.) z amplitudami od 10 do 100 μV. Opazovane frekvence so razdeljene na posebne skupine, saj določena frekvenčna območja bolj izražajo določena možganska stanja. Najpomembnejši so alfa (8-12 Hz) in beta (12-30 Hz) valovi. Alfa valovi so značilni za budno, vendar sproščeno duševno stanje in so najbolj vidni v parietalnem predelu in okcipitalnem režnju. Visoka aktivnost alfa je povezana tudi z neaktivnostjo možganov. Aktivnost beta je povezana z aktivnim stanjem duha, ki je najbolj izrazit v frontalni skorji in nad drugimi področji med intenzivno osredotočeno duševno dejavnostjo. [6]

2.2 Prejšnje raziskave

Pomembna je raziskava Hoekstre in Janssena, ki sta za analizo bio signalov uporabila napravo BraInquiry PET EEG [7]. Leta 2000 je Choppin z nevronskimi mrežami analiziral signale EEG in uporabil nevronske mreže, da bi jih razvrstil v šest čustev na podlagi čustvenih valenc in vzburjenosti, s 64-odstotno uspešnostjo [8]. Njegova magistrska naloga vsebuje dragocen vpogled v metodologijo in pomembne značilnosti EEG za prepoznavanje čustev. K. Takahashijev članek prikazuje predhodne rezultate za razvrščanje čustev na podlagi signalov iz treh elektrod [9].

Pomembno je, da so zaradi omejenega števila razpoložljivih elektrod vplivne značilnosti EEG zaznane med različnimi čustvenimi dražljaji : (I) Valenca: pozitivna čustva povzročajo večjo frontalno koherenco v alfa, višjo desno parietalno koherenco v alfa ter višjo desno parietalno koherenco v alfa in beta v primerjavi z negativnimi čustvi. (II) Vzburjenost: vzburjenje je predstavljalo višjo beta

koherenco v parietalnem režnju in nižjo alfa aktivnost. (III) Dominantnost: moč čustva se v EEG izrazi kot povečanje razmerj aktivnosti beta/alfa frekvenc v čelnem režnju ter povečanje beta aktivnost v parietalnem režnju. [5]

2.3 Nabor podatkov SEED-V

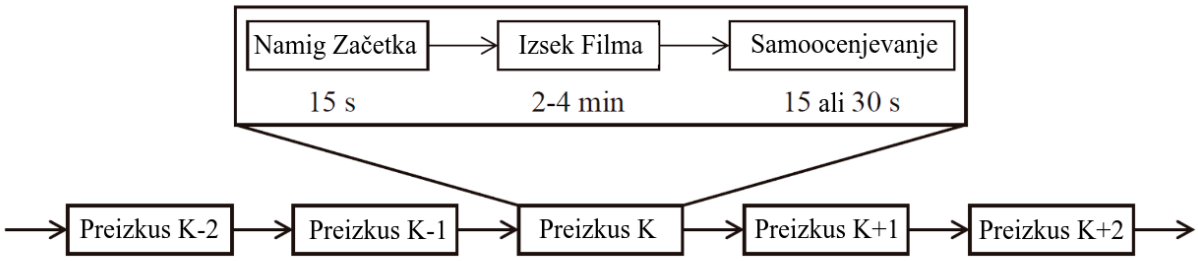
V raziskovalni nalogi sva uporabila zbirko podatkov SEED-V (kratica SJTU Emotion EEG Dataset V) [10].

2.3.1 Metoda vzbujanja čustev

Metoda vzbujanja čustev, uporabljena pri snemanju te podatkovne zbirke, je vzbujanje s stimulativnim slikovnim materialom. Z drugimi besedami, podatki so bili posneti med tem, ko oseba gleda določen čustveno stimulativen posnetek, ki naj bi vzbudil ustrezno čustveno stanje.

Da bi preučili stabilnost prepoznavanja čustev in zagotovili učinkovitost stimulacije, je moral vsak udeleženec v poskusu sodelovati trikrat, med poskusi pa so bili najmanj trije dnevi razmika. V vsakem poskusu so si morali udeleženci ogledati 15 delov stimulativnega gradiva, 3 dele za vsako vrsto čustva. Da bi zagotovili učinkovitost dražljaja in preprečili, da bi se preiskovanci dolgočasili, so bili materiali, ki so si jih preiskovanci ogledovali, vsakič popolnoma drugačni, skupni čas za ogled materialov v vsakem poskusu pa je bil omejen na približno 50 minut. V enem poskusu je udeleženec gledal enega od filmskih posnetkov, medtem ko so bili njegovi signali EEG zbrani s 62-kanalnim sistemom ESI NeuroScan[1].

Vsa spodbujevalna gradiva imajo pred predvajanjem 15 sekund časa za predstavitev ozadja gradiva in čustev, ki jih želijo vzbuditi. Po predvajanju spodbujevalnega gradiva sledi 15 sekund ali 30 sekund samoocenjevanja in počitka, odvisno od vrste gradiva. Če je bila vrsta dražljaja gnus ali strah, je bil čas počitka 30 sekund, čas za srečo, nevtralnost in žalost pa 15 sekund.



Slika 2: Potek ogleda posnetkov za čustveno vzburjanje (SEED V)

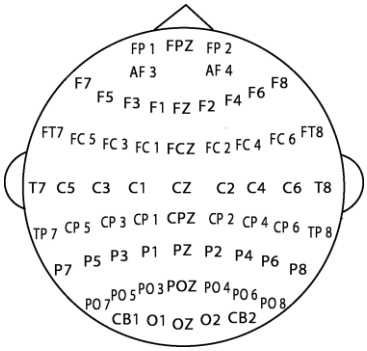
[1] Glej 2.3.3 Merjenje EEG signalov

2.3.2 Udeleženci

V tem poskusu so bili zbrani podatki za 16 oseb, od tega 6 moških in 10 žensk. Vsi udeleženci so bili študenti Univerze Jiao Tong v Šanghaju z normalnim sluhom ter vidom, dominantno desno roko in stabilnim duševnim stanjem. Bili so stari od 19 do 23 let.

2.3.3 Merjenje EEG signalov

V tem poskusu je merjenje EEG signalov potekalo z 62 kanali. Elektrode so označene z oznakami: črke ustrezajo kortikalnim lokacijam: frontalni (F), temporalni (T), parietalni (P), okcipitalni (O), centralni (C), cerebralni (CB) in njihove kombinacije za označevanje vmesnih lokacij (Fp pomeni frontalni polarni in ustreza sprednjemu pred frontalnem režnju). Številke elektrod označuje lateralnost: lihe številke so na levi, sode na desni, črka "Z" (kar pomeni nič) pa je na srednji črti. Na sliki 3 so prikazani vsi kanali. [11]



Slika 3: Imena kanalov pri snemanju EEG signala

2.3.3 Oznake čustev

Udeleženci so doživljali 5 osnovnih čustev, slednja so pa bila kronološko shranjena v vektorjih z enakimi časovnimi intervali kakor surovi podatki[2]. Posameznemu čustvu je prirejeno število, ki ga zaznamuje (preglednica 1). Vsak udeleženec je bil izpostavljen takšnem stimulativnem materialu, da je posamezno čustvo doživel 9 krat (preglednica 2). Za vsakega udeleženca imamo torej 45 meritev čustev.

Preglednica 1: Oznake čustev

Čustvo	**Predpisano število**
Gnus	0
Strah	1
Žalost	2
Nevtralno	3
Sreča	4

[2] Glej 2.2.5 Struktura surovih podatkov

Preglednica 2: Kronološka razvrstitev materiala po čustvenih karakteristikah

Preizkus						
1	Čustva: Seja 1	Strah	Nevtralno	Sreča	Žalost	Gnus
	Čustva: Seja 2	Strah	Nevtralno	Sreča	Gnus	Žalost
	Čustva: Seja 3	Strah	Nevtralno	Sreča	Gnus	Žalost
2	Čustva: Seja 1	Sreča	Strah	Nevtralno	Žalost	Gnus
	Čustva: Seja 2	Sreča	Gnus	Nevtralno	Žalost	Strah
	Čustva: Seja 3	Sreča	Gnus	Nevtralno	Žalost	Strah
3	Čustva: Seja 1	Sreča	Strah	Nevtralno	Žalost	Gnus
	Čustva: Seja 2	Nevtralno	Sreča	Strah	Žalost	Gnus
	Čustva: Seja 3	Nevtralno	Sreča	Strah	Žalost	Gnus

2.3.4 Struktura surovih podatkov

Surovi podatki posameznega udeleženca so razdeljeni na 62 kanalov, za vsak kanal je vsako milisekundo izmerjena električna aktivnost[3]. Dolžina posamezne matrike s podatki je (62, *d*), kjer *d* predstavlja dolžino celotnega procesa pridobivanja podatkov v milisekundah. Za uporabo je torej potrebno iz matrike izluščiti zgolj podatke, ki so bili pridobljeni med predvajanjem stimulativnega materiala.

Celotna količina posnetih podatkov je zelo velika (skupaj 40,5 GB), preden pa jih lahko uporabimo za učenje napovednih modelov s strojnim učenjem, moramo iz surovih podatkov izračunati značilke. S tem tudi radikalno zmanjšamo količino podatkov (na 73 MB; 0,17 % velikosti surovih podatkov).

2.4 Izražanje značilk iz surovih podatkov

Izražanje značilk je del postopka zmanjševanja razsežnosti, pri katerem se začetni niz neobdelanih podatkov razdeli in zmanjša na lažje obvladljive skupine. Najpomembnejša značilnost velikih zbirk podatkov je, da imajo veliko število spremenljivk. Izražanje značilk pomaga pridobiti najboljše značilnosti iz velikih podatkovnih nizov z izbiro in združevanjem spremenljivk v značilnosti, s čimer se učinkovito zmanjša količina podatkov. Te značilke so enostavne za obdelavo in lahko še vedno natančno kot tudi izvirno opišejo dejanski nabor podatkov.

Postopek je uporaben, kadar je treba zmanjšati število virov, potrebnih za obdelavo, ne da bi pri tem izgubili pomembne ali relevantne informacije. Z izražanjem značilk lahko tudi zmanjšamo količino

[3] Glej poglavje 2.3.3 EEG

odvečnih podatkov za določeno analizo. Poleg tega zmanjšanje števila podatkov poveča hitrost učenja in posploševanja v procesu strojnega učenja. [12]

2.4.1 Značilke SEED V podatkov

Za obdelavo signalov EEG so bili neobdelani podatki EEG najprej zmanjšani na frekvenco vzorčenja 250 Hz. Za filtriranje šuma in odstranitev artefaktov so bili podatki EEG nato obdelani s filtrom pasovne prepustnosti med 1 Hz in 75 Hz. V vsakem segmentu so bile izluščene značilnosti diferencialne entropije (DE) v petih frekvenčnih pasovih[11]:

1) delta: 1 ~ 4 Hz;

2) theta: 4 ~ 8 Hz;

3) alfa: 8 ~ 14 Hz;

4) beta: 14 ~ 31 Hz;

5) gama: 31 ~ 50 Hz.

$$DE = -\int_{-\infty}^{\infty} P(x)\, ln\, ln\left(P(x)\right) dx(x))dx$$

Predpostavljamo, da signali EEG ustrezajo normalni porazdelitvi: $x \sim N(\mu, \sigma^2)$. Naj bo $g(x)$ Gaussova porazdelitvena funkcija verjetnosti s srednjo vrednostjo μ in standardnim odklonom σ, $f(x)$ pa poljubna Gaussova porazdelitvena funkcija z enakim standardnim odklonom. Ker je diferencialna entropija translacijsko invariantna, lahko vedno predpostavimo, da ima $f(x)$ enako povprečno vrednost μ kot $g(x)$. Upoštevajmo še Kullback-Leiblerjevo divergenco med obema porazdelitvama. Nato je mogoče poenostaviti izračun značilk DE:

$$DE = -(\int_{-\infty}^{\infty} f(x)ln(g(x))dx) = -(\int_{-\infty}^{\infty} (\frac{1}{\sqrt{2\pi\sigma}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})})ln(\frac{1}{\sqrt{2\pi\sigma}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})})dx)$$

$$DE = -(\int_{-\infty}^{\infty} (\frac{1}{\sqrt{2\pi\sigma}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})})ln(\frac{1}{\sqrt{2\pi\sigma}})dx + ln(e)\int_{-\infty}^{\infty} (\frac{1}{\sqrt{2\pi\sigma}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})})(-\frac{(x-\mu)^2}{2\sigma^2})dx))$$

$$DE = -(-\frac{1}{2} ln(2\pi\sigma^2) - ln(e)\frac{\sigma^2}{2\sigma^2}) = \frac{1}{2} ln(2\pi\sigma^2) + ln(e)\frac{1}{2}$$

$$DE = \frac{1}{2} ln\,(2\pi e\sigma^2)$$

2.4.2 Struktura podatkov po izračunavi značilk diferencialne entropije

Struktura podatkov po izračunavi značilk je matrika velikosti (310 x 29168), torej 310 kanalov (62 kanalov, vsak razdeljen na 5 frekvenčnih pasov) in 29168 različnih vnosov povezanih z različnimi čustvi[4].

[4] Glej poglavje 2.3.3 Merjenje EEG signalov

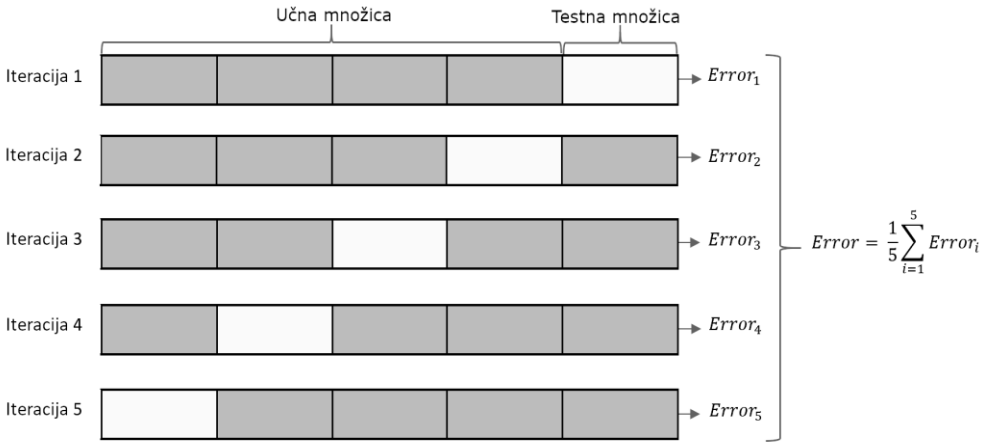
3 METODOLOGIJA

3.1 Metode

Za raziskovanje in grajenje modelov sva uporabljala orodje VS Code, zgoraj opisano množico podatkov SEED-V[5] in različne knjižice za strojno učenje, kot so Tensorflow in Scikit learn. Za učinkovito sodelovanje in programiranje sva uporabljala orodji Git in Github, na slednjem sva tudi ustvarila repozitorij. Vse zgrajene napovedne modele sva ovrednotila s postopkom 10-kratnega prečnega preverjanja.

3.1.1 Prečno preverjanje

Pri prečnem preverjanju se učni podatki razdelijo na k enakih podmnožic, na k-1 podmnožicah se model uči, na preostalem pa se testira. Postopek se ponovi k-krat in na koncu dobimo napovedi za celotno učno množico (slika 4). Stratificirano prečno preverjanje izvedemo tako, da hkrati poskrbimo, da je v vsaki podmnožici približno enak delež razredov kot v celotni učni množici [13].



Slika 4: Prečno preverjanje

3.1.2 Scikit learn

Scikit-learn je odprtokodna programska knjižnica za strojno učenje v programskem jeziku Python. Vsebuje različne algoritme za klasifikacijo, regresijo in razvrščanje v podskupine, vključno z metodo podpornih vektorjev, naključnimi gozdovi, ansambelskimi metodami, metodama za razvrščanje k-means in DBSCAN ter drugimi. Zasnovan je na osnovi numeričnih in znanstvenih knjižnic NumPy in SciPy [14].

3.1.3 GitHub

GitHub je ponudnik internetnega gostovanja za razvoj programske opreme in nadzor različic z uporabo sistema Git. Ponuja funkcionalnost porazdeljenega nadzora različic in upravljanja izvorne

[5] Opisan v poglavju 2.3 Nabor podatkov SEED-V

kode (SCM) sistema Git ter lastne funkcije. Za projekt zagotavlja nadzor dostopa in več funkcij za sodelovanje, kot so sledenje hroščem, zahteve za funkcije, upravljanje nalog, neprekinjena integracija in podobno [15].

3.1.4 Tensorflow

TensorFlow je celovita odprtokodna platforma za strojno učenje. Ima celovit in prilagodljiv ekosistem orodij, knjižnic in virov skupnosti, ki raziskovalcem omogoča napredovanje na področju strojnega učenja, razvijalcem pa enostavno izdelavo in uporabo aplikacij, ki temeljijo na strojnem učenju [16].

3.2 Metode strojnega učenja

Poznamo različne metode strojnega učenja, ki se razlikujejo po razumljivosti, točnosti naučenih modelov in časovni zahtevnosti. Sprva sva izbrala metode Naive Bayes, kNN, odločitvena drevesa in nevronsko mrežo. Prva težava, na katero naletimo, je priprava učne množice, ki naj bi se v metodi prečnega preverjanja spreminjala, hkrati pa ne želimo spremeniti časovnega zaporedja podatkov, to je rešeno že s samim izražanjem značilk, saj so izražene tako, da se ohranja časovna vrsta dolga 4 sekunde.[6]

3.2.1 kNN

Metoda k najbližjih sosedov (kNN) je preprost algoritem strojnega učenja, pogosto uporabljan za naloge klasifikacije in regresije. K predstavlja (po navadi liho) število najbližjih sosedov, ki delujejo kot glavni odločujoči faktor za napoved ciljne vrednosti nove podatkovne točke. Da izračunamo napoved novega primera, je potrebna celotna učna množica, zato potrebuje veliko spomina, ker učenja pravzaprav ni.

3.2.2 Nevronska mreža

Nevronska mreža je metoda, ki temelji na množici med sabo povezanih vozlišč, ki med učenjem avtomatsko izluščijo pomembne značilnosti učne množice za klasifikacijo. Točnost nevronskih mrež je po navadi odvisna od njihove velikosti, z velikostjo pa hkrati narašča tudi časovna zahtevnost učenja.

3.2.3 Odločitvena drevesa

Odločitvena drevesa so eden od pristopov napovednega modeliranja, ki se uporablja v statistiki, podatkovnem rudarjenju in strojnem učenju. V notranjih vozliščih drevesa se nahajajo logični pogoji glede vrednosti vhodnih spremenljivk, v listih pa so vrednosti ciljne spremenljivke. Ciljno vrednost novega primera dobimo tako, da v drevesu sledimo logičnim pogojem od korena drevesa do listov. Drevesni modeli, pri katerih lahko ciljna spremenljivka zavzame diskretno množico vrednosti, se imenujejo klasifikacijska drevesa; v teh drevesnih strukturah listi predstavljajo oznake razredov, veje

[6] Opisana v poglavju 3.2.2 Struktura surovih podatkov.

pa povezave funkcij, ki vodijo do teh oznak razredov. Odločitvena drevesa so zaradi svoje razumljivosti in preprostosti med najbolj priljubljenimi algoritmi strojnega učenja. [17]

3.2.4 Naive Bayes

Metoda naivni Bayes nosi ime po statistiku Thomasu Bayesu in je verjetnostna tehnika strojnega učenja. Metodo sva izbrala, saj je pogosto uporabljena za klasifikacijo in nam pogosto da dokaj dobre rezultate.

3.3 Grajenje osnovnih modelov s knjižnico Scikit learn

Osnovni modele sva zgradila s knjižico Scikit learn. Odločila sva se uporabiti:

I. Metodo k-nearest neighbours: število sosedov je 10, uporabljena je evklidska metrika, uteži najbližjih sosedov pa so enakomerne.
II. Nevronsko mrežo z enim slojem s 100 nevroni (poleg vhodnega in izhodnega sloja), aktivacijska funkcija je tanh, uporabljen je optimizacijski algoritem Adam, ki se lahko uporablja namesto klasičnega stohastičnega postopka gradientnega spuščanja za iterativno posodabljanje uteži omrežja na podlagi podatkov o usposabljanju [17].
III. Binarno odločitveno drevo, minimalno število primerkov na listih je 2, podmnožic manjših od 5 ne razdelimo, največja globina drevesa je 100, učenje drevesa pa se ustavi, ko doseže točnost 95 %.
IV. Metodo naivni bayes nima posebnih parametrov, ki bi jih lahko spreminjali.

3 REZULTATI IN RAZPRAVA

4.1 Modeli za napovedovanje čustev izbrane osebe, testirani s prečnim preverjanjem

Posamezni model je naučen in testiran na podatkih posamezne osebe. Točnost ocenimo s stratificiranim desetkratnim prečnim preverjanjem[7]. Gradimo modele: naivni Bayes, kNN, nevronske mreže in odločitvena drevesa.

Preglednica 3: Točnost modelov za posamezno osebo (izraženo v odstotkih)

	kNN - CA	**NM - CA**	**Drevo- CA**	**NB - CA**
Povprečje	88,0	84,8	84,3	62,5

V preglednici 3 vidimo, da sva dobila modele z relativno visoko točnostjo. V tem primeru je število podatkov, porabljenih za učenje in testiranje modelov, med 1800 in 2000. Pri tem primeru sva dobila najvišjo točnost, za ostale lahko pričakujemo nižje točnosti, ker bodo modeli morali upoštevati tudi razlike med posameznimi osebami.

4.2 Enotni model za napovedovanje čustev poljubne osebe, testiran s prečnim preverjanjem

V tem razdelku smo izvedli učenje modelov in testiranje na enak način kot zgoraj, le da ne ločujemo med podatki različnih oseb in podatke vseh oseb združimo v eno učno množico, na kateri se naučimo skupni model za vse osebe.

Preglednica 4: Točnost enotnih modelov, ocenjena z desetkratnim prečnim preverjanjem (izraženo v odstotkih)

kNN - CA	**NM - CA**	**Drevo - CA**	**NB - CA**
33,1	30,2	28,1	24,3

Podatki v preglednici 4 nakazujejo, da se je točnost občutno zmanjšala pri vseh vrstah modelov. Zmanjšanje točnosti v primerjavi z modeli v prejšnjem podrazdelku je pričakovano, saj je zaradi razlik med osebami naloga, ki jo zdaj rešujemo, bistveno težja[8].

[7] Glej poglavje 3.1.1

[8] Glej poglavje 2.1.1

4.3 Enotni model za napovedovanje čustev poljubne osebe, testirane s prečnim preverjanjem, pri katerem vsakič izpustimo eno osebo

Preglednica 5: Točnost enotnih modelov, ocenjena s prečnim preverjanjem, pri katerem vsakič izpustimo eno osebo (izraženo v odstotkih).

	kNN - CA	**NN - CA**	**Drevo- CA**	**NB - CA**
Povprečje	25,4	28,6	24,0	24,4

Pri tako ovrednotenih modelih smo pričakovali najnižje točnosti, saj je naloga, ki jo rešujemo, najtežja. Takšno vrednotenje namreč posnema uporabo modela, ki se ga naučimo na učnih podatkih, uporabljamo pa ga na osebah, ki niso bile vključene v učne podatke. Vsi modeli v povprečju presegajo točnost naključne izbire.

4.4 Preizkus z upoštevanjem EEG odtisa

Prečno preverjanje z desetimi stratificiranimi pregibi, grajeno na k-1 osebkih in prvih petih čustev preostalega osebka, ki so pomnoženi tolikokrat (16), da zadostujejo za 10 % vseh vnosov. Tako dobimo model, pri katerem upoštevamo unikatnost EEG pri različnih posameznikih. Model nato testiramo na k-tem udeležencu.

Preglednica 6: Točnost pri različnih modelih (izraženo v odstotkih)

	kNN - CA	**NN - CA**	**Tree - CA**	**NB - CA**
Povprečje	42,4	45,1	32,1	29,3

Rezultat, ki sva ga dobila, je fascinanten, saj utemeljuje teorijo unikatnosti signalov, EEG točnost modelov je znatno višja. Ne presegajo pa rezultatov, ki sva jih dobila v preglednici 3. Na primer pri modelih kNN in nevronska mreža se je točnost povečala za kar 17 odstotkov v primerjavi z modelom, pri katerem nisva upoštevala unikatnost EEG signalov.

4.5 Model, zgrajen na petih čustvih osebka, preizkušen na vseh ostalih

V preglednici 7 opazimo, da je natančnost modelov razmeroma visoka, vendar je težava zelo majhen nabor podatkov pri grajenju modela, kar povzroča visoko varianco točnosti. V primerjavi z natančnostjo v preglednici 6 so precej nižje in standardni odklon med podatki je precej višji. V primerjavi z točnostmi v preglednici 3 so pričakovano točnosti manjše, saj je na voljo veliko manj učnih podatkov. Vsi modeli so bili uspešni.

Preglednica 7: Točnost pri različnih modelih (izraženo v odstotkih)

	kNN - CA	NN - CA	Tree - CA	NB - CA
Povprečje	42,2	36,1	29,1	37,7

4.6 Optimiziran model

Za večjo točnost sva optimizirala model nevronske mreže, in sicer tako, da sva ga razširila na globoko nevronsko mrežo s 4 skritimi plastmi, v posamezni pa 64 nevronov z aktivacijsko funkcijo relu. Model je grajen s knjižico tensorflow[9]. Nevroni zadnje, izhodne plasti pa imajo 5 nevronov. Pred treniranjem modela sva značilke skalirala s standardnim skaliranjem. Tako sva dosegla pri prečnem preverjanju z desetimi straticificiranimi pregibi na vseh podatkih točnost 92.5 %. Pri grajenju modela na k-1 osebkih in testiranje na preostalem osebku sva dobila točnost modela 42.6 %. Pri upoštevanju EEG odtisa pa dobimo točnost 54.5 %. Model bi lahko izboljšali z večjim naborom podatkov. Prav tako bi lahko model izboljšali z uporabo izboljšanega postopka za učenje nevronske mreže, s pomočjo katere bi preprečila preveliko prilagoditev učnim podatkom.

[9] Opisana v poglavju 3.1.4

4 ZAKLJUČEK

Pri raziskovanju, ali so modeli, ki uporabljajo strojno učenje, zmožni uspešno napovedovati čustva, sva ugotovila, da so preprosti modeli zmožni določiti doživljeno čustvo bolje kot naključna izbira. Prav tako sva ugotovila, da je zaradi razlik med posameznimi osebami lažje zgraditi model za določeno osebo, kot pa model, ki napoveduje čustva za poljubno osebo. Najuspešnejša enostavna modela sta kNN in nevronska mreža. Model kNN je veliko hitrejši, saj ne potrebuje faze učenja ter potrebuje manj računalniških virov, medtem ko je nevronska mreža počasnejša, predvsem pri učni fazi, hkrati pa potrebuje veliko računalniških virov. Edina težava modela kNN je, da ga je težko optimizirati in izboljšati. Zato sva za optimizacijo izbrala nevronsko mrežo. Ugotovila sva, da lahko optimiziran model v več kot polovici primerov pravilno določi občuteno čustvo. Za pridobivanje le-tega je potrebnih približno 9 minut merjenja EEG signalov posameznika. Izmerjene EEG podatke v obliki časovne vrste sva z izračunom diferencialne entropije pretvorila v značilke, ki zajemajo podatke iz 4 sekund meritev. Značilke bi lahko izračunala tudi drugače, kar bi morda pozitivno vplivalo na točnost modela. Verjetno bi lahko bolj točne modele dobila z uporabo rekurzivnih nevronskih mrež. Zaradi najinega laičnega znanja so modeli osnovni in nedovršeni. Hkrati bi točnost modelov povečal večji nabor podatkov ter upoštevanje dejstva, da se čustva ne spreminjajo na kratkih časovnih intervalih. Model je primeren za pomoč tistim, ki svoja čustva težje izražajo. Odgovor na raziskovalno vprašanje: “Ali lahko za osebo, ki ni bila vključena v nabor podatkov pri učenju modela, uspešno napovemo njeno čustveno stanje?” je da, vendar pa je točnost bistveno manjša, kot če imamo za to osebo na voljo ustrezne učne podatke.

5 LITERATURE

[1] Paul L. Nunez and Ramesh Srinivasan (2007), Scholarpedia, 2(2):1348.

[2] Mohammadi, Parisa Shoushtari, Behnam Molaee Ardekani and Mohammad B. Shamsollahi: Person Identification by Using AR Model for EEG Signals Gelareh

[3] Maria V. Ruiz Blondet, Sarah Laszlo, Zhanpeng Jin: Assessment of Permanence of Non-volitional EEG Brainwaves as a Biometric

[4] C. P. Niemic. Studies of emotion: A theoretical and empirical review of psychophysiological studies of emotion. Journal of Undergraduate Research, 1:15–18, 2002

[5] Danny Plass-Oude Bos: EEG-based emotion recognition

[6] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. Principles of Neural Science. Mc Graw Hill, 2000.

[7] A. Hoekstra and J. Janssen. Linking bio-signals to transfer of knowledge: Towards mind-reading ecas? Capita Selecta, 2006.

[8] A. Choppin. Eeg-based human interface for disabled individuals: Emotion expression with neural networks.

[9] K. Takahashi. Remarks on emotion recognition from bio-potential signals. 2nd International Conference on Autonomous Robots and Agents, pages 186–191, 2004.

[10] Wei Liu, Jie-Lin Qiu, Wei-Long Zheng and Bao-Liang Lu, Comparing Recognition Performance and Robustness of Multimodal Deep Learning Models for Multimodal Emotion Recognition, IEEE Transactions on Cognitive and Developmental Systems, 2021.

[11] Podjetje gtec medical engineering GmbH

[12] What is Feature Extraction? Feature Extraction in Image Processing; By Sampriti Chatterjee -Oct 29, 2020

[13] Distribution-Balanced Stratied Cross-Validation for Accuracy Estimation Xinchuan Zeng and Tony R. Martinez Computer Science Department, Brigham Young University, Provo, Utah

[14] The scikit-learn Open Source Project on Open Hub: Languages Page. Open Hub. Dostopano dne 14 Julij 2018.

[15] GitHub: Where the world builds software · GitHub." https://github.com Dostopano dne 13 dec. 2021

[16] "TensorFlow" https://www.tensorflow.org/ Dostopano dne 10 feb. 2022

[17] Diederik P. Kingma, Jimmy Ba Adam: A Method for Stochastic Optimization

.

6 PRILOGE

```
-------------------------------------------------------------------------
-

npz_to_csv.py
-------------------------------------------------------------------------import numpy as
np
import pickle

for i in range(1,17):
    with open("csvData/participant{}".format(i) + ".csv", "a") as f:
        data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(i))
        data = pickle.loads(data_npz['data'])
        labels = pickle.loads(data_npz['label'])
        for key, value in data.items():
            # ustvarimo vektor (value) ki hrani podatke značilk
            # nato vektor (arr) ki hrani labele
            arr = np.array(labels[key])
            # reshaping arr into a vertical vector
            arr_reshaped = np.reshape(arr,(len(arr),1))
            solution = np.concatenate((value,arr_reshaped), axis=1)
            np.savetxt(f, solution)

with open("csvData/participant_all" + ".csv", "a") as f:
    for i in range(1,17):
        data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(i))
        data = pickle.loads(data_npz['data'])
        labels = pickle.loads(data_npz['label'])
        for key, value in data.items():
            # ustvarimo vektor (value) ki hrani podatke značilk
            # nato vektor (arr) ki hrani labele
            arr = np.array(labels[key])
            # reshaping arr into a vertical vector
            arr_reshaped = np.reshape(arr,(len(arr),1))
            # merging the two vectors together
            solution = np.concatenate((value,arr_reshaped), axis=1)
            np.savetxt(f, solution)
for b in range(1,17):
    with open("csvData/participant_all-{}".format(b) + ".csv", "a") as f:
        for i in range(1,b):
            data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(i))
            data = pickle.loads(data_npz['data'])
            labels = pickle.loads(data_npz['label'])
            for key, value in data.items():
                # ustvarimo vektor (value) ki hrani podatke značilk
                # nato vektor (arr) ki hrani labele
                arr = np.array(labels[key])
                # reshaping arr into a vertical vector
                arr_reshaped = np.reshape(arr,(len(arr),1))
                # merging the two vectors together
                solution = np.concatenate((value,arr_reshaped), axis=1)
                np.savetxt(f, solution)
        for i in range(b+1,17):
            data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(i))
            data = pickle.loads(data_npz['data'])
            labels = pickle.loads(data_npz['label'])
            for key, value in data.items():
                # ustvarimo vektor (value) ki hrani podatke značilk
                # nato vektor (arr) ki hrani labele
                arr = np.array(labels[key])
                # reshaping arr into a vertical vector
```

```
                arr_reshaped = np.reshape(arr,(len(arr),1))
                # merging the two vectors together
                solution = np.concatenate((value,arr_reshaped), axis=1)
                np.savetxt(f, solution)



for b in range(1,17):
    with                open("csvData/participant_all-k+first5x16/participant_all-
{}+first5x16".format(b) + ".csv", "a") as f:
        for i in range(1,b):
            data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(i))
            data = pickle.loads(data_npz['data'])
            labels = pickle.loads(data_npz['label'])
            for key, value in data.items():
            # ustvarimo vektor (value) ki hrani podatke značilk
            # nato vektor (arr) ki hrani labele
                arr = np.array(labels[key])
                # reshaping arr into a vertical vector
                arr_reshaped = np.reshape(arr,(len(arr),1))
                # merging the two vectors together
                solution = np.concatenate((value,arr_reshaped), axis=1)
                np.savetxt(f, solution)
        for i in range(b+1,17):
            data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(i))
            data = pickle.loads(data_npz['data'])
            labels = pickle.loads(data_npz['label'])
            for key, value in data.items():
                # first creating a vector (value) wich cointains all features data
                # then creating a another vector(arr) wich cointains the labels
                arr = np.array(labels[key])
                # reshaping arr into a vertical vector
                arr_reshaped = np.reshape(arr,(len(arr),1))
                # merging the two vectors together
                solution = np.concatenate((value,arr_reshaped), axis=1)
                np.savetxt(f, solution)
        data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(b))
        data = pickle.loads(data_npz['data'])
        labels = pickle.loads(data_npz['label'])
        for b in range(1,17):
             for i in range(0,5):
                # first creating a vector (value) wich cointains all features data
                # then creating a another vector(arr) wich cointains the labels
                arr = np.array(labels[i])
                # reshaping arr into a vertical vector
                arr_reshaped = np.reshape(arr,(len(arr),1))
                # merging the two vectors together
                solution = np.concatenate((data[i],arr_reshaped), axis=1)
                np.savetxt(f, solution)

for b in range(1,17):
    with                                              open("csvData/participant_all-
k+first5x16/participant{}first5x16".format(b) + ".csv", "a") as f:

        data_npz = np.load("Data/EEG_DE_features/{}_123.npz".format(b))
        data = pickle.loads(data_npz['data'])
        labels = pickle.loads(data_npz['label'])

        for b in range(1,17):
             for i in range(0,5):
            # ustvarimo vektor (value) ki hrani podatke značilk
            # nato vektor (arr) ki hrani labele
                arr = np.array(labels[i])
                # reshaping arr into a vertical vector
                arr_reshaped = np.reshape(arr,(len(arr),1))
```

```
                # merging the two vectors together
                solution = np.concatenate((data[i],arr_reshaped), axis=1)
                np.savetxt(f, solution
-----------------------------------------------------------------
model alldata.ipynb
-----------------------------------------------------------------

import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold

from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier

KNNmodelaccuracies = []
n_splits = 10

accuracy = []
dataset = pd.read_csv("csvData/allparticipants.csv",delimiter=' ',header=None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

skf = StratifiedKFold(n_splits=n_splits, random_state=None)
skf.get_n_splits(X, y)

for train_index, test_index in skf.split(X, y):
    # print("TRAIN:", train_index, "TEST:", test_index)
    X1_train, X1_test = X[train_index], X[test_index]
    y1_train, y1_test = y[train_index], y[test_index]
    clf = KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p = 2)
    clf.fit(X1_train, y1_train)
    prediction = clf.predict(X1_test)
    score = accuracy_score(prediction,y1_test)
    accuracy.append(score)

KNNmodelaccuracies.append(accuracy)
accuracy = np.array(accuracy)
print(" %0.2f accuracy z standardno deviacijo %0.2f" % (accuracy.mean(),
accuracy.std()))

NBmodelaccuracies = []
n_splits = 10

accuracy = []
dataset = pd.read_csv("csvData/allparticipants.csv",delimiter=' ',header=None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

skf = StratifiedKFold(n_splits=n_splits, random_state=None)
skf.get_n_splits(X, y)

for train_index, test_index in skf.split(X, y):
    # print("TRAIN:", train_index, "TEST:", test_index)
    X1_train, X1_test = X[train_index], X[test_index]
    y1_train, y1_test = y[train_index], y[test_index]
    clf = GaussianNB()
    clf.fit(X1_train, y1_train)
    prediction = clf.predict(X1_test)
    score = accuracy_score(prediction,y1_test)
    accuracy.append(score)
```

```
NBmodelaccuracies.append(accuracy)
accuracy = np.array(accuracy)
print("%0.2f accuracy z standardno deviacijo %0.2f" % (accuracy.mean(),
accuracy.std()))


print(NBmodelaccuracies)

Treemodelaccuracies = []
n_splits = 10

accuracy = []
dataset = pd.read_csv("csvData/allparticipants.csv",delimiter=' ', header=None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

skf = StratifiedKFold(n_splits=n_splits, random_state=None)
skf.get_n_splits(X, y)

for train_index, test_index in skf.split(X, y):
    # print("TRAIN:", train_index, "TEST:", test_index)
    X1_train, X1_test = X[train_index], X[test_index]
    y1_train, y1_test = y[train_index], y[test_index]
    clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
    clf.fit(X1_train, y1_train)
    prediction = clf.predict(X1_test)
    score = accuracy_score(prediction, y1_test)
    accuracy.append(score)

Treemodelaccuracies.append(accuracy)
accuracy = np.array(accuracy)
print("%0.2f accuracy z standardno deviacijo %0.2f" % (accuracy.mean(),
accuracy.std()))

print(Treemodelaccuracies)

NNmodelaccuracies = []
n_splits = 10

accuracy = []
dataset = pd.read_csv("csvData/allparticipants.csv",delimiter=' ',header=None)
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

skf = StratifiedKFold(n_splits=n_splits, random_state=None)
skf.get_n_splits(X, y)

for train_index, test_index in skf.split(X, y):
    # print("TRAIN:", train_index, "TEST:", test_index)
    X1_train, X1_test = X[train_index], X[test_index]
    y1_train, y1_test = y[train_index], y[test_index]
    clf                                                                    =
MLPClassifier(activation="tanh",solver="adam",max_iter=1000,hidden_layer_sizes=(10
0,),random_state=1)
    clf.fit(X1_train, y1_train)
    prediction = clf.predict(X1_test)
    score = accuracy_score(prediction,y1_test)
    accuracy.append(score)

NNmodelaccuracies.append(accuracy)
accuracy = np.array(accuracy)
print("%0.2f accuracy z standardno deviacijo %0.2f" % (accuracy.mean(),
accuracy.std()))
```

```
 ------------------------------------------------------------------------------
 ------------------------------------------------
modelnaposamezniku.ipynb
 ------------------------------------------------------------------------------
 ------------------------------------------------
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold

from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier

KNNmodelaccuracies = []
n_splits = 10

for i in range(1,17):
    accuracy = []
    dataset     =     pd.read_csv("csvData/participant{}.csv".format(i),delimiter='
',header=None)
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    skf = StratifiedKFold(n_splits=n_splits, random_state=None)
    skf.get_n_splits(X, y)


    for train_index, test_index in skf.split(X, y):
       # print("TRAIN:", train_index, "TEST:", test_index)
        X1_train, X1_test = X[train_index], X[test_index]
        y1_train, y1_test = y[train_index], y[test_index]
        clf = KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p = 2)
        clf.fit(X1_train, y1_train)
        prediction = clf.predict(X1_test)
        score = accuracy_score(prediction,y1_test)
        accuracy.append(score)

    modelaccuracies.append(accuracy)
    accuracy = np.array(accuracy)
    print("Udeleženec%0.0f  %0.2f  accuracy  z  standardno  deviacijo  %0.2f"  %  (i,
accuracy.mean(), accuracy.std()))
In [4]:
for i in range(len(modelaccuracies)):
    print()

NBmodelaccuracies = []
n_splits = 10

for i in range(1,17):
    accuracy = []
    dataset     =     pd.read_csv("csvData/participant{}.csv".format(i),delimiter='
',header=None)
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    skf = StratifiedKFold(n_splits=n_splits, random_state=None)
    skf.get_n_splits(X, y)


    for train_index, test_index in skf.split(X, y):
       # print("TRAIN:", train_index, "TEST:", test_index)
```

```
        X1_train, X1_test = X[train_index], X[test_index]
        y1_train, y1_test = y[train_index], y[test_index]
        clf = GaussianNB()
        clf.fit(X1_train, y1_train)
        prediction = clf.predict(X1_test)
        score = accuracy_score(prediction,y1_test)
        accuracy.append(score)

    NBmodelaccuracies.append(accuracy)
    accuracy = np.array(accuracy)
    print("Udeleženec%0.0f %0.2f accuracy z standardno deviacijo %0.2f" % (i,
accuracy.mean(), accuracy.std()))

print(NBmodelaccuracies)

Treemodelaccuracies = []
n_splits = 10

for i in range(1,17):
    accuracy = []
    dataset     =     pd.read_csv("csvData/participant{}.csv".format(i),delimiter='
',header=None)
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    skf = StratifiedKFold(n_splits=n_splits, random_state=None)
    skf.get_n_splits(X, y)


    for train_index, test_index in skf.split(X, y):
       # print("TRAIN:", train_index, "TEST:", test_index)
        X1_train, X1_test = X[train_index], X[test_index]
        y1_train, y1_test = y[train_index], y[test_index]
        clf = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
        clf.fit(X1_train, y1_train)
        prediction = clf.predict(X1_test)
        score = accuracy_score(prediction,y1_test)
        accuracy.append(score)

    Treemodelaccuracies.append(accuracy)
    accuracy = np.array(accuracy)
    print("Udeleženec%0.0f %0.2f accuracy z standardno deviacijo %0.2f" % (i,
accuracy.mean(), accuracy.std()))

print(Treemodelaccuracies)

NNmodelaccuracies = []
n_splits = 10

for i in range(1,17):
    accuracy = []
    dataset     =     pd.read_csv("csvData/participant{}.csv".format(i),delimiter='
',header=None)
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    skf = StratifiedKFold(n_splits=n_splits, random_state=None)
    skf.get_n_splits(X, y)

    for train_index, test_index in skf.split(X, y):
       # print("TRAIN:", train_index, "TEST:", test_index)
        X1_train, X1_test = X[train_index], X[test_index]
        y1_train, y1_test = y[train_index], y[test_index]
```

```
        clf                                                                    =
MLPClassifier(activation="tanh",solver="adam",max_iter=1000,hidden_layer_sizes=(10
0,),random_state=1)
        clf.fit(X1_train, y1_train)
        prediction = clf.predict(X1_test)
        score = accuracy_score(prediction,y1_test)
        accuracy.append(score)

    NNmodelaccuracies.append(accuracy)
    accuracy = np.array(accuracy)
     print("Udeleženec%0.0f  %0.2f  accuracy  z  standardno  deviacijo  %0.2f"  %  (i,
 accuracy.mean(), accuracy.std()))

 ----------------------------------------------------------------------
optimizirana_nn.py
 ----------------------------------------------------------------------

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import itertools
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold


for x in range(1:17):
    dataseteeg  =  pd.read_csv("Data/all{}.csv".format(x),delimiter='  ',  header  =
None)
    firstfive  =  pd.read_csv("Data/first5{}.csv".format(x),delimiter='  ',  header  =
None)
    testseteeg      =      pd.read_csv("Data/particip{}.csv".format(x),delimiter='
',header=None)

    X = dataseteeg.iloc[:, :-1].values
    Xfive = firstfive.iloc[:, :-1].values

    X_test= testseteeg.iloc[:,:-1].values
    y_test= testseteeg.iloc[:,-1].values

    y = dataseteeg.iloc[:, -1].values
    yfive = firstfive.iloc[:, -1].values

    for x in range(3):
        X = np.concatenate((X,Xfive), axis=0)
        y= np.concatenate((y,yfive), axis=0)

    scaler = preprocessing.StandardScaler()
    X = scaler.fit_transform(X)

    inputs = tf.keras.Input(shape=(X.shape[1],))

    l = tf.keras.layers.Dense(64,activation="relu")(inputs)
    l = tf.keras.layers.Dense(64,activation="relu")(l)
    l = tf.keras.layers.Dense(64,activation="relu")(l)
    l = tf.keras.layers.Dense(64,activation="relu")(l)
```

```
    outputs = tf.keras.layers.Dense(5, activation='softmax')(l)

    model = tf.keras.Model(inputs=inputs, outputs=outputs)

    model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
    )

    history = model.fit(
        X,
        y,
        validation_split=0.2,
        batch_size=64,
        epochs=50,
        callbacks=[
            tf.keras.callbacks.EarlyStopping(
                monitor='val_loss',
                patience=5,
                restore_best_weights=True
            )
        ]
    )

    model_acc = model.evaluate(X_test, y_test, verbose =0)
    y_pred = np.array(list(map(lambda x: np.argmax(x) , model.predict(X_test))))
    label_dict = {"Gnus":0, "strah":1, "žalost":2 , "nevtralno":3, "veselje":4}
    cm = confusion_matrix(y_test,y_pred)

dataseteeg = pd.read_csv("Data/all.csv".format(x),delimiter=' ', header = None)

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

accuracy = []
n_splits = 10
skf = StratifiedKFold(n_splits=n_splits, random_state=None)
skf.get_n_splits(X, y)

for train_index, test_index in skf.split(X, y):
        X1_train, X1_test = X[train_index], X[test_index]
        y1_train, y1_test = y[train_index], y[test_index]

        model = tf.keras.Model(inputs=inputs, outputs=outputs)

        model.compile(
        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
        )

        history = model.fit(
            X1_train,
            y1_test,
            validation_split=0.2,
            batch_size=64,
            epochs=50,
            callbacks=[
                tf.keras.callbacks.EarlyStopping(
                    monitor='val_loss',
                    patience=5,
                    restore_best_weights=True
                )
            ]
```

```
)

prediction = model.predict(X1_test)
score = accuracy_score(prediction,y1_test)
accuracy.append(score)
```