SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 936

KLASIFIKACIJA STRESA METODAMA STROJNOG UČENJA TEMELJENA NA BIOMEDICINSKIM VREMENSKIM NIZOVIMA S PRIJENOSNOG UREĐAJA

Florijan Sandalj

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 936

KLASIFIKACIJA STRESA METODAMA STROJNOG UČENJA TEMELJENA NA BIOMEDICINSKIM VREMENSKIM NIZOVIMA S PRIJENOSNOG UREĐAJA

Florijan Sandalj

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 936

Pristupnik:	**Florijan Sandalj (0036530775)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Klasifikacija stresa metodama strojnog učenja temeljena na biomedicinskim vremenskim nizovima s prijenosnog uređaja**

Opis zadatka:

Stres je jedan od glavnih uzročnika zdravstvenih problema. Pojačani stres i stalna izloženost stresu česti su u današnje vrijeme, a stres dugoročno dovodi do različitih problema u organizmu. Izloženost stresu moguće je ustanoviti na temelju biomedicinskih vremenskih nizova koji se mogu neinvazivno mjeriti preko kože osobe. U ovom radu potrebno je proučiti i opisati načine mjerenja stresa putem prijenosnih uređaja. Potom je potrebno izabrati jedan ili više klasifikatora strojnog učenja koji mogu raspoznati više različitih vrsta stresa (fizički stres, kognitivni stres, emocionalni stres). Model klasifikacije stresa potrebno je naučiti i ispitati na javno dosupnom skupu podataka Non-EEG Dataset for Assessment of Neurological Status (https://www.physionet.org/content/noneeg/1.0.0/). U izradi modela potrebno je iskoristiti postojeće knjižnice strojnog učenja po izboru. U radu je potrebno detaljno opisati ovaj skup podataka i način na koji je naučen model strojnog učenja. Potrebno je prikazati mjere uspješnosti klasifikacije na testnom skupu.

Rok za predaju rada: 9. lipnja 2023.

Zahvaljujem se mentoru izv. prof. dr. sc. Alanu Joviću na pomoći pri izradi rada te roditeljima i prijateljima na pomoći oko svega ostalog.

Sadržaj

Uvod

Većina ljudi svakodnevno se susreće s različitim vrstama stresa te je on jedan od glavnih uzročnika zdravstvenih problema. Pojačani stres i stalna izloženost stresu česti su u današnje vrijeme, a dugoročno mogu dovesti do različitih problema u organizmu. Izloženost stresu moguće je ustanoviti na temelju biomedicinskih vremenskih nizova koji se mogu neinvazivno mjeriti preko kože osobe. Razvojem tehnologije naprave koje mjere takve biomedicinske vremenske nizove postaju sve praktičnije i dostupnije velikom broju ljudi. Bilježenjem vrste i količine stresa kroz koju osoba prolazi u danu može ukazati na preveliku izloženost stresu te pomoći pri očuvanju zdravlja i prevenciji zdravstvenih tegoba.

Tema ovog završnog rada je korištenje strojnog učenja za izgradnju klasifikatora stresa. Odabrano je nekoliko algoritama strojnog učenja te su modeli naučeni na javno dostupnom skupu podataka Non-EEG Dataset for Assesment of Neurological Status [1]. Modeli su potom testirani i uspoređene su njihove performanse.

1. Korištene tehnologije

1.1. WFDB (Waveform Database)

Waveform Database je datotečni standard za čitanje i zapisivanje fizioloških signala i pripadnih anotacija [2]. Skup podataka korišten za učenje i testiranje modela u ovom radu dostupan je u WFDB formatu. Uz vlastiti programski paket, WFDB je podržan i u programskom jeziku Python te platformom MATLAB.

1.2. MATLAB

MATLAB je platforma za numeričko programiranje koja sadrži razne alate za analizu podataka, razvijanje algoritama i izgradnju modela, a podržava i čitanje podataka u wfdb formatu te je iz tog razloga odabran kao platforma za izgradnju modela [3].

1.2.1. MATLAB Diagnostic Feature Designer

MATLAB Diagnostic Feature Designer je aplikacija unutar MATLAB-a koja omogućava učitavanje i vizualizaciju podataka, ekstrakciju odabranih značajki iz skupa podataka, vizualizaciju značajki i njihove učinkovitosti u razdvajanju podataka te uspoređivanje učinkovitosti statističkim testovima i konačno izvoz značajki za daljnju upotrebu izvan programa [4].

1.2.2. MATLAB Classification Learner

MATLAB Classification Learner je aplikacija unutar MATLAB-a namijenjena učenju modela za klasifikaciju podataka. Omogućuje učitavanje značajki, specifikaciju postupka validacije, učenje i testiranje modela. Moguće je automatizirati učenje i pronaći algoritam koji najbolje odgovara danim podacima. Algoritmi na raspolaganju su stabla odluke, diskriminantna analiza, metoda potpornih vektora, logistička regresija, najbliži susjedi, naivni Bayes, jezgrene funkcije i neuronske mreže. [5]

2. Skup podataka

U ovom radu za učenje i testiranje modela korišten je javno dostupan skup podataka Non-EEG Dataset for Assesment of Neurological Status [1] koji sadrži fiziološke vremenske nizove prikupljene od dvadeset subjekata na Teksaškom Sveučilištu u Dallasu. Slijedi detaljniji opis.

2.1. Korištene naprave

Praćenje neurološke aktivnosti pacijenta tradicionalno se i efektivno provodi prikupljanjem EEG (elektroencefalogram) signala površinskim ili umetnutim elektrodama. Međutim, takav način prikupljanja podataka nije niti ugodan niti praktičan za svakodnevnu primjenu. Stoga su podaci za ovaj skup prikupljeni korištenjem prijenosnih uređaja koje subjekt nosi na zapešću, poput pametnog sata. Prva takva korištena naprava jest Affectiva Q prikazana na slici 2.1.1 Ona prikuplja podatke o kretanju zapešća u obliku akceleracije na x, y i z osi, temperaturu te EDA (elektrodermalnu aktivnost) izraženu u vodljivosti kože koja ukazuje na aktivnost simpatičkog živčanog sustava. Drugi korišteni uređaj je Nonin 3150 Wireless WristOx2 prikazan je na slici 2.1.2 i korišten je za mjerenje brzine otkucaja srca na zapešću te dodatno koristi produžetak koji na prstu subjekta mjeri zasićenost krvi kisikom. Neke tvrtke trenutno razvijaju naprave koje mogu mjeriti zasićenost kisikom sa zapešća, ali u vrijeme prikupljanja ovih podataka takve naprave nisu bile dostupne [6].

Slike 2.1.1 (lijevo) prikazuje uređaj Affectiva Q, dok slika 2.1.2 (desno) prikazuje Nonin 3150 Wireless WristOx2

2.2. Prikupljanje podataka

Podaci su prikupljani na 20 subjekata od kojih je 14 muškog i 6 ženskog spola. Osmišljen je 35-minutni eksperiment koji se sastojao od sljedećih aktivnosti:

1) Prvo mirovanje u trajanju od pet minuta
2) Fizički stres, koji se sastoji od minute stajanja u mjestu, hodanja na traci brzinom 1.6 km/h (1 mph) dvije minute te hodanja/trčanja na traci brzinom 4.8 km/h (3 mph) tri minute, ukupno trajanje fizičkog stresa je pet minuta
3) Drugo mirovanje u trajanju od pet minuta
4) 1. Emotivni stres. U ovom periodu od četrdesetak sekundi subjektima su pročitane upute za sljedeći zadatak koji je namijenjen induciranju kognitivnog stresa, preciznije pročitane su upute za „matematički“ dio zadatka. Iako to nije bilo planirano, provoditeljima testa bilo je jasno da su mnogi subjekti u ovom periodu pokazivali znakove emotivnog stresa.

   2. Kognitivni stres. Subjekti su prvo trebali tri minute brojati unazad po 7 počevši od 2485. Zatim su dvije minute rješavali inačicu Stroopovog testa.
5) Treće mirovanje u trajanju od pet minuta
6) Emotivni stres, subjekti su gledali petominutni isječak iz horor filma
7) Četvrto mirovanje u trajanju od 5 minuta.

2.3. Format podataka

Prikupljeni podaci za svakog subjekta spremljeni su u dvije WFDB datoteke; jedna sadrži očitanja akceleracije, temperature i elektrodermalne aktivnosti koje je prikupljao uređaj Affectiva Q u vremenskim razmacima od 0,125 sekundi (8 Hz), a druga sadrži očitanja brzine otkucaja srca te zasićenosti kisikom, koje je uređaj Nonina 3150 prikupljao frekvencijom 1 Hz, te su stoga morali biti odvojeni u zasebnu datoteku. U prvoj datoteci također se nalazi anotacija koja određuje redni broj očitanja u kojem započinje novo stanje pacijenta te naziv tog stanja.

3. Priprema podataka

Da bi mogli pretvoriti podatke iz WFDB formata u format prikladan za obradu u MATLAB-u, prvo je potrebno instalirati WFDB Toolbox for MATLAB [7] te ga staviti na raspolaganje MATLAB-u funkcijom addpath. Funkcije koje od tamo koristimo su:

- wfdb2mat – za pretvaranje .dat datoteka u .mat format
- rdmat – za čitanje .mat datoteka generiranih wfdb2mat funkcijom
- rdann – za čitanje .ann datoteka

Za svakog subjekta koristimo te funkcije da bi signale iz obje datoteke učitali u varijable signal i signal2, te redne brojeve očitanja u kojima započinje novo stanje i nazive tih stanja u varijable ann i comments. Zatim za svako stanje i za svaku mjerenu varijablu u tom stanju kreiramo vremensku tablicu i spremamo ju u vektor vremenskih tablica te varijable. Dodatno stvaramo i vektor u koji spremamo nazive stanja redoslijedom kojim ih obilazimo. Nakon toga vektore je potrebno transponirati i spojiti u jednu tablicu. Time smo podatke prebacili u format prikladan za daljnju analizu. Kod kojim smo sve ovo učinili prikazan je na slikama 3.1 i 3.2., a tablica koju smo dobili kao rezultat prikazana je slikom 3.3. Kako je svaki subjekt prošao kroz 4 stanja mirovanja te dva stanja emotivnog stresa, zapisa o stanjima mirovanja i emotivnog stresa imamo više od ostalih. Kako bi svako stanje bilo jednako reprezentirano, uklanjamo višak stanja mirovanja i emotivnog stresa, i to na način da ostavljamo podatke samo za prvi period mirovanja i drugi period emotivnog stresa, budući da njih smatramo najboljim reprezentacijama tih stanja. Kod za ovu akciju prikazan je na slici 3.4. Sada je svako stanje ravnopravno predstavljeno u podacima i možemo krenuti s ekstrakcijom značajki.

```
addpath 'C:\Users\FLORIJAN\Documents\FER\6.Semestar\ZAVRŠNI\non-eeg-dataset-for-assessment-of-n

k = 0;
for j = 1:20
    wfdb2mat(sprintf('Subject%d_AccTempEDA', j));
    wfdb2mat(sprintf('Subject%d_SpO2HR', j))
    [tm,signal,Fs,labels]=rdmat(sprintf('Subject%d_AccTempEDAm', j));
    [tm2,signal2,Fs2,labels2]=rdmat(sprintf('Subject%d_SpO2HRm', j));
    [ann,anntype,subtype,chan,num,comments]=rdann(sprintf('Subject%d_AccTempEDA', j), 'atra');
    ann2 = round(ann / 8);
    ann(length(ann) + 1) = length(signal) + 1;
    ann2(length(ann2) + 1) = length(signal2) + 1
    ann2(1) = 1;
    for i = 1:length(ann) - 1
        ax{k + i} = timetable(signal(ann(i):ann(i + 1) - 1, 1), 'SampleRate', Fs);
        ay{k + i} = timetable(signal(ann(i):ann(i + 1) - 1, 2), 'SampleRate',Fs);
        az{k + i} = timetable(signal(ann(i):ann(i + 1) - 1, 3), 'SampleRate', Fs);
        temp{k + i} = timetable(signal(ann(i):ann(i + 1) - 1, 4), 'SampleRate', Fs);
        eda{k + i} = timetable(signal(ann(i):ann(i + 1) - 1, 5), 'SampleRate', Fs);
        spo2{k + i} = timetable(signal2(ann2(i):ann2(i + 1) - 1, 1), 'SampleRate',Fs2);
        hr{k + i} = timetable(signal2(ann2(i):ann2(i + 1) - 1, 2), 'SampleRate',Fs2);
        faultCode{k + i} = comments(i);
    end
    k = k + i;

end
```

Slika 3.1

```
ax = transpose(ax);
ay = transpose(ay);
az = transpose(az);
temp = transpose(temp);
eda = transpose(eda);
spo2 = transpose(spo2);
hr = transpose(hr);
faultCode = transpose(faultCode);
faultcode = string(faultCode);
data = table(ax, ay, az, temp, eda, spo2, hr, faultcode);
```

Slika 3.2

data

160x8 table

	1 ax	2 ay	3 az	4 temp	5 eda	6 spo2	7 hr	8 faultcode
1	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	299x1 timetable	299x1 timetable	"Relax"
2	2625x1 timetable	2625x1 timetable	2625x1 timetable	2625x1 timetable	2625x1 timetable	328x1 timetable	328x1 timetable	"PhysicalStr...
3	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	301x1 timetable	301x1 timetable	"Relax"
4	321x1 timetable	321x1 timetable	321x1 timetable	321x1 timetable	321x1 timetable	40x1 timetable	40x1 timetable	"Emotional...
5	2913x1 timetable	2913x1 timetable	2913x1 timetable	2913x1 timetable	2913x1 timetable	364x1 timetable	364x1 timetable	"CognitiveS...
6	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	300x1 timetable	300x1 timetable	"Relax"
7	2881x1 timetable	2881x1 timetable	2881x1 timetable	2881x1 timetable	2881x1 timetable	360x1 timetable	360x1 timetable	"Emotional...
8	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	307x1 timetable	307x1 timetable	"Relax"
9	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	299x1 timetable	299x1 timetable	"Relax"
10	2617x1 timetable	2617x1 timetable	2617x1 timetable	2617x1 timetable	2617x1 timetable	327x1 timetable	327x1 timetable	"PhysicalStr...
11	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	301x1 timetable	301x1 timetable	"Relax"
12	321x1 timetable	321x1 timetable	321x1 timetable	321x1 timetable	321x1 timetable	40x1 timetable	40x1 timetable	"Emotional...
13	2841x1 timetable	2841x1 timetable	2841x1 timetable	2841x1 timetable	2841x1 timetable	355x1 timetable	355x1 timetable	"CognitiveS...
14	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	300x1 timetable	300x1 timetable	"Relax"
15	2945x1 timetable	2945x1 timetable	2945x1 timetable	2945x1 timetable	2945x1 timetable	368x1 timetable	368x1 timetable	"Emotional...
16	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	307x1 timetable	307x1 timetable	"Relax"
17	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	2400x1 timetable	299x1 timetable	299x1 timetable	"Relax"
18	2593x1 timetable	2593x1 timetable	2593x1 timetable	2593x1 timetable	2593x1 timetable	324x1 timetable	324x1 timetable	"PhysicalStr...
19	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	301x1 timetable	301x1 timetable	"Relax"
20	321x1 timetable	321x1 timetable	321x1 timetable	321x1 timetable	321x1 timetable	40x1 timetable	40x1 timetable	"Emotional...
21	2833x1 timetable	2833x1 timetable	2833x1 timetable	2833x1 timetable	2833x1 timetable	354x1 timetable	354x1 timetable	"CognitiveS...
22	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	2401x1 timetable	300x1 timetable	300x1 timetable	"Relax"

Slika 3.3 tablica 'data'

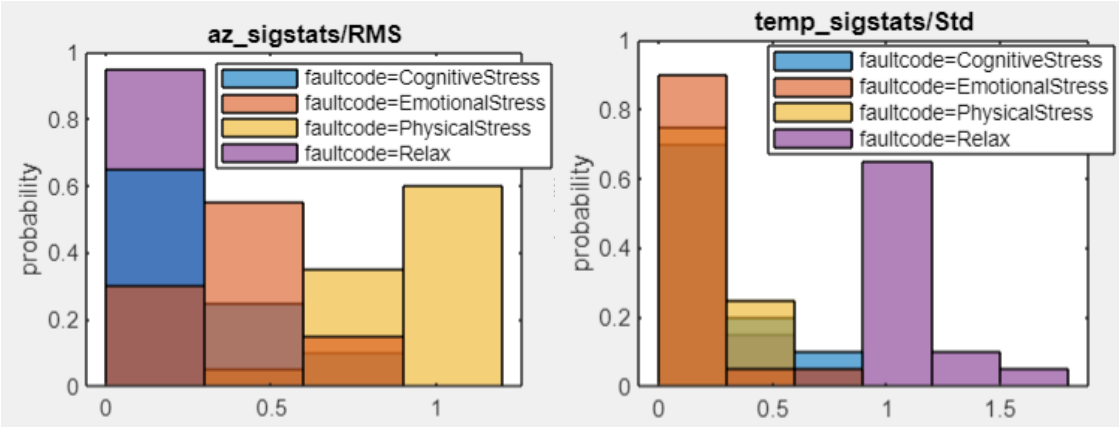
```
data2 = data;
for i = 0:19
    data2(i*8 + 3 - i*3,:) = [];
    data2(i*8 + 5 - i*3,:) = [];
    data2(i*8 + 6 - i*3,:) = [];
end


for i = 0:10:9*10
data2(i + 5 - (i/10)*2,:) = [];
data2(i + 7 - (i/10)*2, :) = [];
end
```

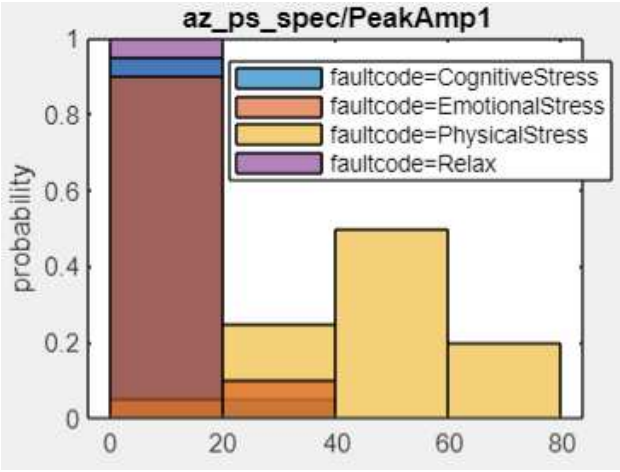
Slika 3.4 Uklanjanje viška podataka

4. Ekstrakcija značajki

Tablicu signala koju smo kreirali prosljeđujemo aplikaciji Diagnostic Feature Designer. Ona nudi mogućnost generiranja značajki signala u vremenskoj i frekvencijskoj domeni. Možemo ručno odabrati koje značajke želimo izračunati, a možemo i koristiti opciju automatskog generiranja velikog skupa značajki iz vremenske i frekvencijske domene za sve signale. Nakon toga, provodi se jednofaktorska ANOVA kako bi se značajke rangirale po svojoj sposobnosti da ispravno odvoje podatke. Nakon provedene analize, rezultati ukazuju da varijable akceleracije u z osi i temperatura najbolje razdvajaju podatke, budući da se značajke izvedene od tih varijabli nalaze u samom vrhu poretka. Neke od značajki dobivenih iz temperature su minimalna vrijednost i standardna devijacija, a neke značajke dobivene iz akceleracije su vrhunac amplitude i korijen srednje kvadratne vrijednosti. Grafovi koji prikazuju kako neke od ovih značajke razdvajaju podatke nalaze se na slikama 4.1-4.3. Na horizontalnoj osi nalaze se vrijednosti koje poprima značajka, a na vertikalnoj označena je vjerojatnost. Svaka boja predstavlja jedno stanje, a stupac u toj boji predstavlja vjerojatnost da će značajka u tom stanju poprimiti vrijednost iz određenog intervala na horizontalnoj osi.



Slika 4.1 (lijevo) prikazuje korijen srednje kvadratne vrijednosti akceleracije dok slika 4.2 (desno) prikazuje standardnu devijaciju temperature



Slika 4.3 prikazuje vrhunac amplitude za akceleraciju

Deset najbolje rangiranih značajki izvodimo u MATLAB radni prostor u obliku tablice značajki prikazane na slici 4.2. Značajke su u tom obliku spremne za sljedeće korake: učenje, validaciju i testiranje modela. Izvedene su značajke redom:

1) temp_sigstats/Std – standardna devijacija temperature
2) az_res_tsmodel/AIC – Akaikeov informacijski kriterij modela vremenskog niza rezidualnog signala (signal umanjen za svoju srednju vrijednost) akceleracije u z osi
3) az_ps_spec/PeakAmp1 – maksimalna amplituda na spektru snage akceleracije u z osi
4) az_sigstats/RMS – kvadrat srednje vrijednosti korijena akceleracije u z osi
5) az_tsproc_tsmodel/AIC – Akaikeov informacijski kriterij modela stacionarnog vremenskog niza akceleracije u z osi
6) az_tsmodel/AIC – Akaikeov informacijski kriterij modela vremenskog niza akceleracije u z osi
7) temp_sigstats/ClearanceFactor – omjer maksimalne vrijednosti i kvadrata srednje vrijednosti korijena apsolutnih amplituda temperature
8) temp_sigstats/ImpulseFactor – omjer maksimalne vrijednosti i srednje apsolutne vrijednosti temperature
9) temp_sigstats/CrestFactor – omjer maksimalne vrijednosti i kvadrata srednje vrijednosti korijena temperature
10) temp_tsfeat/Minimum – minimalna vrijednost vremenskog niza temperature

Akaikeov informacijski kriterij (engl. AIC) je procjenitelj greške u predviđanju, odnosno relativne kvalitete statističkih modela za dani skup podataka.

FeatureTable1

80x11 table

	1 faultcode	2 az_res_tsmo	3 az_tsproc_ts	4 az_ps_spec/	5 temp_sigs	6 temp_sigs	7 temp_sigs	8 temp_sigs	9 temp_tsfeat	10 az_sigstats	11 az_tsmodel
1	"Relax"	-9.0147	-10.2923	0.6819	1.0267	1.0265	1.0266	0.5650	30.1000	0.1041	-10.2906
2	"PhysicalStr...	-4.4141	-4.4005	51.3996	1.0118	1.0118	1.0118	0.1381	32.2000	0.9020	-4.3971
3	"Emotional...	-3.4240	-3.3856	3.6400	1.0091	1.0091	1.0091	0.0289	32.6000	0.6143	-3.4004
4	"CognitiveS...	-6.4392	-6.4393	1.9699	1.0181	1.0181	1.0181	0.1075	32.2000	0.2268	-6.4138
5	"Relax"	-9.1392	-11.0567	0.4404	1.0303	1.0300	1.0302	0.5884	28.9001	0.0842	-11.0225
6	"PhysicalStr...	-3.6490	-3.6259	59.6566	1.0138	1.0137	1.0138	0.0888	30.8001	1.0104	-3.6224
7	"CognitiveS...	-6.5828	-6.7312	2.3705	1.0367	1.0364	1.0366	0.7348	32.2000	0.2032	-6.7305
8	"Emotional...	-4.8418	-4.8610	1.4854	1.0144	1.0144	1.0144	0.1616	31.6001	0.2972	-4.8612
9	"Relax"	-8.5647	-9.3443	0.5206	1.0313	1.0310	1.0312	0.5505	28.5000	0.0943	-9.3430
10	"PhysicalStr...	-5.4951	-5.5139	50.2337	1.0095	1.0095	1.0095	0.1322	30.4001	0.8967	-5.5057
11	"Emotional...	-4.6143	-4.7115	2.2570	1.0001	1.0001	1.0001	0.0193	31.9999	0.5040	-4.5958
12	"CognitiveS...	-6.9282	-7.5619	9.6058	1.0125	1.0125	1.0125	0.0930	31.0000	0.3837	-7.2827
13	"Relax"	-7.6258	-7.8979	2.2869	1.0556	1.0545	1.0552	1.0603	25.8001	0.2107	-7.8444
14	"PhysicalStr...	-4.0388	-4.0362	59.3771	1.0182	1.0181	1.0182	0.3750	29.7000	0.9960	-4.0230
15	"CognitiveS...	-5.7971	-5.7867	9.9802	1.0098	1.0098	1.0098	0.1149	31.7999	0.4858	-5.8506
16	"Emotional...	-6.1076	-6.1796	12.6512	1.0128	1.0128	1.0128	0.1262	32.8999	0.4877	-6.1667
17	"Relax"	-9.0756	-10.5824	0.0100	1.0427	1.0419	1.0424	0.9597	27.9001	0.0163	-10.5687
18	"PhysicalStr...	-6.2431	-6.2963	52.1402	1.0119	1.0119	1.0119	0.1744	31.4000	0.9159	-6.2956
19	"Emotional...	-5.1812	-5.1661	1.4186	1.0062	1.0062	1.0062	0.0222	32.0000	0.3803	-5.1449

Slika 4.2 Tablica značajki

5. Algoritmi strojnog učenja

5.1. Strojno učenje

Strojno učenje je programiranje računala da optimiraju određeni kriterij korištenjem testnih primjera ili prošlog iskustva. Model je definiran nekim svojim hiperparametrima, a učenje je izvođenje programa koji optimira ostale parametre modela koristeći podatke i prijašnje iskustvo kako bi se dobio najbolji rezultat s obzirom na zadani kriterij. S obzirom na vrstu podataka koji su nam na raspolaganju, strojno učenje možemo podijeliti na nadzirano učenje (skup podataka je označen, tj. znamo koju vrijednost izlazne funkcije želimo dobiti za svaki primjer) i nenadzirano učenje (skup podataka nije označen). Kod nadziranog učenja, ovisno o vrsti izlazne vrijednosti razlikujemo klasifikaciju (izlazna funkcija poprima diskretne/nebrojčane vrijednosti) i regresiju (izlazna funkcija poprima kontinuirane/brojčane vrijednosti). U našem slučaju, podaci su označeni, a izlazna funkcija poprima jednu od četiri vrijednosti, pa govorimo o klasifikaciji. Klasifikacija se u grubo može podijeliti na tri slijedna koraka: učenje, validacija i testiranje. Skup podataka koji nam je na raspolaganju dijelimo u nekom omjeru (npr. 40:30:30) na skup za učenje, skup za provjeru i skup za testiranje.

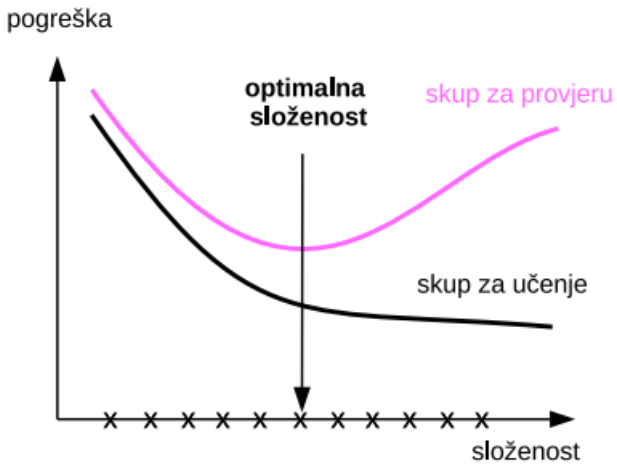
5.1.1. Učenje

Detalji faze učenja razlikuju se od algoritma do algoritma, ali ono što je zajedničko svim algoritmima jest da se u fazi učenja parametri modela prilagođavaju podacima u skupu za učenje kako bi se optimirala ispravnost predviđanja modela nad tim istim podacima. Ključno je da u fazi učenja algoritam ne koristi podatke iz skupa za provjeru i skupa za testiranje. Drugim riječima, ti skupovi moraju biti nezavisni. Time se osigurava nepristrana provjera i testiranje.

5.1.2. Provjera

Glavni cilj strojnog učenja je izgraditi model koji dobro generalizira. Ako je model jako složen, on se može dobro naučiti na podatke iz skupa za učenje i za njih davati gotovo savršene rezultate. Ako je model presložen, tada se neće dobro snalaziti u prije neviđenim podacima, drugim riječima loše će generalizirati [8]. Provjera (ili validacija) je proces kojim

određujemo složenost modela na način da minimiziramo prenaučenost. Provjera se provodi na način da se korištenjem skupa za učenje grade modeli s različitim hiperparametrima odnosno različite kompleksnosti koji se onda provjeravaju na skupu za provjeru. Kako kompleksnost modela raste, pogreška na skupu za učenje će uvijek opadati, dok će pogreška na skupu za provjeru padati do određene razine kompleksnosti, nakon čega će zbog prenaučenosti modela ponovno rasti, kao što je prikazan o na slici 5.1. Tu razinu složenosti uzimamo kao optimalnu složenost te se ona primjenjuje na model koji uči koristeći cijeli skup za učenje, zajedno sa skupom za provjeru.



Slika 5.1 Prenaučenost modela, preuzeto iz [8]

5.1.3. Testiranje

U fazi testiranja, našem modelu prvi puta dajemo na ulaz podatke iz skupa za testiranje. Model za svaki ulaz daje svoju predikciju te se rezultati na kraju uspoređuju s točnim rješenjima. Postoji više mjera uspješnosti modela, a neke od popularnijih su točnost i matrica zbunjenosti. Točnost je jednostavno postotak točno klasificiranih primjera. Matrica zbunjenosti daje nam malo bolji uvid u performanse modela. U matrici zbunjenosti redci predstavljaju stvarne vrijednosti, dok stupci predstavljaju vrijednosti koje je naš model predvidio. Vrijednost ćelije predstavlja broj slučajeva u kojima je naš model podatak koji pripada klasi retka klasificirao u klasu stupca. Idealno, sve vrijednosti osim na dijagonali su nula i tada govorimo o 100% točnosti, kao što je prikazanu na tablici 5.1. Matrica zbunjenosti

daje nam dodatne informacije o tome kako naš model klasificira podatke i ima li tendenciju griješiti na određenim klasama više nego na drugima.

	Predviđena klasa 1	**Predviđena klasa 2**	**Predviđena klasa 3**
Stvarna klasa 1	3	0	0
Stvarna klasa 2	0	3	0
Stvarna klasa 3	0	0	3

Tablica 5.1 Matrica zbunjenosti sa 100% točnosti

5.2. Bayesov naivni klasifikator

Bayesov klasifikator je jednostavan algoritam nadziranog strojnog učenja koji se temelji na Bayesovom teoremu. Bayesov teorem bavi se uvjetnim vjerojatnostima. Ako je H neka hipoteza, a E neki događaj, Bayesovo pravilo govori nam sljedeće:

$$P(H|E) = \frac{P(H)\,P(E|H)}{P(E)} = \frac{P(H)\,P(E|H)}{P(H)\,P(E|H) + P(\neg H)\,P(E|\neg H)} \qquad (1)$$

gdje je $P(H)$ apriorna vjerojatnost hipoteze, $P(E|H)$ uvjetna izglednost događaja ako je ispunjena hipoteza, a $P(H|E)$ aposteriorna vjerojatnost hipoteze. Ako imamo više hipoteza, i njima pokrivamo sve moguće događaje, onda vrijedi:

$$P(H_i|E) = \frac{P(H_i)\,P(E|H_i)}{\sum_j P(H_j)\,P(E|H_j)} \qquad (2)$$

Ako dodatno imamo i više događaja, formula glasi:

$$P(H_i|E_1, E_2 \dots E_n) = \frac{P(H_i)\,P(E_1, E_2 \dots E_n|H_i)}{\sum_j P(H_j)\,P(E_1, E_2 \dots E_n|H_j)} \qquad (3)$$

Ako pretpostavimo da su događaji $E_1, E_2 \dots E_n$ međusobno uvjetno nezavisni uz danu hipotezu, formula poprima sljedeći oblik:

$$P(H_i|E_1, E_2 \dots E_n) = \frac{P(H_i)\,P(E_1|H_i)\,P(E_2|H_i) \dots P(E_n|H_i)}{\sum_j P(H_j)\,P(E_1|H_i)\,P(E_2|H_i) \dots P(E_n|H_i)} \qquad (4)$$

Formulu (4) možemo direktno primijeniti kao klasifikator ako kao hipoteze uzmemo klase (vrijednosti izlazne varijable), a kao događaje vrijednosti koje poprimaju značajke u ulaznom skupu. Apriorne vjerojatnosti klasa i izglednosti značajki možemo izračunati iz skupa za učenje te tražimo onu klasu za koju je aposteriorna vjerojatnost maksimalna. Drugim riječima, uz zadane značajke tražimo klasu kojoj primjer najvjerojatnije pripada. Sada je jasno zašto nam je bitna pretpostavka o uvjetnim nezavisnostima događaja $E_1, E_2 \dots E_n$ ili u našem slučaju – značajki: kada one ne bi bile nezavisne, morali bi izračunati izglednost klase $P(E_1, E_2 \dots E_n|H_i)$ za bilo koju kombinaciju vrijednosti značajki na ulazu i za svaku klasu na izlazu. Pretpostavka o nezavisnosti pojednostavljuje račun, a pokazuje se da se dobivaju dobri rezultati čak i onda kada značajke nisu u potpunosti nezavisne [9]. Nadalje, za zadani skup značajki svaka će aposteriorna vjerojatnost klase imati isti nazivnik, pa je ono što tražimo:

$$klasa = argmax_y\, P(y) \prod_i P(x_i|y)$$

Apriorne vjerojatnosti klasa računamo kao udio primjera iz skupa za učenje koji pripadaju toj klasi, dok izglednosti značajke računamo ovisno o njihovom tipu. Ako značajke

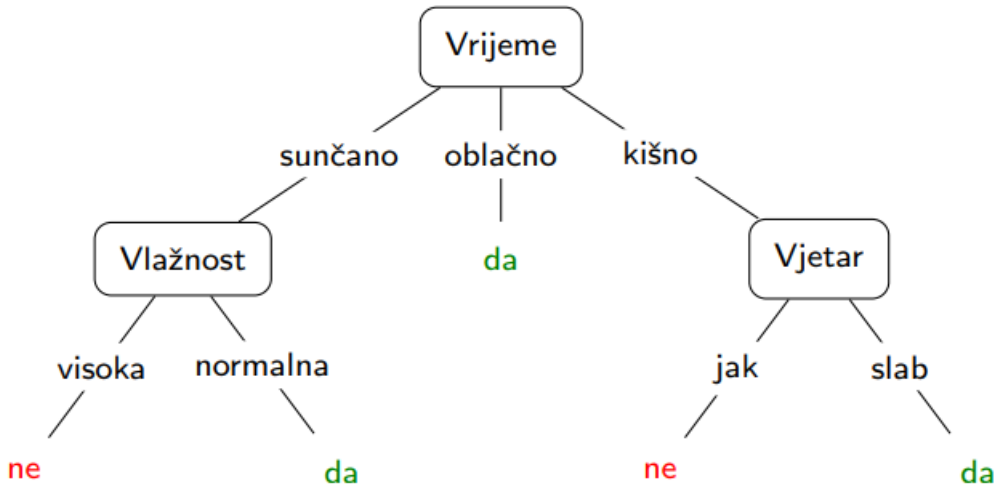
poprimaju kategoričku vrijednost, onda $P(x_i = v \mid y)$ možemo računati kao udio primjera u kojima značajka x_i poprima vrijednost v u svim primjerima koji pripadaju klasi *y*. Međutim, ako značajke poprimaju kontinuirane vrijednosti, izglednost moramo modelirati na drugačiji način. Jedan od njih je da pretpostavimo da su značajke normalno distribuirane. Iz podataka za svaku klasu izračunamo srednju vrijednost μ i standardnu devijaciju σ koju poprima svaka od značajki. Iz toga slijedi da izglednost značajke računamo formulom:

$$P(x_i = x \mid y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\sigma)^2}{2\sigma^2}}$$

Ostatak postupka ostaje isti. Taj algoritam nazivamo Gaussov naivni Bayesov klasifikator [10].

5.3. Stabla odluke

Stabla odluke jednostavan su i intuitivan način klasifikacije podataka. Predikcija se temelji na stablu odluke, strukturi u kojoj čvorovi predstavljaju značajke, a grane njihove vrijednosti. Predikcija započinje u korijenu stabla koji sadrži prvu značajku, a grane koje vode prema čvorovima sljedeće razine sadrže vrijednosti koje poprima ta značajka. Ovisno o vrijednosti koju značajka poprima u primjeru koji želimo klasificirati krećemo se granom te vrijednosti do sljedećeg čvora. Postupak ponavljamo dok ne dođemo do lista – on sadrži naziv klase u koju ćemo smjestili naš primjer, ili dok ne možemo nastaviti kretanje stablom jer naš primjer nema definiranu vrijednost značajke tog čvora. Jednostavan primjer stabla odluke nalazi se na slici 5.2. Svaki put od korijenskog čvora do lista stabla zapravo je konjunkcija uvjeta [8].



Slika 5.2 Jednostavno stablo odluke, preuzeto iz [8]

Gradnja stabla odluke rekurzivan je postupak koji je najbolje opisan ID3 algoritmom, a njegov (malo pojednostavljen) pseudokod nalazi se ispod:

```
ID3(skup primjera, skup značajki, najčešća klasa roditelja)
   Ako je skup primjera prazan
      vrati list s najčešćom klasom iz roditeljskog čvora
   Pronađi najčešću klasu u skupu primjera
   Ako je skup značajki prazan ili ako su svi primjeri iste klase
      Vrati čvor s najčešćom klasom
   Odaberi značajku po kojoj će se dijeliti podaci
   Ukloni tu značajku iz skupa značajki
   Za svaku vrijednost te značajke
      Pozovi funkciju ID3 i predaj joj samo one primjere u kojima
      značajka poprima tu vrijednost, skup značajki (bez te značajke) i
      najčešću klasu trenutnog čvora
      Rezultat će biti čvor/list i on postaje dijete trenutnog čvora
   Vrati trenutni čvor
```

Sada jedino preostaje pitanje kako odabrati značajku po kojoj će se podaci dijeliti. Kao i kod svih algoritama klasifikacije, želimo što jednostavniji model koji dobro generalizira i nije prenaučen na podatke iz skupa za učenje. Kod stabla odluke to možemo postići minimiziracijom broja čvorova i dubine stabla. U principu želimo u svakom čvoru odabrati značajku takvu da skupovi nastali podjelom budu što manje uređeni. Maksimalnu uređenost skupa dobili bi kada bi sve klase bile jednako zastupljene u primjerima, a minimalnu onda

kada bi svi primjeri pripadali istoj klasi. Postoji više načina na koje možemo odrediti uređenost skupa, a algoritam ID3 koristi minimizaciju entropije. Ako se skup *D* sastoji od primjera iz *k* različitih klasa, onda je njegova entropija definirana formulom:

$$E(D) = -\sum_{i=1}^{i=k} p(y = y_k) \log\, p(y = y_k)$$

U nekom čvoru biramo značajku tako da ona maksimizira informacijsku dobit. Informacijska dobit definirana je formulom:

$$IG = E(D) - \sum_{v\, \varepsilon\, V(x)} \frac{|D_{x=v}|}{|D|} E(D_{x=v})$$

gdje je *V*(*x*) skup svih vrijednosti koje poprima značajka *x*. Još jedna popularna mjera uređenosti skupa jest Ginijev indeks, a računa se formulom

$$GI(D) = 1 - \sum_{i} P^2(y = y_i)$$

Na sličan način bira se značajka *x* koja minimizira sljedeću vrijednost:

$$GI(D) \; - \sum_{v\, \varepsilon\, V(x)} \frac{|D_{x=v}|}{|D|} GI(D_{x=v})$$

Do sada smo pokrili slučaj u kojem značajke poprimaju kategoričke vrijednosti, no što ako su one kontinuirane? Podjelu možemo raditi određivanjem jedne ili više graničnih vrijednosti te stvaranjem novog čvora za svaki interval određen tim graničnim vrijednostima. Intervale možemo odrediti podjelom sortiranog skupa vrijednosti koje značajka poprima na *n* podjednakih skupova, te tražiti značajku za koju takva podjela rezultira najmanjom

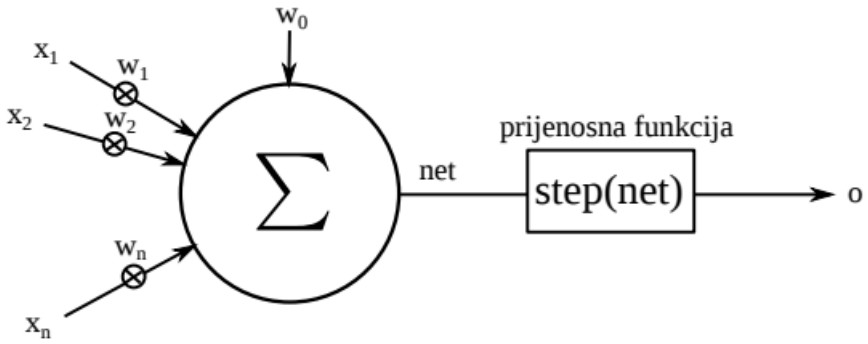
uređenošću. Kompliciraniji pristup uključuje primjenu strojnog učenja, točnije grupiranja za određivanje intervala. Ako želimo jednostavnije stablo s manje čvorova, bolji je pristup odrediti samo jednu graničnu vrijednost i podijeliti podatke na one iznad te vrijednosti i ostatak. Kako odabrati kandidate za graničnu vrijednost? Jednostavniji pristupi su uzeti srednju vrijednost ili medijan svih primjera, ili odabrati graničnu vrijednost tako da nastali skupovi nakon podjele budu podjednaki. Zahtjevniji način je sortiranje svih vrijednosti koje značajka poprima i isprobavanjem srednje vrijednosti svake dvije susjedne vrijednosti. Trebalo bi odabrati onu graničnu vrijednost za koju smo maksimizirali informacijsku dobit ili neku drugu mjeru uređenosti primjera. To bi značilo da postupak moramo provesti za svaku značajku i za svake dvije susjedne vrijednosti te značajke. Kako bi ubrzali cijeli postupak, možemo se poslužiti heuristikom, npr. isprobavati samo srednje vrijednosti koje dobivamo od primjera koji pripadaju različitim klasama [11].

Još jedan način na koji možemo smanjiti složenost modela i poboljšati generalizaciju kod stabla odluke je podrezivanje, tj. ograničavanje dubine stabla. Kada stablo dosegne određenu dubinu, grananje se prekida i u list se sprema ona klasa koja je u tom skupu primjera najčešća. Podrezivanje osobito pomaže kada je skup za učenje opširan i kada na raspolaganju imamo puno značajki.

5.4. Neuronske mreže

Neuronske mreže su složene strukture korištene u strojnom učenju koje su građene po uzoru na ljudski mozak. To je ideja koja slijedi iz konektivističkog pristupa umjetnoj inteligenciji koji se bazira na stvaranju sustava koji samostalno uče na temelju iskustva. Umjetna neuronska mreža je zapravo skup jednostavnih međusobno povezanih procesnih elemenata koji oponašaju neurone u ljudskom mozgu i služe distribuiranoj paralelnoj obradi podataka. Neuronske mreže nešto su kompliciranije od ostalih metoda strojnog učenja opisanih u ovom radu i nije ih lako vizualizirati te slijediti njihov postupak odlučivanja.

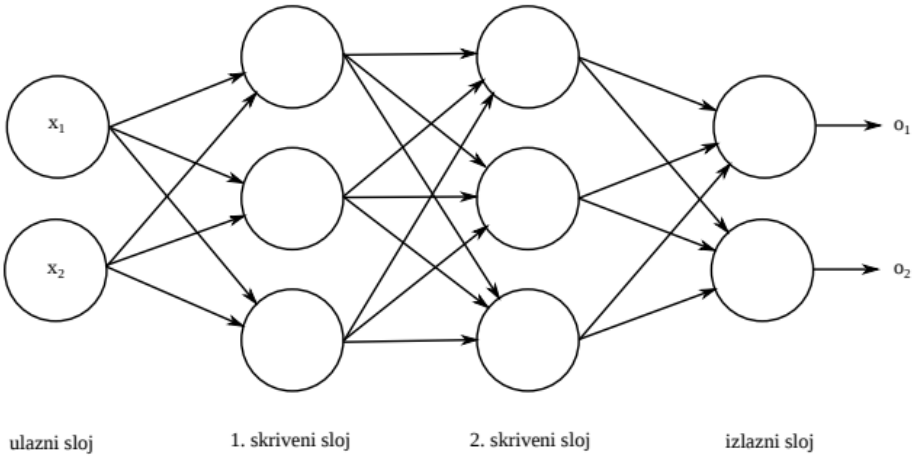
Struktura umjetnog neurona definirana je još 1943. i prikazana je na slici 5.3. Neuron ima više ulaza x_i. Svaki se ulaz množi s pripadnom osjetljivošću w_i te se umnošci sumiraju. Sumi se dodaje pomak w_0. Tako dobivamo akumuliranu vrijednost $\sum_i x_i w_i + w_0$. Ta se vrijednost konačno propušta kroz prijenosnu funkciju neurona i tako dobivamo izlaz iz neurona.



Slika 5.3 Model TLU-perceptrona, preuzeto iz [12]

Neuronsku mrežu definirali smo kao skup međusobni povezanih neurona. Objasnimo sada na koji su način oni povezani i kako međusobno surađuju.

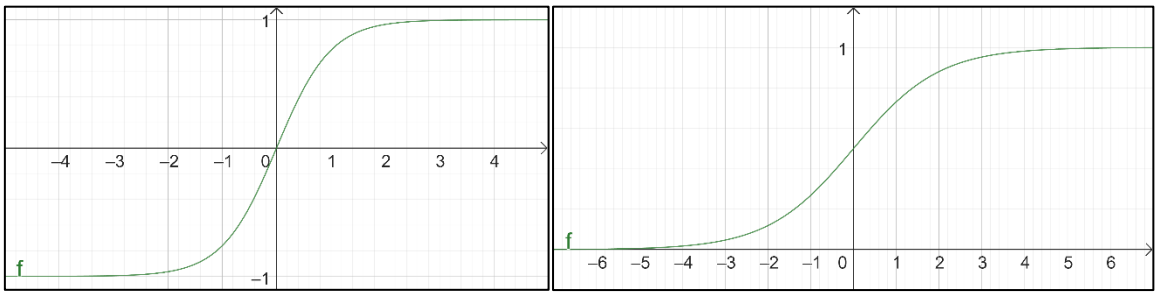
Arhitektura neuronske mreže definira broj slojeva i broj neurona u pojedinom sloju. Jednostavna 2x3x3x2 neuronska mreža prikazana je na slici 5.4. Prvi sloj nazivamo ulazni sloj, posljednji sloj nazivamo izlazni sloj a slojeve između nazivamo skrivenim slojevima. Za neuronsku mrežu kažemo da je unaprijedna ako informacija unutar mreže putuje isključivo od ulaza prema izlazu, odnosno ne postoje povratne veze između neurona. Također kažemo da je mreža potpuno povezana ako se izlaz svakog neurona u jednom sloju prenosi na ulaz svakog neurona sljedećeg sloja, kao na slici 5.4 [12].



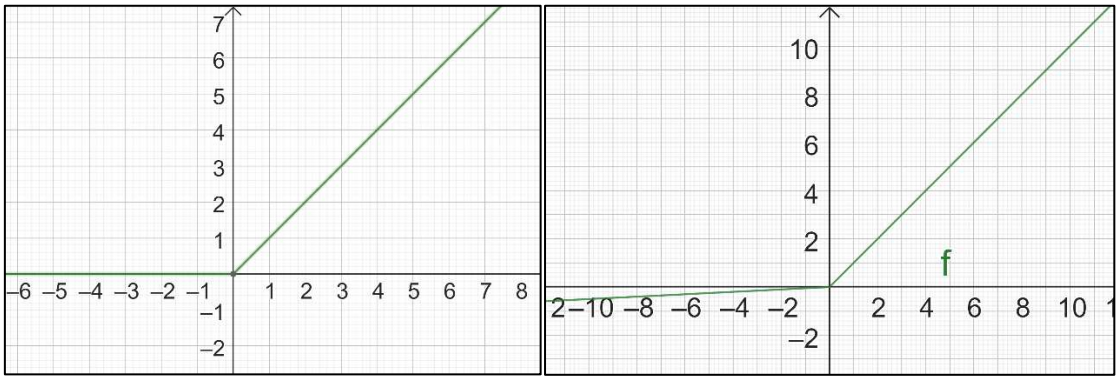
Slika 5.4 Neuronska mreža, preuzeto iz [12]

Kada prijenosne funkcije ne bi postojale, odnosno kada bi prijenosna funkcija svih neurona u mreži bila funkcija identiteta, izlaz iz neuronske mreže bila bi kombinacija linearnih transformacija, što je u konačnici opet linearna transformacija, te bi funkciju cijele mreže mogao obavljati samo jedan neuron. Zato koristimo prijenosne funkcije, da bi povećali ekspresivnost neuronske mreže i modelirali nelinearne odnose. Neke od često korištenih prijenosnih funkcija (slike 5.5, 5.6) su:

1) Sigmoidalna funkcija $\text{sigm}(x) = \frac{1}{1+e^{-x}}$
2) Tangens hiperbolni $\tanh(x) = 2sigm(2x) - 1$
3) Zglobnica $f(x) = \max(0, x)$
4) Propusna zglobnica $f(x) = \begin{cases} \alpha x, x < 0 \\ x, x \geq 0 \end{cases}$
5) Funkcija skoka. $\begin{cases} 0, x \leq 0 \\ 1, x > 0 \end{cases}$



Slika 5.5 tangens hiperbolni (lijevo) i sigmoidalna funkcija (desno)



Slika 5.6 Funkcija zglobnica (lijevo) i propusna zglobnica (desno)

Koji ćemo rezultat dobiti na izlazu iz neuronske mreže određuju njene težine i pragovi. U fazi učenja mreže namještamo težine i pragove tako da dobijemo što bolje rezultate na izlazu.

Tijekom nadziranog učenja skup podataka sastoji se od parova vrijednosti koje šaljemo na ulaz mreže i vrijednosti koje očekujemo na izlazu, u obliku

$$\{(x_{1,1}, \dots, x_{1,N_0}) \to (t_{1,1}, \dots, t_{1,N_1}), \dots, (x_{N,1}, \dots, x_{N,N_0}) \to (t_{N,1}, \dots, t_{N,N_1})\}$$

gdje je N broj primjera, N_0 broj ulaznih varijabli i N_1 broj izlaznih varijabli. Kako bi došli do algoritma za učenje mreže, prvo moramo odrediti funkciju pogreške. Često korištena funkcija je polovična suma srednjeg kvadratnog odstupanja:

$$E = \frac{1}{2} \sum_{i=1}^{N} \frac{1}{N_1} \sum_{j=1}^{N_1} (t_{i,j} - o_{i,j})^2$$

Ako je odabrana funkcija pogreške derivabilna, možemo ju derivirati po svim težinama i pragovima i vidjeti kako će njihovo mijenjanje utjecati na pogrešku. Na tome se temelji postupak učenja neuronskih mreža koji nazivamo propagacijom pogreške unatrag. Računamo parcijalne derivacije funkcije pogreške i na temelju njih korigiramo težine i pragove neuronske mreže. Slijedi opis algoritma u slučaju kada je prijenosna funkcija sigmoida:

1) Sve težine i pragovi postavljaju se na slučajne vrijednosti
2) Dok nije zadovoljen uvjet zaustavljanja:
    1) Za svaki uzorak $(x_{s,1}, \dots, x_{s,N_0}) \to (t_{s,1}, \dots, t_{s,N_1})$ iz skupa primjera
        (1) Računa se izlaz iz mreže za podatak $(x_{s,1}, \dots, x_{s,N_0})$ na ulazu, označimo ga $(o_{s,1}, \dots, o_{s,N_0})$
        (2) Određuje se pogreška neurona izlaznog sloja
        $$\delta_i^K = o_{s,i} \cdot (1 - o_{s,i}) \cdot (t_{s,i} - o_{s,i})$$
        (3) Kretanjem unatrag sloj po sloj do početka mreže računa se pogreška za i-ti neuron u k-tom sloju po formuli:
        $$\delta_i^{(k)} = y_i^{(k)} \cdot \left(1 - y_i^{(k)}\right) \cdot \sum_d w_{i,d} \cdot \delta_d^{(k+1)}$$
        (4) Radi se korekcija svih težina. Težina $w_{i,j}^{(k)}$ korigira se prema izrazu
        $$w_{i,j}^{(k)} \leftarrow w_{i,j}^{(k)} + \eta \cdot y_i^{(k)} \cdot \delta_j^{(k+1)}$$
        a prag $w_{0,j}^{(k)}$ prema izrazu
        $$w_{0,j}^{(k)} \leftarrow w_{0,j}^{(k)} + \eta \cdot \delta_j^{(k+1)}$$

U gore opisanom algoritmu još treba definirati simbol η koji zovemo stopom učenja. On definira u kojoj mjeri će se težine korigirati i bitno je da ne bude premalen kako postupak učenja ne bi bio prespor, a isto tako i ne prevelik kako učenje ne bi divergiralo. Najčešće se koristi pozitivan broj između 0,001 i 0,5. Također, formule su napisane s pretpostavkom da je korištena prijenosna funkcija sigmoide. U slučaju korištenja druge prijenosne funkcije potrebno je dijelove algoritma koji su posljedica derivacije sigmoide ($f \cdot (1 - f)$) zamijeniti odgovarajućom derivacijom korištene prijenosne funkcije.

6. Vrednovanje modela i rezultati

6.1. Stvaranje sjednice u aplikaciji Classification Learner

Za generiranje modela strojnog učenja za klasifikaciju u MATLAB-u koristimo aplikaciju Classification Learner.

6.1.1. Uvođenje podataka

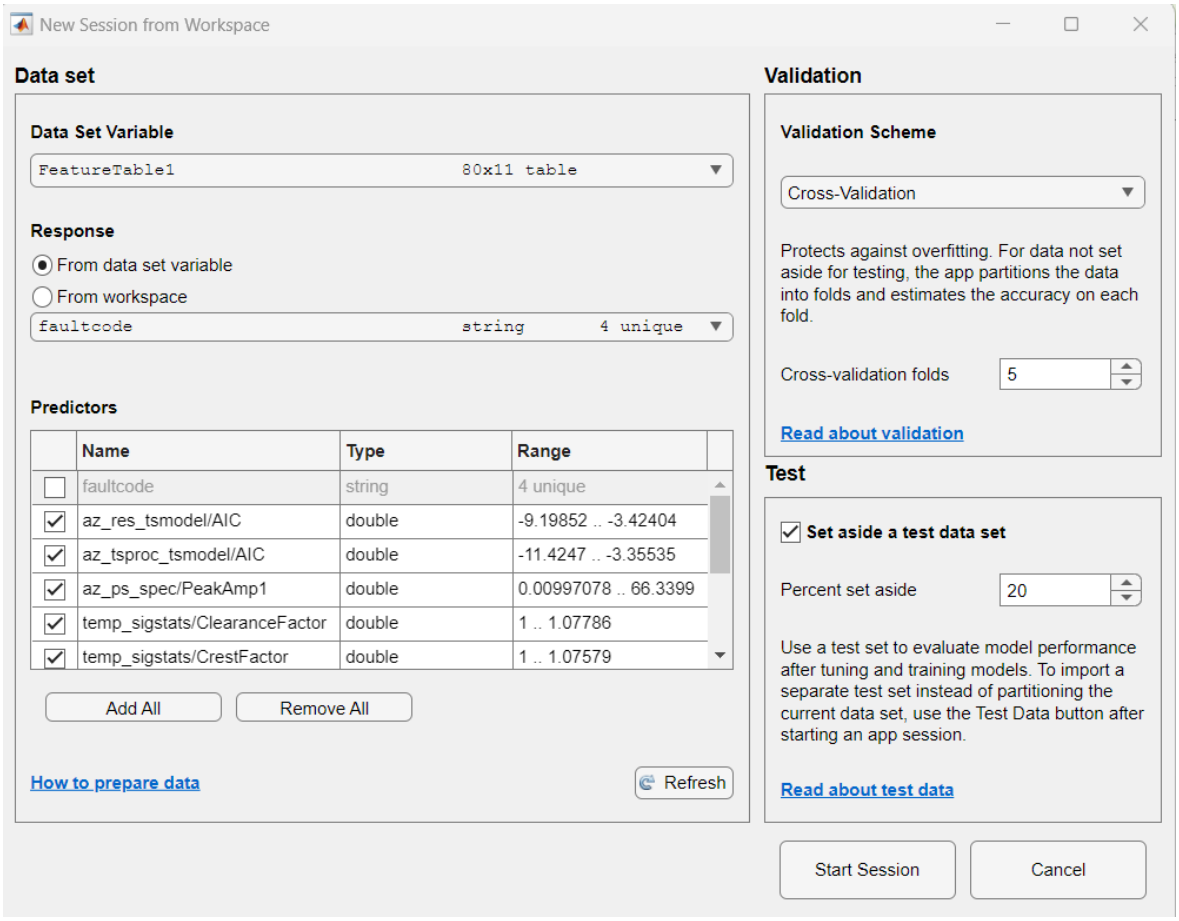
Pokretanjem aplikacije otvara se prozor u kojem biramo opciju "New session", gdje treba odabrati skup podataka s kojim ćemo raditi. Ti su podaci značajke koje smo ranije ekstrahirali i spremili u tablicu značajki. U našem slučaju stupac `faultcode` sadrži tekstualne vrijednosti pa ga Classification Learner prepoznaje kao izlaznu varijablu, tj. varijablu klase. Ostale stupce prepoznaje kao značajke tipa `double` koje će koristiti za predviđanje (slika 6.1).

6.1.2. Odabir načina validacije

Classification Learner također nam nudi da odaberemo način validacije. Osim načina opisanih u poglavlju 5.1.2, Classification Learner nudi nam unakrsnu validaciju u k preklopa (slika 6.1). Skup za učenje dijeli se na k particija, te se u svakoj iteraciji k-ta particija uzima kao skup za provjeru, a ostatak kao skup za učenje. Na kraju se za optimalnu složnost uzima srednja vrijednost k složenosti, točnost na skupu za validaciju uzima se kao srednja vrijednost točnosti preko svih iteracija i model se uči s tom složenošću nad cijelim skupom za učenje. Budući da je skup podataka relativno malen, a dio podataka moramo odvojiti i za testiranje, u ovom radu korištena je unakrsna validacija u 5 preklopa kako bi što bolje odredili složenost modela. Dakle, u svakoj iteraciji skup za provjeru bio je $\frac{1}{k} \cdot 100\%$ skupa za učenje.

6.1.3. Odvajanje podataka za testiranje

Konačno, Classification Learner omogućuje odvajanje dijela podataka sa strane u skup za testiranje. Kod velikih skupova podataka uobičajeno je odvojiti 30-40% podataka u skup za testiranje, no budući da radimo s manjim skupom odvojit ćemo samo 20% podataka, kako bi ostalo dovoljno podataka za učenje modela (slika 6.1).



Slika 6.1 Stvaranje sjednice Classification Learnera

6.2. Stabla odluke

Prva vrsta modela koje ćemo izgraditi su stabla odluke. U Classification Learneru, stabla odluke definirana su sljedećim hiperparametrima:

- Maksimalan broj čvorova, slično maksimalnoj dubini stabla ovaj parametar utječe na kompleksnost modela
- Kriterij za odabir čvora: možemo specificirati kriterij po kojem model odlučuje po kojoj će se značajki podaci dijeliti u svakom čvoru, ponuđeni kriteriji su Ginijev indeks, minimizacija entropije i twoing indeks.
- Korištenje surogatnih značajki: ako su podaci nepotpuni, te primjer koji pokušavamo klasificirati nema definiranu značajku čvora na kojoj se nalazimo, treba li model samo klasificirati primjer u klasu najčešću za taj čvor ili imati definiranu drugu, tzv. surogatnu značajku po kojoj će nastaviti podjelu.

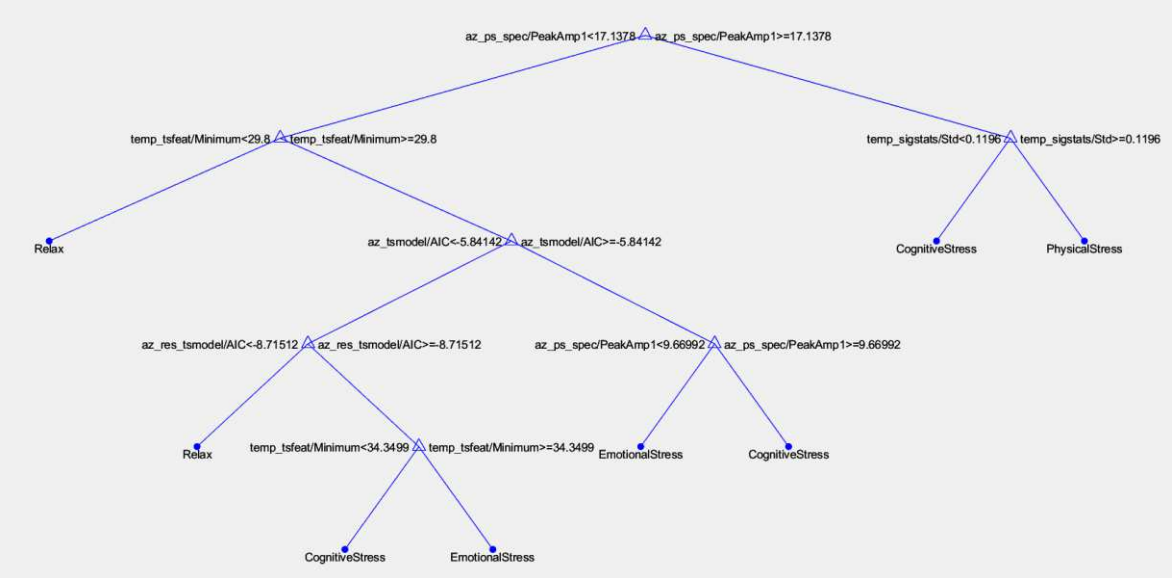
Također su nam ponuđena tri tipa stabla koja možemo automatski generirati:

- *Fine tree*: maksimalan broj čvorova je 100, kao kriterij odabira čvora koristi Ginijev indeks i ne koristi surogatne značajke
- *Medium tree*: maksimalan broj unutarnjih čvorova je 20, kao kriterij odabira čvora koristi Ginijev indeks i ne koristi surogatne značajke
- *Coarse tree*: maksimalan broj unutarnjih čvorova je 4, kao kriterij odabira čvora koristi Ginijev indeks i ne koristi surogatne značajke

Izgradnjom ovih modela možemo vidjeti njihovu uspješnost na skupu za validaciju. *Medium tree* i *fine tree* imaju postotak točnosti na skupu za validaciju od 78,1%, dok *coarse tree* ima nešto veću točnost od 79,1%. S obzirom na jednak rezultat, postoji mogućnost da su *medium* i *fine tree* zapravo jednaka stabla, što je moguće jer je gornja granica na broj čvorova za *medium tree* jednaka 20, a teško da je na ovako malom skupu podataka moguće izgraditi tako veliko stablo. Nakon izvoza modela u MATLAB-ov radni prostor naredbom

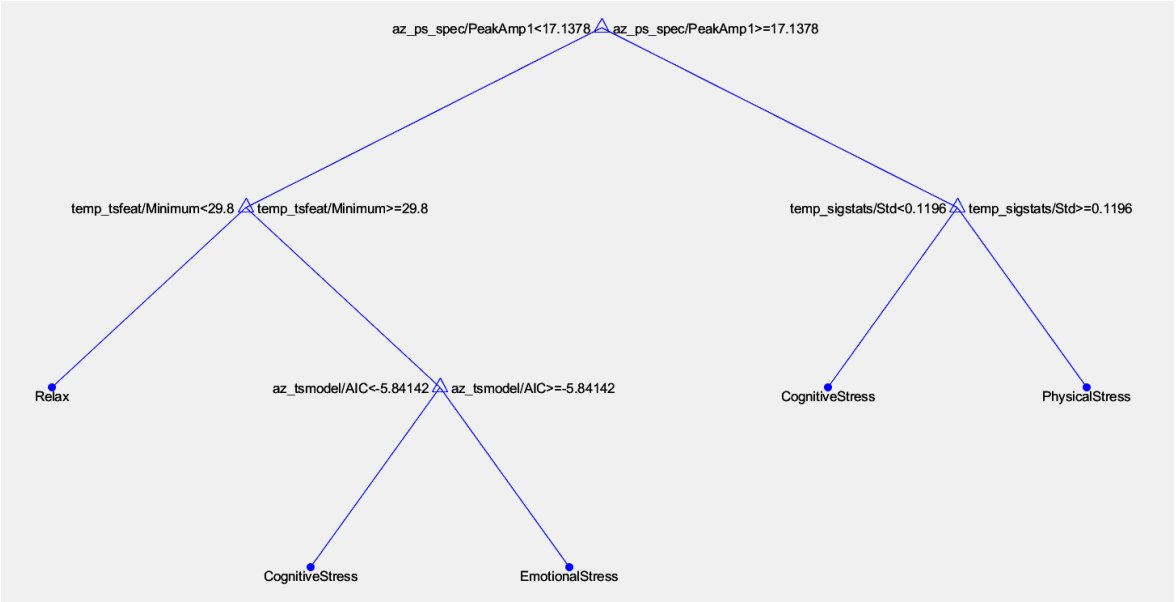
```
>> view(treeModel.ClassificationTree,"Mode","graph")
```

Možemo dobiti grafički prikaz stabla te se uvjeriti da su *fine tree* i *medium tree* generirali identično stablo koje je prikazano na slici 6.2. Stablo nije pretjerano složeno, sastoji se od svega sedam unutarnjih čvorova i osam listova.



Slika 6.2 *Fine tree* i *medium tree*

S druge strane, *coarse tree* ograničeno je na maksimalno četiri unutarnja čvora i generirano stablo prikazano je na slici 6.3. Očekivano, radi se o podrezanom stablu sa slike 6.2 koje ide do dubine 3. Model je jednostavniji i nadamo se da će to rezultirati boljom generalizacijom.

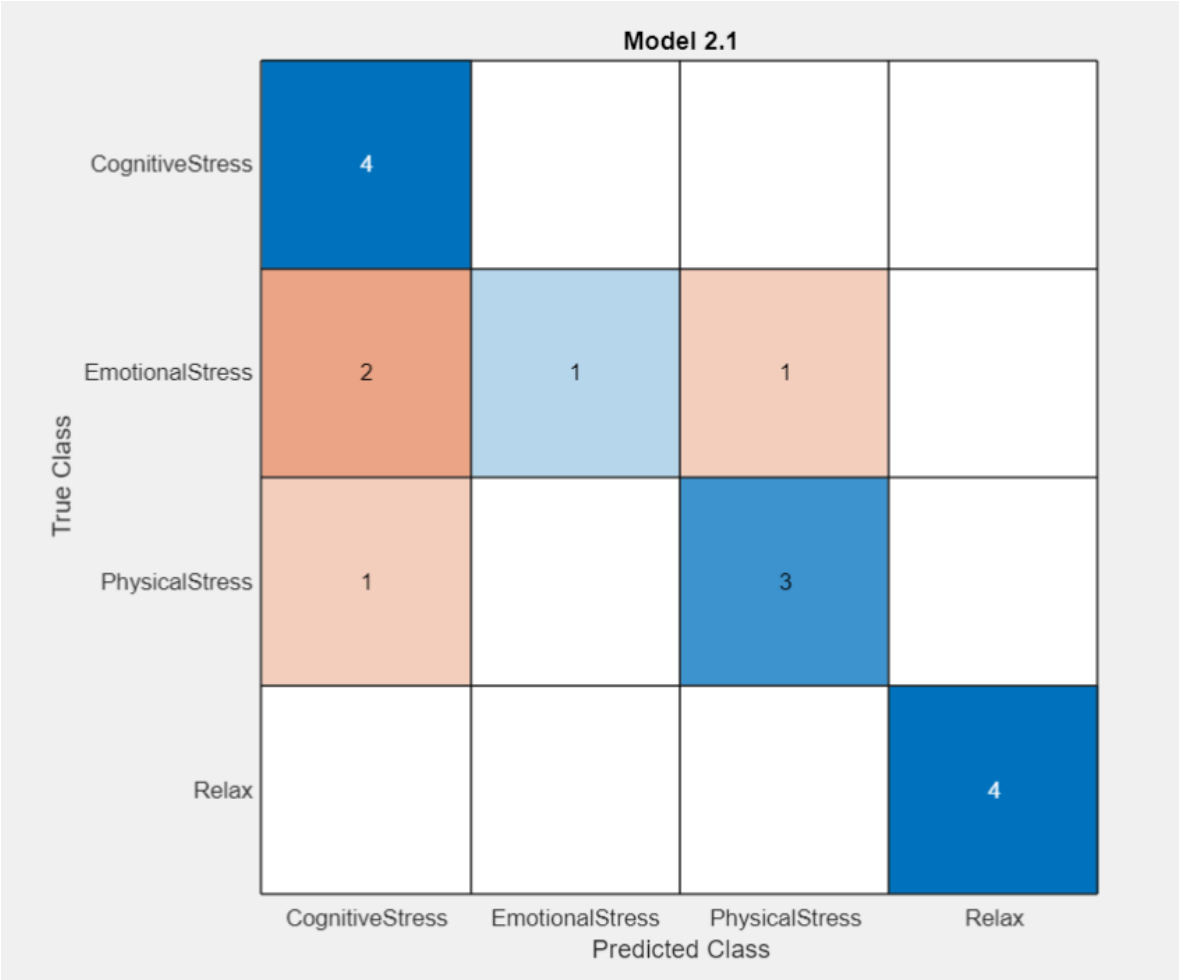


Slika 6.3 *Coarse tree*

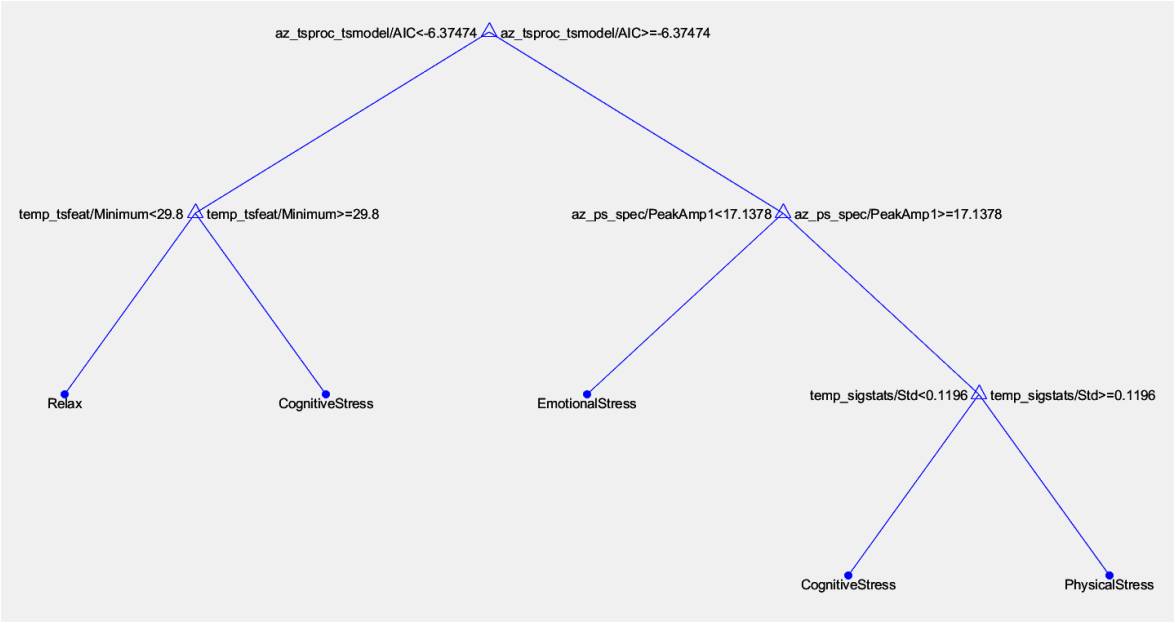
Pri stvaranju sjednice odvojili smo 20% podataka u skup za testiranje. Sada ćemo te podatke iskoristiti da bi testirali naše modele. Krenut ćemo od modela *medium tree* budući da je jednak modelu *fine tree* i imao je nešto nižu točnost na skupu za validaciju. Testiranjem dobivamo točnost 75% i matricu zbunjenosti sa slike 6.4. Provođenjem testa na modelu *coarse tree* koji je jednostavniji nažalost ne dobivamo ništa bolje rezultate, štoviše rezultati su potpuno jednaki: 75% točnosti i jednaka matrica konfuzije.

Classification Learner nudi nam još jednu opciju: optimirana stabla. Odabirom ove opcije možemo odabrati hiperparametre modela koje želimo optimirati, a aplikacija će izgraditi nekoliko modela s različitim vrijednostima tih hiperparametara i reći nam koji je ostvario najbolje rezultate na testu za validaciju. Za optimizaciju su odabrani hiperparametri maksimalan broj čvorova i kriterij za odabir čvora. Izgrađeno je stablo s četiri čvora, a čvorovi su birani kriterijem minimizacije entropije, što je rezultiralo u drugačijem stablu od do sada prikazanih (slika 6.5). Točnost na skupu za validaciju bila je ponovno 79,7%, no točnost na skupu za testiranje poboljšala se na 81,2%, matrica zbunjenosti prikazana je na slici 6.6. Analiziramo li značajke korištene u pojedinim čvorovima, na nekim mjestima

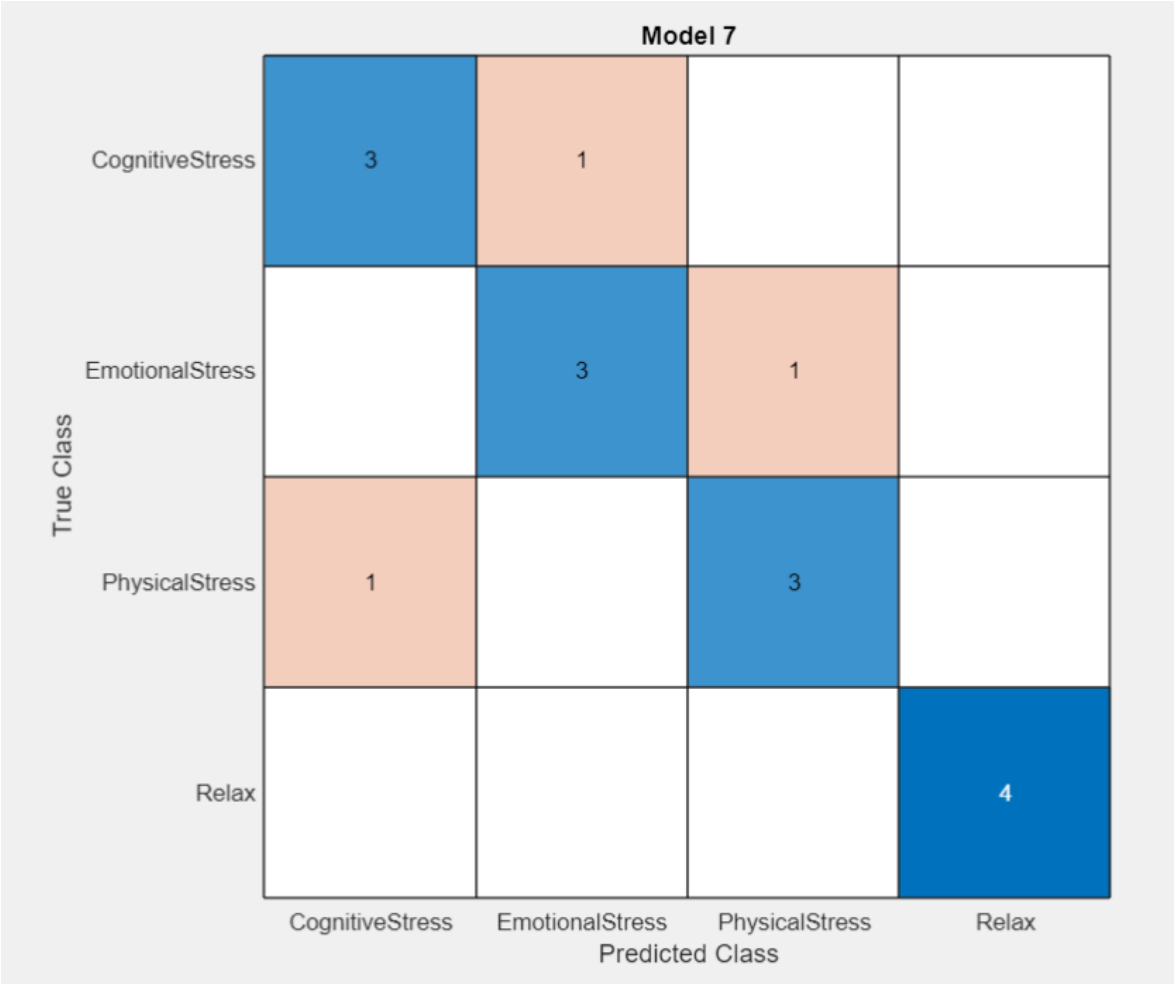
možemo pronaći dosta smislene odluke. Na primjer, na lijevoj strani optimiranog stabla u ovisnosti o minimalnoj temperaturi stanje se klasificira u mirovanje u slučaju niže ili kognitivni stres u slučaju više temperature, što ima smisla budući da bi tijelo u stanju mirovanja trebalo doseći nižu minimalnu temperaturu nego kada je pod stresom. Temperatura se ponovno koristi u krajnjem desnom čvoru istog stabla, ovaj puta se na osnovi standardne devijacije odlučuje između kognitivnog u slučaju manje i fizičkog stresa u slučaju veće vrijednosti. Očekujemo da će tijelo u fazi fizičkog stresa doživjeti veće promijene temperature nego u kognitivnom stresu, pa odluka ponovno ima smisla.



Slika 6.4 Matrica konfuzije za tri modela stabla odluke, točnost 75%



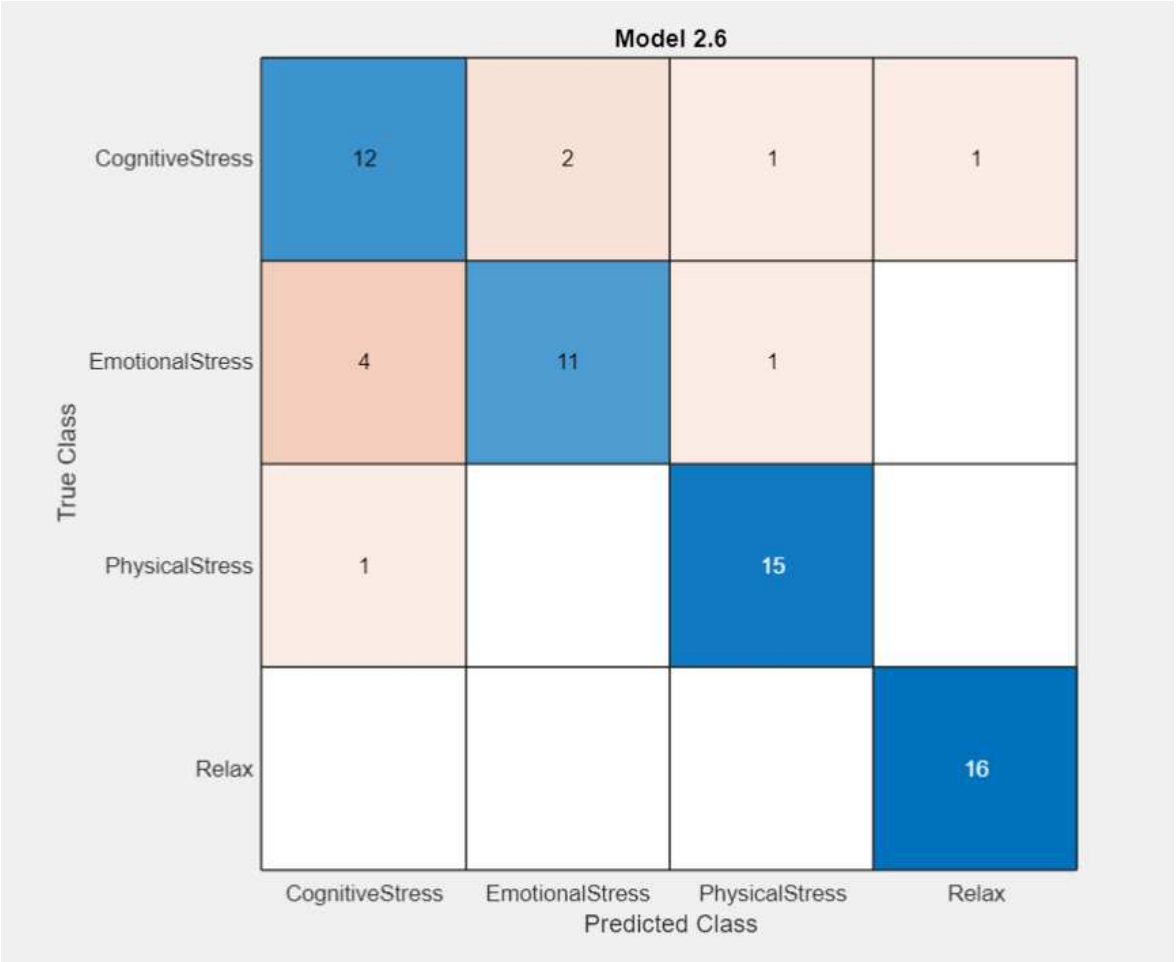
Slika 6.5 Optimirano stablo odluke



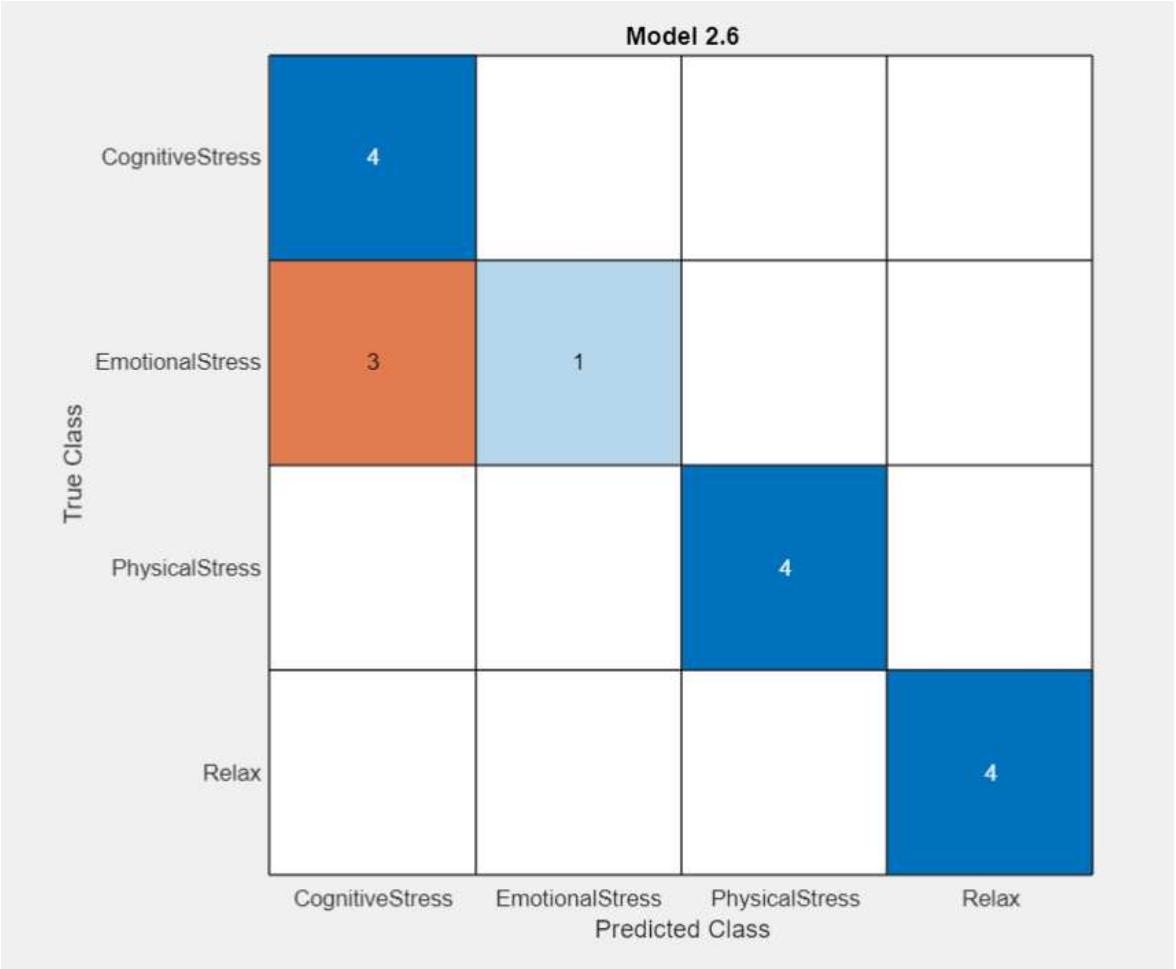
Slika 6.6 Matrica zbunjenosti za optimirano stablo odluke, točnost 81,2%

6.3. Gaussov naivni Bayes

Sljedeći model koji ćemo izgraditi je naivni Bayesov klasifikator korištenjem normalne distribucije. Ovaj model je puno manje fleksibilan, štoviše ne možemo mijenjati (pa tako ni optimirati) niti jedan njegov hiperparametar. Unatoč tome, generirani model dao je prilično dobre rezultate na skupu za validaciju: točnost od 84,4%. Matrica zbunjenosti na skupu za validaciju prikazana je na slici 6.7. Čini se da model ima problem s razlikovanjem kognitivnog i emocionalnog stresa: šest od deset pogrešaka upravo su zamjene tih dviju klasa, a četiri od tih šest slučajeva su emocionalni stresovi pogrešno klasificirani kao kognitivni. Testiranjem modela uvjeravamo se da je ovo zaista problem. Klasifikator je ostvario za sada najbolju točnost od 81,2%, a sve tri pogreške bile su upravo emocionalni stresovi pogrešno klasificirani kao kognitivni. Kada bi na raspolaganju imali veći skup podataka bilo bi zanimljivo vidjeti je li ovakvo ponašanje posljedica nedostatka primjera ili nečeg drugog, npr. sličnost između tih vrsta stresa ili načina na koji su subjekti dovedeni u stresno stanje te mjernih instrumenata.



Slika 6.7 Matrica zbunjenosti na skupu za validaciju naivnog Bayesovog klasifikatora



Slika 6.8 Matrica zbunjenosti na testnom skupu naivnog Bayesovog klasifikatora

6.4. Neuronske mreže

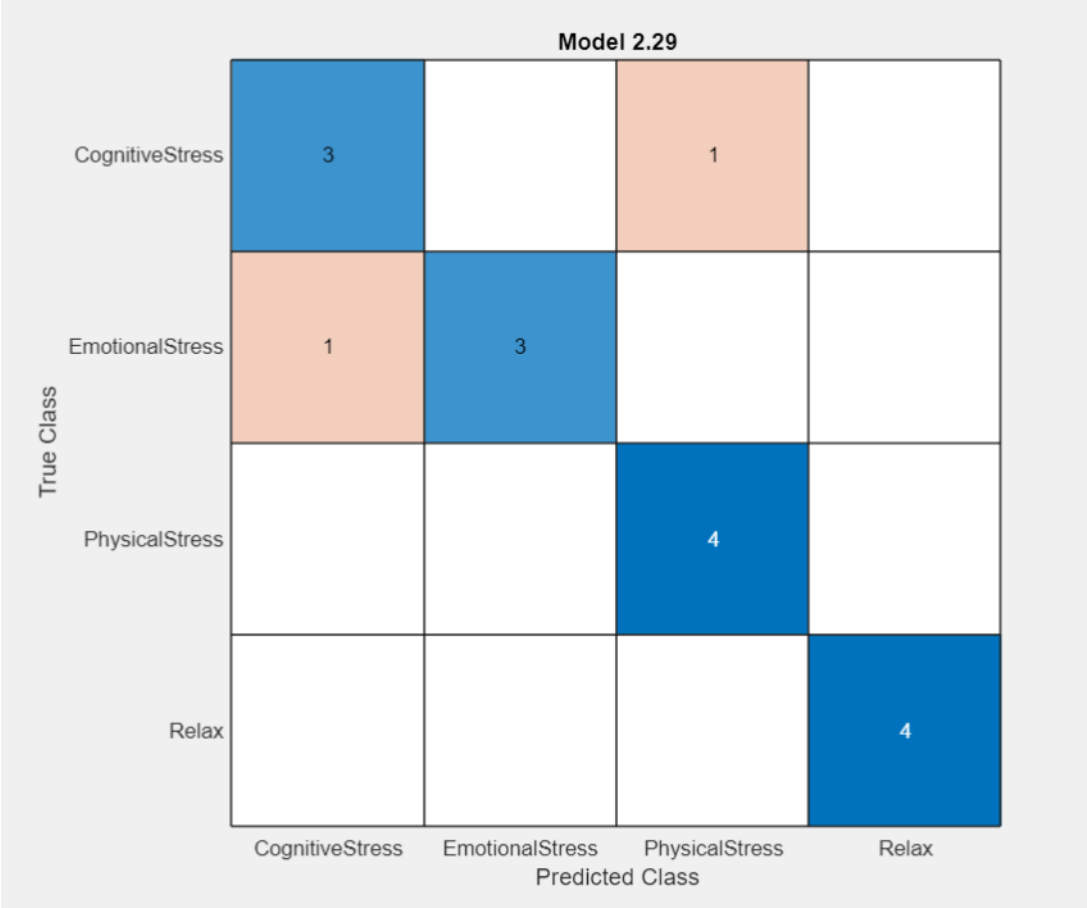
Classification Learner nudi nam velik izbor opcija u radu s neuronskim mrežama. Neuronske mreže mnogo su fleksibilnije od ranije opisanih klasifikatora te su hiperparametri koje možemo mijenjati:

- Broj skrivenih slojeva neuronske mreže, cijeli broj između 1 i 3 uključeno.
- Broj neurona u svakom od skrivenih slojeva
- Izlazna funkcija, odabir između ReLU, sigmoide, tangensa hiperbolnog i funkcije identiteta
- Maksimalan broj iteracija
- Hoće li se podaci standardizirati

Ponuđeno nam je nekoliko predefiniranih mreža:

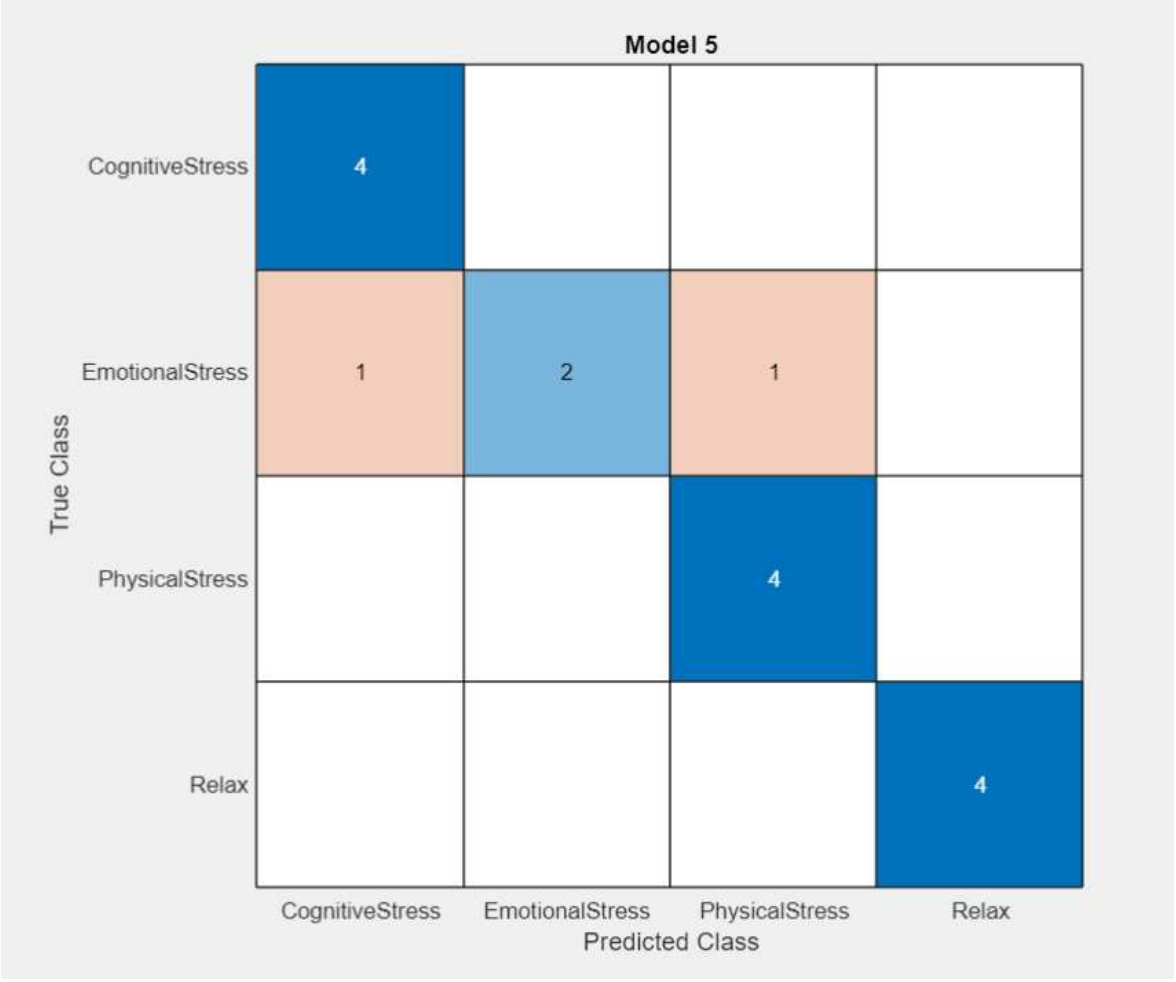
- Uska neuronska mreža: jedan skriveni sloj od 10 neurona
- Srednja neuronska mreža: jedan skriveni sloj od 25 neurona
- Široka neuronska mreža: jedan skriveni sloj od 100 neurona
- Dvoslojna neuronska mreža: dva skrivena sloja od po 10 neurona
- Troslojna neuronska mreža: tri skrivena sloja od po 10 neurona

Sve predefinirane mreže koriste prijenosnu funkciju zglobnice (ReLU) te standardiziraju podatke. Na skupu za validaciju najbolje su se pokazale široka i srednja neuronska mreža s točnošću od 82,8%. Nešto manju točnost ostvaruju dvoslojna i troslojna mreža s 81,2% točnosti, dok je najmanju točnost od 79,7% imala uska mreža. No, budući da su sve točnosti relativno blizu, testirat ćemo svih 5. Najbolje rezultate na testnom skupu ostvarila je troslojna neuronska mreža s 87,5%. Njena je matrica zbunjenosti prikazana na slici 6.9. Sve su ostale mreže ostvarile jednaku točnost od 81,2%, ali s različitim matricama zbunjenosti.



Slika 6.9 Matrica zbunjenosti troslojnog stabla na testnom skupu

Još je ostalo pokušati optimirati hiperparametre mreže. Hiperparametri koje ćemo optimirati su broj skrivenih slojeva, broj neurona u pojedinom skrivenom sloju te prijenosnu funkciju. Optimizacija je rezultirala izgradnjom mreže s jednim skrivenim slojem od 11 neurona koji koriste izlaznu funkciju sigmoide. Točnost na skupu za provjeru bila je 84,4%, dok je točnost na skupu za testiranje ponovno bila visokih 87,5%. Matrica zbunjenosti prikazana je na slici 6.10. Iako jednoslojna mreža s 10 neurona nije davala najbolje rezultate, dodavanje jednog neurona i korištenje druge prijenosne funkcije oni su se znatno poboljšali.



Slika 6.10 Matrica zbunjenosti optimirane mreže na testnom skupu

Zaključak

Cilj ovog rada bio je korištenje algoritama strojnog učenja za klasifikaciju biomedicinskih signala u neurološka stanja stresa. Korišten je javno dostupan skup podataka Non-EEG Dataset for Assessment of Neurological Status [1]. Opisan je način na koji su podaci prikupljani i kako su transformirani u željeni oblik. Objašnjene su ideje i standardne metode strojnog učenja Proučavan je manji skup algoritama strojnog učenja i to preciznije, klasifikacije: stabla odluke, naivni Bayesov klasifikator i neuronske mreže. Za učenje modela korištena je platforma MATLAB. Tamo su podaci uvedeni, transformirani te su iz njih izlučene značajke. Modeli su generirani, provjereni te konačno testirani nad podacima i njihova uspješnost je uspoređena. Zaključak koji mogu izvesti po završetku pisanja ovog rada je da strojno učenje ima velik potencijal, ali bitno je znati ga iskoristiti na pravi način. Sama količina različitih metoda koje su na raspolaganju, u kombinaciji sa svim načinima na koje se parametri modeli mogu podešavati daje nam sliku o tome koliko mogućnosti imamo. Naravno, problem je u tome što pri odabiru modela ne možemo isprobati sve kombinacije jer je vremenski neisplativo, a i neizvedivo. Dobro poznavanje teorije i matematičkih podloga algoritama u kombinaciji s iskustvenim znanjem jedino je što nam daje mogućnost iskorištavanja punog potencijala strojnog učenja.

Literatura

[1] PhysioNet. Non-EEG Dataset for Assessment of Neurological Status. Poveznica: https://www.physionet.org/content/noneeg/1.0.0/ pristupljeno 2023-04-21

[2] Waveform Database. wfdb.github.io. Poveznica: https://wfdb.io/ pristupljeno 2023-04-21

[3] Mathworks. MATLAB. Poveznica: https://www.mathworks.com/products/matlab.html pristupljeno 2023-05-05

[4] Mathworks. Diagnostic Feature Designer. Poveznica: https://www.mathworks.com/help/predmaint/ref/diagnosticfeaturedesigner-app.html pristupljeno 2023-05-14

[5] Mathworks. Classification Learner. Poveznica: https://www.mathworks.com/help/stats/classificationlearner-app.html pristupljeno 2023-05-14

[6] J. Birjandtalab, D. Cogan, M. B. Pouyan, M. Nourani *A non-EEG Biosignals Dataset for Assesment and Visualisation of Neurological Status*. 2016 IEEE International Workshop on Signal Processing Systems, pp. 110-114, doi: 10.1109/SiPS.2016.27

[7] PhysioNet. WFDB Toolbox for Matlab. Poveznica: https://archive.physionet.org/physiotools/matlab/wfdb-swig-matlab/new_version.shtml#:~:text=For%20Quick%20installation%20of%20the,%3E%3E%20unzip('WFDB_Toolbox_0_0_3 pristupljeno 2023-05-05

[8] B. D. Bašic, M. Čupić, J. Šnajder, Uvod u umjetnu inteligenciju, Materijali s predavanja, Sveucilište u Zagrebu, Fakultet elektrotehnike i računarstva, 10. Strojno učenje, Akademska godina 2019./2020.

[9] Mathworks. Help. Poveznica: https://www.mathworks.com/help/stats/choose-a-classifier.html#bunt0ky pristupljeno 2023-05-05

[10] V. Rohan. Gaussian Naive Bayes: What You Need To Know, Upgrad, (2022, veljača). Poveznica: https://www.upgrad.com/blog/gaussian-naive-bayes/ pristupljeno 2023-05-20

[11] P. Jadhav. Handling Continous Variables in Decision Trees. Poveznica https://medium.com/geekculture/handling-continuous-attributes-in-decision-trees-bbc044986621 pristupljeno 2023-05-25

[12] B. D. Bašic, M. Čupić, J. Šnajder, Uvod u umjetnu inteligenciju, Materijali s predavanja, Sveucilište u Zagrebu, Fakultet elektrotehnike i računarstva, 11. Umjetne neuronske mreže, Akademska godina 2019./2020.

Sažetak

U ovom radu korišteni su algoritmi strojnog učenja za klasifikaciju biomedicinskih signala u neurološka stanja stresa. Korišten je javno dostupan skup podataka Non-EEG Dataset for Assessment of Neurological Status [1]. Opisan je način na koji su podaci prikupljani i kako su transformirani u željeni oblik. Objašnjene su ideje i standardne metode strojnog učenja Proučavan je manji skup algoritama strojnog učenja i to preciznije, klasifikacije: stabla odluke, naivni Bayesov klasifikator i neuronske mreže. Za učenje modela korištena je platforma MATLAB. Tamo su podaci uvedeni, transformirani te su iz njih izlučene značajke. Biomedicinski signali koji su se pokazali najbitnijima za klasifikaciju signala bili su temperatura i akceleracija u z osi te njihove značajke poput minimuma i standardne devijacije (izvedenih iz temperature) te vrhunca amplitude na spektru snage i Akaikeovog kriterija informacije (izvedenih iz akceleracije u z osi). Modeli su naučeni, provjereni te konačno testirani nad podacima i njihova uspješnost je uspoređena. Točnost od 87,5% na testnom skupu dala je unaprijedna neuronska mreža s tri skrivena sloja, što je ujedno bio i najbolji rezultat.

Ključne riječi: strojno učenje, klasifikacija, stabla odluke, naivni Bayesov klasifikator, neuronske mreže, biomedicinski signali, stres

Summary

In this paper, machine learning algorithms were used for the classification of biomedical signals into neurological conditions of stress. The publicly available dataset used was the Non-EEG Dataset for Assessment of Neurological Status [1]. The method of data collection and its transformation into the desired format was described. The ideas and standard methods of machine learning were explained. A small set of machine learning algorithms was studied, specifically decision trees, naive Bayes classifier, and neural networks. The MATLAB platform was used for training the models. The data was imported, transformed, and features were extracted from it. The biomedical signals that proved to be crucial for classification were temperature and acceleration on the z-axis and their features such as minimum value and standard deviation (derived from temperature) and peak amplitude on the power spectrum and Akaike's information criterion (derived from acceleration on the z-axis). The models were generated, verified, and finally tested on the data, and their performance was compared. The highest accuracy on the test of 87.5% was achieved by a feedforward neural network with three hidden layers.

Keywords: machine learning, classification, decision trees, naive Bayes classifier, neural networks, biomedical signals, stress