Detekcija prisustva i broja ljudi u Wi-Fi polju

Hrboka, Tonka

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Tonka Hrboka

Detekcija prisustva i broja ljudi u Wi-Fi polju

Diplomski rad

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER ISTRAŽIVAČKI

Tonka Hrboka

Diplomski rad

Detekcija prisustva i broja ljudi u Wi-Fi polju

Voditelj diplomskog rada: doc. dr. sc. Dario Jukić

Suvoditelj diplomskog rada: prof. dr. sc. Hrvoje Buljan

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2023.

Zahvaljujem mentorima na razumijevanju i usmjeravanju tijekom izrade ovoga rada.

Beskrajno sam zahvalna prijateljima fizičarima uz koje je lakše bilo preskočiti sve prepreke, pogotovo hvala Mateju za najbolja zajednička učenja, AMB koja je uvijek čuvala utičnicu u knjižnici, Glogaru koji je slušao kad god sam htjela i Gašpi (+1) bez kojeg se nikad ne bismo družili. Također hvala prijateljima iz Zadra koji su ostali uz mene sve ove godine i trpjeli moje ludosti. Posebno hvala roditeljima na bezuvjetnoj potpori i svemu što su mi omogućili; mami na savjetima, a tati na smirenosti. Bez vas ništa ne bi bilo moguće.

Sažetak

Detekcija prisutstva ljudi u zatvorenom prostoru neophodna je za sigurnost u javnim ustanovama ili kućanstvima, a uobičajeno se obavlja kamerama koje imaju niz mana. Predložena je alternativna metoda detekcije ljudi Wi-Fi poljem koja ne narušava privatnost te izvedba nije skupa, štoviše većina ljudi već posjeduje Wi-Fi odašiljače. Koristi se činjenica da je relativna permitivnost ljudskog tijela na Wi-Fi frekvenciji visoka, stoga ljudi samim stajanjem u Wi-Fi polju mijenjaju njegova svojstva, bez potrebe za nošenjem nekog drugoog uređaja.

Strojnim učenjem obrađeni su podatci dobiveni eksperimentom u kojem su za različite veličine skupina ljudi u prostoriji mjerene RSSI vrijednosti Wi-Fi polja. Cilj je dobiti model koji će biti u mogućnosti predvidjeti broj ljudi u prostoriji na temelju neviđenih RSSI vrijednosti. Iznesena je fizika elektromagnetskih valova i njihove interakcije s materijom te osnove propagacije valova u komunikacijskim sustavima. Opisana je teorija algoritama plitkog učenja: logističke regresije, stroja potpornih vektora, *k* najbližih susjeda, stabla odluke te algoritma slučajnih šuma, a opisane su i osnove rada neuralnih mreža.

Dobivene su vrlo visoke točnosti za predviđanje broja ljudi u prostoriji svih korištenih algoritama, preko 96 % za sve korištene plitke modele te preko 98 % za sve korištene neuralne mreže, a najveća točnost klasifikacije na neviđenim podatcima iznosi 99.2 %.

Ključne riječi: strojno učenje, elektromagnetski val, dielektrik, Wi-Fi polje, RSSI vrijednost, nadzirano učenje, klasifikacija, linearni diskriminativni model, neparametarski model, neuralna mreža.

Detection of the presence and number of people in the Wi-Fi field

Abstract

Detection of the presence of people indoors is imperative for security in both public institutions and households. This is typically achieved through the use of cameras, which possess several drawbacks. An alternative method for detecting people using a Wi-Fi field is proposed. This approach respects privacy, is cost-effective, particularly given that the majority of individuals already possess Wi-Fi transmitters. It leverages the fact that the human body exhibits a high relative permittivity at Wi-Fi frequencies. Thus, merely by standing in a Wi-Fi field, individuals inadvertently alter its properties, obviating the need for additional devices.

The data gathered from an experiment measuring RSSI values of the Wi-Fi field for various group sizes in a room were subjected to machine learning. The aim is to develop a model capable of predicting the number of people in the room based on unseen RSSI values. The text provides an overview of the physics of electromagnetic waves, their interaction with matter, and the fundamentals of wave propagation in communication systems. It also outlines the theory of shallow learning algorithms, such as logistic regression, support vector machines, k nearest neighbors, decision trees, and the random forest algorithm. Furthermore, it offers an introduction to the basics of neural networks.

The results indicate very high prediction accuracies for the number of people in the room using all the mentioned algorithms. Specifically, all shallow models achieved accuracies exceeding 96 %, while neural networks surpassed 98 %. The highest classification accuracy on unseen data reached 99.2 %.

Keywords: machine learning, electromagnetic wave, dielectric, Wi-Fi field, RSSI value, supervised learning, classification, linear discriminative model, non-parametric model, neural network.

Sadržaj

1 Uvod

Današnja tehnologija dosegla je velike uspjehe u detekciji predmeta i ljudi te se već rutinski koristi u autonomnim vozilima, videonadzoru pa čak i u sportu. Metode detekcije ljudi uglavnom se temelje na kamerama i računalnom vidu, na LiDAR i radar senzorima koji su cjenovno nedostupni za kućanstva ili GPS senzorima, prvenstveno namijenjenima za uporabu na otvorenom prostoru. U zatvorenim prostorima kao jedini adekvatan način detekcije ljudi preostaje kamera, čija uporaba ima niz mana. Vidno polje kamere može biti ograničeno, pogotovo u zatvorenom prostoru jer je u pravilu gustoća predmeta veća. Za rad kamere potrebna je svjetlost, a njihovo korištenje u svrhu videonadzora nije toliko popularno i zbog narušavanja privatnosti. Ovaj rad bavit će se alternativnom metodom detekcije ljudi u zatvorenom prostoru koja nema prethodno navedene mane; ideja je koristiti Wi-Fi polje. Wi-Fi odašiljači široko su rasprostranjeni, polje je dovoljno velikog dosega, prolazi nepromijenjeno kroz predmete, ali se značajno mijenja prolaskom kroz ljudsko tijelo što ga čini savršenim za ovakvu uporabu. Bitno je naglasiti da ljudi ne moraju na sebi nositi nikakav uređaj kako bi bili detektirani, što širi mogućnosti praktične uporabe ove tehnologije, od detekcije uljeza u školama, zdravstvenim ustanovama i privatnim domovima, do prebrojavanja ljudi na okupljanjima.

Poznata je činjenica da ljudsko tijelo na Wi-Fi frekvenciji ima visok koeficijent dielektrične permitivnosti te se ta pojava široko istraživala jer može stvarati problem za primarnu uporabu Wi-Fi polja gdje ljudsko tijelo zasjenjuje signal pri prijenosu informacija[1, 2, 3]. Osim što je ljudsko tijelo u Wi-Fi polju istraženo u kontekstu smetnje, postoje i novija istraživanja koja se bave detekcijom ljudi te se uglavnom dijele na one koji koriste RSSI vrijednosti Wi-Fi polja[4, 5, 6] i one koji koriste CSI vrijednosti[7, 8]. Osim same detekcije ljudi, provedena su i naprednija istraživanja u kojima se korištenjem Wi-Fi polja prepoznaju pozicije ljudskog tijela u prostoru i njihovog stava[9, 10], detektiralo se jesu li ljudi u opasnosti pomoću njihovog stava[11], a postignuta je i identifikacija ljudi[12, 13].

U radu je provedena analiza eksperimentalno dobivenih podataka RSSI vrijednosti Wi-Fi polja u prostoriji u kojoj su se nalazili različiti brojevi ljudi. Promatralo se koliko se uspješno neki od osnovnih algoritama strojnog učenja nose s problemom detekcije ljudi na temelju tih podataka.

2 Teorijska pozadina

2.1 Fizika elektromagnetskih valova

Elektromagnetski valovi su oscilacije električnih i magnetskih polja koje se mogu širiti kroz vakuum i razne materijale. Nisu samo čest predmet proučavanja, već i ključan alat u tehnološkoj industriji i u znanosti. Osim u komunikaciji, elektromagnetski valovi koriste se i u medicinskoj dijagnostici i zahvatima, satelitskoj navigaciji, mikrovalnim pećnicama, sigurnosnim skenerima na aerodromima te imaju široku primjenu u fizici materijala, nuklearnoj fizici i astronomiji.

2.1.1 Valna jednadžba

Temelj svakog problema u elektrodinamici su Maxwellove jednadžbe:

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0} \tag{2.1}$$

$$\nabla \cdot \vec{B} = 0 \tag{2.2}$$

$$\nabla \times \vec{E} = -\partial_t \vec{B} \tag{2.3}$$

$$\nabla \times \vec{B} = \mu_0 \left(J + \epsilon_0 \partial_t \vec{E} \right) \tag{2.4}$$

Korištenjem definicije vektorskog laplasijana[14]:

$$\Delta \vec{X} = \nabla \cdot (\nabla \cdot \vec{X}) - \nabla \times (\nabla \times \vec{X}) \tag{2.5}$$

na električnom polju u vakuumu:

$$\Delta \vec{E} = -\nabla \times (\nabla \times \vec{E}) = \nabla \times (\partial_t \vec{B}) = \partial_t (\nabla \times \vec{B}) = \mu_0 \epsilon_0 \partial_t^2 \vec{E} \tag{2.6}$$

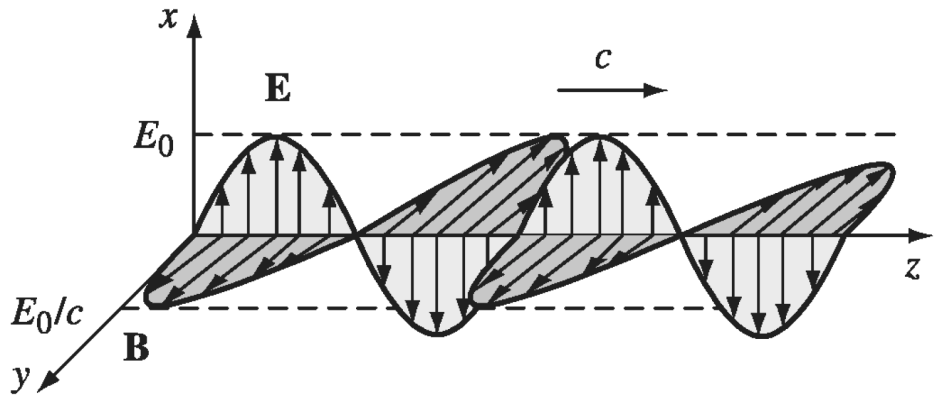
jednostavno se dolazi do valne jednadžbe, koristeći Maxwellove jednadžbe (2.3) i (2.4). Analognim postupkom, koristeći iste jednadžbe suprotnim redoslijedom, dobije se i valna jednadžba za magnetsko polje:

$$\Delta \vec{B} = -\nabla \times (\nabla \times \vec{B}) = -\mu_0 \epsilon_0 \nabla \times (\partial_t \vec{E}) = -\mu_0 \epsilon_0 \partial_t (\nabla \times \vec{E}) = \mu_0 \epsilon_0 \partial_t^2 \vec{B} \tag{2.7}$$

Konačno se može uvrstiti i definicija brzine svjetlosti $c = \frac{1}{\sqrt{\epsilon_0 \mu_0}}$:

$$\begin{aligned} \partial_t^2 \vec{E} &= c^2 \Delta \vec{E} \\ \partial_t^2 \vec{B} &= c^2 \Delta \vec{B} \end{aligned} \tag{2.8}$$

Za ravne valove može se pokazati da su smjerovi električnog i magnetskog polja međusobno okomiti te su također okomiti na smjer propagacije, prikazano na Slici 2.1.



Slika 2.1: Prikaz ravnog elektromagnetskog vala u vakuumu, preuzeto iz [15]

2.1.2 Dielektrici

Dielektrici su materijali u kojima su elektroni snažno vezani za kristalnu rešetku, stoga se ne mogu puno pomicati, odnosno raspodjela naboja može se mijenjati samo unutar atoma ili molekule. Postavljanjem dielektrika u vanjsko električno polje događa se spomenuta preraspodjela naboja te se materijal polarizira, pri čemu je dipolni moment $\vec{p}$ jednog atoma u izotropnoj aproksimaciji proporcionalan nametnutom vanjskom polju $\vec{E}$[15]:

$$\vec{p} = \alpha \vec{E} \tag{2.9}$$

pri čemu je konstanta proporcionalnosti α atomska polarizabilnost i ovisi o konkretnom materijalu, odnosno strukturi atoma koji se promatra.

S druge strane ako se problem promatra na makroskopskoj razini, definiranjem polarizacije $\vec{P}$ kao volumne gustoće električnih dipolnih momenata, za izotropni linearni dielektrik vrijedi relacija[15]:

$$\vec{P} = \epsilon_0 \chi_e \vec{E} \tag{2.10}$$

gdje je konstanta proporcionalnosti χ_e nazvana električna susceptibilnost. Zbog korisnosti u rješavanju praktičnih problema često se uvodi i polje električnog pomaka[15]:

$$\vec{D} = \epsilon_0 \vec{E} + \vec{P} = \epsilon_0 \vec{E} + \epsilon_0 \chi_e \vec{E} = \epsilon_0 (1 + \chi_e) \vec{E} = \epsilon \vec{E} \tag{2.11}$$

pri čemu prirodno dolazi do definicije konstante permitivnosti materijala $\epsilon = \epsilon_0(1 + \chi_e)$. Konačno se tada definira bezdimenzionalna konstanta najbitnija za ovaj rad:

$$\epsilon_r = \frac{\epsilon}{\epsilon_0} = 1 + \chi_e \tag{2.12}$$

koja se naziva relativna permitivnost ili dielektrična konstanta.

Dodatno se definirana mikroskopska konstanta α može povezati s relativnom permitivnosti Clausius-Mossottijevom relacijom[14]:

$$\alpha = \frac{3\epsilon_0}{n} \frac{\epsilon_r - 1}{\epsilon_r + 2} \tag{2.13}$$

pri čemu je n volumna koncentracija atoma.

2.1.3 Elektromagnetski valovi u materijalima

Elektromagnetski valovi u homogenom izotropnom dielektriku zadovoljavaju jednadžbe (2.6) i (2.7) uz zamjenu $\epsilon_0 \longrightarrow \epsilon$ i $\mu_0 \longrightarrow \mu$. Tada brzina propagacije vala postaje:

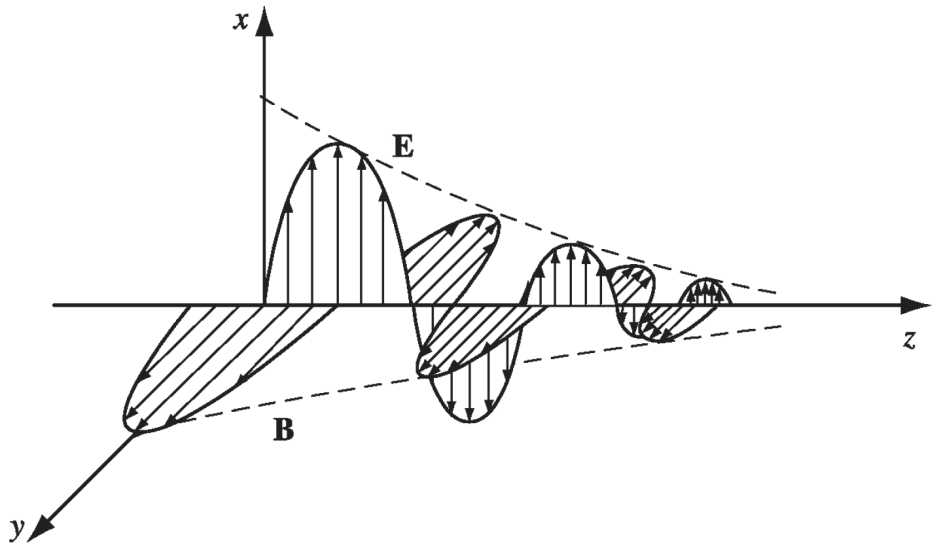
$$v = \frac{1}{\sqrt{\epsilon \mu}} = \frac{1}{\sqrt{\epsilon_0 \epsilon_r \mu_0 \mu_r}} = \frac{c}{n} \tag{2.14}$$

uz definiciju indeksa loma $n = \sqrt{\epsilon_r \mu_r}$.

Ewald-Oseenov teorem objašnjava interakciju elektromagnetskih valova i dielektrika, govori o iznesenoj činjenici smanjenja brzine valova, ali u pozadini je i ostalih pojava, na primjer promjene smjera propagacije vala. Kao što je spomenuto u poglavlju 2.1.2, električno polje u dielektricima stvara dipole koji proizvode svoje vlastito polje. Promatrajući mikroskopske jednadžbe za elektromagnetski val koji iz vakuuma

ulazi u dielektrik, teorem pokazuje da se dipolni doprinos polju sastoji od dva člana, od kojih jedan zadovoljava valnu jednadžbu u vakuumu i poništava upadni val, a drugi član zadovoljava valnu jednadžbu s brzinom propagacije c/n.[16]

Za elektromagnetske valove u vodičima u valnoj se jednadžbi pojavljuje i faktor gušenja proporcionalan s vodljivošću, stoga rješenja valne jednadžbe eksponencijalno trnu te elektromagnetski valovi ne dopiru duboko u unutrašnjost vodiča, a dodatno se događa i pomak u fazi između električnog i magnetskog polja, prikazano na Slici 2.2.



Slika 2.2: Prikaz ravnog elektromagnetskog vala u vodiču, preuzeto iz [15]

2.2 Wi-Fi polje

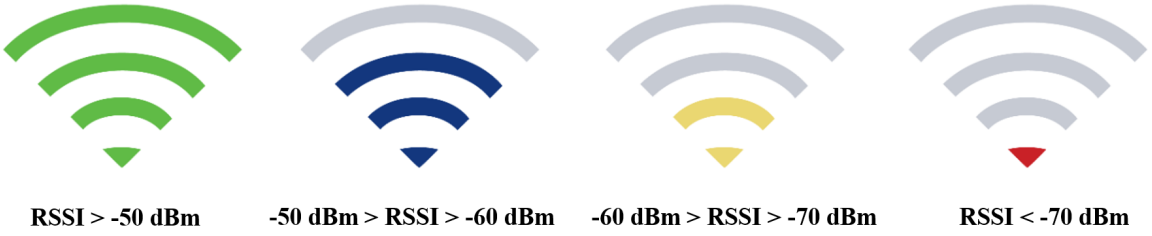
Širenje bežičnih komunikacijskih tehnologija relativno nedavno revolucioniralo je moderno povezivanje omogućavajući brzu i kvalitetnu razmjenu podataka te široku dostupnost mreže. Među ovim inovacijama, Wi-Fi polje (*engl. Wireless Fidelity*) izdvaja se kao kamen temeljac, pružajući sveprisutan način prijenosa informacija na kratkim udaljenostima bez ograničenja fizičkog povezivanja. U okviru fizike, Wi-Fi polje elektromagnetski je val u radio dijelu spektra, frekvencijskih pojaseva 2.4 ili 5 GHz. Iako je na frekvenciji 5 GHz prijenos podataka brži, u ovom radu korisniji će biti pojas 2.4 GHz jer je na toj frekvenciji pokriveno veće područje te su uređaji cjenovno pristupačniji.[17]

2.2.1 RSSI i CSI vrijednosti

RSSI vrijednost (*engl. Received Signal Strength Indicator*) jedna je od glavnih mjera u Wi-Fi komunikaciji te govori o snazi signala. Matematički RSSI vrijednost definira se logaritamski[18]:

$$\text{RSSI} = 10 \log \frac{P_1}{P_0} \tag{2.15}$$

pri čemu je P_1 snaga elektromagnetskog polja na promatranom mjestu, P_0 konstantna referentna vrijednost snage te se ovako definirana vrijednost mjeri u decibelima (dB). Ako je odabrana referentna vrijednost $P_0 = 1$ mW, mjerna jedinica RSSI vrijednosti naziva se dBm. Na Slici 2.3 prikazano je kako su vrijednosti RSSI u dBm povezane s konkretnom jačinom Wi-Fi signala.



Slika 2.3: Klasifikacija jačine signala u odnosu na RSSI vrijednost, od najjačeg signala prema najslabijem, s lijeva na desno, preuzeto iz [19]

Neki proizvođači RSSI vrijednost drugačije definiraju birajući proizvoljnu referentnu vrijednost i skaliranje u odnosu na specifične karakteristike njihovih uređaja pa jedna RSSI vrijednost bez mjerne jedinice može odgovarati nizu vrijednosti dBm, ovisno o konvenciji.[20]

Snaga se može iskazati kao[21]:

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^n} \tag{2.16}$$

gdje indeksi *t* i *r* redom predstavljaju odašiljač signala (*engl. transmiter*) i prijemnik (*engl. receiver*), stoga su P_t i P_r njihove snage, a G_t i G_n su dobitci antena. Dalje u izrazu λ predstavlja valnu duljinu, d je udaljenost od odašiljača, a n faktor atenuacije koji ovisi o okruženju.

CSI vrijednost (*engl. Channel State Information*) naprednija je metrika za analizu Wi-Fi signala. Pri prijenosu informacija Wi-Fi mrežom signal se dijeli na niz frekvencijskih komponenti unutar frekvencijskog pojasa te ova metrika sadrži informacije o

iznosu i fazi svake komponente.

2.2.2 Propagacija radio valova

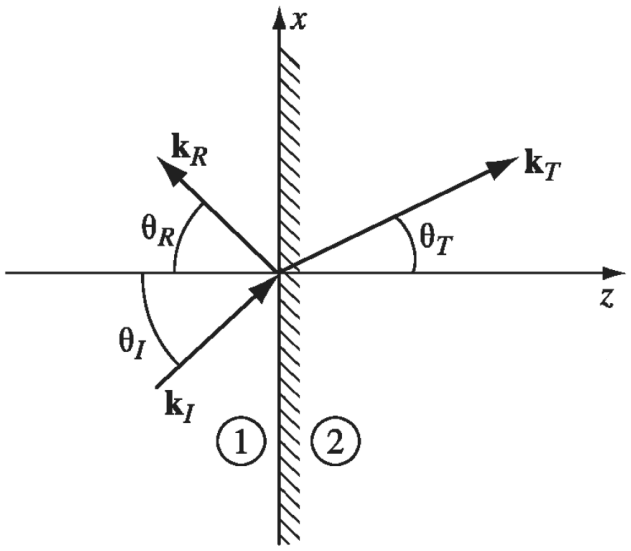
U komunikacijskim sustavima razlikuju se tri osnovna mehanizma propagacije elektromagnetskih valova: refleksija, raspršenje i difrakcija.[22]

Refleksija se događa nailaskom vala na prepreku velikih dimenzija u odnosu na valnu duljinu samog vala. Promatrajući problem nailaska vala na ravnu granicu dvaju homogenih sredstava, prikazano na Slici 2.4, mogu se izvesti glavni zaključci povezani s refleksijom. Upadni, reflektiran i transmitiran valni vektor, redom k_I, k_R, k_T, pripadaju istoj ravnini, kao i normala granice sredstava. Upadni kut i kut refleksije jednaki su (θ_I=θ_R).

Ako se radi o dielektricima, koeficijenti relativne permitivnosti ϵ_r i relativne permeabilnosti μ_r ovise o materijalu pa se na granici dva medija mijenja smjer širenja vala po Snellovom zakonu:

$$\frac{\sin\theta_I}{\sin\theta_T} = \frac{n_T}{n_I} \tag{2.17}$$

pri čemu je θ_I upadni kut (*engl. angle of incidence*), θ_T kut transmisije te su $n = \sqrt{\epsilon_r \mu_r}$ indeksi loma (koji ovise o sredstvu).



Slika 2.4: Prikaz refleksije i transmisije elekromagnetskog vala na granici dva homogena dielektrika, preuzeto iz [15]

Za dielektrike se odnos amplituda upadnog, reflektiranog i transmitiranog vala određuje Fresnelovim relacijama[23]:

$$
\begin{aligned}
r_s = &\frac{E_r}{E_i} = \frac{n_i \cos\theta_i - n_t \cos\theta_t}{n_i \cos\theta_i + n_t \cos\theta_t} \\
t_s = &\frac{E_t}{E_i} = \frac{2n_i \cos\theta_i}{n_i \cos\theta_i + n_t \cos\theta_t}
\end{aligned}
\tag{2.18}
$$

$$
\begin{aligned}
r_p = &\frac{E_r}{E_i} = \frac{n_i \cos\theta_t - n_t \cos\theta_i}{n_i \cos\theta_t + n_t \cos\theta_i} \\
t_p = &\frac{E_t}{E_i} = \frac{2n_i \cos\theta_i}{n_i \cos\theta_t + n_t \cos\theta_i}
\end{aligned}
\tag{2.19}
$$

Dva navedena slučaja odgovaraju iznosima polja za s i p polarizaciju.

Raspršenje se događa nailaskom vala na prepreku malih dimenzija, usporedivih s valnom duljinom, ili na nepravilnim površinama. Zbog ove je pojave jačina signala često veća nego što se dobije modeliranjem samo refleksije i difrakcije jer se energija raspršenjem širi u svim smjerovima.[22]

Difrakcija je pojava skretanja valova iza prepreka, objašnjava se Huygensovim načelom koje govori da se svaka točka valne fronte može smatrati točkastim izvorom sekundarnog vala te svi sekundarni valovi interferiraju i stvaraju novu valnu frontu. Iako se snaga polja brzo smanjuje u zasjenjenom području, difrakcijski valovi često su dovoljni za proizvodnju korisnog signala.[22]

2.3 Wi-Fi i ljudsko tijelo

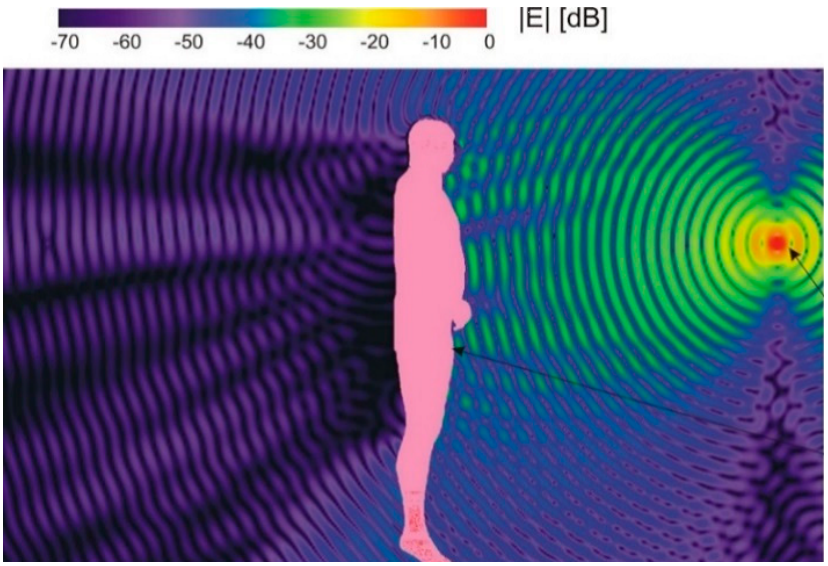
Zanimljiva i korisna činjenica jest da je ljudsko tijelo dielektrik s velikim koeficijentom permitivnosti na Wi-Fi frekvenciji, prikazano u Tablici 2.1.

materijal	ϵ_r na 2.45 GHz
gipsana ploča	2.94–0.1607i
beton	5.31–0.4947i
stropna ploča	1.5–0.0104i
ljudski mišić	52.729–13.0410i

Tablica 2.1: Usporedba relativne permitivnosti ljudskog tijela s relativnim permitivnostima građevinskih materijala na Wi-Fi frekvenciji, preuzeto iz [1]

Ljudi značajno mijenjanju električno polje u kojem se nalaze u skladu s navedenim mehanizmima; polje je atenuirano te se mijenja smjer širenja valova, prikazano na Slici 2.5. Ideja je mjerenjem iznosa i faze električnog polja duž prostorije

odrediti broj ljudi koji se nalaze u prostoriji. Uvođenjem niza pretpostavki mogle bi se izračunati očekivane vrijednosti polja za dani raspored ljudi, rješavanjem kompleksnog elektrodinamičkog problema. Za obrnut postupak rekreacije broja ljudi na temelju vrijednosti polja potrebno je koristiti strojno učenje.



Slika 2.5: Simulacija iznosa električnog polja u blizini ljudskog tijela, preuzeto iz [2]

3 Strojno učenje

Termin strojno učenje odnosi se na automatsku detekciju značajnih uzoraka u podatcima te je relativno nedavno postalo glavni alat u zadatcima koji iziskuju dobivanje informacija iz velikog broja podataka: koristi se u filtriranju elektroničke pošte, sortiranju rezultata u pretraživanju, personalizaciji reklama, detekciji prevara u kartičnim transakcijama, prepoznavanju lica i govora, doziranju lijekova, predikciji cijena, procjeni rizika, ali i u znanosti. Svim primjerima zajedničko je da su uzorci koje je potrebno prepoznati iznimno složeni te programer ne može eksplicitno napisati naredbe koje bi zadatak izvršile korak po korak, već po uzoru na ljude program "uči" svoje ponašanje na temelju iskustva, odnosno danih primjera.[24, 25]

3.1 Terminologija

Ovisno o zadatku koji je potrebno riješiti i dostupnim podatcima, pojam strojno učenje može se podijeliti na nadzirano i nenadzirano učenje. Nenadziranim učenjem traže se pravilnosti i uzorci u samim podatcima, a u nadziranom učenju dostupan je niz primjera s već označenim vrijednostima te je cilj programa na temelju tih primjera odrediti oznake za nove, neviđene primjere. Dalje u nadziranom učenju postoje regresijski i klasifikacijski problemi, što se odnosi na to jesu li moguće oznake brojčane kontinuirane vrijednosti ili se radi o svrstavanju primjera u grupe koje se nazivaju klase.

Već spomenuti "primjer" naziv je za ulaz algoritma strojnog učenja. Primjer se prikazuje kao vektor vrijednosti relevantnih za taj primjer - značajki. Primjeri razapinju ulazni prostor dimenzije koja odgovara broju značajki. Svrha je nadziranog strojnog učenja odrediti funkciju koja primjerima dodjeljuje oznake; ta funkcija naziva se hipoteza. Unaprijed egzaktna funkcija nije poznata, u suprotnom ne bi postojala svrha učenja, ali potrebno je odlučiti se za oblik funkcije, odnosno model, uzimajući u obzir dostupno znanje o problemu. Odabrana funkcija definirana je do na parametre koje određuje sam program.

Svaki algoritam strojnog učenja sastoji se od tri glavne komponente koje čine[26]:

- model
- funkcija pogreške
- optimizacija.

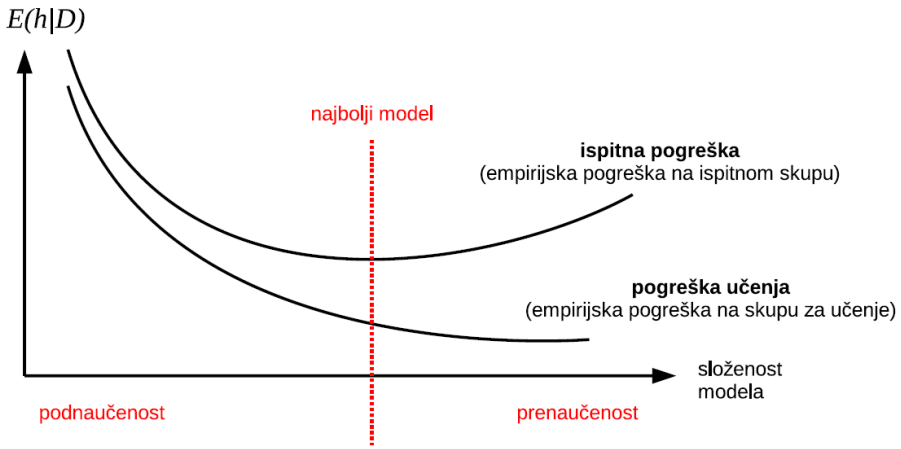
Model je naziv za skup svih mogućih hipoteza koje se razlikuju do na iznos parametara. Funkcija pogreške numerički govori o tome koliko se razlikuje predviđanje modela od označenog skupa primjera. Optimizacija je postupak kojim se unutar modela pronalazi hipoteza koja minimizira pogrešku.

Među modelima nadziranog strojnog učenja postoji i podjela na parametarske i neparametarske modele. Složenost parametarskih modela ne ovisi o broju primjera, već ovisi o pretpostavljenoj teorijskoj raspodjeli; na primjer to može biti linearna funkcija što će biti detaljnije opisano u sljedećem poglavlju. Učenje se tada svodi na pronalazak parametara koji definiraju tu raspodjelu. Ovim modelima suprostavljeni su neparametarski, koji unatoč nazivu imaju parametre, ali ti parametri ne ovise o nekoj pretpostavljenoj distrubuciji, već njihov broj raste s brojem primjera, a time raste i složenost.

3.2 Odabir modela

Očiti kriterij za odabir najboljeg modela bio bi minimalna greška modela na dostupnim podatcima. Pritom problem stvara šum koji se javlja u svim stvarnim problemima te sprječava da primjeri savršeno odgovaraju pozadinskom teorijskom modelu. Drugi problem stvara činjenica da je dostupan set podataka samo podskup ukupne domene problema. Traženje modela s najmanjom greškom na dostupnom skupu podataka bilo bi ekvivalentno memoriranju, što znači da se ne događa nikakvo učenje. Cilj je odabrati model koji unatoč iznesenim preprekama opisuje stvarni trend na cijeloj domeni i s visokom točnošću pridjeljuje oznake neviđenim primjerima, odnosno dobro generalizira.[25]

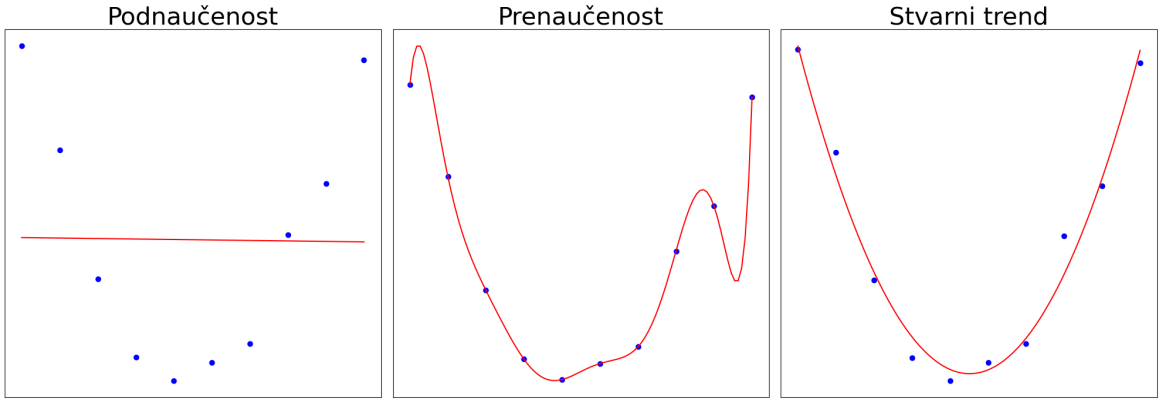
Za provođenje takvog odabira, skup dostupnih primjera dijeli se na skup za učenje i ispitni skup, optimizacija modela provodi se samo na primjerima iz skupa za učenje te dobiveni model predviđa oznake za primjere iz ispitnog skupa. Greške se računaju na oba skupa te je predviđeno ponašanje u ovisnosti o složenosti modela prikazano na



Slika 3.1: Teorijska ovisnost pogreške o složenosti modela na skupu za učenje i ispitnom skupu, preuzeto iz [26]

Slici 3.1. Situacija kada je model presložen, odnosno previše se prilagođava skupu za učenje, naziva se prenaučenost, a situacija kada je model prejednostavan i rezultira lošim rezultatima na oba skupa naziva se podnaučenost. Optimalan model je onaj s minimalnom greškom na ispitnom skupu.

Na Slici 3.2 prikazan je jednostavan primjer prenaučenosti i podnaučenosti na polinomijalnoj regresiji. Generirani su primjeri polinoma drugog stupnja sa šumom te se vidi da pravac nije dovoljno složen model za opis takve relacije, a polinom previsokog stupnja savršeno se prilagodio dostupnim podatcima, ali neće raditi dobre predikcije.



Slika 3.2: Podnaučenost i prenaučenost na primjeru polinomijalne regresije

3.3 Podjela podataka za nadzirano učenje

Dostupan skup podataka dijeli se na tri disjunktna skupa:

- skup za učenje - koristi se za treniranje modela
- skup za validaciju - koristi se za procjenu greške kod optimizacije parametara
- skup za testiranje - koristi se za procjenu greške konačnog, optimalnog modela.

Skup svih podataka			
Učenje i validacija			**Testiranje**

1. skup	2. skup	3. skup	4. skup
Validacija	**Učenje**	**Učenje**	**Učenje**
Učenje	**Validacija**	**Učenje**	**Učenje**
Učenje	**Učenje**	**Validacija**	**Učenje**
Učenje	**Učenje**	**Učenje**	**Validacija**

Učenje	**Testiranje**

Tablica 3.1: Primjer podjele podataka za unakrsnu provjeru s četiri preklopa

Skup podataka najprije se dijeli na dva skupa: skup za učenje i validaciju te na skup za testiranje koji se ne koristi dok nije odabran konačan model. Prvi skup višestruko se dijeli na dva podskupa, odnosno provodi se stratificirana k-struka unakrsna provjera. U metodi k-struke unakrsne provjere skup se dijeli na k disjunktnih podskupova te skup za učenje postaje k-1 podskupova, a k-ti podskup je skup za validaciju. Proces učenja i validacije ponavlja se k puta, dok svaki podskup nije jednom tretiran kao skup za validaciju; podjela je prikazana u Tablici 3.1 Točnost modela računa se kao srednja vrijednost točnosti pojedinih k rezultata, a pogreška točnosti je standardna devijacija podijeljena s $\sqrt{k}$. Pridjev stratificirano znači da podjela skupa primjera na skup za učenje i skup za validaciju zrcali pravu razdiobu primjera u kompletnom skupu podataka, odnosno u svakom od k podskupa očuvan je omjer primjera iz pripadajućih klasa.

Najbolji parametri biraju se odabirom modela s najvećom točnošću te se takav optimalan model trenira na kompletnom skupu za učenje i validaciju te se greška generalizacije računa na skupu za testiranje.

3.4 Vrednovanje modela

Već spomenuta točnost, koja se računa na testnom skupu podataka, pri klasifikaciji se jednostavno računa kao omjer broja točno klasificiranih primjera i ukupnog broja primjera:

$$T = \frac{1}{N}\mathbb{1}\left\{y^{stv} = y^{pred}\right\} \tag{3.1}$$

pri čemu je N broj primjera, y^{stv} stvarna oznaka primjera, y^{pred} oznaka koju predviđa model, a $\mathbb{1}$ je funkcija koja Booleove vrijednosti pretvara u 0 ili 1:

$$\mathbb{1}\left\{P\right\} = \begin{cases} 1, & \text{ako je uvjet P ispunjen,} \\ 0, & \text{inače} \end{cases} \tag{3.2}$$

Iako je za početak točnost najbolja mjera uspješnosti modela, za dublju analizu i vrednovanje modela moguće je, a nekad i potrebno, izračunati i druge veličine. Na primjer, ako dostupan skup podataka sadrži većinski jednu klasu, trivijalan model koji sve primjere označava tom većinskom klasom imati će visoku točnost, ali neće biti dobar model strojnog učenja.

U binarnoj klasifikaciji postoje četiri mogućnosti za klasifikaciju jednog primjera:

- stvarno pozitivni TP (*engl. true positive*) - stvarna oznaka i predikcija su 1
- lažno negativni FN (*engl. false negative*) - stvarna oznaka je 1, a predikcija 0
- lažno pozitivni FP (*engl. false positive*) - stvarna oznaka je 0, a predikcija je 1
- stvarno negativni TN (*engl. true negative*) - stvarna oznaka i predikcija su 0.

Ti slučajevi sažeto se mogu prikazati u matrici koja se naziva matrica zabune (*eng. confusion matrix*):

$$\begin{array}{cc} & \begin{array}{cc} y^{pred}=1 & y^{pred}=0 \end{array} \\ \begin{array}{c} y^{stv}=1 \\ y^{stv}=0 \end{array} & \left[\begin{array}{cc} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{array}\right] \end{array}$$

pri čemu su elementi matrice ukupan broj odgovarajućeg tipa primjera, konkretnije element matrice TP je broj primjera u skupu za testiranje koji su stvarno pozitivni.

Sada se mogu definirati neke od alternativnih mjera vrednovanja[27]:

- Preciznost (*engl. precision*) - govori koliko je predviđenih pozitivnih primjera stvarno pozitivno:
$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.3}$$
- Odziv (*engl. recall*) - govori koliko je stvarno pozitivnih primjera točno klasificirano:
$$\text{R} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.4}$$
- F_1 mjera (*engl. F_1 score*) - harmonijska sredina mjere i odziva, s obzirom da su preciznost i odziv međusobno komplementarne mjere direktno su suprotstavljene pa je F_1 mjera korisna kad je uspješnost modela potrebno prikazati jednim brojem:
$$\text{F}_1 = \frac{2\text{PR}}{\text{P} + \text{R}}. \tag{3.5}$$

Vrednovanje se lako poopćava na probleme s više klasa tako da se mjere računaju za svaku klasu na način da se ta klasa smatra pozitivnom, a sve ostale klasifikacije su tada negativne u odnosu na odabranu klasu.

4 Linearni diskriminativni modeli

Naziv linearni diskriminativni modeli opisuje klasifikacijske algoritme s linearnom granicom. Linearni modeli jedni su od najkorisnijih općenito u strojnom učenju, mnogo algoritama korištenih u praksi oslanja se na njih prvenstveno zbog efikasnosti učenja, ali i zbog relativno jednostavne interpretacije rezultata te zbog činjenice da se velik broj problema strojnog učenja može razumno dobro linearno aproksimirati.[24]

4.1 Model

Diskriminativni modeli modeliraju granicu koja diskriminira, odnosno razdvaja klase tako da parametri modela opisuju tu granicu i ništa osim te granice, stoga će modeli imati onoliko parametara koliko je potrebno da bi se definirala granica. Diskriminativni modeli suprotstavljeni su drugoj velikoj familiji modela o kojima će biti riječi kasnije, generativnim modelima, koji općenito imaju više parametara i modeliraju više od granice, a granica se modelira indirektno.[28]

Model $h(\vec{x})$ je afina funkcija s $n+1$ parametara w_i, pri čemu je n dimenzija prostora značajki:

$$h(\vec{x}) = \vec{w} \cdot \vec{x} + w_0 \tag{4.1}$$

gdje je $\vec{x}$ oznaka n-dimenzionalnog primjera. Komponente vektora $\vec{w}$ nazivaju se težinama (*engl. weights*) jer govore o tome kolikim udjelom, odnosno težinom, određena značajka doprinosi rezultatu hipoteze.

Ovakav model može se promatrati kao binaran klasifikacijski model na način da se ulazni prostor dijeli na dva potprostora hiperravninom $h(\vec{x}) = 0$ te su primjeri jedne klase oni za koje vrijedi $h(\vec{x}) > 0$, a primjeri druge klase oni za koje vrijedi $h(\vec{x}) < 0$, prikazano na Slici 4.1.

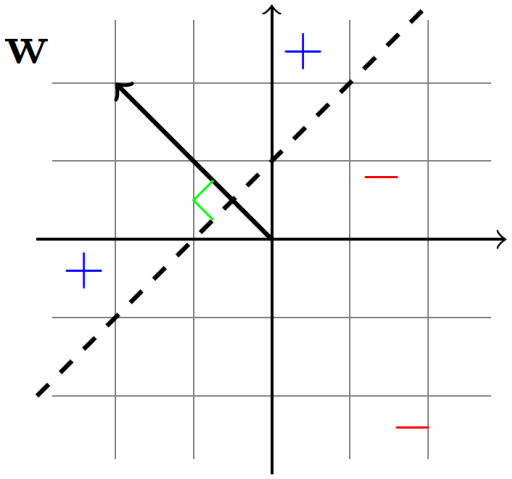
Udaljenost nekog primjera i hiperravnine bitna je veličina jer govori o pouzdanosti klasifikacije, veća udaljenost odgovara većoj sigurnosti da je primjer točno klasificiran te je iznos određen s[28]:

$$d = \frac{h(\vec{x})}{\|\vec{w}\|} = \frac{\vec{w} \cdot \vec{x} + w_0}{\|\vec{w}\|} \tag{4.2}$$

pri čemu je norma vektora $\|\vec{w}\|$ klasična euklidska:

$$\|\vec{w}\| = \sqrt{\vec{w} \cdot \vec{w}} = \sqrt{\sum_{i=1}^{n} w_i^2} \tag{4.3}$$

Definirana udaljenost d može biti pozitivna i negativna, ovisno o tome kojoj klasi primjeri pripadaju, odnosno ovisno o tome s koje strane hiperravnine se nalaze, prikazano na Slici 4.1. Bitna je činjenica da postoji beskonačno mogućih kombinacija parametara w_i koji definiraju istu hiperravinu, odnosno vektor (w_0,$\vec{w}$) može se proizvoljno skalirati.



Slika 4.1: Primjer binarne klasifikacije u dvodimenzionalnom prostoru značajki, preuzeto iz [24]

4.2 *Preslikavanje u prostor više dimenzije*

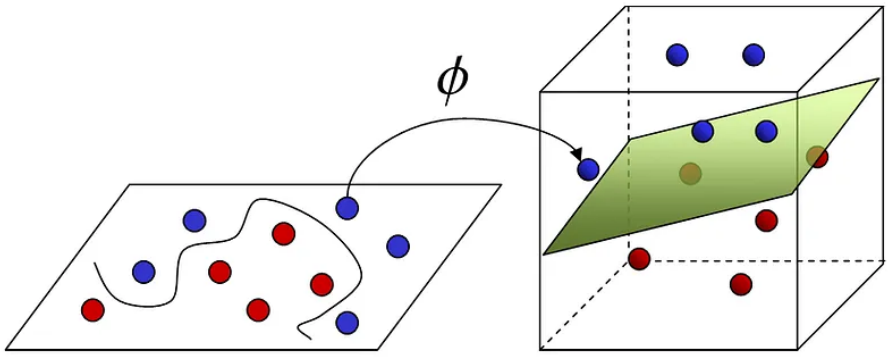
Linearni modeli, suprotno svom nazivu mogu modelirati i nelinearne granice što je vrlo korisna činjenica. Ideja je ne mijenjati model, već promijeniti podatke tako da postanu linearno separabilni, što je ekvivalentno nelinearnoj granici u početnom prostoru, prikazano na Slici 4.2.

Potrebno je definirati nelinearnu funkciju preslikavanja $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}^{m+1}$ pri čemu je n dimezija početnog ulaznog prostora, a $m + 1$ dimenzija konačnog prostora:

$$\phi(\vec{x}) = (\phi_0(\vec{x}), \phi_1(\vec{x}), ..., \phi_m(\vec{x})) \tag{4.4}$$

gdje se funkcije ϕ_i nazivaju bazne funkcije, a funkcija preslikavanja je vektor dobiven primjenom pojedinačnih baznih funkcija na primjer $\vec{x}$. Nakon preslikavanja

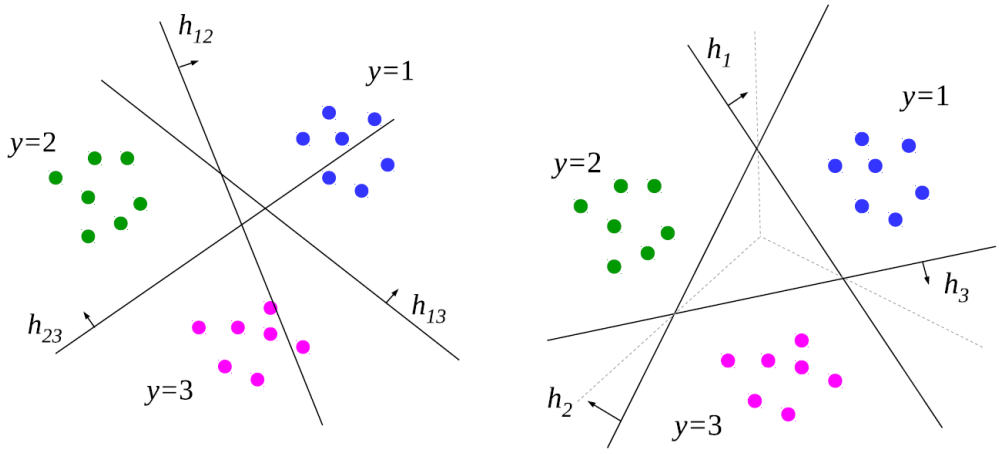
u početnom modelu dovoljno je zamijeniti vektor $\vec{x}$ s vektorom ϕ. Dogovorno se nulta funkcija preslikavanja definira kao $\phi_0 = 1$ jer takva definicija omogućava da se slobodna konstanta linearnih modela jednostavnije implementira u daljnje račune. Tipično se događa preslikavanje u prostor veće dimenzije jer je tada vjerojatnije da će primjeri postati linearno separabilni.[29]



Slika 4.2: Primjer preslikavanja ϕ iz dvodimenzionalnog u trodimenzionalni prostor pri čemu primjeri postaju linearno odvojivi, preuzeto iz [30]

4.3 Višeklasni problemi

Razmatranja koja su se dosad odnosila na binarnu klasifikaciju lako se poopće na višeklasne probleme. S obzirom da su linearni diskriminativni modeli inherentno binarni, jednostavnije je promijeniti problem nego model, odnosno promatrati višeklasnu klasifikaciju kao niz binarnih klasifikacija. Dva su glavna načina podjele; OVO (*engl. one-vs-one*) i OVR (*engl. one-vs-rest*).[28] Kao što sama imena govore, pri OVO klasifikaciji radi se binarna klasifikacija između svakog para klasa što je ukupno $\binom{K}{2}$ nezavisnih klasifikacija, pri čemu je K broj klasa, a pri OVR klasifikaciji binarno se klasificira jedna klasa naspram svih ostalih zajedno, što se svodi na samo K klasifikacija. Očita prednost OVR nad OVO jest manji ukupni broj klasifikacija što rezultira manjom računalnom složenošću koja može biti značajna za veći broj primjera i klasa. Nedostatak OVR jest problem neuravnoteženosti klasa koji također dolazi do izražaja za veći broj primjera i klasa jer u svakoj binarnoj klasifikaciji jedna klasa ima značajno manje primjera nego druga, s čim se linearni diskriminativni modeli u pravilu ne nose dobro. Primjer OVO i OVR klasifikacije za 3 klase prikazan je na Slici 4.3.



Slika 4.3: Primjer višeklasne klasifikacije, lijevo provedena OVO pristupom, a desno OVR pristupom, preuzeto iz [28]

4.4 Logistička regresija

4.4.1 Komponente algoritma

Logistička regresija je klasifikacijski algoritam čiji je model za binarnu klasifikaciju linearna funkcija omotana aktivacijskom funkcijom sigmoidom:

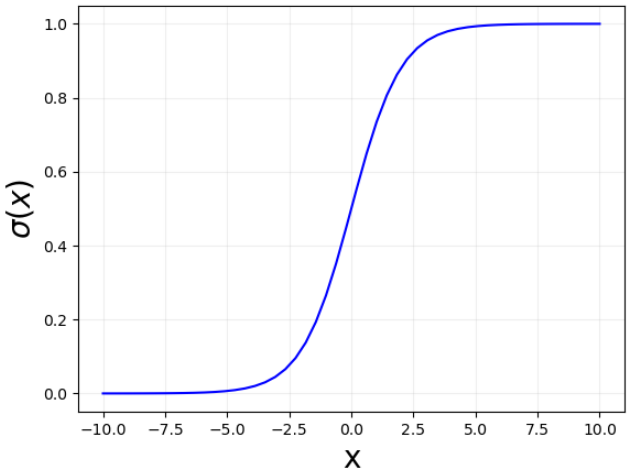
$$h(\vec{x};\vec{w}) = \sigma(\vec{x}\cdot\vec{w}) = \frac{1}{1+e^{-\vec{x}\cdot\vec{w}}} \tag{4.5}$$

gdje je $\vec{x}$ vektor značajki, a $\vec{w}$ vektor parametara, odnosno težina. Iako je izlaz modela (4.5) kontinuiran, radi se o binarnom klasifikacijskom modelu pri čemu je granica među klasama definirana hiperravninom $h(\vec{x}) = 0$. Kontinuirane vrijednosti zadržavaju se jer one daju infomaciju o tome koliko je klasifikacija pouzdana. Sigmoida je prikladna funkcija jer ulazne vrijednosti normalizira na raspon od nula do jedan, što omogućava probabilističku interpretaciju izlaza; izgled sigmoide prikazan na Slici 4.4.

Funkcija gubitka izvodi se probabilistički kao negativan logaritam izglednosti te se konačan izraz naziva gubitak unakrsne entropije:

$$L(y, h(\vec{x})) = -y\ln(h(\vec{x})) - (1-y)\ln(1-h(\vec{x})) \tag{4.6}$$

pri čemu su y oznake označenih primjera te mogu poprimiti vrijednosti 0 ili 1. Poopćenje binarne logističke regresije na više klasa naziva se multinomijalna logistička regresija, u modelu se koristi zaseban vektor težina $\vec{w}_k$ za svaku klasu te



Slika 4.4: Izgled sigmoidalne funkcije na proizvoljnoj domeni, bitno je primjetiti da je funkcija omeđena nulom i jedinicom

se skalarni umnožak $\vec{w}_k \cdot \vec{x}$ stavlja u funkciju softmax:

$$h_k(\vec{x}; \vec{w_1}, ..., \vec{w_K}) = \frac{e^{\vec{w}_k \cdot \vec{x}}}{\sum_{j=1}^{K} e^{\vec{w}_j \cdot \vec{x}}} = P(y = k; \vec{w_1}, ..., \vec{w_K}) \tag{4.7}$$

koja normalizira vrijednosti radi mogućnosti probabilističke interpretacije te povećava velike vrijednosti, a smanjuje male.

Funkcija gubitka definira se analogno kao za binarni problem, pri čemu je oznaka y_k jednaka 1 ako primjer pripada klasi k, a inače je jednaka nuli:

$$L(y, h_k(\vec{x})) = -\sum_{k=1}^{K} y_k \ln(h_k(\vec{x})) \tag{4.8}$$

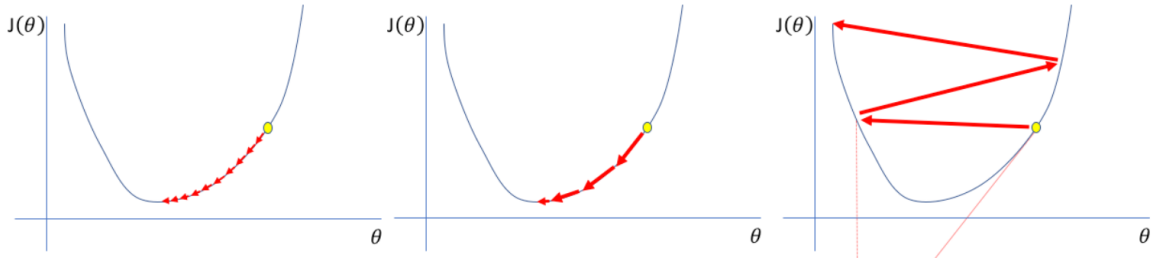
4.4.2 Standardni gradijentni spust

Gradijentni spust jedan je od najpoznatijih optimizacijskih postupaka, a koristi se i u logističkoj regresiji. Odabire se početni vektor težina $\vec{w}$ te se on ažurira pomicanjem u smjeru suprotnom od gradijenta u danoj točki niz površinu funkcije pogreške L za stopu η:

$$\vec{w} \longleftarrow \vec{w} - \eta \vec{\nabla} L \tag{4.9}$$

Koraci se ponavljaju dok se ne pronađe optimalni vektor težina $\vec{w^*}$ koji odgovara globalnom minimumu funkcije pogreške. Poželjno je definirati konveksnu funkciju pogreške, kako se optimizacija ne bi zaustavila na lokalnom minimumu. Također je

bitno dobro odabrati stopu učenja; prevelika stopa može rezultirati divergencijom, a premala uzrokuje presporu konvergenciju, prikazano na Slici 4.5. Postoje razne metode za određivanje stope, poput linijskog pretraživanja u kojem se traži točka na pravcu u smjeru negativnog gradijenta koja odgovara minimumu funkcije ili optimizacije drugog reda pri kojima se računa i promjena gradijenta, poput Newtonovog postupka.



Slika 4.5: Ovisnost greške o parametru modela, prikazuje odabir stope pri gradijentnom spustu, s lijeva na desno: premala stopa, optimalna stopa i prevelika stopa, preuzeto iz [31]

4.5 Stroj potpornih vektora

Drugi primjer linearnog diskriminativnog modela koji će se detljnije opisati jest stroj potpornih vektora.

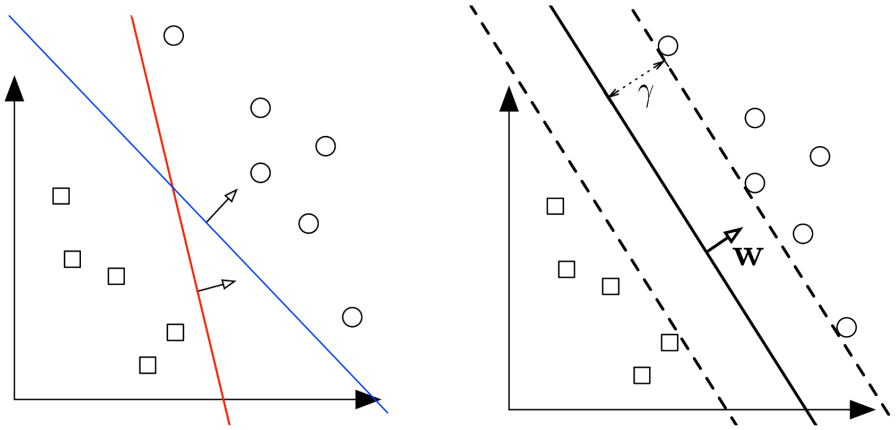
4.5.1 Problem maksimalne margine

Neka je $S = \{(\vec{x_i}, y_i)\}$ skup označenih primjera, pri čemu je primjer $\vec{x_i} \in \mathbb{R}^d$ s brojem značajki d te su oznake $y_i \in \{\pm 1\}$. Skup primjera je linearno separabilan ako postoji potprostor $(\vec{w}, w_0)$ takav da za svaki i vrijedi $y_i = \text{sgn}(\vec{w} \cdot \vec{x_i} + w_0)$. Konciznije se uvjet može zapisati kao:

$$\forall i : y_i(\vec{w} \cdot \vec{x_i} + w_0) > 0 \tag{4.10}$$

Promatrajući model (4.1), hiperravnina koja odvaja primjere definira klasifikaciju, a za linearno separabilan skup primjera na domeni realnih brojeva postoji beskonačno mnogo takvih hiperravnina, odnosno beskonačno mnogo modela koji savršeno klasificiraju primjere s pogreškom 0. Radi konzistentnosti i želje za maksimizacijom točnosti na neviđenim primjerima potrebno je odabrati dodatan kriterij za odabir

modela; u tu svrhu se uvodi pojam margine. Na Slici 4.6 prikazana su dva moguća odabira pravca za podjelu primjera u dvodimenzionalnom prostoru.[24]



Slika 4.6: Odabir hiperravnine u dvodimenzionalnom prostoru značajki, hiperravnine lijevo točno klasificiraju primjere te postoji beskonačno takvih mogućih ravnina, a desno je prikazana hipperavnina s maksimalnom marginom koja se odabire u SVM metodi, preuzeto iz [32]

Margina je udaljenost hiperravnine i njoj najbližeg primjera. Dodatkom šuma na svaki postojeći primjer klasifikacija će ostati ista ako je margina dovoljno velika, stoga je intuitivno jasno da će najbolje generalizirati hiperravnina koja maksimizira marginu. Koristeći izraz (4.2) za udaljenost te uvjet (4.10), zahtjev optimizacije je:

$$\underset{\vec{w},w_0}{\operatorname{argmax}} \left\{ \frac{1}{\|\vec{w}\|} \min_i \left\{ y_i(\vec{w} \cdot \vec{x_i} + w_0) \right\} \right\} \tag{4.11}$$

Odnosno, između i primjera traži se onaj s najmanjom udaljenošću do hiperravnine pa se ta udaljenost maksimizira. O obzirom da se vektor (w_0,$\vec{w}$) može proizvoljno skalirati, kao što je rečeno u poglavlju 4.1, jednadžba (4.11) može se značajno pojednostaviti odabirom da za primjer najbliži hiperravnini vrijedi $y_i(\vec{w} \cdot \vec{x_i} + w_0) = 1$, odnosno tada za svaki primjer vrijedi:

$$y_i(\vec{w} \cdot \vec{x_i} + w_0) \geq 1 \tag{4.12}$$

Optimizacijski problem sveo se na zahtjev:

$$\underset{\vec{w},w_0}{\operatorname{argmax}} \frac{1}{\|\vec{w}\|} \tag{4.13}$$

pri čemu treba imati da umu da mora vrijediti uvjet (4.12). Maksimizacija izraza

(4.13) ekvivalentna je minimizaciji izraza $\|\vec{w}\|$ što je matematički jednostavnije pa se koristi takva formulacija problema. Radi se o minimizaciji funkcije uz ograničenja koja moraju biti zadovoljena, preciznije se ovakav problem u teoriji optimizacije naziva problem kvadratnog programiranja - to su problemi kojima je ciljna funkcija kvadratna, a ograničenja afine funkcije.

4.5.2 Lagrangeova dualnost

Problem se može riješiti koristeći metodu Lagrangeove dualnosti pri čemu se optimizacijski problem preformulira tako da su ograničenja ugrađena u ciljnu funkciju koja se minimizira te se tako definira Lagrangeova funkcija:

$$L(\vec{x}, \vec{\alpha}) = f(\vec{x}) + \sum_{i=1}^{m} \alpha_i g_i(\vec{x}) \tag{4.14}$$

Ovdje je $f(\vec{x})$ početna funkcija, $g_i(\vec{x})$ su ograničenja nejednakosti, a koeficijenti α_i nazivaju se Lagrangeovi multiplikatori. Rješenje originalnog problema je točka u kojoj gradijent Lagrangeove funkcije iščezava te se može pokazati da je ta točka sedlo, konkretnije minimum funkcije po $\vec{x}$, a maksimum funkcije po $\vec{\alpha}$. Definira se dualna Lagrangeova funkcija:

$$\tilde{L}(\vec{\alpha}) = \min_{\vec{x}} L(\vec{x}, \vec{\alpha}) \tag{4.15}$$

čijom se maksimizacijom po Lagrangeovim multiplikatorima dolazi do rješenja problema, odnosno do točke sedla. Konačno model stroja potpornih vektora u dualnoj formulaciji glasi:

$$h(\vec{x}) = \sum_{i=1}^{N} \alpha_i y_i \vec{x}\vec{x_i} + w_0 \tag{4.16}$$

Sada su parametri modela α_i te je njihov broj jednak broju primjera što znači da je problem postao neparametarski. Dodatno se može pokazati i da su koeficijenti α_i različiti od nule samo ako primjer leži točno na margini, odakle dolazi i naziv algoritma. U jednadžbi (4.16) jedini neiščezavajući pribrojnici su vektori koji leže na margini pa se ti vektori nazivaju potporni.

Skalarni umnožak vektora primjera u jednadžbi (4.16) može se shvatiti kao mjera sličnosti dva primjera što otvara niz mogućnosti za primjenu u kompliciranijim pri-

mjerima. Mjera sličnosti može se prikazati kao jezgrena funkcija, što je korisno u problemima gdje nije jasno kako primjer prikazati kao vektor značajki, na primjer u obradi jezika ili klasifikaciji DNK nizova. Za model je dovoljno koristiti mjeru sličnosti primjera, oni ni ne moraju biti esplicitno prikazani.

Iznesena je početna teorijska ideja za model potpornih vektora, zahtjev linearne odvojivosti može se ublažiti uvođenjem meke margine, a optimizacija se alternativno može raditi i gradijentnim spustom gdje se za empirijsku grešku uzima gubitak zglobnice:[33]

$$L(y\vec{w} \cdot \vec{x}) = \max(0, 1 - y\vec{w} \cdot \vec{x}) \tag{4.17}$$

5 Neparametarski modeli

Neparametarski modeli podrazumijevaju slabije pretpostavke o podatcima za razliku od parametarskih, stoga je očekivano da rezultati daju veću točnost ako je znanje o problemu ograničeno. Koriste se kada je dostupno dovoljno podataka i kada se želi izbjeći nametanje pretpostavki. Prednost neparametarskih modela je i lokalnost; u parametarskim modelima pretpostavljena je jedna funkcija, stoga svaki primjer ima globalan utjecaj na izgled konačnog modela, a u neparametarskim modelima rade se lokalne aproksimacije u prostoru primjera na temelju sličnosti s bliskim primjerima pa se može dogoditi da se točnost razlikuje unutar faznog prostora ovisno o tome koliko se primjera gdje nalazi; rezolucija hipoteze prilagođava se primjerima.

5.1 K najbližih susjeda

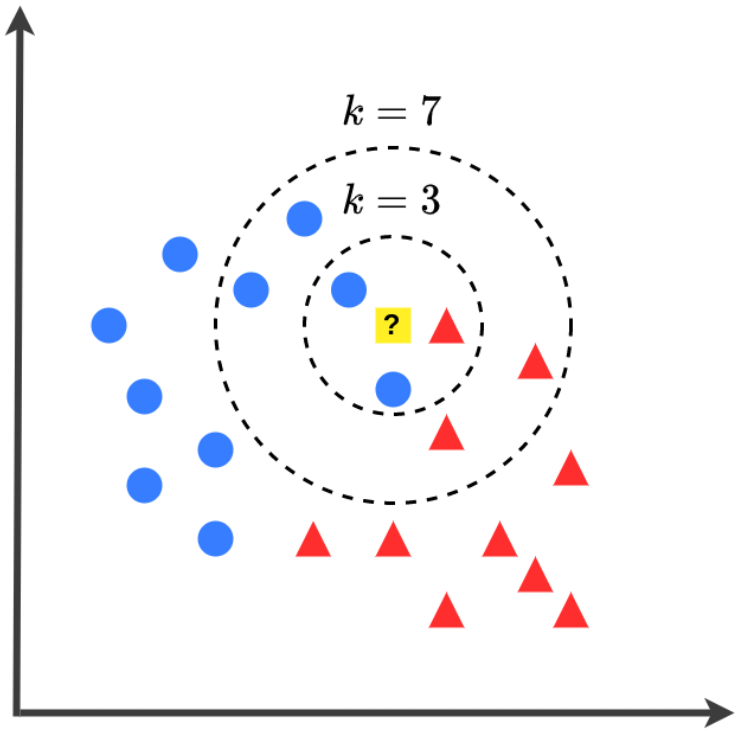
Algoritmi najbližih susjeda jedni su od najjednostavnijih u strojnom učenju; ideja je da se skup za učenje memorira, a nove oznake se pridjeljuju na temelju označenih najbližih susjeda u prostoru značajki, odnosno koristi se pretpostavka da su značajke primjera relevantne za oznaku na taj način da slične značajke daju istu oznaku. Neophodno je definirati na koji se način računa udaljenost, odnosno odabrati neku metriku ρ. Na primjer, na domeni realnih brojeva metrika može biti Euklidska udaljenost:

$$\rho(\vec{x}, \vec{x'}) = \left\| \vec{x} - \vec{x'} \right\| = \sqrt{\sum_{i=1}^{d} (x_i - x'_i)^2} \tag{5.1}$$

Ako je $S = \{(\vec{x_1}, y_1), ..., (\vec{x_m}, y_m)\}$ skup označenih primjera za učenje, za svaki novi primjer $\vec{x}$ može se definirati niz oznaka $\pi_1(\vec{x}), ..., \pi_m(\vec{x})$ koje rangiraju primjere za učenje u odnosu na njihovu udaljenost od primjera $\vec{x}$ tako da je:

$$\rho(\vec{x}, \vec{x}_{\pi_i(\vec{x})}) \leq \rho(\vec{x}, \vec{x}_{\pi_{i+1}(\vec{x})}) \tag{5.2}$$

gdje π_i poprima vrijednosti iz skupa $\{1, 2, ..., m\}$. Na primjer, konkretno oznaka $\pi_1(\vec{x})$ je cijeli broj koji je u početnom skupu označavao najbliži primjer nekoj referentnoj točki, odnosno primjeru koji se treba klasificirati. Parametar modela je broj *k* koji označava koliko će se najbližih susjeda uzimati u obzir za označavanje novog primjera, ti susjedi čine skup $NN_k(\vec{x}) = \left\{ y_{\pi_i(\vec{x})} : i \leq k \right\}$.[24] Za jednostavniju



Slika 5.1: Primjer algoritma najbližih susjeda u dvodimenzionalnom prostoru značajki; žuti primjer u središtu moguće je označiti na dva načina, ovisno o odabiru parametra k, preuzeto iz [34]

vizualizaciju; na Slici 5.1 prikazan je odabir dva različita parametra *k* za binarnu klasifikaciju u dvodimenzionalnom prostoru značajki. Oznaka novog primjera dodjeljuje se na temelju većinske oznake *k* najbližih primjera:

$$h(\vec{x}) = \underset{j \epsilon \{0,\ldots,K-1\}}{\operatorname{argmax}} \sum_{(\vec{x_i}, y_i) \epsilon N N_k(\vec{x})} \mathbb{1}\{y_i = j\} \tag{5.3}$$

pri čemu je j oznaka klase za višeklasni problem s K klasa.

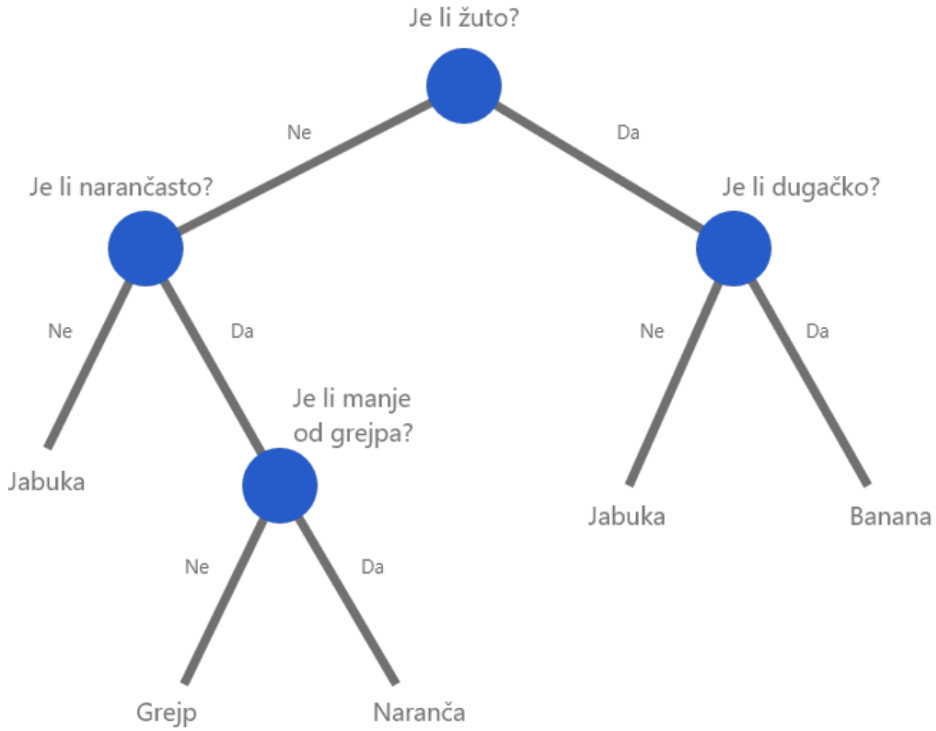
Odabir parametra algoritma k ponovo se svodi na balans između prenaučenosti i podnaučenosti pri čemu manji odabir k odgovara složenijim modelima. Ako algoritam uzima u obzir samo jednog najbližeg susjeda, sigurno će postojati situacije u kojima je taj najbliži primjer šum i novi primjeri biti će prilagođeni šumu.

Algoritam se teško nosi s primjerima velikih dimenzija, odnosno s previše značajki, zbog matematičke činjenice da porastom dimenzije prostora točke u prostoru postaju međusobno vrlo udaljene te su time udaljenosti nediskriminativne; ova pojava naziva se prokletstvo dimenzionalnosti i općenito se javlja u algoritmima u visokodimenzij-

skim prostorima.

5.2 Stablo odluke

Stabla odluke mogu tretirati i klasifikacijske i regresijske probleme, ali kao i dosad prvenstveno će se promatrati klasifikacija. Algoritam radi na način da se primjeri dijele na podskupe ovisno o značajkama dok podjela konačno ne rezultira izoliranim klasama; ovakvim postupkom podatci se organiziraju u strukturu stabla. Svaka podjela podataka čini čvor stabla, postupak započinje s korijenskim čvorom (*engl. root node*) gdje se ukupan skup primjera dijeli na podskupe ovisno o iznosu neke značajke te se dalje svaki taj podskup dijeli u odnosu na neku drugu značajku čime se stvaraju unutrašnji čvorovi i grane stabla, a konačni čvor naziva se listom. Idealno, svaki list sadrži primjere samo jedne klase, ali u stvarnosti se oznaka lista bira većinskom klasom pripadajućih primjera.[35]



Slika 5.2: Grafički prikaz ideje algoritma stablo odluke na primjeru klasifikacije voća, preuzeto iz [36]

Primjer jednostavnog stabla prikazan je na Slici 5.3. Postupak se može promatrati i na način da svaki čvor odgovara pitanju te niz odgovora, odnosno odluka, vodi do konačne klasifikacije, odakle dolazi i naziv algoritma. Ovim je opisan jednostavan postupak klasifikacije novog primjera koristeći već izrađeno stablo, ali preostaje

problem pronalaska najboljeg stabla koristeći primjere za učenje.

Porastom broja značajki, broj mogućih stabala raste eksponencijalno, stoga nije komputacijski moguće pronaći najoptimalnije moguće stablo, ali postoje načini kojima se postiže prihvatljiva točnost u razumnom vremenu. Uglavnom se koriste takozvani pohlepni algoritmi koji grade stablo donoseći lokalno optimalne odluke na svakom čvoru, ne uzimajući u obzir posljedice za buduće podjele. Potrebno je odabrati kriterij za računanje točnosti podjele jednog čvora; u radu se koristila Gini nečistoća, a alternativni odabiri su entropija, klasifikacijska pogreška ili logaritamska greška. Izraz za Gini nečistoću glasi:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{K} [p(i|t)]^2 \tag{5.4}$$

pri čemu je $p(i|t)$ udio primjera u čvoru t koji pripadaju klasi i, u problemu s K klasa.[37]

Stabla odluke sklona su prenaučenosti te imaju visoku varijancu što znači da mala promjena skupa primjera za učenje može dovesti do drastično različitog stabla. Sljedeći algoritam bolje će se nositi s tim glavnim problemima.[38]

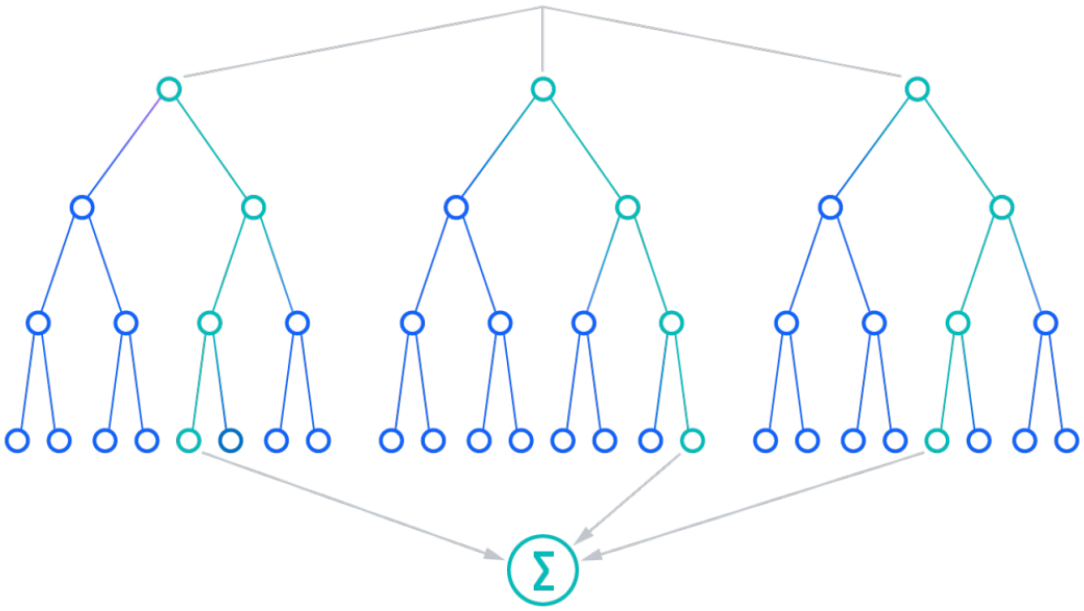
5.3 Algoritam slučajnih šuma

Algoritam slučajnih šuma kombinira niz stabala odluke i uprosječuje njihove rezultate. Uz osnovnu konstrukciju stabla odluke, algoritam unosi dvije ključne modifikacije. Prvo, prilikom izgradnje svakog stabla koristi se ”bootstrapping” tehnika, što znači da se uzorci iz skupa podataka biraju nasumično s ponavljanjem, stvarajući više podskupa koji su manji od originalnog skupa. Ova varijabilnost unosi raznolikost među stablima i pomaže smanjiti varijancu modela. Drugo, svako stablo koristi samo podskup značajki prilikom donošenja odluka. Na taj način svako stablo u ansamblu donosi odluke temeljene na različitim karakteristikama, dodatno povećavajući raznolikost.

Prilikom predviđanja, algoritam slučajnih šuma donosi konačne odluke putem glasanja; svako stablo daje svoj vlastiti glas za klasu koju predviđa kao rezultat za određeni ulazni primjer, prikazano na Slici . Klasa koja dobiva najveći broj glasova postaje konačna klasifikacija. U nekim implementacijama može se koristiti i težinsko glasanje, što znači da se svakom stablu može dodijeliti težina ovisno o njegovoj

točnosti ili performansama. Stabla s boljim performansama dobivaju veće težine, što ih čini važnijima prilikom glasanja.

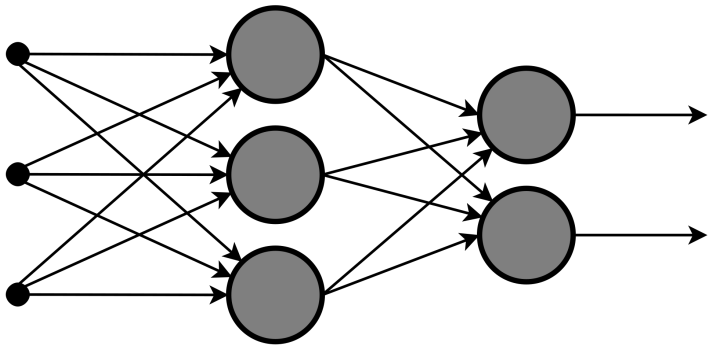
Ovaj algoritam pokazuje mnoge prednosti, uključujući visoku točnost, otpornost na šum i sposobnost rada s velikim skupovima podataka. Također omogućuje procjenu važnosti značajki, što je korisno za razumijevanje doprinosa svake značajke modelu.[39]



Slika 5.3: Grafički prikaz ideje algoritma slučajnih šuma, sastoji se od više stabala, prikazana su tri, koji pridonose ukupnom rezultatu, preuzeto iz [40]

6 Neuralne mreže

Neuralna mreža je algoritam modeliran po uzoru na ljudski mozak; sastoji se od jednostavnih operativnih jedinica neurona koji su međusobno povezani. Neuroni se postavljaju u slojeve tako da izlazi jednog sloja predstavljaju ulaze sljedećem sloju, kao što je prikazano na Slici 6.1. Neuron računa linearnu kombinaciju ulaznih vrijednosti s pripadnim vrijednostima težina koje se određuju procesom učenja te primjenjuje aktivacijsku funkciju. Odabir aktivacijske funkcije ovisi o podatcima i zadatku koji se izvršava; neki primjeri su već spomenute sigmoida i softmax, funkcija zglobnica ili tangens hiperbolni. Postoje i neke izvedbe koje ne koriste linearnu kombinaciju poput rezidualne mreže i transformer mreže.

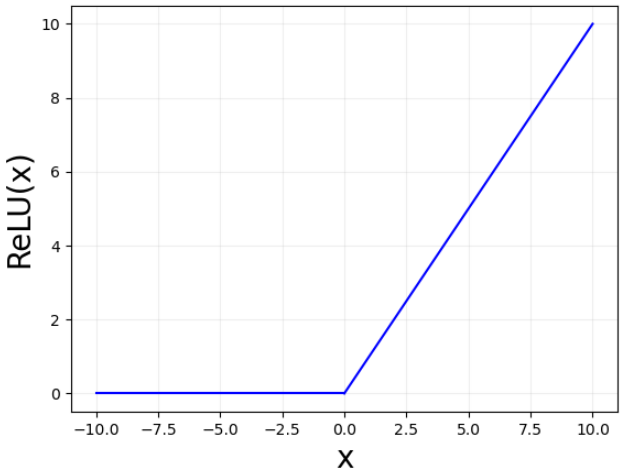


Slika 6.1: Shema rada neuronske mreže, dvoslojna mreža s 3 i 2 neurona, preuzeto iz [41]

6.1 Aktivacijska funkcija

Povijesno su se performanse neuralnih mreža značajno poboljšale uvođenjem po dijelovima linearnih aktivacijskih funkcija umjesto sigmoide, spomenute u poglavlju 4.4.1. Funkcije čuvaju svojstva linearnih modela koja ih čine jednostavnim za optimizaciju temeljenu na gradijentnom spustu, a također se prenose svojstva zbog kojih linearni modeli dobro generaliziraju.[42] Primjer takve funkcije je funkcija ReLU (*Rectified Linear Unit*), koja je i korištena u praktičnom dijelu ovog rada, prikazana na Slici 6.2.

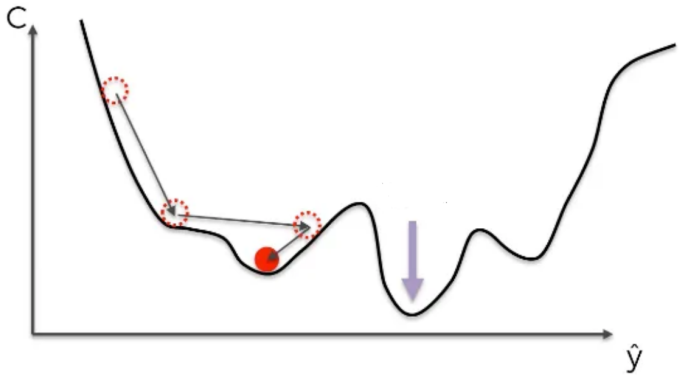
$$\text{ReLU}(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{6.1}$$



Slika 6.2: Izgled ReLU funkcije

6.2 *Stohastički gradijentni spust*

Gradijentni spust opisan u poglavlju 4.4.2 dobro radi za konveksne funkcije gubitka, ali kod neuralnih mreža funkcija gubitka poprima razne oblike te se standardnim gradijentnim spustom može dogoditi da optimizacija stane u lokalnom minimumu, prikazano na Slici 6.3.

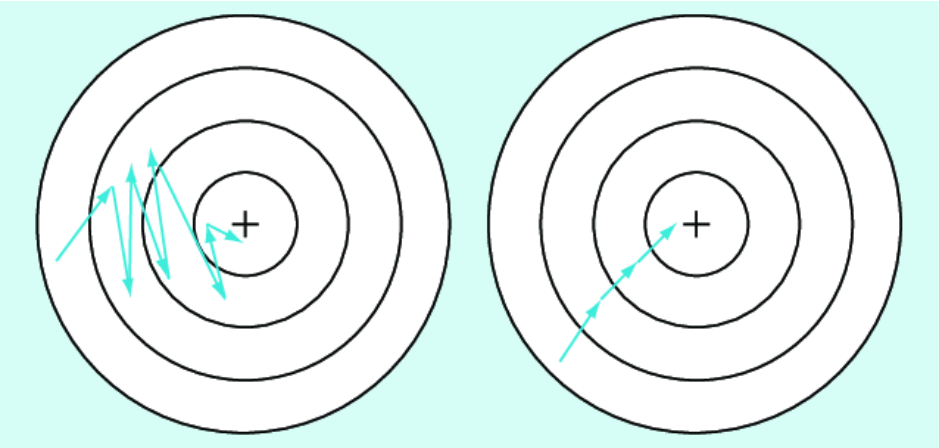


Slika 6.3: Prikaz funkcije gubitka koja nije konveksna; gradijentnim spustom može se dogoditi da se optimizacija zaustavi u lokalnom minimumu, označeno crvenom točkom, dok postoji povoljnija kombinacija parametara koja rezultira manjim gubitkom, označeno strelicom, preuzeto iz [43]

Pri standardnom gradijentnom spustu težine se ažuriraju nakon što je izračunata pogreška na cijelom skupu podataka te se model u prostoru značajki ažurira direktno u smjeru minimuma. U stohastičkom gradijentnom spustu težine se ažuriraju za svaki primjer pojedinačno, što znači da se događa spuštanje prema minimumu

u više koraka, prikazano na Slici 6.4. Stohastički spust manje je računalno zahtjevan od standardnog jer nije potrebno spremati sve podatke o greškama u memoriju kada se težine odmah ažuriraju pa se u istom vremenu može napraviti više koraka. Krivudanje pri spustu također pomaže u navedenom problemu izbjegavanja lokalnih minimuma.

U neuralnim mrežama koristi se kompromisno rješenje između standarnog i stohastičkog spusta pri kojem se težine ažuriraju za odabranu podgrupu primjera.



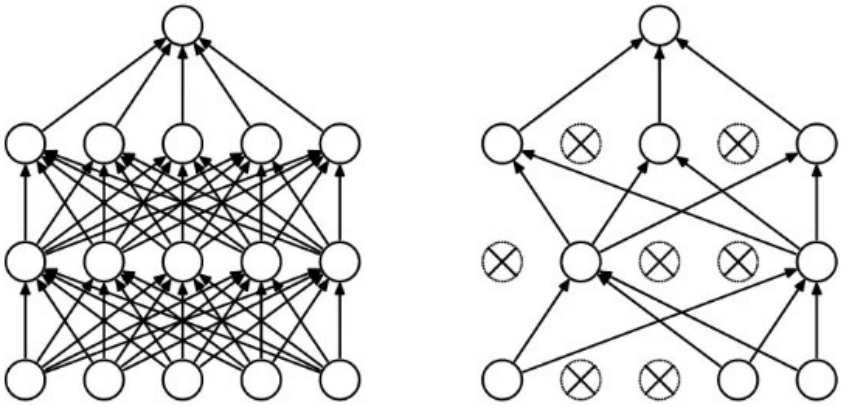
Slika 6.4: Shema rada gradijentnog spusta u dvodimenzionalnom prostoru značajki. Kružnice predstavljaju izokonture funkcije pogreške; lijevo je prikazan stohastički spust, a desno standardni, preuzeto iz [44]

6.3 *Postupak učenja neuralne mreže*

Učenje neuralnih mreža provodi se u epohama. Kao i u svakom problemu nadziranog strojnog učenja, ulaz algoritma je skup podataka za učenje s označenim primjerima. U prvoj epohi događa se najprije unaprijedna propagacija (*engl. forward pass*) pri kojoj se polazeći od primjera provode operacije koje su definirane arhitekturom neuralne mreže te se određuju oznake. Zatim se računa gubitak modela usporedbom sa stvarnim oznakama te se odvija unazadna propagacija (*engl. backpropagation*) pri kojoj se ažuriraju težine pojedinih neurona s ciljem minimizacije pogreške. Postupak unaprijedne i unazadne propagacije odvija se u manjim grupama primjera (*engl. batch*), a jedna epoha gotova je kada se postupak provede na cijelom skupu za učenje. Sveukupno učenje neuralnih mreža sastoji se od niza epoha tako da neuralna mreža više puta procesuira isti set podataka, što pomaže modelu da poboljša rezultate i prilagodi se uzorcima u podatcima.

6.4 *Dropout metoda*

Dropout metoda koristi se u neuralnim mrežama u svrhu sprječavanja prenaučenosti modela. Osnovna ideja je da se tijekom učenja modela nasumično isključuju određeni neuroni u neuronskom sloju. Dropout koeficijent poprima vrijednosti od 0 do 1 i označava vjerojatnost da će se neuron isključiti. U svakoj iteraciji neki neuroni se ”gase” i ne doprinose konačnom rezultatu. Neuralne mreže su vrlo fleksibilne i sposobne za pamćenje šuma i specifičnih karakteristika u podatcima pa dropout služi kao svojevrsna regularizacija modela, prisiljavajući ga da se prilagodi širokom spektru informacija u trening podatcima umjesto da se specifično prilagodi pojedinim uzorcima. Na Slici 6.5 prikazan je primjer standardne neuralne mreže te mreže s uključenim dropoutom.

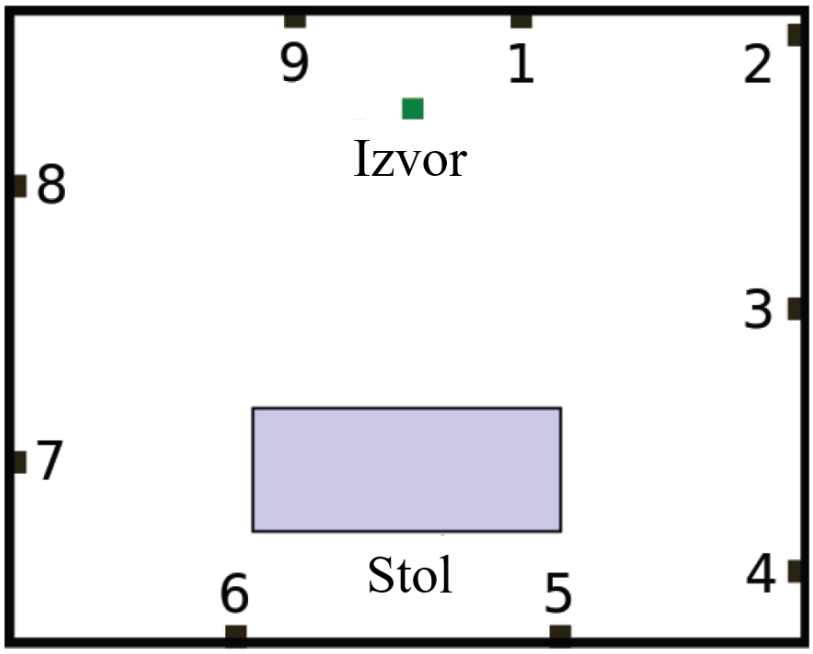


Slika 6.5: Shema neuralnih mreža; lijevo potpuno povezana mreža, a desno mreža s dropoutom, preuzeto iz [45]

7 Obrada podataka

7.1 Eksperimentalni postav

Provedeni eksperiment izgledao je tako da je u prostoriji postavljen izvor Wi-Fi polja te 9 detektora raspoređenih po zidovima; tlocrt prikazan na Slici 7.1. Ljudi u grupama različitih veličina hodali su po prostoriji ili sjedili te se cijelo vrijeme bilježila RSSI vrijednost.

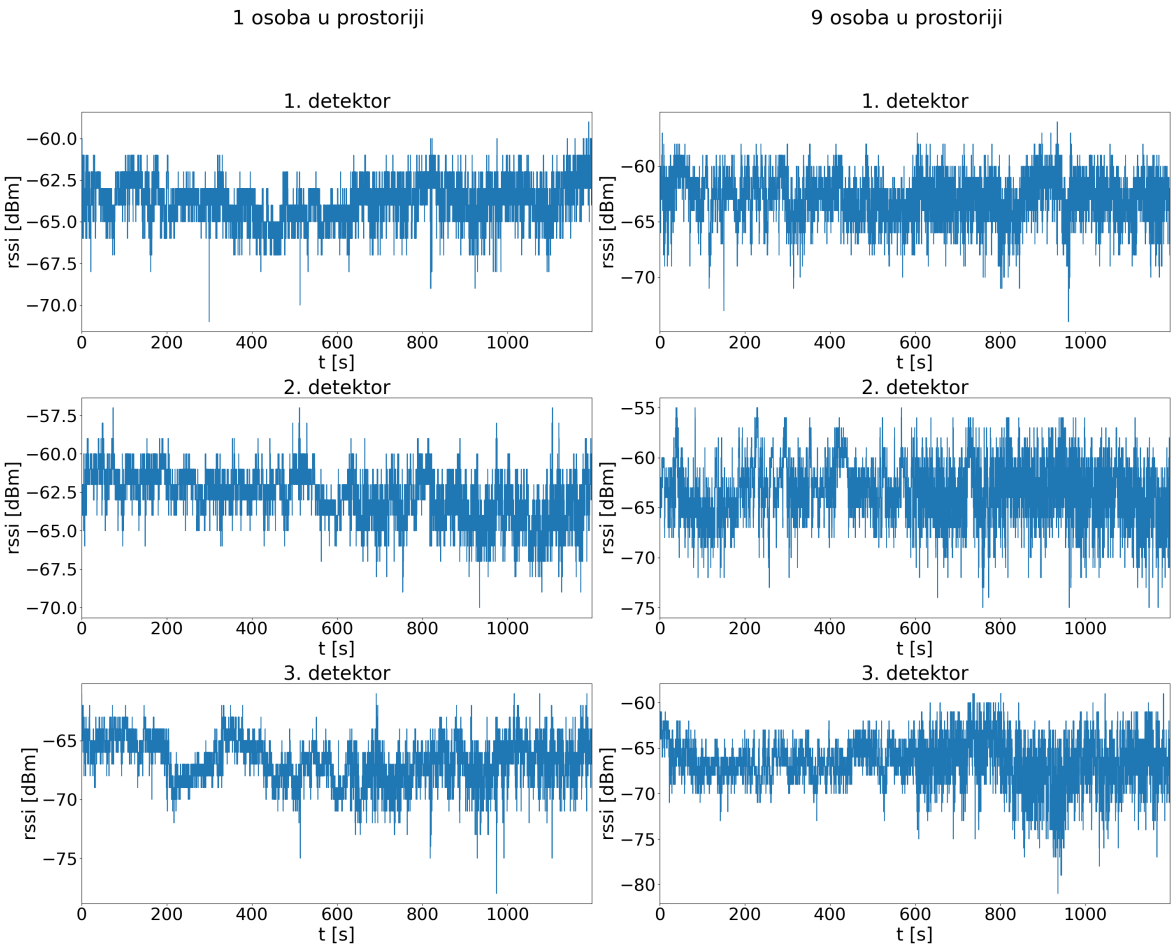


Slika 7.1: Tlocrt prostorije u kojoj se provodio eksperiment

7.2 Podatci

Set podataka sastoji se od mjerenja na 9 detektora napravljenih za 5 različitih veličina grupa ljudi u prostoriji. Svako mjerenje (jedan detektor i jedna grupa ljudi) sadrži oko 11750 RSSI vrijednosti s odgovarajućim timestampom. Preciznost timestampa je mikrosekunda, vrijednosti se bilježe svaku desetinku sekunde, a svako mjerenje traje 20 minuta. Primjer mjerenih podataka prikazan je na Slici 7.2.

Za obradu podataka korišten je programski jezik Python. Podatci se učitavaju iz .csv datoteke korištenjem `pandas` biblioteke te se spremaju u listu. Konačni rezultat je lista koja sadrži 5 lista koje odgovaraju različitim veličinama skupina ljudi; svaka od tih lista sadrži 9 lista za odgovarajuće detektore koje sadrže zabilježene RSSI vrijednosti, implementacija je navedena u Kodu 7.1.



Slika 7.2: Primjer izmjerenih podataka na 3 detektora za 1 i 9 osoba u prostoriji

```
ln9=[]
for j in range(1,10,2):
    data=[]
    for i in range(1,10):
        l=pd.read_csv(str(j)+"Osoba/2022-10-19[ESP"+str(i)+"].csv")
        data.append(l.to_numpy()[:,:2])
    ln9.append(data)
```

Kod 7.1: Učitavanje podataka

7.3 Metoda

Ideja je koristiti mjerenja električnog polja u istom trenutku na svim detektorima, kako bi se dobila informacija o trenutnom stanju u prostoriji, međutim timestampovi nisu identični u različitim datotekama, odnosno ne postoji istovremeno mjerenje na svim detektorima pa se na eksperimentalnim podatcima ne može dobiti točno takav primjer. Stoga je odabrana tehnika koja se često koristi u obradi time seriesa[46, 47] u kojoj se primjer sastoji od nekoliko uzastopnih mjerenja za svaki detektor pa se

očitanja na detektorima mogu smatrati istovremenima s obzirom da obuhvaćaju veći vremenski period. Značajke se određuju računajući određene statističke veličine iz te grupe podataka, a veličina grupe tretira se kao parametar.

Slijedi (Kod 7.2) skup funkcija kojima se učitani podatci pripremaju na opisani način, koristeći funkciju `X,y=fun(ln,c)` pri čemu je `ln` učitana lista podataka, a c je broj podgrupa na koje se dijeli lista RSSI vrijednosti za jedan detektor i jednu veličinu skupine ljudi. Konačni rezultat je skup primjera `X`, dvodimenzionalna numpy lista čiji su redovi primjeri, a stupci značajke. Broj primjera je tada 5·c jer je dostupno 5 skupina ljudi različitih veličina te je svaka od pripadnih datoteka podijeljena na c poskupa koji će predstavljati primjere, a broj značajki je 12·9 jer je odabrano 12 statističkih veličina koje se računaju za svaki od 9 detektora, kasnije u analizi broj značajki će biti manji kada se budu obrađivali podatci za manji broj detektora.

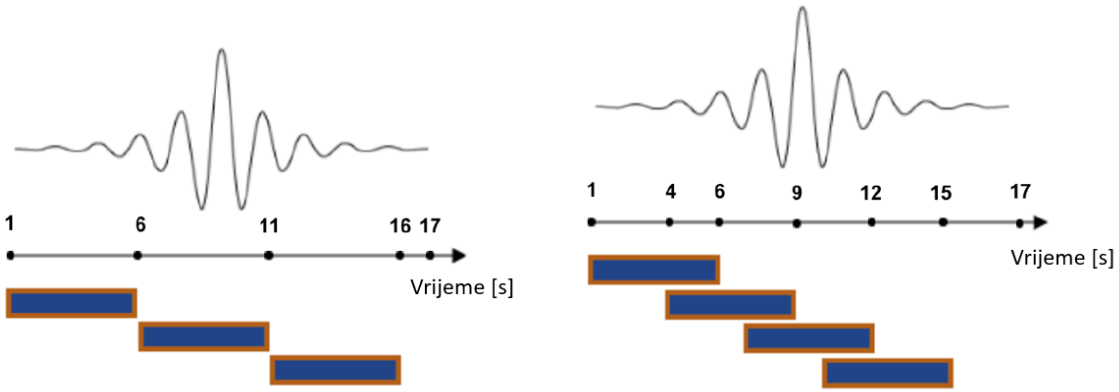
Funkcija `split(l,c)` dijeli listu na c dijelova s tim da će početni dijelovi biti duži za jedan element ako dužina liste nije djeljiva s c; time se osigurava neophodan uvjet da je broj primjera jednak za svaki detektor (zato što su značajke jednog primjera statističke veličine na svim detektorima), iako dužine lista RSSI vrijednosti nisu jednake. Povoljna činjenica u ovom pristupu jest da se ne gube informacije, sve izmjerene vrijednosti uzimaju se u obzir, za razliku od mogućosti da se sve liste skrate na istu duljinu brisanjem elemenata.

```
def split(l, c):
    k=len(l)// c
    m=len(l)%c
    return [l[i * k + min(i, m):(i + 1) * k + min(i + 1, m)] for i in
   range(c)]

def features(x):
    l=[]
    mean = np.mean(x); l.append(mean)
    std = np.std(x); l.append(std)
    variance = np.var(x); l.append(variance)
    min_val = np.min(x); l.append(min_val)
    max_val = np.max(x); l.append(max_val)
    percentile_25 = np.percentile(x, 25); l.append(percentile_25)
    percentile_50 = np.percentile(x, 50); l.append(percentile_50)
    percentile_75 = np.percentile(x, 75); l.append(percentile_75)
    skewness = skew(x); l.append(skewness)
    kurt = kurtosis(x); l.append(kurt)
    entropy =  entr(x); l.append(entropy)
    f, psd = welch(x, nperseg=len(x)//8)
    spectral_entropy = entr(psd); l.append(spectral_entropy)
    return l

def entr(data):
    total_count = len(data)
    probabilities = np.array(list(collections.Counter(data).values()))
    / total_count
```

```
26      entropy = -np.sum(probabilities * np.log(probabilities))
27      return entropy
28
29  def shaper(data, j,c):
30      X = np.concatenate([np.vstack([features(chunk) for chunk in split(
        row[:, 1], c)]) for row in data], axis=1)
31      y = np.repeat(j, X.shape[0])
32      return (X,y)
33
34  def fun(ln,c):
35      X, y = zip(*[shaper(ln[i], 2*i+1,c) for i in range(len(ln))])
36      X, y = np.concatenate(X), np.concatenate(y)
37      return X,y
```

Kod 7.2: Funkcije za obradu podataka

U funkcijama je korišten nefizikalan parametar c kako bi se najjednostavnije izbjegao problem različitih dužina podlista. U analizi rezultata je obavljen prelazak na smisleniji parametar τ koji predstavlja vremensko trajanje jednog primjera te se u literaturi takav standardni parametar naziva vremenski prozor.



Slika 7.3: Primjer odabira nepreklapajućih vremenskih prozora lijevo i preklapajućih prozora desno, preuzeto iz [48]

Ovdje je donesen i odabir da se vremenski prozori ne preklapaju, što je računalno manje zahtjevno, ali može dovesti do gubitka podataka, na primjer ako se neki uzorak ili događaj odvije u trenutku koji je presječen prozorom. Preklapajući vremenski prozori omogućavaju veću fleksibilnost, moguće je koristiti prozore različitih duljina i razmatrati potpunu kauzalnost događaja te detektirati kratkotrajne uzorke u podatcima, a u konačnici dobiti i veći broj primjera za učenje, iako je nedostatak to što će ti primjeri biti zavisni u nekoj mjeri. Međutim, s obzirom da je konkretan problem statičan, u smislu da cilj nije detektirati neke aktivnosti ili posebne događaje u prostoriji, već samo broj ljudi u prostoriji koji se ne mijenja duž jednog mjerenja,

ovaj slučaj prikladniji je za nepreklapajuće prozore.[48, 49, 50] Na Slici 7.3 grafički je prikazan primjer odabira preklapajućih i nepreklapajućih vremenskih prozora te se već i na prikazanoj maloj domeni vidi da je broj primjera u slučaju preklapajućih prozora veći.

U nastavku su navedene veličine koje su odabrane za značajke uz njihove definicije, ako su x_i elementi jedne podliste, a n dužina podliste:

- aritmetička sredina:
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{7.1}$$

- standardna devijacija:
$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{7.2}$$

- varijanca:
$$V = \sigma^2 \tag{7.3}$$

- minimum i maksimum skupa

- 25., 50., i 75. percentil daju vrijednost koja odvaja taj postotak podataka

- koeficijent asimetrije (*engl. skewness*) definira se kao Fisher-Pearsonov koeficijent[51]:
$$g_1 = \frac{m_3}{m_2^{3/2}} \tag{7.4}$$
pri čemu je:
$$m_i = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^i \tag{7.5}$$

- koeficijent spljoštenosti (*engl. kurtosis*), koristi se Fisherova definicija pri kojoj je spljoštenost jednaka nuli za normalnu raspodjelu, uz koeficijente m_i definirane izrazom (7.5)[52]:
$$k = \frac{m_4}{m_2^2} - 3 \tag{7.6}$$

- entropija, konkretnije koristi se definicija Shannon entropije iz informacijske teorije:
$$H = -\sum_{i=1}^{c} p_i \cdot \log p_i \tag{7.7}$$

gdje su p_i vjerojatnosti da je element liste poprimio jednu od c vrijednosti.[53]

- spektralna entropija računa se kao entropija spektralne gustoće snage (*engl. power spectral density*) koja se aproksimira Welchovom metodom.[54, 55]

Osim standarnih `numpy` funkcija korištene su i funkcije `skew` i `kurtosis` iz biblioteke `scipy.stats` te funkcija `welch` iz biblioteke `scipy.signal`, a funkcija za računanje entropije je implementirana vlastitom funkcijom `entr` koja koristi funkciju `Counter` iz biblioteke `collections`.

8 Implementacija algoritama

8.1 Plitki modeli

8.1.1 Oblik modela

Korišteni su gotovi algoritmi strojnog učenja iz `sklearn` biblioteke. Odabrani modeli spremljeni su u rječnik (Kod 8.1) radi jednostavnijih iteracija po modelima i parametrima:

```
models = {
    'Logisticka regresija': LogisticRegression(),
    'Stroj potpornih vektora': SVC(),
    'K najblizih susjeda': KNeighborsClassifier(n_neighbors=3),
    'Stablo odluke': DecisionTreeClassifier(random_state=random_state),
    'Algoritam slucajnih suma': RandomForestClassifier(n_estimators=100, random_state=random_state)
}
```

Kod 8.1: Odabrani modeli

Za logističku regresiju zadani parametri funkcije `LogisticRegression` iz `sklearn.linear_model` odgovaraju problemu, automatski se koristi multinomijalna regresija te `lbfgs` optimizacijski algoritam što je u suštini gradijentni spust u kojem se stopa bira aproksimacijom druge derivacije funkcije pogreške.

Funkcija `SVC` iz biblioteke `sklearn.svm` odabrana je za izvršavanje algoritma stroj potpornih vektora te su također zadani parametri odgovarajući, ali najbitniji parametar za naglasiti je odabrana jezgrena fukncija, koja već spomenuta u poglavlju 4.5.2 Radi se o `'rbf'` jezgri, odnosno radijalnim baznim funkcijama (*engl. radial basis functions*). Taj naziv općenito odgovara jezgrama koje sličnost primjera mjere na temelju neke funkcije euklidske udaljenosti među primjerima.[56] U ovom slučaju konkretno je funkcija definirana gausijanski[57]:

$$\kappa(\vec{x}, \vec{x'}) = e^{-\gamma \left\| \vec{x} - \vec{x'} \right\|^2} \tag{8.1}$$

što znači da su modelirane nelinearne granice među klasama u prostoru značajki. Ovakav odabir je računalno zahtjevniji od linearnog, ali daje bolje razultate u još uvijek razumnom vremenu. Parametar γ određuje širinu Gaussove raspodjele, ili intuitivnije, parametar govori o tome koliko daleko u prostoru značajki doseže utjecaj primjera; računa se kao recipročna vrijednost umnoška broja značajki i varijance primjera duž svake značajke.

Pri implementaciji algoritma k najbližih susjeda korištena je funkcija `KNeighborsClassifier` iz biblioteke `sklearn.neighbors` te je odabran parametar modela `n_neighbors=3`, što znači da algoritam uzima u obzir 3 najbliža susjeda, kao što je opisano u poglavlju 5.1.

Odabrana je globalna varijabla `random_state=42` za algoritme temeljene na šumama jer imaju komponente nasumičnosti, a poželjno je da rezultat bude isti u svakom pokretanju programa te je tako omogućena vjerodavnija usporedba različitih algoritama. Za stablo odluke korištena je funkcija `DecisionTreeClassifier` iz biblioteke `sklearn.tree`, a za algoritam slučajnih šuma `RandomForestClassifier` iz `sklearn.ensemble`. Zadani kriterij za procjenu kvalitete podjele je Gini nečistoća, kao što je opisano u poglavlju 5.2 Još je jedan bitan kriterij broj stabala odluke u algoritmu nasumičnih šuma koji je postavljen na 100.

8.1.2 Treniranje modela

```
def train(chunks,ln):

    results = {model: ([], []) for model in models}

    for c in chunks:
        X, y = fun(ln, c)

        X_scaled = scaler.fit_transform(X)
        X_train_val, X_test, y_train_val, y_test = train_test_split(
    X_scaled, y, test_size=test_size,random_state=random_state)

        for model_name, model in models.items():
            scores = cross_val_score(model, X_train_val, y_train_val,
    cv=kfold, scoring=scoring)
            results[model_name][0].append(np.mean(scores)*100)
            results[model_name][1].append((np.std(scores) / np.sqrt(
    kfold.n_splits))*100)

    return results
```

Kod 8.2: Treniranje plitkih modela

Treniranje modela provodi se funkcijom prikazanom u Kodu 8.2 postupkom opisanim u poglavlju 3.3. Najprije je odvojeno $20\,\%$ podataka koji će služiti za testiranje, funkcijom `train_test_split` iz biblioteke `sklearn.model_selection`. Zatim se za svaki parametar c i za svaki model provodi unakrsna k-struka provjera na skaliranim podatcima. Podatci se skaliraju funkcijom `StandardScaler` iz biblioteke `sklearn.preprocessing`, a za unakrsnu provjeru služi funkcija `cross_val_score` iz biblioteke `sklearn.model_selection`. Odabran je parametar $k = 5$, odnosno u svakom

koraku $20\,\%$ podataka čini validacijski set, konkretnije `cv` parametar u argumentu funkcije odabran je kao:

```
    kfold = StratifiedKFold(n_splits=k, shuffle=True, random_state=
random_state)
```

Kod 8.3: Odabir varijable koja određuje parametre unakrsne validacije

pri čemu je `StratifiedKFold` funkcija koja vraća indekse podjele skupova za učenje i testiranje u stratificiranoj unakrsnoj provjeri, također iz biblioteke `sklearn.model_selection`. U rječnik se konačno spremaju srednje vrijednosti točnosti s pripadajućim greškama.

8.1.3 Testiranje modela

Funkcija korištena za testiranje podataka prikazana je u Kodu 8.4. Konačna točnost modela računa se za jedan parametar τ, koji je pojedinačno odabran za svaki model kao onaj za koji je točnost unakrsne validacije maksimalna. Model se ponovo trenira na kompletnom skupu za učenje i validaciju te se računa točnost na skupu za testiranje. Fiksirana varijabla `random_state` osigurava jednaku podjelu podataka pri treniranju i testiranju. Drugi dio koda odnosi se na analizu točnosti po klasama, funkcijama iz biblioteke `sklearn.metrics` računaju se matrice zabune, preciznost, odziv i F_1 mjera.

```

def table(results, ln):
    acc_test = {}
    confusion_matrices = {}
    precision = {}
    recall = {}
    f1 = {}

    for model_name, model in models.items():
        c = chunks[np.argmax(results[model_name][0])]
        X, y = fun(ln, c)
        X_scaled = scaler.fit_transform(X)
        X_train, X_test, y_train, y_test = train_test_split(X_scaled,
    y, test_size=test_size, random_state=random_state)
        model.fit(X_train, y_train)
        acc_test[model_name] = accuracy_score(y_test, model.predict(
    X_test))

        y_pred = model.predict(X_test)
        confusion_matrices[model_name] = confusion_matrix(y_test,
    y_pred)
        precision[model_name] = precision_score(y_test, y_pred,average
    =None)
        recall[model_name] = recall_score(y_test, y_pred,average=None)
        f1[model_name] = f1_score(y_test, y_pred,average=None)

```

```
23     return acc_test, confusion_matrices,[precision,recall,f1]
```

Kod 8.4: Testiranje plitkih modela

8.2 Neuralne mreže

8.2.1 Oblik modela

Za implementaciju neuralnih mreža korištena je biblioteka `tensorflow.keras`. Trenirano je šest različitih mreža pri čemu su varirani broj slojeva mreže i dropout koeficijent. Korištena je dvoslojna, troslojna i četveroslojna mreža s dropout koeficijentima 0.2 i 0.4, odabir broja čvorova u svakom sloju prikazan je u Tablici 8.1. Za zadnji sloj svih mreža odabran je broj čvorova 10 jer je najveća oznaka klase 9, a odabrana je funkcija pogreške koja tretira izlaze kao cijele brojeve pa je potrebno imati dovoljno čvorova na izlazu. S obzirom da su neuralne mreže sklone prenaučenosti, arhitektura mreža pažljivo je odabrana promatranjem grešaka pri učenju na većem skupu mogućih mreža, u nastojanju da se pronađe optimalna ravnoteža s maksimalnom točnošću na testnom skupu podataka.

oznaka mreže	konstrukcija mreže	dropout koeficijent
1	40-10	0.2
2	40-10	0.4
3	30-60-10	0.2
4	30-60-10	0.4
5	30-60-60-10	0.2
6	30-60-60-10	0.4

Tablica 8.1: Arhitektura odabranih neuralnih mreža

U Kodu 8.5 prikazana je implementacija jedne od odabranih mreža, a ostale su implementacije priložene u Dodatku A; modeli su kao i kod plitkih modela spremani u rječnik. Za izgradnju mreže korištena je funkcija `Sequential` koja linearno dodaje slojeve redom navedenim unutar argumenta. Funkcija `Dense` stvara potpuno povezani sloj mreže s brojem čvorova navedenim u argumentu te je korištena aktivacijska funkcija ReLU, opisana u poglavlju 6.1. Funkcija `BatchNormalization` služi za normalizaciju, a funkcija `Dropout` postavlja navedeni postotak ulaza na vrijednost nula, čime se isključuje djelovanje tih čvorova.

```
model32 = tf.keras.Sequential([
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(120, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
    ])
```

Kod 8.5: Primjer implementacije jedne neuralne mreže s tri sloja i dropout koeficijentom 0.2 u svakom sloju

8.2.2 Treniranje modela

Ponovo se za učenje koristi metoda peterostruke unakrsne validacije prikazano u Kodu 8.7, ali u ovom slučaju pomoću funkcije `kfold.split`, pri čemu je kfold definiran kao u Kodu 8.3.

```
def ucenje(chunks,ln,models):

    results = {model: ([], []) for model in models}

    for c in chunks:
        X, y = fun(ln, c)

        X_scaled = scaler.fit_transform(X)
        X_train_val, X_test, y_train_val, y_test = train_test_split(X_scaled, y, test_size=test_size,random_state=random_state)

        for model_name, model in models.items():
            scores = []
            for train_index, val_index in kfold.split(X_train_val, y_train_val):
                X_train, X_val = X_train_val[train_index], X_train_val[val_index]
                y_train, y_val = y_train_val[train_index], y_train_val[val_index]
                model.fit(X_train, y_train, batch_size=20,epochs=30, validation_data=(X_val, y_val), callbacks=[early_stopping],shuffle=True)
                val_loss, val_accuracy = model.evaluate(X_val, y_val)
                scores.append(val_accuracy)
            acc=np.mean(scores)
            results[model_name][0].append(acc*100)
            results[model_name][1].append((np.std(scores) / np.sqrt(kfold.n_splits))*100)

    return results
```

Kod 8.6: Treniranje neuralnih mreža

Unutar funkcije `fit` odabran je broj epoha 30 i veličina podgrupe primjera za koje se težine ažuriraju 20. Odabrani su i kriteriji ranog zaustavljanja mreže:

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

Kod 8.7: Kriterij ranog zaustavljanja učenja

koristeći funkciju `EarlyStopping` iz modula `callbacks` biblioteke `keras` u čijim je argumentima odabrano da se prati iznos gubitka na skupu za validaciju (parametar `monitor`), zatim je za pararametar `patience` odabran iznos 5, što znači da će se učenje zaustaviti ako se gubitak na skupu za validaciju ne popravi nakon pet iteracija, te je konačno posljednji argument `restore_best_weights` postavljen na `True`, što znači da će se parametri modela nakon učenja vratiti na najbolje dobivene parametre tijekom učenja.

Učenje se pokreće funkcijom `compile` (Kod 8.8) u čijem je argumentu odabrana funkcija pogreške prikladna za rad s više klasa čije su oznake cijeli brojevi `SparseCategoricalCrossentropy` te optimizator `Adam` (*engl. adaptive moment estimation*) koji je proširena verzija stohastičkog gradijentnog spusta, komputacijski nije zahtjevan, ne zauzima puno memorije i dobro radi na primjerima velikog broja značajki.[58]

```
for model_name, model in models.items():
    model.compile(optimizer='adam', loss=tf.keras.losses.
    SparseCategoricalCrossentropy(from_logits=True), metrics=['
    accuracy'])
```

Kod 8.8: Pokretanje učenja neuralnih mreža

8.2.3 Testiranje modela

Testiranje se provodi na identičan način kao za plitke modele, birajući optimalni parametar c za svaki model i ponovo trenirajući model na cijelom skupu za učenje i validaciju te se točnost konačno računa odvojeno na skupu za testiranje, prikazano u Kodu 8.9.

```
def test(results, ln,models):
    acc_test = {}
    confusion_matrices = {}
    precision = {}
    recall = {}
    f1 = {}

    for model_name, model in models.items():
        c = chunks[np.argmax(results[model_name][0])]
        X, y = fun(ln, c)
        X_scaled = scaler.fit_transform(X)
        X_train, X_test, y_train, y_test = train_test_split(X_scaled,
    y, test_size=test_size, random_state=random_state)

        model.fit(X_train, y_train, batch_size=20, epochs=30,
    validation_data=(X_test, y_test), callbacks=[early_stopping])
        y_pred_probs = model.predict(X_test)
        y_pred_labels = np.argmax(y_pred_probs, axis=1)
        test_loss, test_accuracy = model.evaluate(X_test, y_test)
        acc_test[model_name]=test_accuracy

```

```
20          confusion_matrices[model_name] = confusion_matrix(y_test,
    y_pred_labels)
21          precision[model_name] = precision_score(y_test, y_pred_labels,
    average=None)
22          recall[model_name] = recall_score(y_test, y_pred_labels,
    average=None)
23          f1[model_name] = f1_score(y_test, y_pred_labels,average=None)
24
25      return acc_test, confusion_matrices,[precision,recall,f1]
```

Kod 8.9: Testiranje neuralnih mreža

9 Rezultati

9.1 Ovisnost točnosti o broju detektora

S obzirom da su na originalnom dostupnom skupu podataka dobivene visoke točnosti (veće od $95\,\%$ za svaki algoritam), algoritmi su korišteni i za različite podskupove podataka, odnosno za smanjeni broj detektora u prostoriji kako bi se moglo odrediti koji je broj detektora dovoljan za sigurnu predikciju broja ljudi u prostoriji. Algoritmi su testirani na setovima podataka dobivenim na 3, 4, 5, 7 i 9 detektora. Korišteni detektori odabrani su na način prikazan u Tablici 9.1, tako da su maksimalno udaljeni u prostoriji prikazanoj na Slici 7.1:

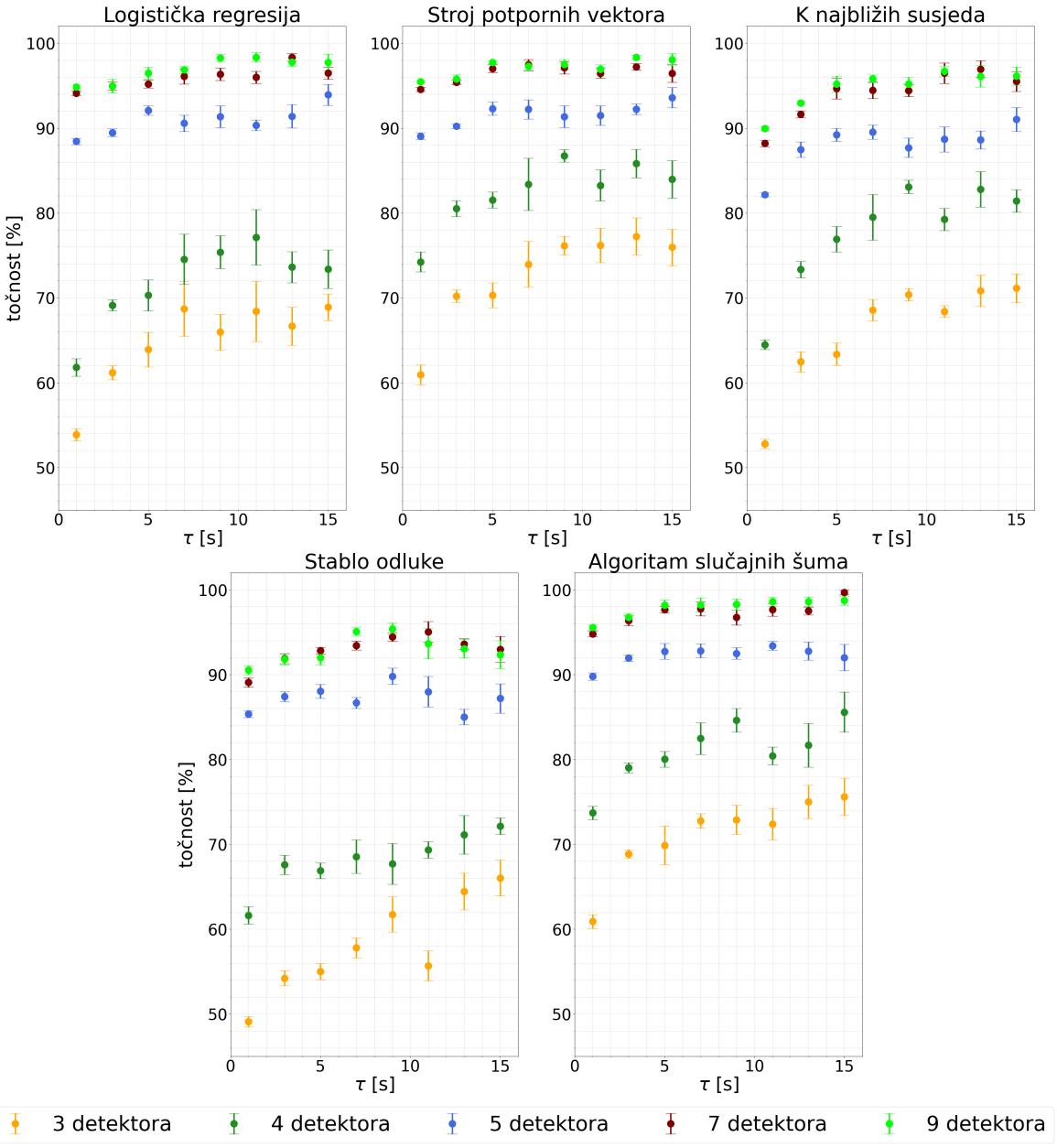
broj korištenih detektora	oznake detektora
3	3,7,9
4	3,5,7,9
5	2,4,5,7,9
7	1,3,4,6,7,8,9

Tablica 9.1: Odabir pojedinih detektora za korištenje podskupa podataka

9.1.1 Plitki modeli

Na Slici 9.1 prikazane su dobivene srednje vrijednosti točnosti s pripadajućim greškama unakrsne validacije. Testirani su algoritmi s iznosima parametra τ od 1 do 15 sekundi u koracima po dvije sekunde. Radi prikaza trenda, u Dodatku B.1.1 prikazani su i rezultati točnosti na dužoj vremenskoj skali, ali ti rezultati nisu korišteni u daljnjoj analizi jer odabir većih iznosa parametra τ nema smisla. Potrebno je ipak unaprijed izabrati razuman raspon u kojem se parametar može nalaziti, nije dovoljno tražiti samo najveću točnost na bilo kojoj domeni. Za odabir visoke vrijednosti parametra τ svi algoritmi daju točnost 100 %, ali to nema smisla za praktičnu uporabu. Algoritmi u tom slučaju daju savršenu točnost jer je skup primjera drastično smanjen pa su sami primjeri više razmaknuti u prostoru značajki i moguće ih je jednostavno razdvojiti, odnosno klasificirati. Konkretnije, već za $\tau = 17$ s ukupan broj primjera je 276, što je značajno smanjenje podataka u odnosu na 4696 primjera za $\tau = 1$ s. Kada bi bila dostupna duža mjerenja, točnost algoritma s takvim odabirom τ bi se smanjila. U praksi je naravno problem i što se očekuje da se primjeri mogu klasificirati u nekom

razumno kratkom roku pa nije optimalno mjeriti polje nekoliko minuta kako bi se odredio broj osoba u prostoriji.



Slika 9.1: Srednja vrijednost točnosti s pripadajućim greškama u ovisnosti o parametru τ za različite algoritme plitkog učenja te za različite odabire podskupa seta podataka, odnosno broja detektora

Na samoj Slici 9.1 vidi se da su odabiri parametra manjeg od 5 s premali, odnosno točnost raste za prvih nekoliko odabira parametara, a nakon toga se ne uočava značajni trend. Kao što je i očekivano u teorijskoj analizi iz poglavlja 5.3, na slici se može uočiti da algoritam slučajnih šuma daje značajno veće točnosti od stabla od-

luke. Stablo odluke daje najgore rezultate općenito na skupu odabranih algoritama, stoga je više prikazan u radu radi ilustracije performansi, nego zbog mogućnosti realne upotrebe. Algoritam *k* najbližih susjeda na skupu svih detetektora ima manje točnosti učenja od ostalih algoritama (osim stabla odluke), ali za tri detektora točnost je već usporediva s točnošću logističke regresije. Tu se očitava svojstvo algoritma *k* najbližih susjeda napomenuto u poglavlju 5.1 da se algoritam teže nosi s primjerima visoke dimenzije, a primjeri u problemu s manje detektora imaju i manje značajki.

Za svaki model i broj detektora numerički je određen optimalan parametar τ koji je korišten za računanje točnosti na testnom skupu podataka, prikazanih u Tablici 9.2.

broj detektora / model	3	4	5	7	9
logistička regresija	60.3 %	81.1 %	88.5 %	96.7 %	98.1 %
stroj potpornih vektora	84.4 %	80.8 %	93.6 %	95.2 %	99.0 %
k najbližih susjeda	70.5 %	80.8 %	89.7 %	94.4 %	99.1 %
stablo odlučivanja	65.4 %	76.9 %	89.2 %	90.6 %	96.2 %
algoritam slučajnih šuma	71.8 %	80.8 %	98.1 %	96.2 %	97.4 %

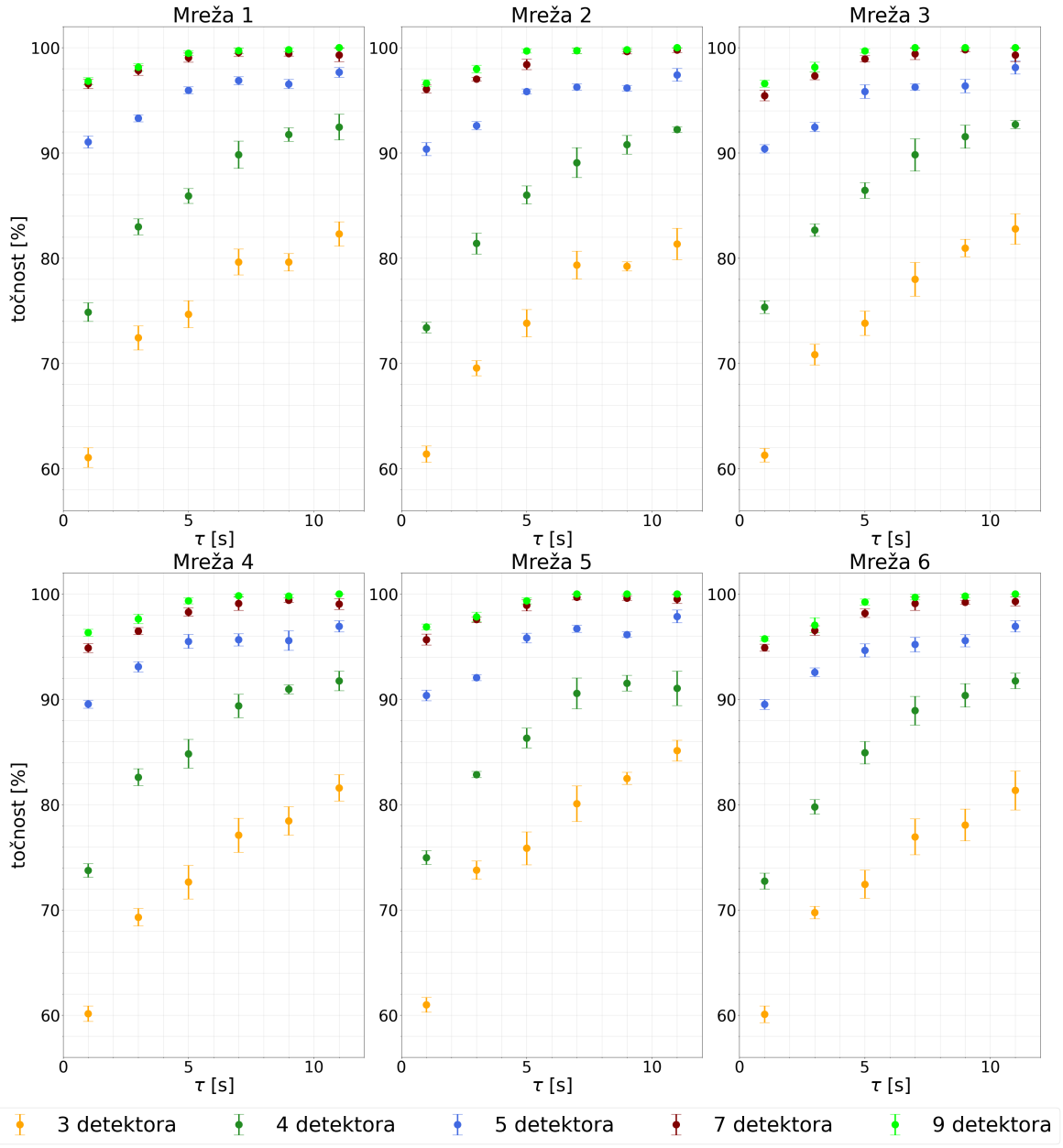
Tablica 9.2: Točnosti istreniranih algoritama na testnom skupu podataka

Najbolje rezultate točnosti na testnom skupu podataka pokazali su algoritmi logistička regresija, stroj potpornih vektora te algoritam slučajnih šuma. Za podskup od 7 detektora ne primjećuje se veliki pad u točnosti klasifikacije svih modela. Algoritam slučajnih šuma na podskupu podataka za 5 detektora još uvijek radi s impresivnom točnošću 98.1 %. Za najmanji skup podataka stroj potpornih vektora pokazuje značajan skok u točnosti naspram ostalih algoritama, konkretno točnost iznosi 84.4 %, što je vrlo dobar rezultat za takve podatke.

9.1.2 Neuralne mreže

Za neuralne mreže odabrana je kraća domena τ nego za plitke modele jer se ovdje zbog veće složenosti modela ranije dogodi prenaučenost za velike parametre τ, prikazano u Dodatku B.1.2. Testirane su neuralne mreže za parametre τ od 1 do 11 sekundi u koracima po dvije sekunde. Na Slici 9.2 prikazane su dobivene srednje vrijednosti točnosti s pripadajućim greškama unakrsne validacije, a arhitektura mreža

s pripadajućim oznakama koje se koriste na grafovima u rezultatima prikazana je u Tablici 8.1.



Slika 9.2: Srednja vrijednost točnosti s pripadajućim greškama u ovisnosti o parametru τ za različite neuralne mreže te za različite odabire podskupa seta podataka, odnosno broja detektora

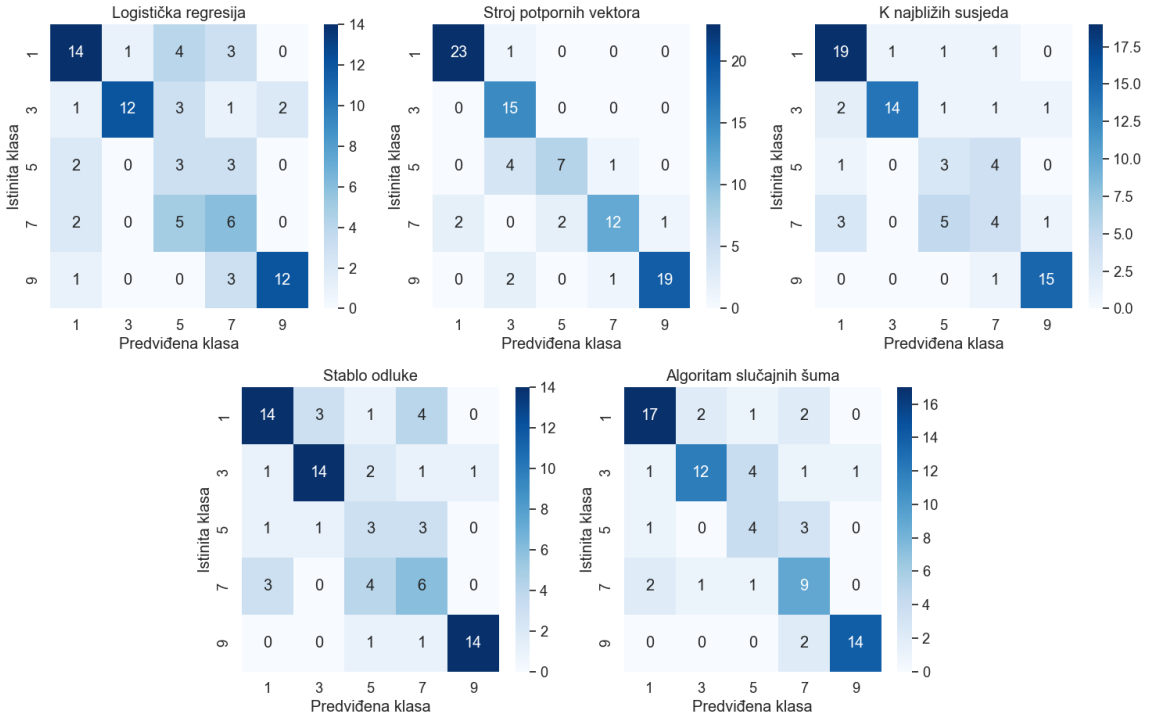
Točnosti istreniranih algoritama na testnom skupu podataka prikazane su u Tablici 9.3. Neuralne mreže dale su bolje konačne rezultate od plitkih modela. Najveća točnost iznosa 99.2 % dobivena je za troslojnu neuralnu mrežu s dropout koeficijentom 0.2.

broj detektora / oznaka mreže	3	4	5	7	9
1	79.2 %	80.0 %	95.3 %	98.1 %	99.1 %
2	76.4 %	86.8 %	94.3 %	99.2 %	98.1 %
3	72.6 %	84.9 %	96.2 %	99.2 %	99.2 %
4	78.3 %	86.8 %	96.2 %	97.7 %	97.2 %
5	73.6 %	89.6 %	95.3 %	99.1 %	98.1 %
6	69.8 %	82.1 %	92.5 %	96.2 %	99.1 %

Tablica 9.3: Točnosti istreniranih algoritama na testnom skupu podataka

9.2 *Raspodjela točnosti po klasama*

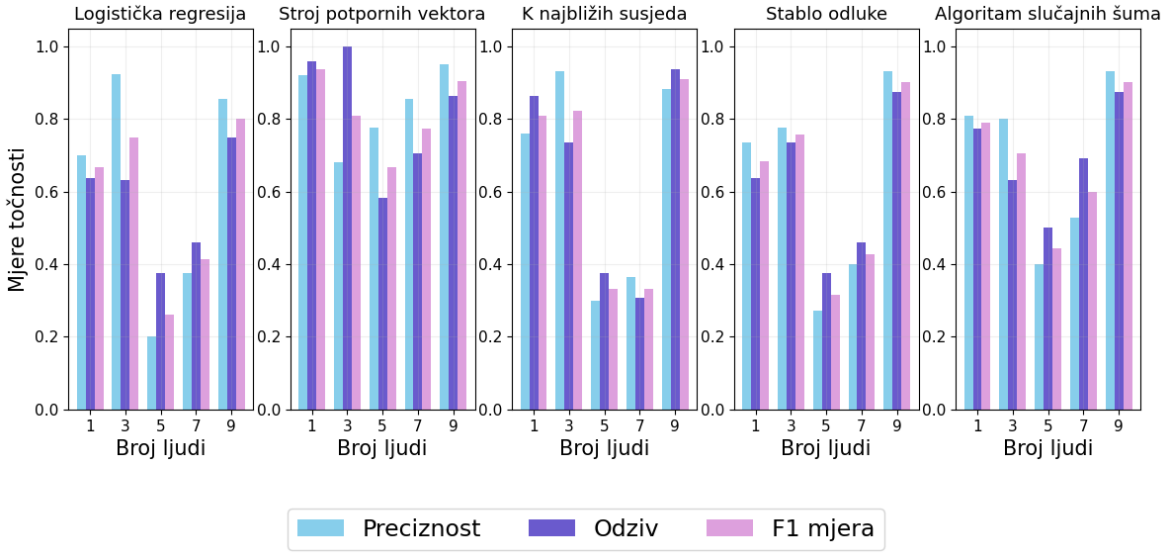
Radi kompletnije analize i boljeg razumijevanja rada algoritama, korisno je detaljnije proučiti rezultate, stoga su prikazane matrice zabune i mjere točnosti predstavljene u poglavlju 3.4. Ova analiza prikazana je samo za rezultate na podskupima podataka za manji broj detektora jer se za veće skupove podataka ne uočava toliko značajan trend zbog visokih točnosti; ti su rezultati radi potpunosti priloženi u Dodatku B.2.



Slika 9.3: Matrice zabune plitkih modela za optimalan odabir τ i za 3 detektora u prostoriji

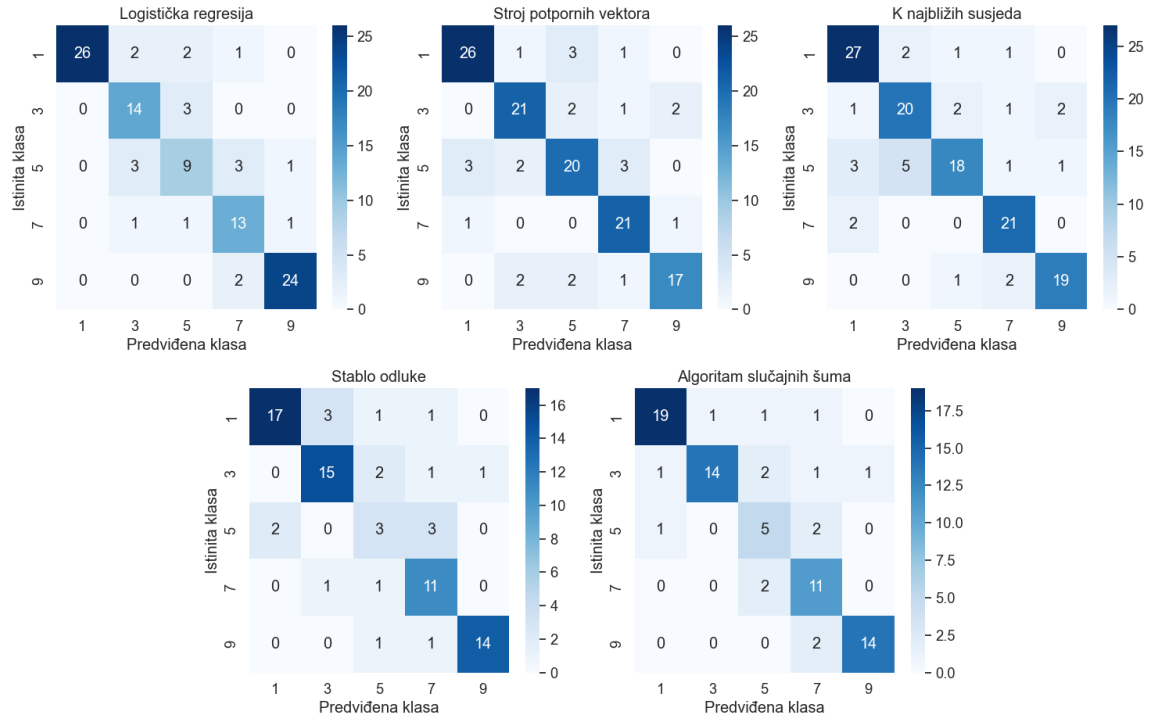
9.2.1 Plitki modeli

Na Slici 9.3 prikazane su dobivene matrice zabune za plitke modele na podskupu podataka za 3 detektora; točnosti iz Tablice 9.2 sada su podijeljene prema klasama. Iz matrica se može uočiti da najbolje rezultate daje stroj potpornih vektora jer ima najmanje nedijagonalnih elemenata različitih od nule, što ima smisla jer je na ovom skupu podataka i točnost algoritma daleko najveća, kao što je spomenuto u poglavlju 9.1.1.

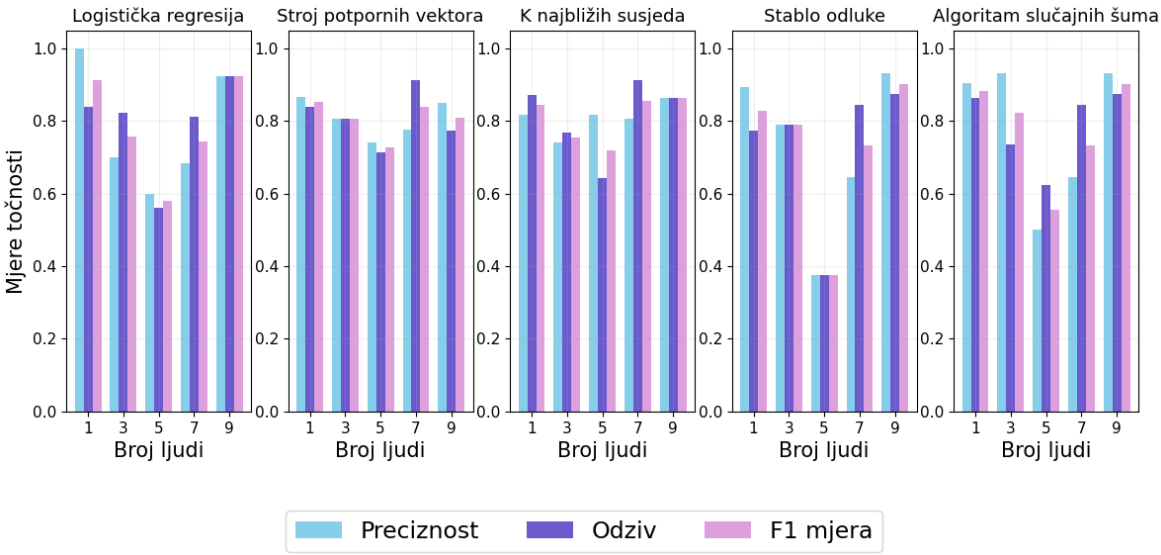


Slika 9.4: Točnosti po klasama plitkih modela za optimalan odabir τ i za 3 detektora u prostoriji

Na Slikama 9.4 i 9.6 prikazane su dobivene mjere točnosti na podskupu za tri i četiri detektora. Za sve plitke modele dobro se može uočiti da su na podskupu četiri detektora mjere točnosti simetričnije raspoređene nego za tri detektora, najmanje su za klasifikaciju 5 ljudi u prostoriji, a za 3 detektora je točnost za 7 ljudi u prostoriji usporediva s točnosti za 5 ljudi.



Slika 9.5: Matrice zabune plitkih modela za optimalan odabir τ i za 4 detektora u prostoriji

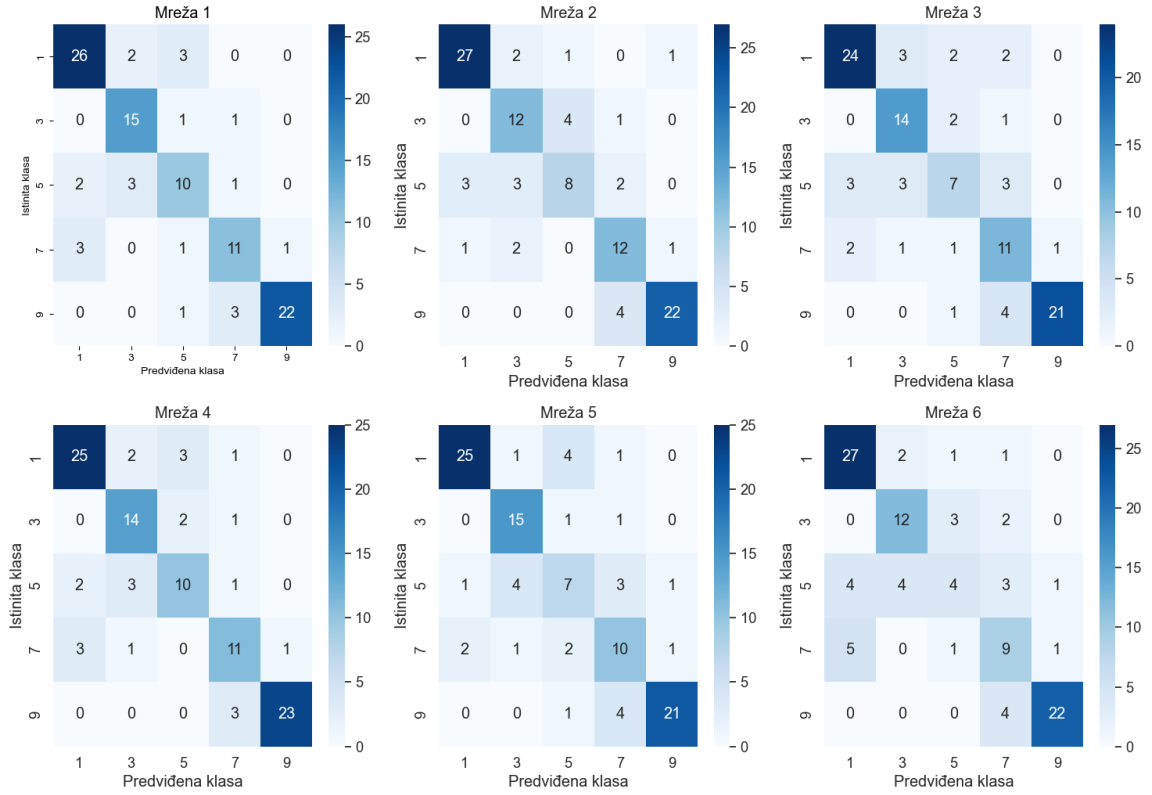


Slika 9.6: Točnosti po klasama plitkih modela za optimalan odabir τ i za 4 detektora u prostoriji

Na Slici 9.5 prikazane su matrice zabune plitkih modela na podskupu četiri detektora. Povezanost matrice zabune i mjera točnosti može se dobro pojasniti na dva

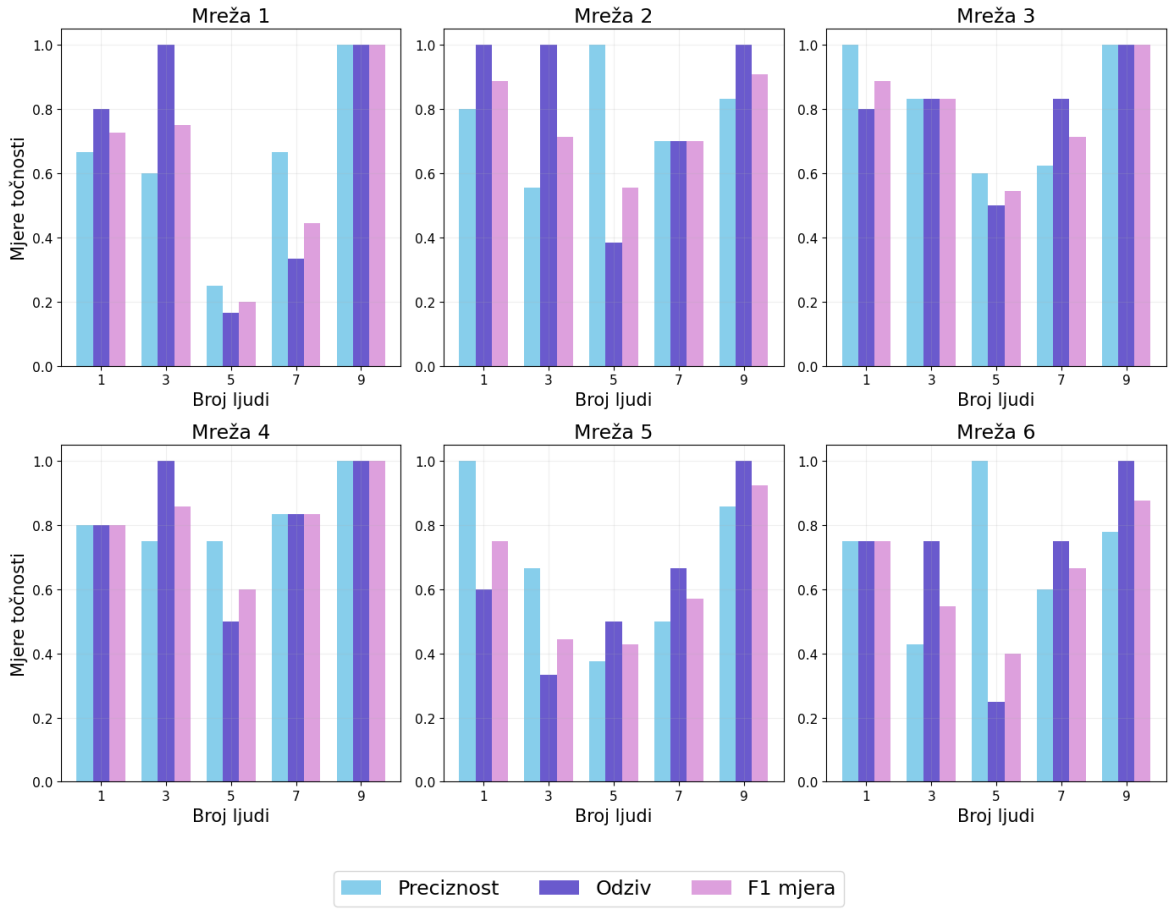
primjera gdje su mjere točnosti maksimalne. Na Slici 9.4 može se vidjeti da je za stroj potpornih vektora i troje ljudi u prostoriji odziv jednak jedan što znači da je svaki primjer u kojem je zaista troje ljudi u prostoriji točno predviđen. U matrici zabune taj se primjer lako uočava; u redu matrice koji odgovara klasi 3 su sve znamenke 0, osim one koja odgovara i stupcu klase 3. Kao što je objašnjeno u poglavlju 3.4, ovaj primjer ne znači da stroj potpornih vektora savršeno klasificira primjer troje ljudi u prostoriji, štoviše u tom primjeru mjera preciznost značajno pati, što se u matrici vidi po brojevima u stupcu koji odgovara trećoj klasi. Ova pojava ne znači da su preciznost i odziv uvijek drastično drugačiji, već primjer služi da bi se naglasila potreba za kompletnom analizom podataka jer je jednostavno doći do krivog zaključka. Protuprimjer gdje su odziv i preciznost sličnog iznosa na podskupu tri detektora mjere su za klasu 1 pri logističkoj regresiji i tri detektora, ili mjere za klasu 9 u algoritmu slučajnih šuma i tri detektora. Primjer gdje je preciznost maksimalna logistička je regresija za klasifikaciju jedne osobe na podskupu četiri detektora, što se u matrici vidi jer je stupac koji odgovara klasi 1 ispunjen nulama, osim u redu koji odgovara klasi 1.

9.2.2 Neuralne mreže



Slika 9.7: Matrice zabune za optimalan odabir τ i za 3 detektora u prostoriji

Na Slikama 9.7 i 9.8 prikazane su dobivene matrice zabune i mjere točnosti za neuralne mreže na podskupu tri detektora. Pri usporedbi mjera točnosti neuralnih mreža i plitkih modela za podskup tri detektora može se uočiti da neuralne mreže bolje raspoznaju klasu sedam ljudi u prostoriji, nego plitki modeli; raspodjela točnosti je simetričnija, s minimumom u središnjoj klasi koja odgovara skupini pet ljudi u prostoriji, pri čemu je točnost klasifikacije skupine pet ljudi vrlo niska.



Slika 9.8: Točnosti po klasama za optimalan odabir τ i za 3 detektora u prostoriji

Za sve prikazane ovisnosti mjere točnosti o broju ljudi uočava se jasan trend da se rubne klase klasificiraju s najvećom točnošću, algoritmi najbolje predviđaju postojanje jedne i devet osoba u prostoriji, a greške se događaju na središnjim veličinama grupa. Pojava je u skladu s očekivanim jer će primjeri s maksimalnim ili minimalnim brojem ljudi u prostoriji imati najmanje sličnih primjera iz druge klase. Za klase 1 i 9 ljudi najveća je vjerojatnost da će se dogoditi da se krivo klasificiraju u klase 2 i 7, a središnje klase mogu se pogrešno klasificirati u dvije bliske susjedne klase. Dodatno bi se moglo pretpostaviti da će točnost klasifikacije jedne osobe biti veća nego točnost klasfikacije devet osoba, ali u rezultatima se ne uočava značajna razlika. Veća količina osoba u prostoriji više će promijeniti Wi-Fi polje pa je za očekivati da će biti teže razlikovati promjene u većim skupinama ljudi, ali moguće je da su skupine od sedam i devet ljudi premale da bi se manifestirao taj problem.

10 Zaključak

Pokazano je da je devet detektora sasvim dovoljno za sigurnu predikciju broja ljudi u jednoj prostoriji, čak i korištenjem jednostavnijih modela strojnog učenja. Za neuralnu je mrežu moguće i sa samo pet detektora imati točnost predviđanja preko 96 %. U daljnjem razvijanju metode bilo bi korisno provesti eksperiment i s parnim brojevima ljudi u prostoriji jer algoritmi najviše griješe pri raspoznavanju među bliskim klasama gdje je u prostoriji pet i sedam ljudi pa bi dodavanje opcije da je u prostoriji šest ljudi moglo dovesti do lošijih rezultata. Kod bi se dalje mogao poboljšavati provođenjem detaljnije analize za odabir parametara svih modela, koristeći više standardnih značajki pri obradi timeseriesa te uvodeći metodu selekcije značajki. U konačnici, rad je ilustrirao teorijsku pozadinu metode, opisan je način rada algoritama i njihova implementacija u Pythonu te su dobivene visoke točnosti, što je dokaz da je predložena metoda detekcije ljudi dobra i možda će se u budućnosti zaista pokazati korisnom u svakodnevnom životu.

Dodatci

Dodatak A Implementacija neuralnih mreža

```
    model22 = tf.keras.Sequential([
    tf.keras.layers.Dense(40, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
    ])

    model24 = tf.keras.Sequential([
    tf.keras.layers.Dense(40, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10)
    ])

    model32 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
    ])

    model34 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10)
    ])

    model42 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
    ])

    model44 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),
```

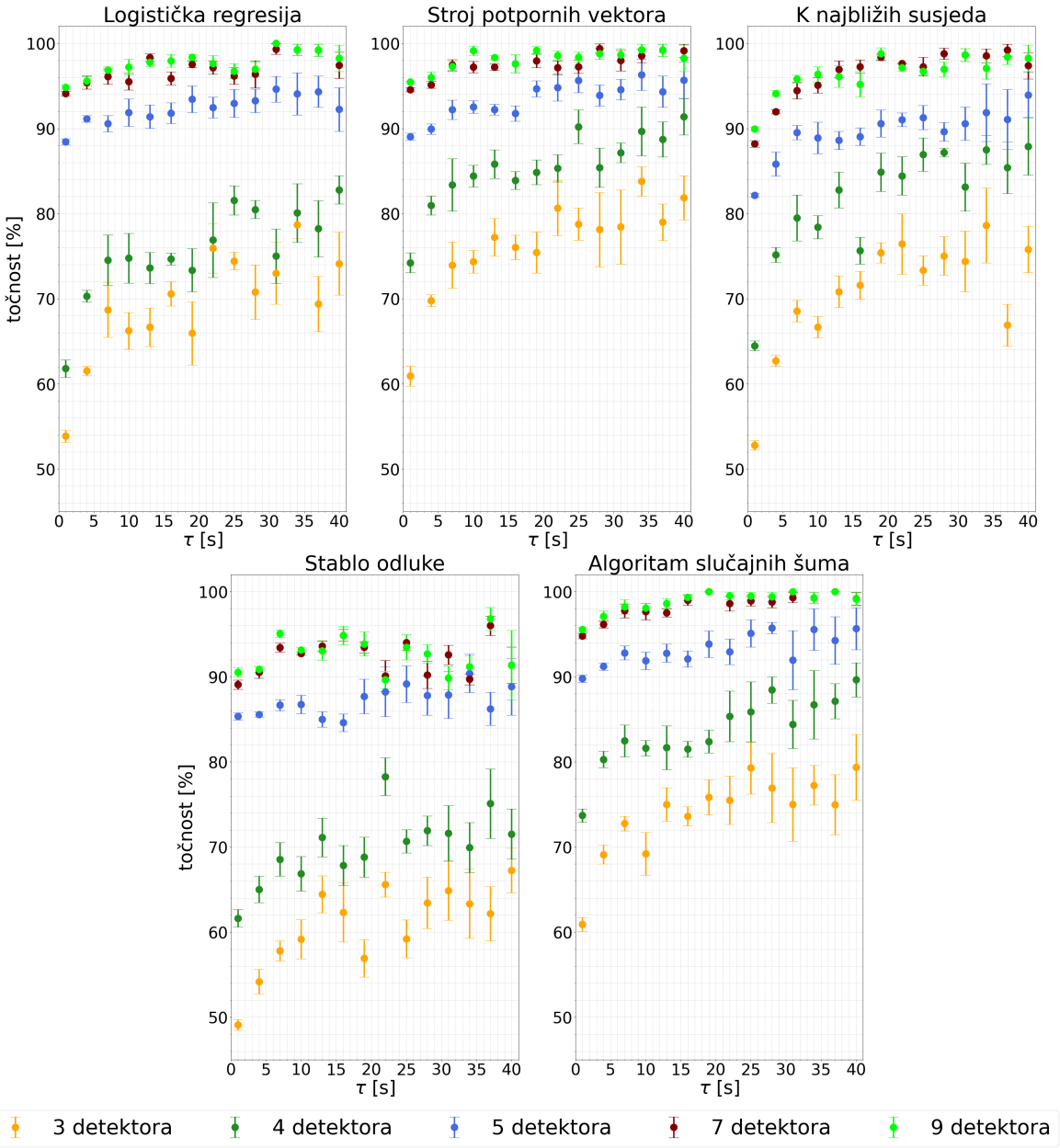
```
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10)
    ])

    models = {
        'neur22': model22,
        'neur24': model24,
        'neur32': model32,
        'neur34': model34,
        'neur42': model42,
        'neur44': model44
    }
```

Kod A.1: Odabrani modeli neuralnih mreža

Dodatak B Rezultati

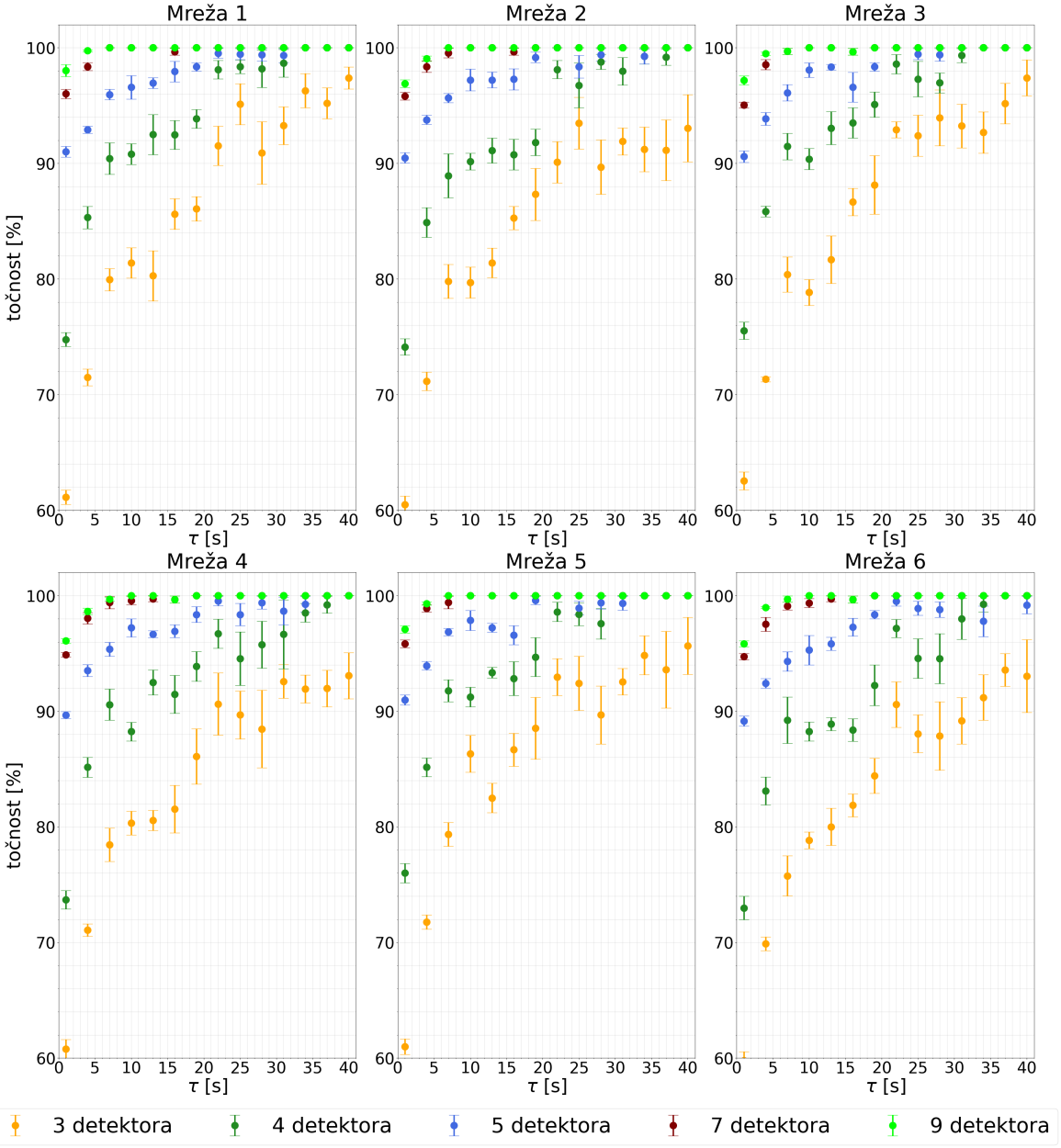
B.1 Ovisnost točnosti o parametru τ na dužoj vremenskoj skali

B.1.1 Plitki modeli



Slika B.1: Srednja vrijednost točnosti s pripadajućim greškama u ovisnosti o parametru τ za različite algoritme plitkog učenja te za različite odabire podskupa seta podataka, odnosno broja detektora

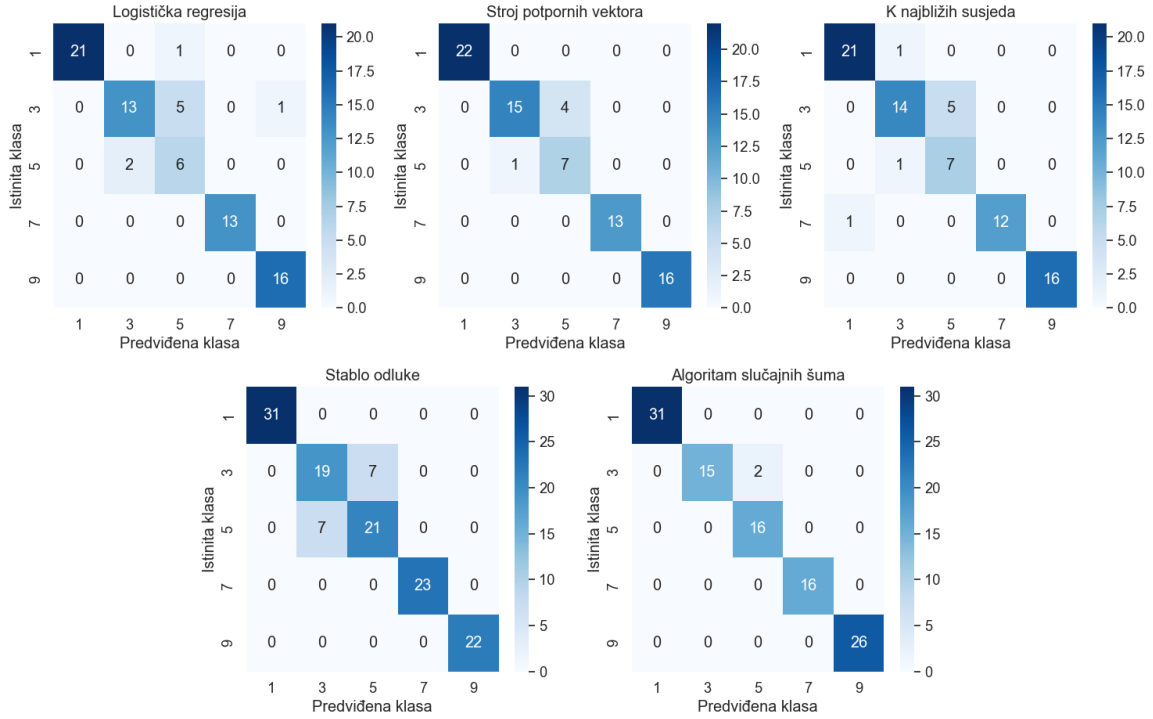
B.1.2 Neuralne mreže



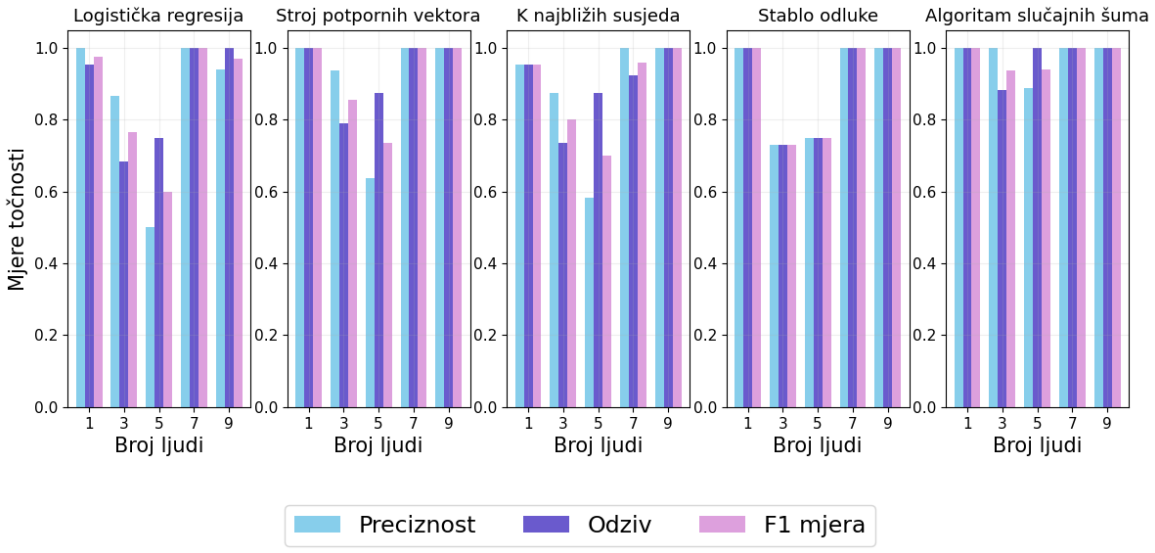
Slika B.2: Srednja vrijednost točnosti s pripadajućim greškama u ovisnosti o parametru τ za različite neuralne mreže te za različite odabire podskupa seta podataka, odnosno broja detektora

B.2 Raspodjela točnosti po klasama za veći broj detektora u prostoriji

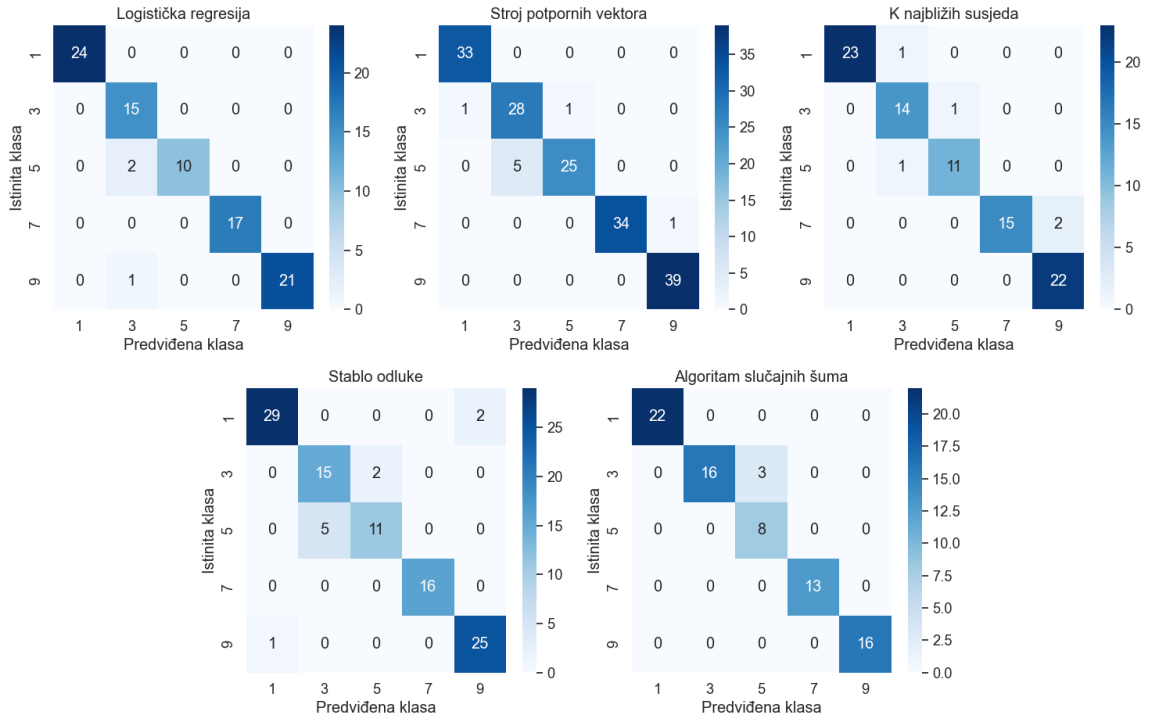
B.2.1 Plitki modeli



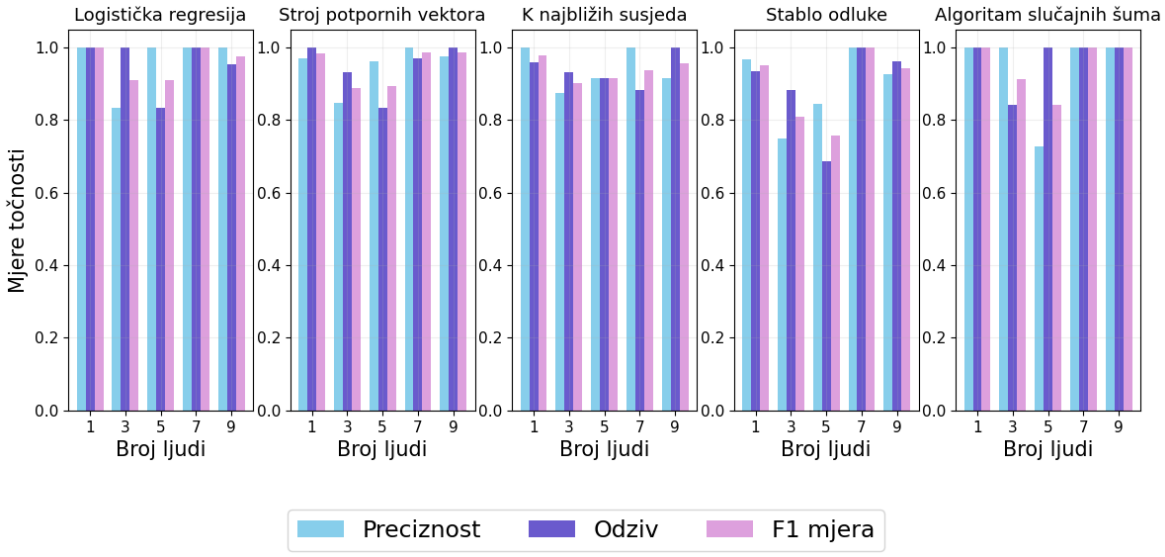
Slika B.3: Matrice zabune za optimalan odabir τ i za 5 detektora u prostoriji



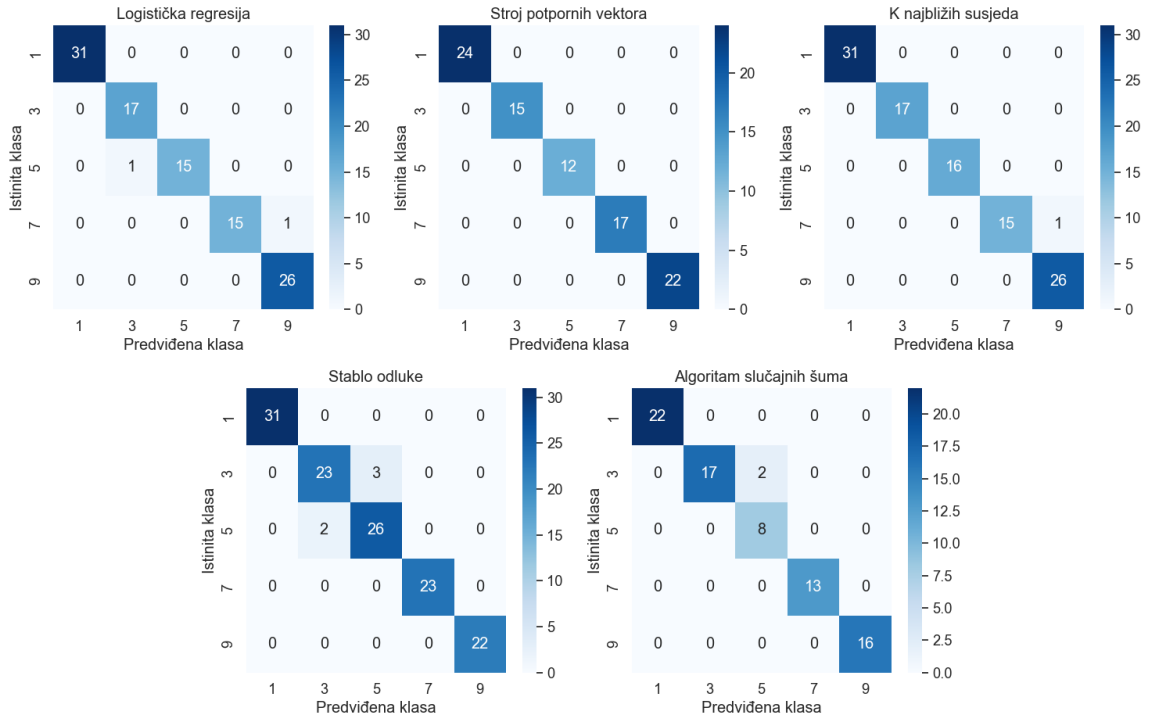
Slika B.4: Točnosti po klasama za optimalan odabir τ i za 5 detektora u prostoriji



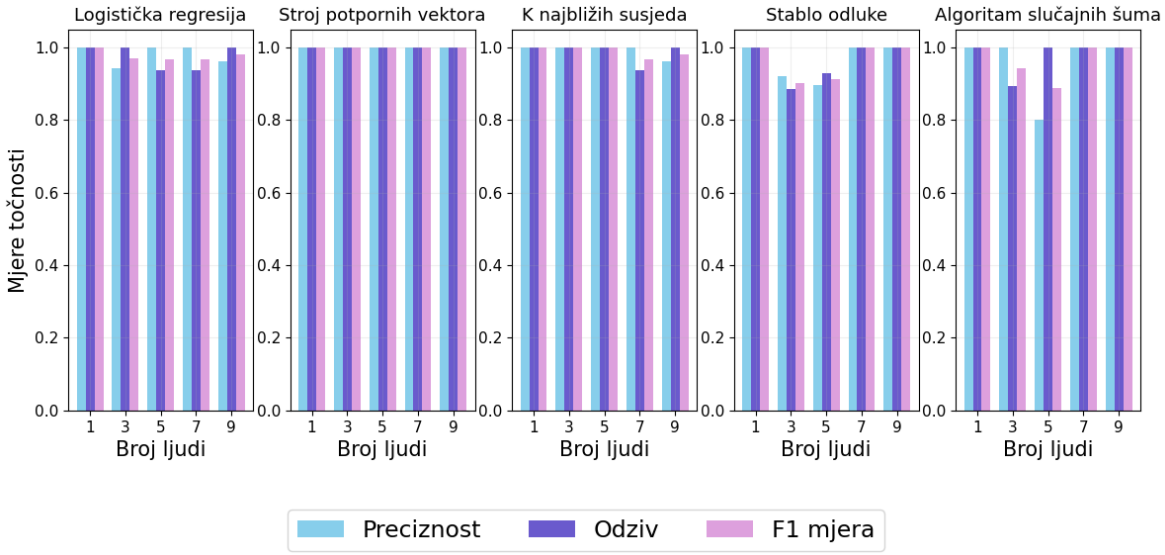
Slika B.5: Matrice zabune za optimalan odabir τ i za 7 detektora u prostoriji



Slika B.6: Točnosti po klasama za optimalan odabir τ i za 7 detektora u prostoriji

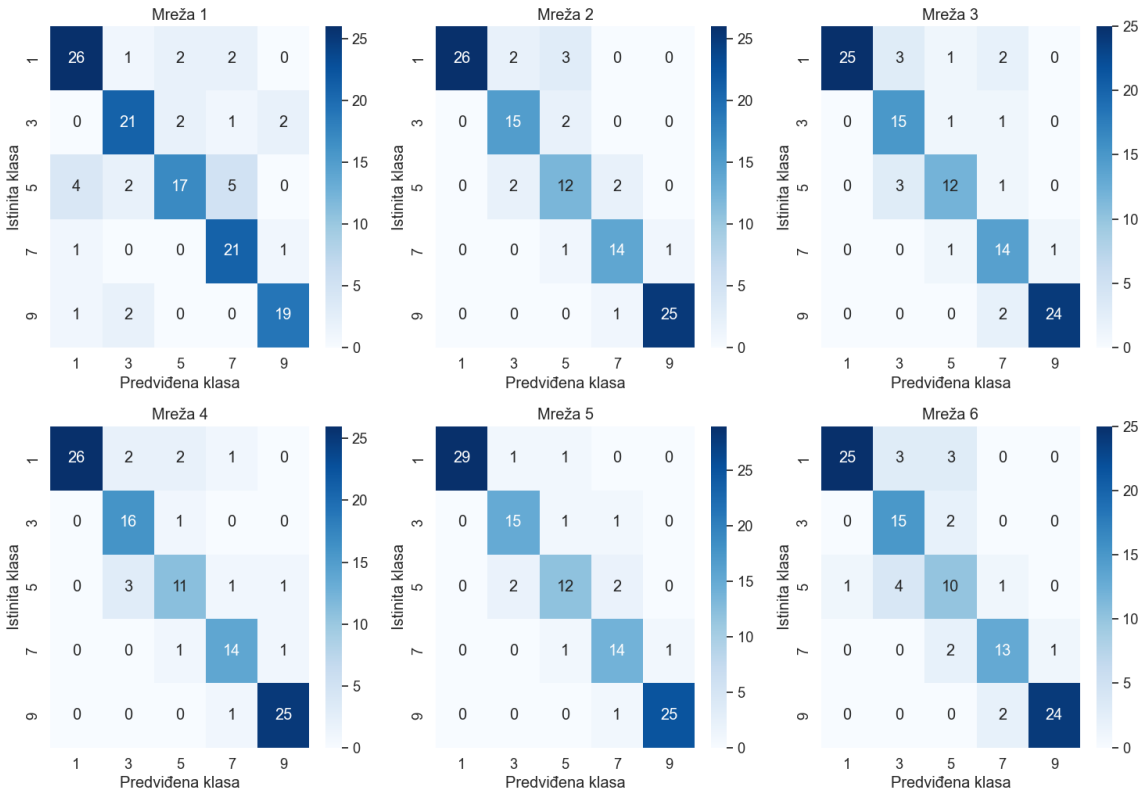


Slika B.7: Matrice zabune za optimalan odabir τ i za 9 detektora u prostoriji

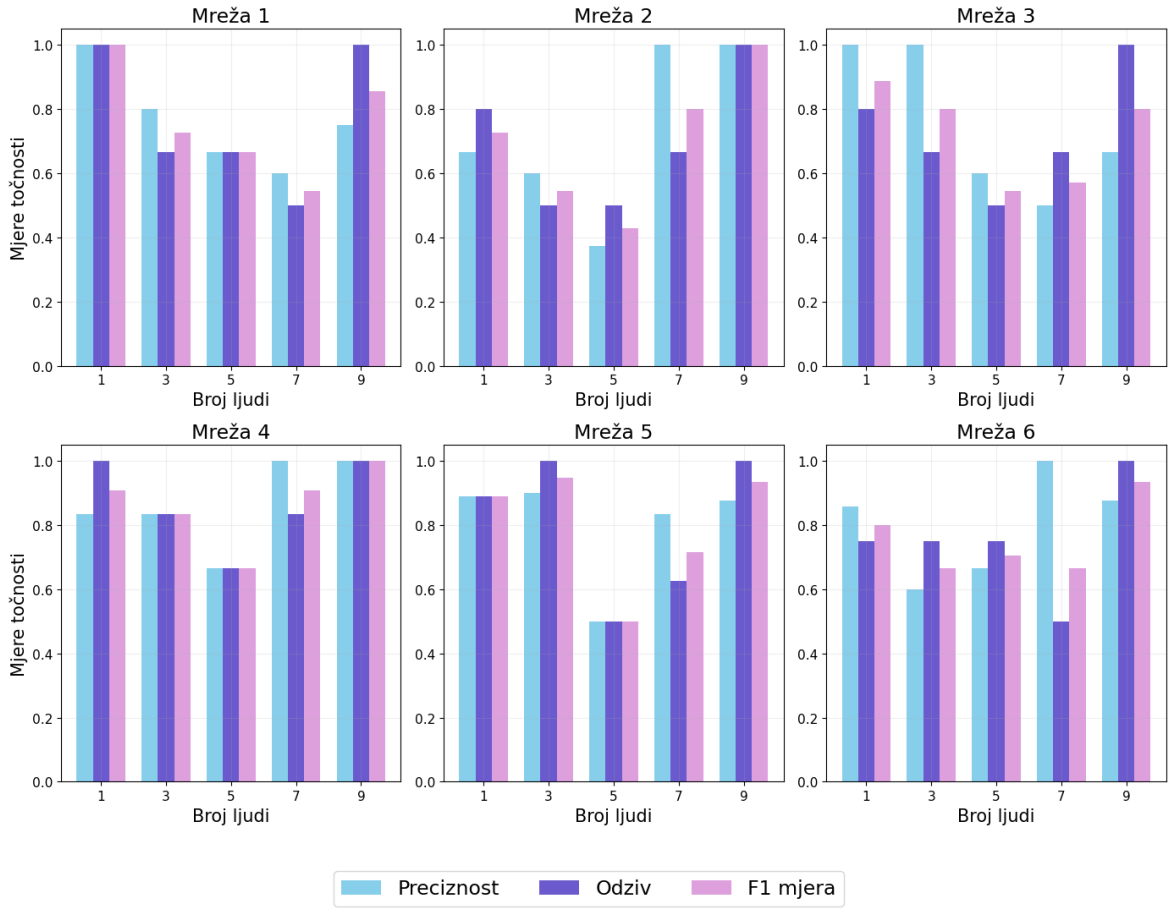


Slika B.8: Točnosti po klasama za optimalan odabir τ i za 9 detektora u prostoriji

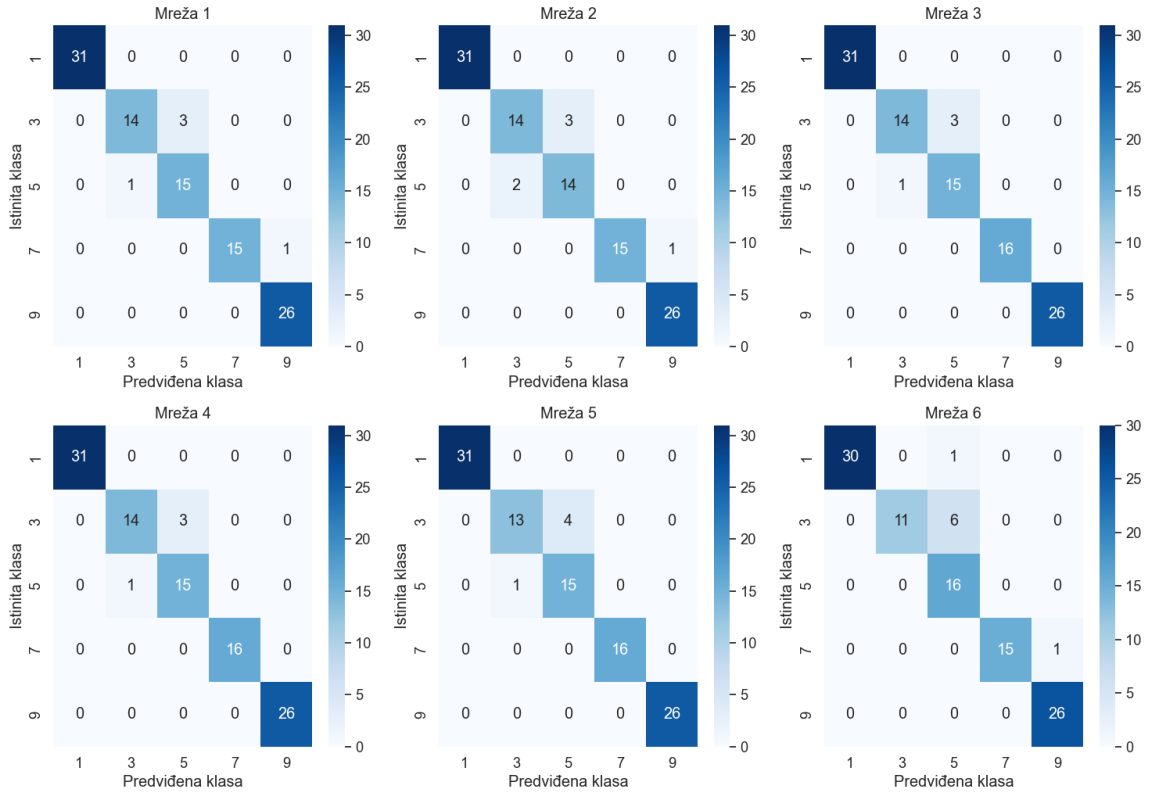
B.2.2 Neuralne mreže



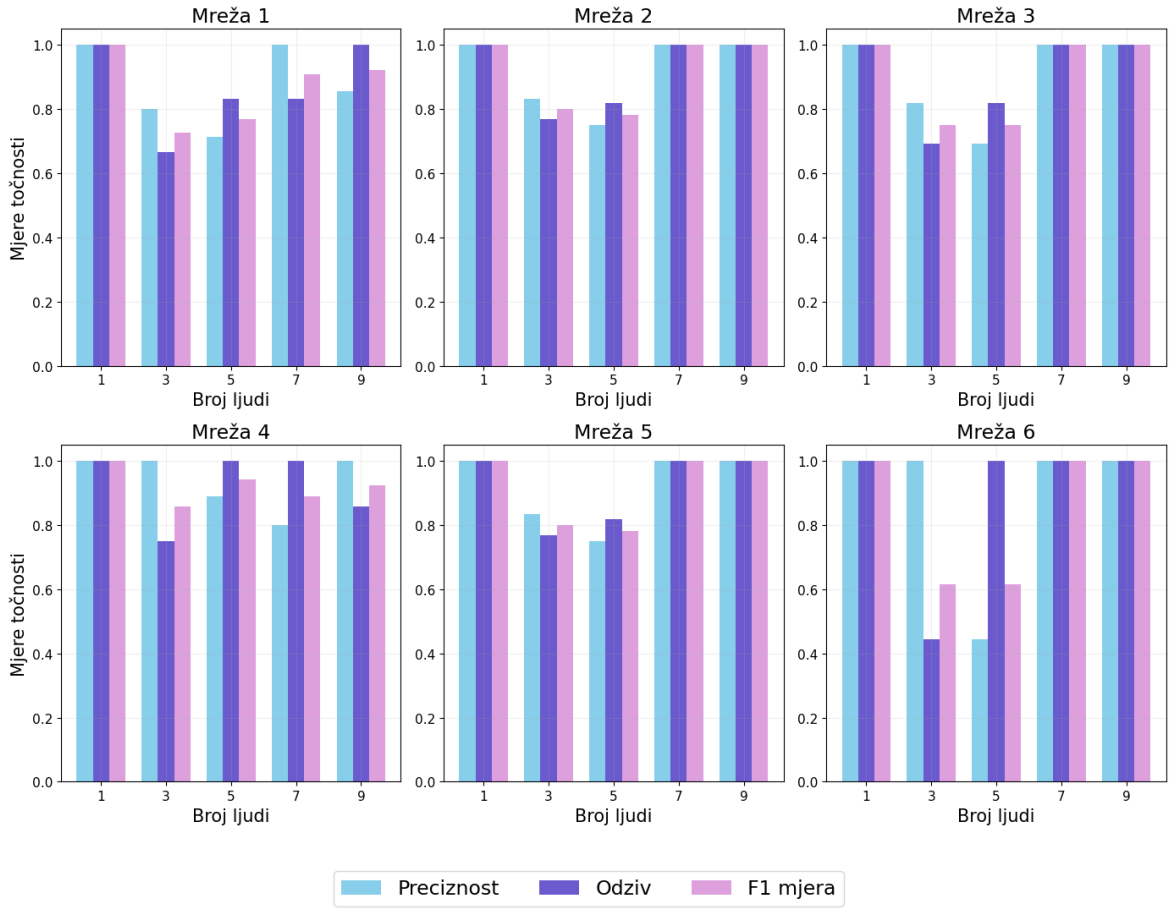
Slika B.9: Matrice zabune za optimalan odabir τ i za 4 detektora u prostoriji



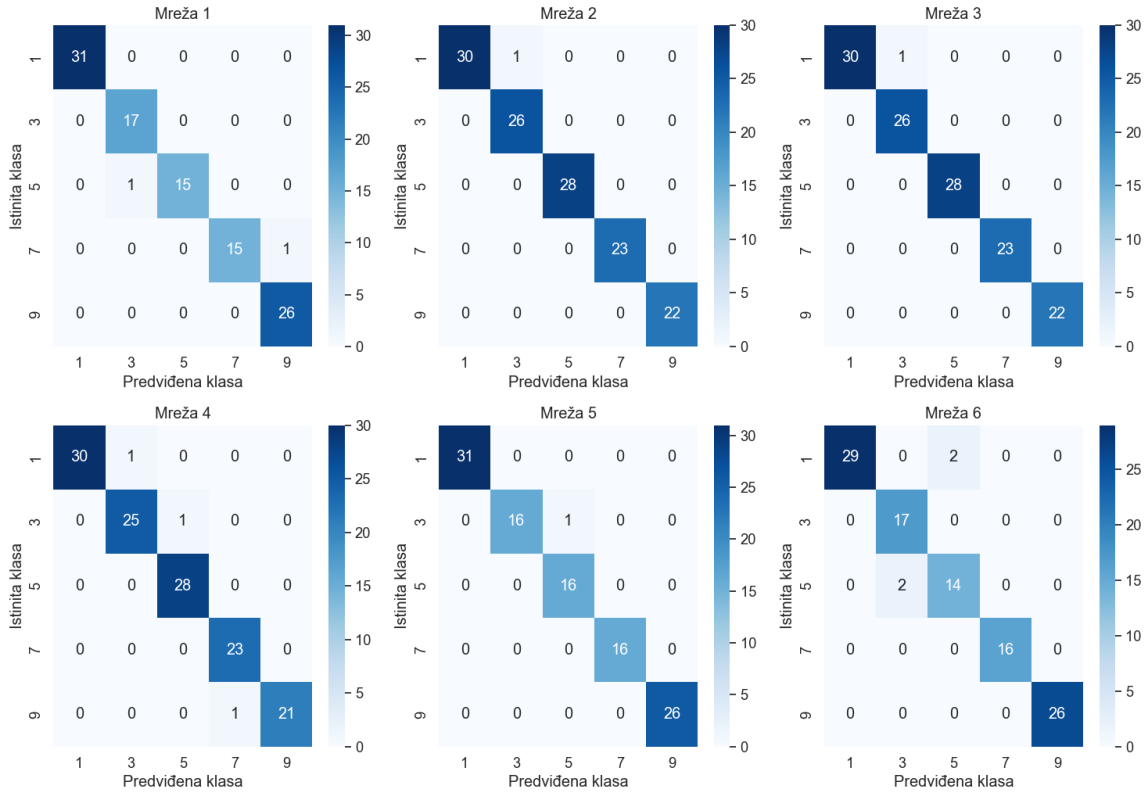
Slika B.10: Točnosti po klasama za optimalan odabir τ i za 4 detektora u prostoriji



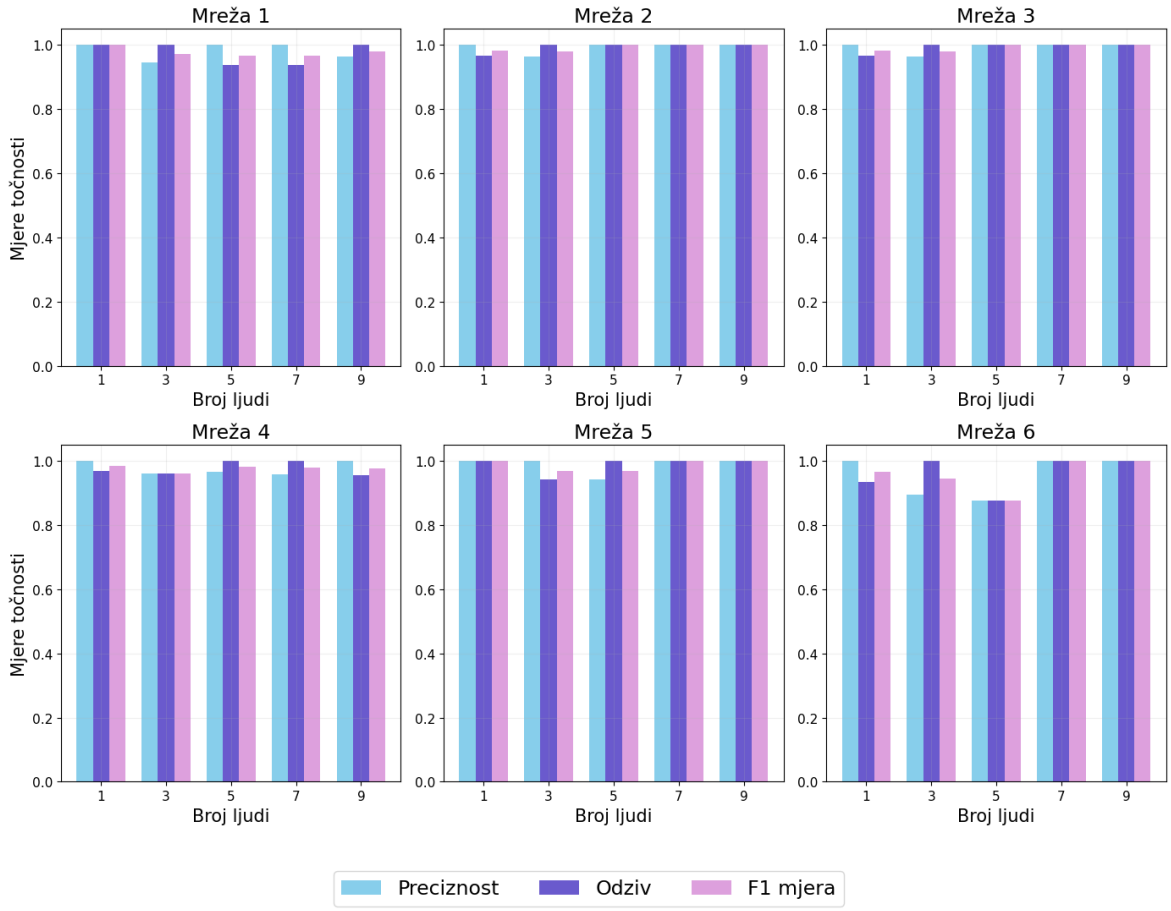
Slika B.11: Matrice zabune za optimalan odabir τ i za 5 detektora u prostoriji



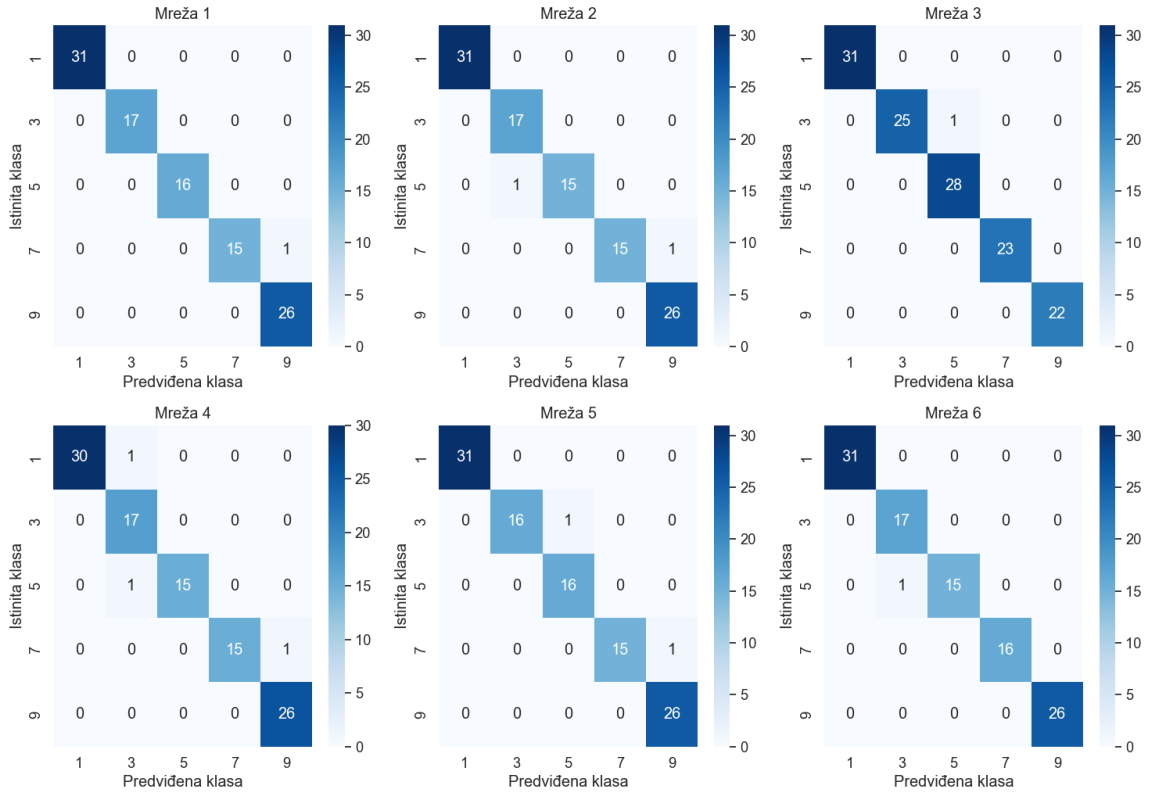
Slika B.12: Točnosti po klasama za optimalan odabir τ i za 5 detektora u prostoriji



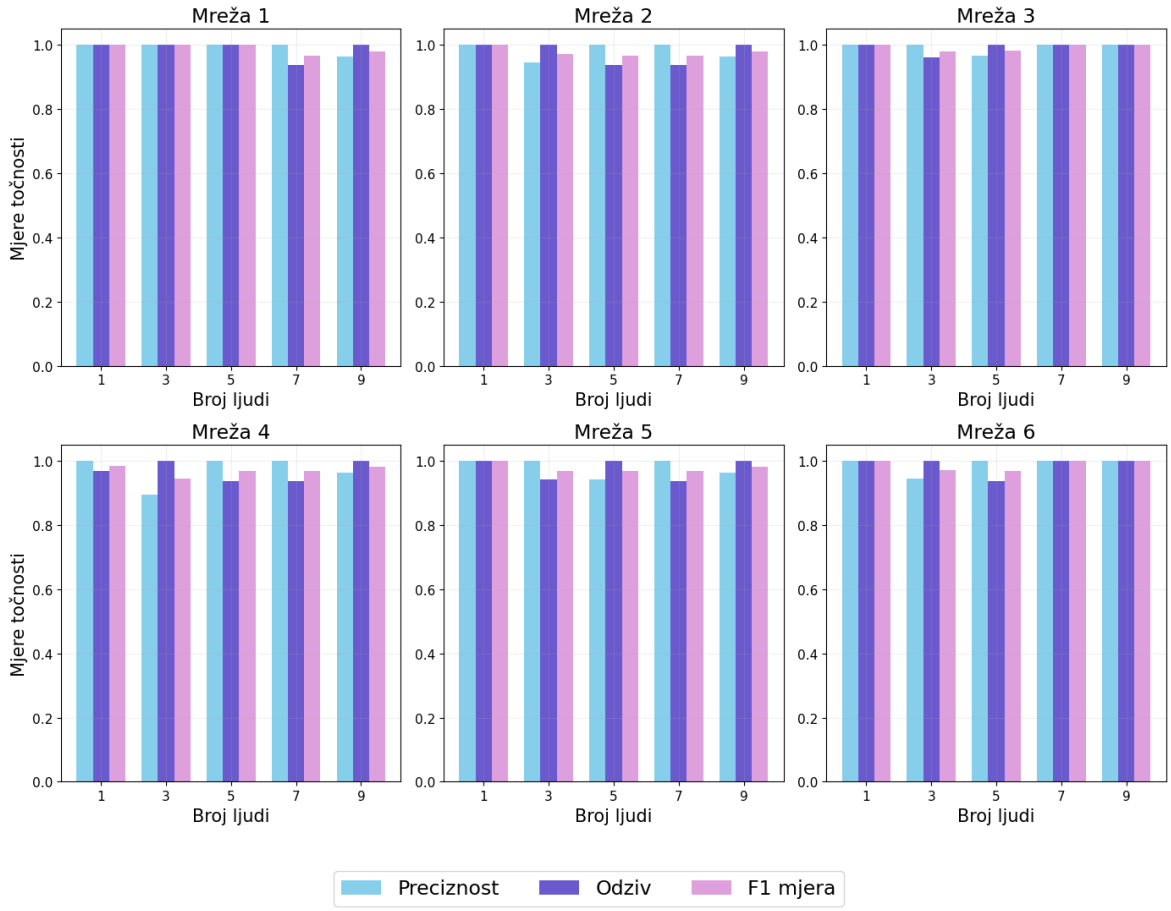
Slika B.13: Matrice zabune za optimalan odabir τ i za 7 detektora u prostoriji



Slika B.14: Točnosti po klasama za optimalan odabir τ i za 7 detektora u prostoriji



Slika B.15: Matrice zabune za optimalan odabir τ i za 9 detektora u prostoriji



Slika B.16: Točnosti po klasama za optimalan odabir τ i za 9 detektora u prostoriji

Literatura

[1] Marshed Mohamed i dr. A dynamic channel model for indoor wireless signals: working around interference caused by moving human bodies. // IEEE Antennas and Propagation Magazine 60.2 (travanj 2018.), str. 82–91. DOI: 10.1109/map.2018.2796022.

[2] Łukasz Januszkiewicz. Analysis of human body shadowing effect on wireless sensor networks operating in the 2.4 GHz band. // Sensors 18.10 (listopad 2018.), str. 3412. DOI: 10.3390/s18103412.

[3] Huo Hongwei i dr. The effect of human activites on 2.4 GHz radio propagation at home environment. // Proceedings of 2009 2nd IEEE International Conference on Broadband Network and Multimedia Technology, IEEE IC-BNMT2009 (listopad 2009.), str. 95–99. DOI: 10.1109/ICBNMT.2009.5347820.

[4] Takuya Yoshida i Yoshiaki Taniguchi. Estimating the number of people using existingWiFi access point in indoor environment. // Proceedings of the 15th International Conference on Evolutionary Computing (EC'15) [and] Proceedings of the European Conference of Computer Science (TCCS'15) (studeni 2015.).

[5] Saandeep Depatla i Yasamin Mostofi. Crowd Counting Through Walls Using WiFi. *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2018., str. 1–10. DOI: 10.1109/PERCOM.2018.8444589.

[6] Chenren Xu i dr. SCPL: Indoor device-free multi-subject counting and localization using radio signal strength. *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 2013., str. 79–90. DOI: 10.1145/2461381.2461394.

[7] Ramón F. Brena i dr. Device-Free crowd counting using Multi-Link Wi-Fi CSI descriptors in Doppler spectrum. // Electronics 10.3 (siječanj 2021.), str. 315. DOI: 10.3390/electronics10030315.

[8] Yongsen Ma, Gang Zhou i Shuangquan Wang. WiFi Sensing with Channel State Information. // ACM Computing Surveys 52.3 (lipanj 2019.), str. 1–36. DOI: 10.1145/3310194.

[9] Jiaqi Geng, Hui Dong i Fernando De La Torre. DensePose from WiFi. // arXiv (Cornell University) (prosinac 2022.). DOI: 10.48550/arxiv.2301.00250.

[10] Mingmin Zhao i dr. Through-Wall Human Pose Estimation Using Radio Signals. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018., str. 7356–7365. DOI: 10.1109/CVPR.2018.00768.

[11] Zizheng Zhang i dr. Danger-Pose detection system using commodity Wi-Fi for bathroom monitoring. // Sensors 19.4 (veljača 2019.), str. 884. DOI: 10.3390/s19040884.

[12] Denis Stijn. Using sub-GHz radio frequency communication for device-free detection, tracking and identification. Disertacija. Faculty of Applied Engineering, University of Antwerp, 2020.

[13] Ronghui Zhang i Xiaojun Jing. Device-Free Human Identification Using Behavior Signatures in WiFi Sensing. // Sensors 21.17 (rujan 2021.), str. 5921. DOI: 10.3390/s21175921.

[14] John David Jackson. Classical Electrodynamics. John Wiley & Sons, kolovoz 1998.

[15] David J. Griffiths. Introduction to Electrodynamics. Cambridge University Press, lipanj 2017.

[16] Max Born i Emil Wolf. Principles of optics. Cambridge University Press, prosinac 2019.

[17] Valeria Magoni. What is the best WiFi frequency between 2.4GHz and 5GHz?: // Towards Data Science (lipanj 2017.). URL: https://medium.com/tanaza/what-is-the-best-wifi-frequency-between-2-4ghz-and-5ghz-538620e16138 (pogledano 2. 10. 2023.).

[18] Rob Flickenger i dr. Wireless Networking in the Developing World Second edition. CreateSpace, ožujak 2005.

[19] URL: https://support.aquascapeinc.com/hc/en-us/articles/360055224992-How-to-Improve-Wi-Fi-Strength-for-Aquascape-Smart-Control-Products (pogledano 25. 9. 2023.).

[20] J. Bardwell. Converting Signal Strength Percentage to dBm Values. // WildPackets Inc. (Studeni 2002.).

[21] Zheng Yang, Zimu Zhou i Yunhao Liu. From RSSI to CSI. // ACM Computing Surveys 46.2 (studeni 2013.), str. 1–32. DOI: 10.1145/2543581.2543592.

[22] Gordon L. Stüber. Principles of mobile communication. Springer Science & Business Media, prosinac 2000.

[23] Daniel Mittleman. Bilješke s predavanja: Lecture 13 - Fresnel's Equations for Reflection and Transmission. Brown University, kolegij Optics. URL: https://www.brown.edu/research/labs/mittleman/sites/brown.edu.research.labs.mittleman/files/uploads/lecture13_0.pdf (pogledano 10. 9. 2022.).

[24] Shai Shalev-Shwartz i Shai Ben-David. Understanding machine learning. Cambridge University Press, svibanj 2014.

[25] John D. Kelleher, Brian Mac Namee i Aoife D'Arcy. Fundamentals of Machine Learning for Predictive data analytics. MIT Press, srpanj 2015.

[26] Jan Šnajder. Bilješke s predavanja: Predavanje 2 - Osnovni koncepti. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, kolegij Strojno učenje 1. URL: https://www.fer.unizg.hr/_download/repository/SU1-2022-P02-OsnovniKoncepti.pdf (pogledano 30. 11. 2022.).

[27] Bex T. Comprehensive Guide to Multiclass Classification Metrics. // Towards Data Science (lipanj 2021.). URL: https://towardsdatascience.com/comprehensive-guide-on-multiclass-classification-metrics-af94cfb83fbd (pogledano 2. 9. 2023.).

[28] Jan Šnajder. Bilješke s predavanja: Predavanje 5 - Linearni diskriminativni modeli. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, kolegij Strojno učenje 1. URL: https://www.fer.unizg.hr/_download/repository/SU1-2022-P05-LinearniDiskriminativniModeli.pdf (pogledano 30. 11. 2022.).

[29] Jan Šnajder. Bilješke s predavanja: Predavanje 4 - Regresija II. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, kolegij Strojno učenje 1. URL: https://www.fer.unizg.hr/_download/repository/SU1-2022-P04-LinearnaRegresija2.pdf (pogledano 30. 11. 2022.).

[30] Drew Wilimitis. The Kernel Trick in Support Vector Classification. // Towards Data Science (prosinac 2018.). URL: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f (pogledano 30. 8. 2023.).

[31] URL: https://duchesnay.github.io/pystatsml/optimization/optim_gradient_descent.html (pogledano 25. 7. 2023.).

[32] Kilian Weinberger. Bilješke s predavanja: Lecture 9 - SVM. Cornell University, kolegij Machine Learning for Intelligent Systems. URL: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html (pogledano 14. 7. 2023.).

[33] Jan Šnajder. Bilješke s predavanja: Predavanje 9 - Stroj potpornih vektora II. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, kolegij Strojno učenje 1. URL: https://www.fer.unizg.hr/_download/repository/SU1-2022-P09-StrojPotpornihVektora2.pdf (pogledano 30. 11. 2022.).

[34] Silvan Ferreira i dr. A Transformer-Based Contrastive Learning Approach for Few-Shot Sign Language Recognition. 2022. arXiv: 2204.02803 [cs.CV].

[35] Carolina Bento. Decision Tree Classifier explained in real-life: picking a vacation destination. // Towards Data Science (lipanj 2021.). URL: https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575 (pogledano 21. 8. 2023.).

[36] Marko Čular. Modeli slučajnih šuma i primjene. Mag. rad. Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, 2020.

[37] Pang-Ning. Tan, Michael Steinbach i Vipin Kumar. Introduction to Data Mining. Pearson, 2014.

[38] Simon Hegelich. Decision Trees and Random Forests: Machine learning techniques to classify rare events. // European policy analysis 2.1 (siječanj 2016.). DOI: 10.18278/epa.2.1.7.

[39] Scott Hartshorn. Machine learning with random forests and decision trees: a visual guide for beginners. 2016.

[40] URL: https://www.ibm.com/topics/random-forest (pogledano 11. 10. 2023.).

[41] URL: https://hr.wikipedia.org/wiki/Neuronska_mre%C5%BEa (pogledano 10. 8. 2023.).

[42] Ian Goodfellow, Yoshua Bengio i Aaron Courville. Deep learning. MIT Press, studeni 2016.

[43] Mo Daoud. Neurons, Activation Functions, Back-Propagation, Epoch, Gradient Descent: What are these?: // Towards Data Science (svibanj 2020.). URL: https://towardsdatascience.com/neurons-activation-functions-back-propagation-epoch-gradient-descent-what-are-these-c80349c6c452 (pogledano 12. 10. 2023.).

[44] Kristy Carpenter i dr. Deep learning and virtual drug screening. // Future Medicinal Chemistry 10 (listopad 2018.). DOI: 10.4155/fmc-2018-0314.

[45] Harsh Yadav. Dropout in Neural Networks. // Towards Data Science (srpanj 2022.). URL: https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9 (pogledano 10. 10. 2023.).

[46] Bilal Esmael i dr. A Statistical Feature-Based Approach for Operations Recognition in Drilling Time Series. // International Journal of Computer Information Systems and Industrial Management Applications. 5 (prosinac 2013.), str. 454–461.

[47] Severin Ionut. Time Series Feature Extraction For Head Gesture Recognition: Considerations Toward HCI Applications. (Listopad 2020.), str. 232–237. DOI: 10.1109/ICSTCC50638.2020.9259741.

[48] Akbar Dehghani i dr. A quantitative comparison of Overlapping and Non-Overlapping sliding Windows for human activity recognition using inertial sensors. // Sensors 19.22 (studeni 2019.), str. 5026. DOI: 10.3390/s19225026.

[49] íScar D. Lara i dr. Centinela: A human activity recognition system based on acceleration and vital sign data. // Pervasive and Mobile Computing 8.5 (listopad 2012.), str. 717–729. DOI: 10.1016/j.pmcj.2011.06.004.

[50] Shilpy Sharma. Identification of Overlapping Features in Time Series Data. Disertacija. The University of Guelph, kolovoz 2015.

[51] URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skew.html (pogledano 20. 8. 2023.).

[52] URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kurtosis.html (pogledano 20. 8. 2023.).

[53] Mirna Patricia Ponce-Flores i dr. Time series complexities and their relationship to forecasting performance. // Entropy 22.1 (siječanj 2020.), str. 89. DOI: 10.3390/e22010089.

[54] URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html (pogledano 20. 8. 2023.).

[55] Ning Jia. Spectral Entropy — An Underestimated Time Series Feature. // Towards Data Science (prosinac 2022.). URL: https://medium.com/towards-data-science/spectral-entropy-an-underestimated-time-series-feature-94e18ae5b958 (pogledano 20. 8. 2023.).

[56] Jan Šnajder. Bilješke s predavanja: Predavanje 10 - Jezgrene metode. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, kolegij Strojno učenje 1. URL: https://www.fer.unizg.hr/_download/repository/SU1-2022-P10-JezgreneMetode.pdf (pogledano 30. 11. 2022.).

[57] URL: https://scikit-learn.org/stable/modules/svm.html.

[58] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization. (Prosinac 2014.). DOI: 10.48550/arxiv.1412.6980.