Mjerenje i analiza korelacija WiFi signala i broja ljudi pomoću strojnog učenja

Kumiša, Karlo

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Karlo Kumiša

MJERENJE I ANALIZA KORELACIJA Wi-Fi SIGNALA I BROJA LJUDI POMOĆU STROJNOG UČENJA

Diplomski rad

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER NASTAVNIČKI

Karlo Kumiša

Diplomski rad

Mjerenje i analiza korelacija Wi-Fi signala i broja ljudi pomoću strojnog učenja

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

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2022.

Zahvaljujem se svojem mentoru Hrvoju Buljanu na savjetima i pomoći pri istraživanju i pisanju ovog rada.
Zahvaljujem se profesoru Dariju Jukiću na savjetima oko korištenja algoritama.
Također veliko hvala profesoru Silviju Domazetu na provedenoj numeričkoj simulaciji kao i kolegi Vinku Kličeku na provedenom eksperimentu.
Zahvaljujem se svim kolegama i prijateljima koji su olakšali i uljepšali ovo iskusto, a najveća hvala ide mojoj obitelji, a pogotovo mami, tati, baki i djevojci na strpljenju i pruženoj podršci.

Sažetak

U ovom radu istražuje se mogućnost predviđanja broja ljudi u danom prostoru kroz jakost Wi-Fi polja u određenim točkama prostora. Predviđanja ostvaruju algoritmi strojnog učenja iz podataka generiranih kroz simulaciju i eksperiment. Iako rezultati na eksperimentalnim podacima nisu zadovoljavajući, istraživanje na podacima iz simulacije daje obećavajuće rezultate s obzirom na to da je najveća točnost predviđanja oko 69%.

Ključne riječi: Wi-Fi, nadzirano učenje, strojno učenje, senzori, stablo odlučivanja, k najbližih susjeda, neuralna mreža, konvolucijska neuralna mreža

Diploma thesis title

Abstract

This paper investigates the possibility of predicting the number of people in a given space through the strength of Wi-Fi signal in certain points of space. Predictions are made by machine learning algorithms from data generated through simulation and experiment. Although the results of the experiment data are not satisfactory, research on simulation data give promising results considering that the highest prediction accuracy is around 69%.

Keywords: Wi-Fi, supervised learning, machine learning, sensors, decision tree, k nearest neighbors, neural network, convolutional neural network

Sadržaj

1 Uvod

Senzori imaju jako veliku ulogu u životu čovjeka. Automobil današnjeg vremena ima u sebi više od 50 različitih senzora koji omogućuju takvom stroju interakciju s okolinom i samostalno motrenje ispravnosti vlastitih dijelova i procesa [1]. Nedavnim povećanjem uplita tehnologije u ljudske živote potreba za naprednijim, pouzdanijim i novim senzorima postaje sve veća. Jedan od problema koji zahtjeva unaprjeđenje starih ili razvoj novih senzora je unutarnja lokalizacija i detekcija pokreta ljudi.

Vanjska lokalizacija pomoću GPS-a daje dobre rezultate, ali unutarnja lokalizacija ostaje problem kojeg valja istražiti. Nije teško razumjeti zašto bi informacija o lokaciji čovjeka u zatvorenim prostorima bila od koristi. Očiti primjer su javne ustanove kao bolnice i škole u kojima se informacija o lokaciji i pokretima pojedinih pacijenta ili učenika mogu koristiti za sprječavanje niza nemilih događaja. Za sada, u rješavanju navedenog problema, se uglavnom koriste kamere koje daju dobre rezultate, ali sa sobom nose nedostatke koji potiču razmatranje alternativa. Neki od tih nedostataka su potreba za izvorom svjetla, narušavanje privatnosti čovjeka te ograničenje područja promatranja objektima u prostoru. Neke od predloženih alternativa koje se nisu pokazale praktičnima su invazivne metode, koje zahtijevaju da čovjek uz sebe konstantno nosi uređaj, ili ne invazivni ultrazvučni i lidar senzori koji opet imaju problem što objekti u prostoru ograničuju područje promatranja, ali također zahtijevaju postavljanje infrastrukture koja ovu alternativu čini skupom. Nedavna istraživanja pokazuju da je iz WiFI signala moguće izvući informacije o prisustvu, lokaciji pa čak i pokretima ljudi [3] [2]. Takva alternativa bi riješila navedene probleme s obzirom na to da ne narušava privatnost ljudi, Wi-Fi signal jednog rutera se može detektirati i kroz više zidova te je cjenovno prihvatljiva s obzirom na široku rasprostranjenost Wi-Fi uređaja. Cilj ovog rada je istražiti mogućnost detektiranja broja ljudi u prostoriji iz danih vrijednosti Wi-Fi signala pomoću algoritama za strojno učenje. Algoritmi su testirani na podacima generiranima u simulaciji i eksperimentu, te će više riječi o tome biti u 3. poglavlju.

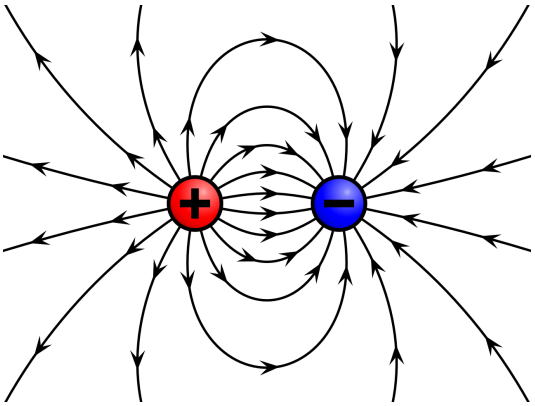
2 Primjena Wi-Fi signala na detekciju ljudi

2.1 Elektromagnetski valovi

Još u doba antičke Grčke ljudi su primijetili da nakon trljanja dva komadića jantara vunom, komadići se međusobno odbijaju. Danas je poznato da trljanje jantara vunom čini jantar električki nabijenim. Kada se neko tijelo okarakterizira kao nabijeno, to podrazumijeva da tijelo ima više elektrona nego protona (negativno nabijeno tijelo) ili više protona nego elektrona (pozitivno nabijeno tijelo). Eksperimentom je ustanovljeno da se istoimeni naboji odbijaju, a raznoimeni privlače. Da bi se dva tijela mogla odbijati ili privlačiti moraju jedan na drugoga djelovati silom. U ovom slučaju ta sila se naziva električna sila. Kvantifikaciju količine električne sile opisao je Charles Augustin de Coulomb (takozvanim) Coulombovim zakonom (jednadžba 2.1).

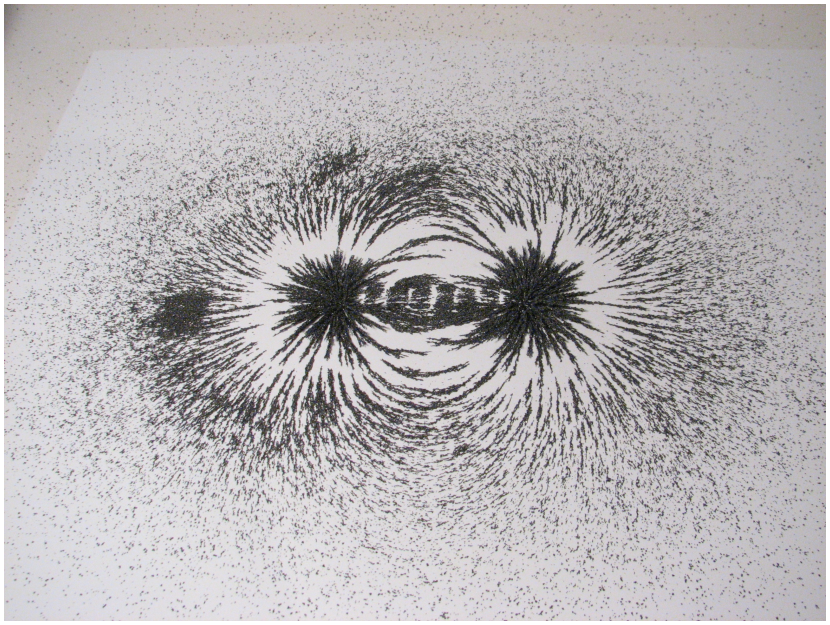
$$F_{el} = \frac{1}{4\pi\epsilon}\frac{q_1 q_2}{r^2} \tag{2.1}$$

Kao i kod Newtonovog zakona gravitacije Coulombov zakon uključuje problem djelovanja sile na daljinu. Taj problem je riješio britanski fizičar Michael Faraday uvodeći pojam polja. Polje je veličina koje ima vrijednost u svakoj točki prostora. Tijelo koje je nabijeno posljedično mijenja svojstva prostora oko sebe, odnosno stvara električno polje. Tako se rješava problem djelovanja sile na daljinu. Nabijeno tijelo ne međudjeluje s drugim nabijenim tijelom direktno nego s električnim poljem koje ono stvara u prostoru. Dakle, pozitivan naboj je izvor električnog polja, a negativan ponor. Skica silnica električnog polja, pozitivnog i negativnog naboja je prikazana na slici (2.1).



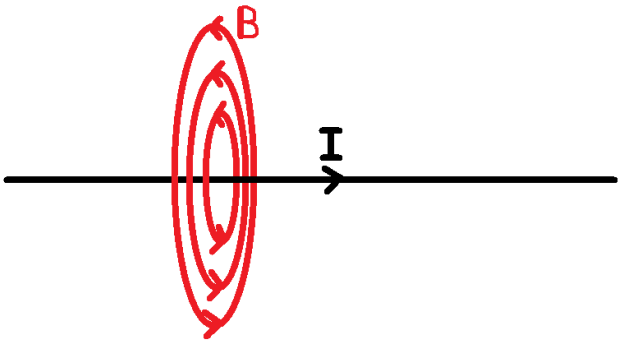
Slika 2.1: Prikaz silnica električnog polja. (slika preuzeta s [15])

Jedan od boljih načina za prikazivanje postojanja polja je pomoću magneta i željezne prašine. Takav eksperiment je prikazan na slici (2.2).



Slika 2.2: Eksperimentalni prikaz silnica magnetskog polja. (slika preuzeta s [16])

U tom slučaju prikazane silnice nisu silnice električnog polja, nego magnetskog. Ljudi su se s pojavama magnetizma upoznali također u doba antičke Grčke primijetivši odbijanje između dva komada željezne rude. Željeznu rudu koja je pokazivala takva svojstva ljudi su nazvali magnet, prema gradu blizu kojega je pronađena (Magnesia). Primijećeno je da ako se magnetski štap pusti da slobodno visi uvijek će se jedan te isti kraj okrenuti prema sjeveru. Tako je magnet podijeljen na sjeverni i južni pol. Također je primijećeno da se istoimeni polovi magneta odbijaju dok se raznoimeni privlače. Takvo međudjelovanje moguće je objasniti kao i kod slučaja električnog naboja, magnetskim poljem. Budući da nije pronađen niti jedan magnet koji bi imao samo jedan pol, silnice magnetskog polja su, za razliku od silnica električnog polja, zatvorene krivulje. To da naboj u gibanju stvara magnetsko polje prvi je primijetio u 19. stoljeću danski znanstvenik Hans Christian Oersted primijetivši da se igla kompasa pomiče ako se nalazi u prisustvu žice kojom teče konstantna struja. Dakle žica kojom teče struja oko sebe stvara magnetsko polje. Silnice tog magnetskog polja prikazane su na slici 2.3.



Slika 2.3: Skica silnica magnetskog polja oko vodiča.

U oba slučaja električno i magnetsko polje su konstantni u vremenu te ih je moguće promatrati neovisno jedno o drugom, ali u slučaju kada se elektično i magnetsko polje mijenjaju u vremenu to više nije slučaj. Faradayev zakon (jednadžba 2.2) govori da promjenjivo magnetsko polje inducira električno polje,

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.2}$$

dok Ampèreov zakon (jednadžba 2.3) govori da promjenjivo električno polje inducira magnetsko polje.

$$\vec{\nabla} \times \vec{B} = \mu_0 \left(\vec{j} + \epsilon_0 \frac{\partial \vec{E}}{\partial t} \right) \tag{2.3}$$

Tako se stvara niz promjenjivih električnih i magnetskih polja koji se šire kroz prostor bez potrebe za medijem. Takav podražaj naziva se elektromagnetski val.

2.2 Interakcija elektromagnetskih valova s materijom

Jednadžbe 2.4, 2.5, 2.6 i 2.7 su Maxwellove jednadžbe za prostor u kojem nema slobodnih naboja niti struja.

$$\vec{\nabla} \cdot \vec{E} = 0 \tag{2.4}$$

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{2.5}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.6}$$

$$\vec{\nabla} \times \vec{B} = \mu_0 \epsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{2.7}$$

Jednostavnom manipulacijom tih jednadžbi dolazi se do izraza znanog kao valna jednadžba.

$$\nabla^2 \vec{E} = \mu_0 \epsilon_0 \frac{\partial^2 \vec{E}}{\partial t^2} \tag{2.8}$$

$$\nabla^2 \vec{B} = \mu_0 \epsilon_0 \frac{\partial^2 \vec{B}}{\partial t^2} \tag{2.9}$$

Iz jednadžbi 2.8 i 2.9 moguće je dobiti informaciju o brzini širenja tih valova.

$$v = \frac{1}{\sqrt{\epsilon_0 \mu_0}} = c \tag{2.10}$$

Proizlazi da je brzina širenja elektromagnetskih valova u vakuumu jednaka brzini svjetlosti. Situacija se mijenja ako elektromagnetski val iz vakuuma dođe u neki dielektrični materijal. U takvom slučaju Maxwellove jednadžbe mijenjaju se na slijedeći način.

$$\vec{\nabla} \cdot \vec{E} = 0 \tag{2.11}$$

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{2.12}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.13}$$

$$\vec{\nabla} \times \vec{B} = \mu \epsilon \frac{\partial \vec{E}}{\partial t} \tag{2.14}$$

Gdje su $\mu = \mu_0 \mu_r$ i $\epsilon = \epsilon_0 \epsilon_r$. Očito je da će ta promjena utjecati na brzinu širenja vala:

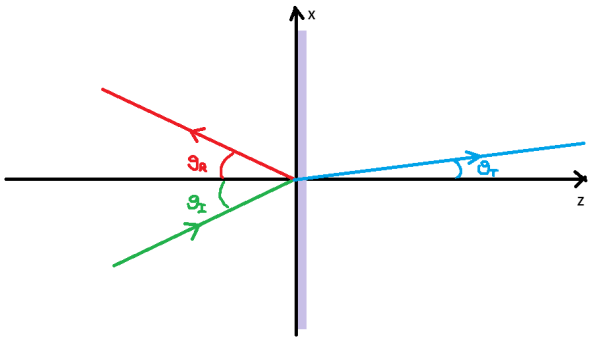
$$v = \frac{1}{\sqrt{\epsilon \mu}}. \tag{2.15}$$

Odnosno brzina se može pisati preko indeksa loma n kao:

$$v = \frac{c}{n}. \tag{2.16}$$

Indeks loma, za gotovo sve materijale, ima vrijednost veću od 1 (postoje umjetno stvoreni materijali s negativnim indeksom loma no takvi materijali u ovom radu neće biti razmatrani). Odnosno u materijalu brzina propagacije vala bit će manja od brzine propagacije vala u vakuumu.

Situacija u kojoj elektromagnetski val upada na granicu između dva sredstva je prikazana na slici 2.4. Elektromagnetski val se u takvoj situaciji može transmitirati, reflektirati ili djelomično i transmitirati i reflektirati. Potpuno transmitirani val će se otkloniti poštujući Snellov zakon (2.17), a reflektirani val će se odbiti na granici sredstva pod istim kutom pod kojim je došao upadni val.



Slika 2.4: Upad elektromagnetskog vala na granicu dva sredstva.

$$n_1 sin\theta_1 = n_2 sin\theta_2 \tag{2.17}$$

Ako se ljudsko tijelo (s obzirom na to da 60% ljudskog tijela čini voda [13]) modelira kao vodeni cilindar (indeks loma n=1,330) i stavi na put elektromagnetskom valu koji propagira kroz zrak (indeks loma n≈1), zbog refleksije i transmisije elektromagnetskog vala na granici dva medija, mijenjat će se vrijednosti jakosti elektromagnetskog polja u određenim točkama prostora. Takvo ponašanje podržava pretpostavku da se detekcija čovjeka može odrađivati pomoću elektromagnetskih valova.

2.3 Wi-Fi

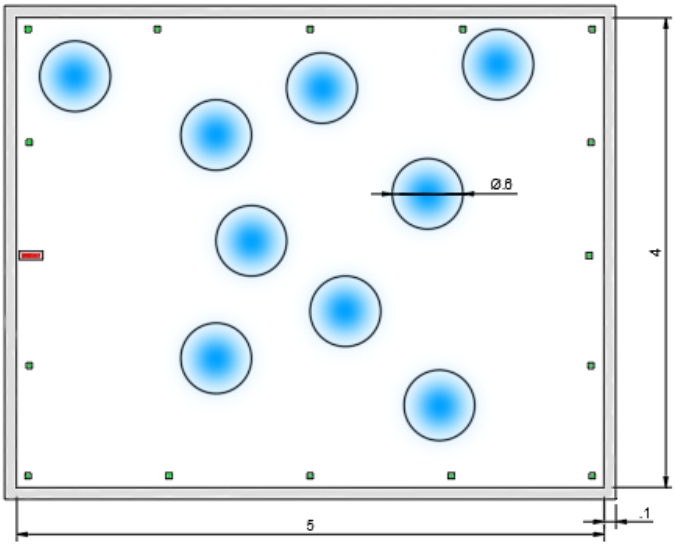
Bežična komunikacija nije novi pojam. Ljudi su koristili bežične načine komunikacije još i prije industrijskog doba. Pa su tako prethodnim dogovorom o značenju različitih kombinacija signala, preko dimnih ili svjetlosnih kombinacija mogli slati složene poruke na udaljenostima veličine dosega ljudskog vida [4]. Daljnji napredak prijenosa informacija povećanjem brzine slanja i udaljenosti dolazi izumom telegrafa 1835.(Samuel F.B. Morse) i telefona 1876. (Alexander Graham Bell) koji su svjetlost i dim zamijenili električnom strujom [5] [6]. Na prijelazu 19. u 20. stoljeće Nikola Tesla primjećuje da uz pomoć svog novog izuma, Tesline zavojnice, može slati i primati radio signal, Nedugo nakon toga je Marconi napravio prvi radio telegraf [8]. Na taj način otvorena je mogućnost bežične komunikacije. Prva radio komunikacija pomoću informacije slane u obliku paketa osmišljena je i realizirana na Hawaiskom sveučilištu 1971. pod nazivom ALOHAnet [7]. Iako je takva tehnologija bila revolucionarna, komercijalno nikad nije zaživjela zbog žičane Ethernet tehnologije koja je omogućavala puno veće brzine prijenosa podataka dostupne od 80-ih godina prošlog stoljeća. Do promjene dolazi 1985. godine kada je Američki FFC odobrio slobodnu uporabu tri frekvencijska pojasa za komunikaciju. Na frekvencijama od 900 MHz, 2.4 GHz i 5.8 GHz. To je zaokupilo pažnju pružatelja telekomunikacijskih usluga te je počeo intenzivan rad na rješavanju problema bežične komunikacije. 1990-ih i dalje su sva računala bila povezana Ethernet kabelom. Naime, najveći problem je bio to što elektromagnetski valovi na granici između dva sredstva mogu doživjeti refleksiju, refrakciju ili apsorpciju te se na prijemniku dobiva superponirani signal pun smetnji. Način na koji je taj problem rješen je slanjem informacijskog paketa manjim brzinama kroz više podnosioca OFDM (orthogonal frequency-division multiplexing) metodom. Na zahtjev pružatelja telekomunikacijskih usluga stvoren je odbor za standarde bežične mreže ”802.11” danas poznatiji pod nazivom Wi-fi [7]. Brzina prijenosa informacija i stabilnost komunikacijske veze su iznimno važni elementi bežične komunikacije te zahtjev za konstantnim unapređenjem istih vodi na stvaranje novih Wi-fi generacija (802.11./b/a/g/n/ac/ax). Za kvantifikaciju stanja signala koriste se dvije vrijednosti koje nose naziv RSSI (Received Signal Strength) i CSI (Channel State Information). Indikator snage signala ili RSSI (engl. Received Signal Strength) je relativna vrijednost koja opisuje snagu signala. Veća RSSI vrijednost označava veću snagu. Vrijednost je relativna jer ne postoji definirani odnos između RSSI vrijednosti

i snage u Watt-ima pa tako proizvođač Wi-Fi kartice sam određuje raspon vrijednosti te osjetljivost iste. Na RSSI vrijednost utječe mnogo faktora gdje su neki od njih vrsta WLAN kartice, udaljenost odašiljača i primatelja, okolina te prisutnost čovjeka [9]. S obzirom na to da prisutnost čovjeka i njegovo kretanje utječu na RSSI vrijednost moguće je iz iste izvući podatke o navedenim radnjama. Kako signal od odašiljača do primatelja dolazi kroz više puteva konačni signal je superpozicija svih signala. Za razliku od RSSI vrijednosti koja prikazuje samo snagu signala, CSI (engl. Channel State Information) prikazuje vrijednost amplitude i faze svakog podnosioca te tako reflektira utjecaj propagacije signala kroz više puteva [10]. Popularnost i napredak Wi-Fi tehnologije omogućava jednostavno dohvaćanje CSI vrijednosti pomoću Wi-fi prijamnika koji prate 802.11.a/g/n/ac/ax standarde.

3 Mjerenja i numeričke simulacije

3.1 Simulacija

Simulacija je provedena u programu Comsol. U programu je simulirano ponašanje elektromagnetskog polja u pravokutnoj prostoriji s obzirom na broj i poziciju ljudi. Simulacija je rađena za 1, 3, 5 ,7 i 9 ljudi te su njihove pozicije mijenjane nasumično. Soba je dimenzija $4\,\mathrm{m} \times 5\,\mathrm{m}$ te se proteže kroz dvije dimenzije. Zidovi sobe su od betona debljine $10\,\mathrm{cm}$, a ljude predstavljaju vodeni cilindri radijusa $30\,\mathrm{cm}$. Izvor elektromagnetskog polja je beskonačno duga žica (koja se proteže okomito na ravninu sobe) kojom teče izmjenična struja sinusoidalnog oblika. U prostoriji se nalazi 15 detektora koji zapisuju kompleksnu vrijednost elektromagnetskog polja. Pozicija detektora (označeni zelenom bojom) i odašiljača (označen crvenom bojom) je prikazana na slici 3.1.



Slika 3.1: Prikaz simulacijskog postava.

3.2 *Baza podataka za simulaciju*

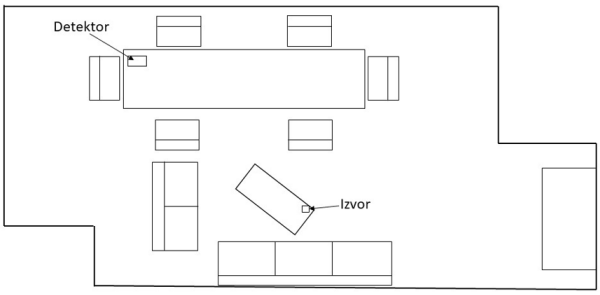
Snimljene vrijednosti nalaze se u 20 ”.csv” datoteka. Kao što je navedeno, simulacija je provedena za pet različitih brojeva ljudi te svakom broju pripadaju 4 seta mjerenja. Svaki set mjerenja sadrži 10 primjera te se tako dolazi do baze podataka od 200 primjera. Iako je mjerenje provedeno samo s petnaest senzora svaki senzor mjeri 2 vrijednosti (realnu i kompleksnu). S obzirom na to da su obje vrijednosti iskorištene kao zasebne značajke, svaki primjer se sastoji od 30 značajki. Tako će matrica koja predstavlja bazu podataka imati 40 stupaca i 200 redaka. Struktura spomenute matrice je prikazana na slici 3.2

$$\begin{bmatrix} x^1_{1\_real} & x^1_{2\_real} & \cdots & x^1_{15\_real} & x^1_{1\_imag} & x^1_{2\_imag} & \cdots & x^1_{15\_imag} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x^{200}_{1\_real} & x^{200}_{2\_real} & \cdots & x^{200}_{15\_real} & x^{200}_{1\_imag} & x^{200}_{2\_imag} & \cdots & x^{200}_{15\_imag} \end{bmatrix}$$

Slika 3.2: Prikaz strukture matrice podataka za mjerenja provedena simulacijom.

3.3 Eksperiment

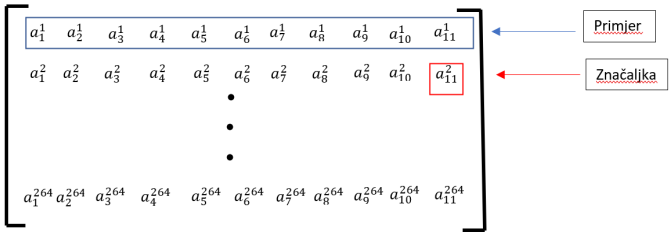
Eksperiment je proveden na sličan način kao i simulacija u smislu mjerenja elektromagnetskog polja u sobi koju ispunjavaju ljudi na različitim pozicijama. Uz odašiljač Wi-Fi signala (mobitel) postoji samo jedan detektor (ESP32) koji mjeri RSSI vrijednost. Signal je mjeren s 1, 2, 3, 4, 5, 6, 7 i 8 ljudi u prostoriji te se njihova lokacija u jednim mjerenjima mijenja u vremenu, a u drugima je konstantna. Soba u kojoj je mjerenje provedeno ima kompliciraniji oblik prikazan na slici 3.3.



Slika 3.3: Prikaz eksperimentalnog postava.

3.4 Baza podataka za eksperiment

Mjerene vrijednosti se nalaze u ".csv" datoteci u obliku dva stupca s 1130 redova. Prvi stupac predstavlja vrijeme mjerenja, a drugi stupac predstavlja vrijednost jakosti Wi-Fi signala (RSSI) u danom trenutku. Mjerenja su napravljena u rasponu od jedne do osam osoba te za svaki broj osoba postoje tri spomenute datoteke. Za treniranje i testiranje modela od svake datoteke je dobiveno 11 primjera s po 100 značajki, tako da je uzeto prvih 1100 vrijednosti jakosti signala i odbačeno zadnjih 30. Tako je dobiveno ukupno 264 primjera. Skica te matrice podataka je prikazana na sljedećoj slici.



Slika 3.4: Prikaz strukture matrice podataka za eksperimentalna mjerenja.

4 Strojno učenje

Budući da je u navedenim podacima jako teško pronaći neku pravilnost, za klasifikaciju novih primjera razmatrat će se korištenje algoritama strojnog učenja.

4.1 Općenito o strojnom učenju

Strojno učenje je grana računalne znanosti, ali čime se ta znanost zapravo bavi najbolje je objasniti definicijama koje su dali informatičari Arthur Samuel (1959.) i Tom Michel (1999.). Samuelova definicija glasi: ”Strojno učenje je grana proučavanja koja daje računalima mogućnost učenja bez eksplicitne naredbe”, dok malo detaljnija Michelova definicija kaže: ”Računalni program uči iz iskustva E s obzirom na neki zadatak T i neku mjeru izvedbe P, ako se njegova izvedba na T, mjerena P, poboljšava s iskustvom E [11]. Ljudi su bili upoznati s idejom strojnog učenja još od sredine 20. stoljeća, ali njezina popularnost je narasla tek početkom 21. stoljeća. Razlog tome je nagli rast količine podataka dostupne zbog interneta kao i razvoj računalne moći [12]. U današnje doba svakodnevno se susrećemo s tehnologijom koja u pozadini koristi strojno učenje. Na primjer personalizirane reklame, Google tražilica ili mobitel koji prepoznaje lica. Algoritme strojnog učenja, prema načinu učenja, moguće je podijeliti u 3 kategorije. To su nadzirano učenje, ne nadzirano učenje i podržano učenje.

4.1.1 Nadzirano strojno učenje

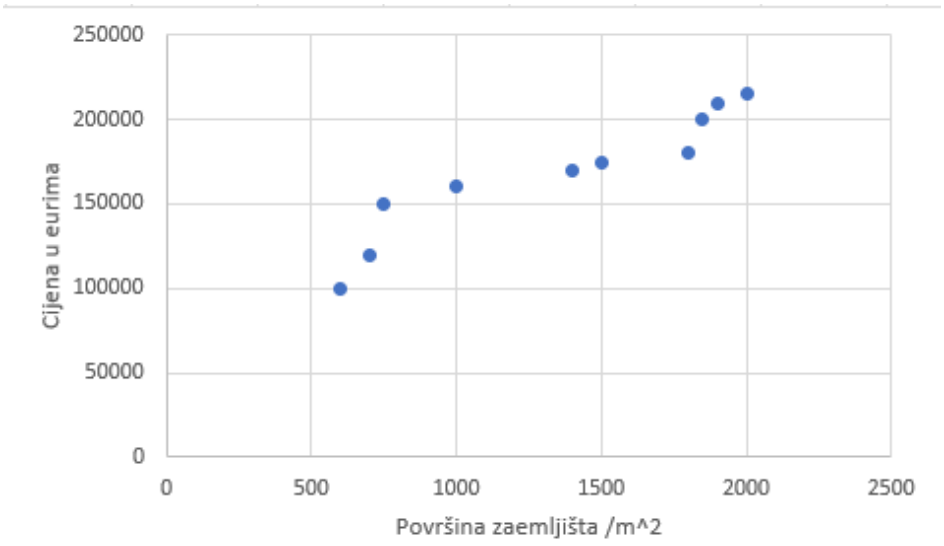
Za ovakav način učenja algoritmu uz odgovarajuće podatke se mora pridružiti i oznaka. Problemi na koje bi se primijenio takav algoritam su prikazani na sljedećim primjerima.

Potrebno je odrediti kolika je cijena zemljišta veličine 2250 m^2 na temelju podataka o prodaji ostalih zemljišta prikazanih u tablici 4.1.

Površina zemljišta / m^2	Cijena u eurima
600	100 000
700	120 000
750	150 000
1000	160 000
1400	170 000
1500	175 000
1800	180 000
1850	200 000
1900	210 000
2000	215 000

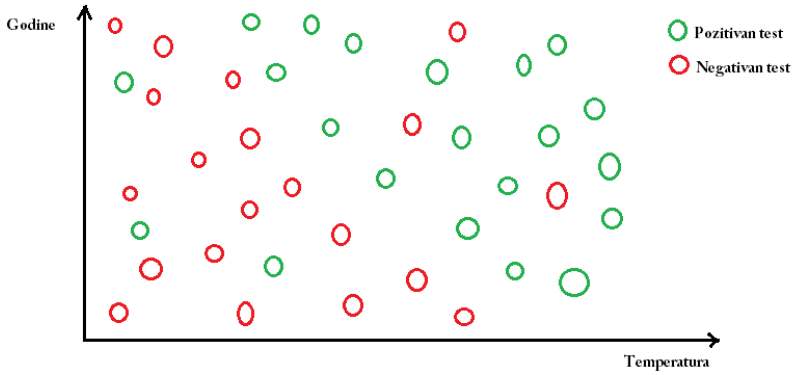
Tablica 4.1: Ovisnost cijene o površini zemljišta.

Na slici 4.1 prikazana je ovisnost danih veličina grafički. Ono što bi algoritam mogao napraviti je ”povući” pravac kroz točke tako da je zbroj apsolutne vrijednosti udaljenosti svih točaka od pravca minimalna. Tako je algoritam pomogao naći linearnu funkciju koja opisuje raspodjelu prikazanih točaka, te predviđanje cijene nekog zemljišta se svodi na uvrštavanje argumenta u funkciju. Ovo je jednostavni primjer linearne regresije.



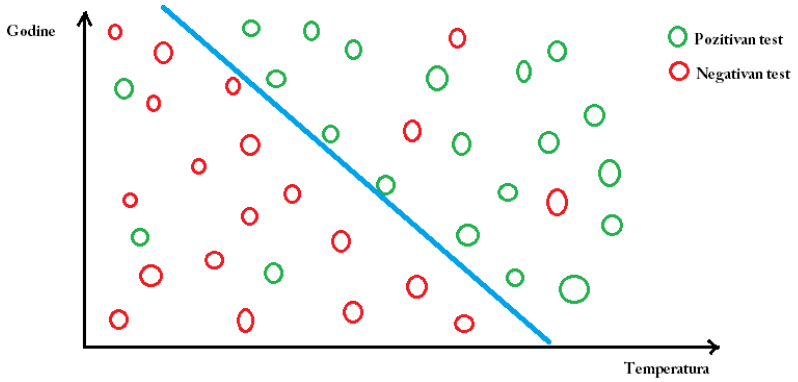
Slika 4.1: Grafički prikaz ovisnosti površine o cijeni.

Sljedeći primjer zahtjeva predviđanje moguće zaraze virusom, s obzirom na starost i temperaturu pacijenta, na temelju podataka prikazanih na slici 4.2.



Slika 4.2: Grafički prikaz pozitivnih i negativnih rezultata testova u ovisnosti o starosti i tjelesnoj temperaturi pacijenata.

Algoritam, kao i u prošlom primjeru, može naći parametre linearne funkcije te je mogući ishod takvog algoritma linearna funkcija prikazana na slici 4.3.



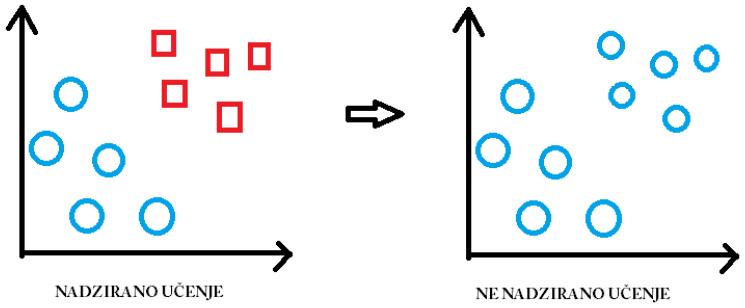
Slika 4.3: Prikaz izleda moguće linearne funkcije.

Tako funkcija predstavlja granicu između dvije klase te prema poziciji točke na grafu, koju određuju značajke, u odnosu na granicu moguće je dobiti predviđanje. Ovo je jednostavan primjer klasifikacijskog algoritma.

U oba problema algoritam je tražio linearnu funkciju, ali to ne mora biti tako. Možda se raspored točaka može bolje opisati kvadratnom funkcijom ili nekim polinomom, ali jedna stvar u oba problema je ista i neće se mijenjati. Algoritmu je za učenje dan skup podataka koji je imao značajke s pripadnom oznakom. Zbog toga oba primjera stavljaju u kategoriju nadziranog učenja.

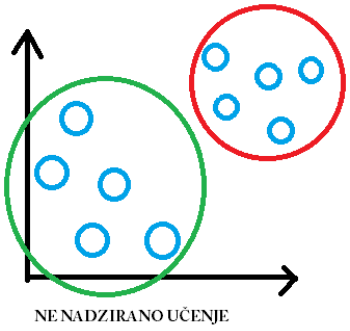
4.1.2 Nenadzirano strojno učenje

Za razliku od nadziranog učenja, kod ne nadziranog učenja podaci koji se predaju algoritmu nemaju nikakvu oznaku. Ta razlika je ilustrirana na slici 4.4.



Slika 4.4: Prikaz razlike između nadziranog i ne nadziranog učenja.

Ne nadzirani tip učenja koristi se kada problem zahtjeva da računalo u skupu podataka samo pronađe strukturu. Na prethodnoj skici bi algoritam mogao podijeliti dane podatke u dvije različite grupe kao što je prikazano na slici 4.5.



Slika 4.5: Prikaz grupiranja ne označenih podataka.

Ovakav tip algoritma za ne nadzirano učenje se naziva algoritam grupiranja i izuzetno je popularan. Može se primijetiti na streaming platformama gdje bi se koristio za grupiranje ljudi sa sličnim ukusom za filmove radi poboljšanja preporuka ili na internetskim tražilicama gdje se koristi za grupiranje stranica sličnog sadržaja. Ovo

je primjer samo jednog tipa problema koji se može rješavati ne nadziranim učenjem, a kao dodatni primjeri mogu se navesti detekcija anomalija ili obrađivanje zvučnih zapisa.

4.1.3 Podržano strojno učenje

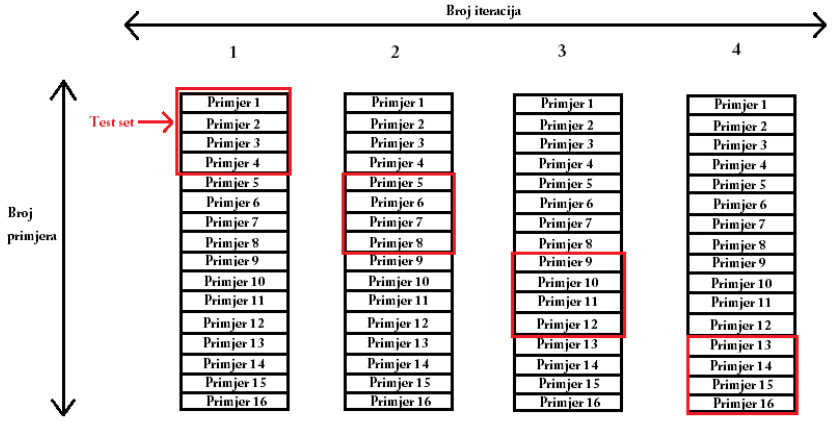
Ovakav način učenja podrazumijeva učenje metodom pokušaja i pogreške u realnom vremenu preko povratne veze iz okoline u kojoj se sustav nalazi. Algoritam dobiva nagradu ili kaznu ovisno o svom postupku, a cilj algoritma je skupiti što više nagrada. Ovakvi algoritmi se uglavnom koriste u računalnim igrama i robotima.

4.2 Korišteni algoritmi

Svi algoritmi korišteni u ovom radu su višeklasni klasifikacijski algoritmi za nadzirano strojno učenje.

4.2.1 Cross Validation

Da bi se mogla saznati moć predviđanja određenog modela potrebno je testirati model na test primjerima. Test primjeri su oni koji nisu sudjelovali u izgradnji trenutnog modela. Zbog toga se mora skup podataka podijeliti na trening dio i test dio. S obzirom na to da će model biti bolji što više ima primjera za treniranje, smanjenje trening primjera se nastoji zaobići. Način na koji se to riješilo je pomoću algoritma za unakrsnu provjeru valjanosti (engl. cross validation), koji je dio scikit learn biblioteke.

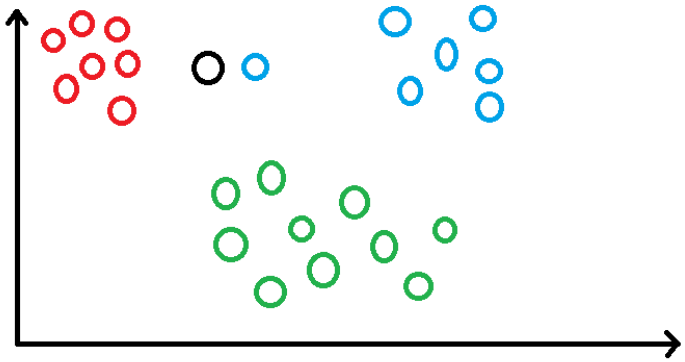


Slika 4.6: Prikaz rada algoritma za unakrsnu provjeru valjanosti.

Ideja algoritma je prikazana na slici 4.6. Algoritam kroz više iteracija uzima različite test setove te se uzima prosjek točnosti svih iteracija. Točnost modela se dobiva kao postotak točnih predviđanja na test setu. Tu varijablu vraća funkcija koja je već prije napisana u biblioteci određenog algoritma (score, evaluate,...).

4.2.2 K najbližih susjeda

Ideja iza ovog algoritma jest da novi primjer klasificira na temelju blizine klasificiranih trening primjera u prostoru značajka. Varijabla k u nazivu označava broj susjeda koje će algoritam uzeti u obzir. Ako je vrijednost k=3 algoritam će danom primjeru pridijeliti klasifikaciju koju imaju njemu 3 najbliža primjera.

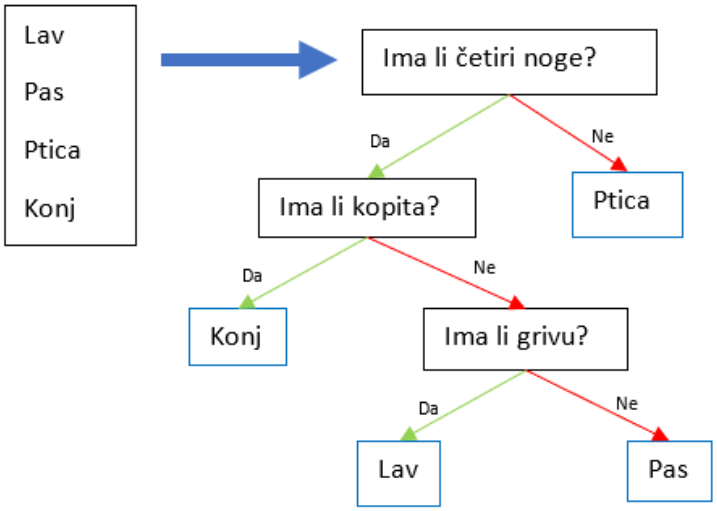


Slika 4.7: Prikaz rada algoritma K najbližih susjeda.

Skica jednog problema je prikazana na slici 4.7. Crni krug predstavlja novi primjer kojeg treba klasificirati, a krugovi u boji su već klasificirani trening podaci. Algoritam mora odrediti koje je boje (kojoj klasi pripada) crni krug. Ako je k=1 algoritam će klasificirati crni krug kao plavi, ako je k=3 algoritam će klasificirati crni krug kao crveni, a ako je k=8 crni krug će biti klasificiran kao zeleni. Ne postoji način na koji se u naprijed može odrediti vrijednost k stoga je potrebno isprobati par različitih vrijednosti i odabrati onu koja daje najbolje rezultate. Za implementaciju ovog koda u Pythonu koristila se scikit learn biblioteka. Implementacija algoritma u Pythonu se može naći u dodatku C.

4.2.3 Stablo odlučivanja

Algoritam podliježe primjer nizu ispita kako bi ga klasificirao. Takav model može se grafički prikazati kao niz čvorova u kojima se primjer ispituje i grana koje predstavljaju odgovor na postavljeno pitanje. Zbog takve strukture algoritam nosi ime stablo odlučivanja. Princip rada algoritma je ilustriran na slici 4.8.



Slika 4.8: Prikaz principa rada algoritma Stablo odlučivanja.

Kod stvaranja modela glavni problem je odabir značajki koje idu u čvorove. Rješenje tog problema je realizirano kroz proces smanjenja nečistoće. Nečistoća se kvantitativno može opisati različitim pojmovima (npr. entropija, informacijski gain, gini omjer, ...), ali za ovo objašnjenje koristit će se gini indeks. Tablica 4.2 pokazuje skup podataka koji prikazuju karakteristike auta koji se prodaju u roku od mjesec dana.

Prvi vlasnik	Klima	Starost	Prodaja
Da	Da	5	Ne
Da	Ne	8	Ne
Ne	Da	10	Da
Ne	Da	12	Da
Da	Da	13	Da
Da	Ne	15	Ne
Ne	Ne	18	Ne

Tablica 4.2: Karakteristike vozila prodanih u mjesec dana.

Cilj je odabrati značajku koja će ići u korijen stabla. Da bi to bilo moguće treba pokazati koliko dobro pojedina značajka predviđa prodaju auta. Prvo se u korijen

stavlja proizvoljno odabrana značajka (npr. Generacija vlasnika) i gleda se pozitivan i negativan slučaj, odnosno koliko se auta prodalo, a koliko ne. Za pozitivan slučaj je prodan 1 auto i 3 nisu, a za negativan slučaj 2 su prodana, a jedan ne. Prema formuli 4.1,

$$G^{(i)} = 1 - \sum_{k=1}^{n} (p_k^{(i)})^2 \tag{4.1}$$

gdje je i broj značajke, n broj različitih vrijednosti koje ta značajka može poprimiti i p vjerojatnost da značajka poprimi tu vrijednost. Za pozitivan slučaj se dobiva:

$$G_n = 1 - (\text{vjerojatnost za Da})^2 - (\text{vjerojatnost za Ne})^2 = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375.$$

Za negativan slučaj istim postupkom dobiva se:

$$G_p = 0.444.$$

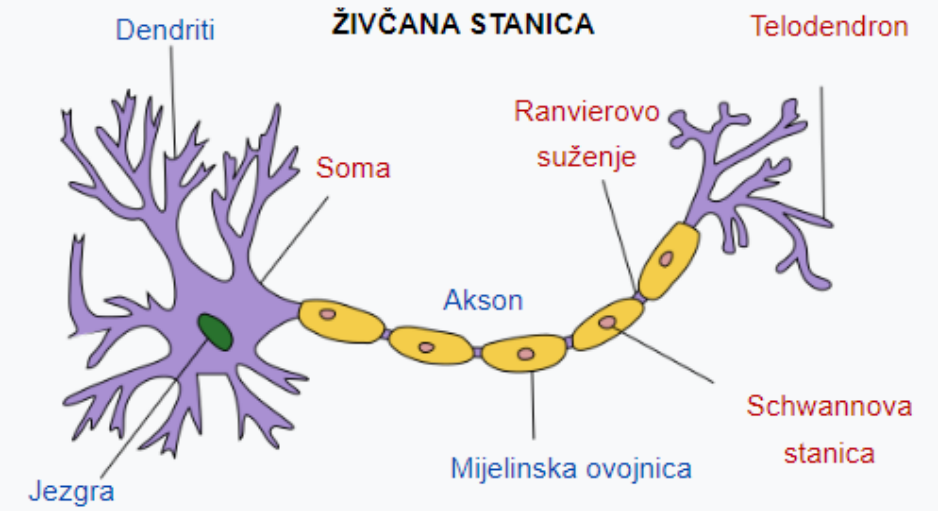
S obzirom na to da se broj vlasnika u pozitivnom i negativnom primjeru razlikuje, ukupni Gini indeks ove značajke je ponderirani zbroj ova dva slučaja. Tako se dobiva ukupni Gini indeks:

$$G_{uk} = \left(\frac{4}{4+3}\right) \cdot 0.375 + \left(\frac{3}{4+3}\right) \cdot 0.444 = 0.405. \tag{4.2}$$

Treća značajka je starost auta koja sadrži numeričke vrijednosti te se gini indeks računa malo drugačije. Prvo se svi primjeri sortiraju od najmanje do najveće vrijednosti značajke, a zatim se računa srednja vrijednost između svih susjednih primjera te se računa gini indeks za svaku srednju vrijednost. Ta vrijednost se dobiva tako da se u korijen stavi dana vrijednost i pita se jesu li sljedeće vrijednost veće ili manje od zadane te se istim algoritmom kao i u prethodnom slučaju proces nastavlja. Za implementaciju ovog koda u Pythonu korištena je scikit learn biblioteka. Implementacija algoritma u Pythonu se može naći u dodatku D.

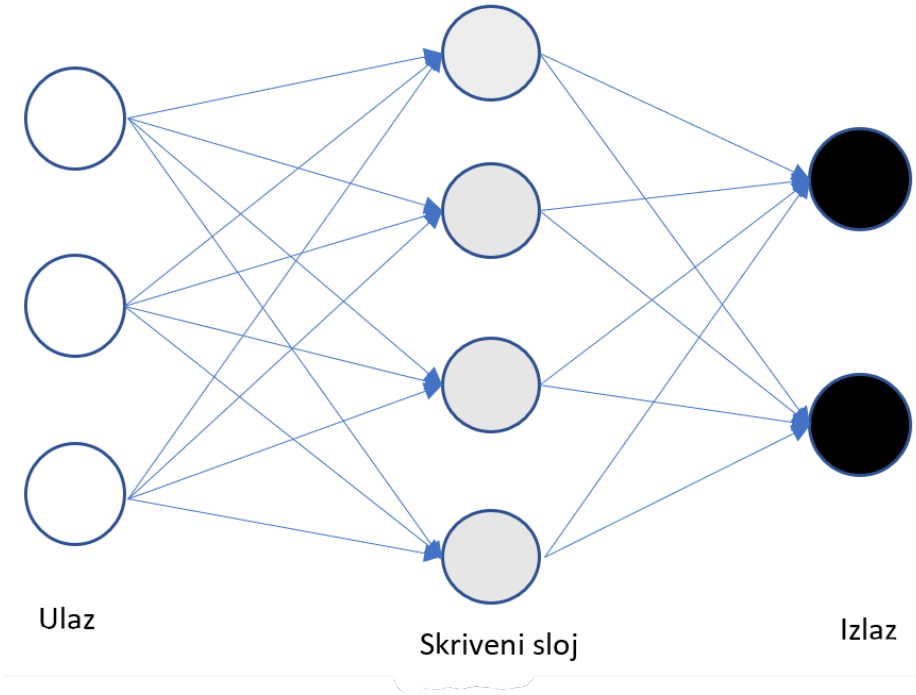
4.2.4 Neuralna mreža

Neuralnom mrežom se naziva skup međusobno povezanih neurona. Kao takva može biti biološka (formirana prirodno u mozgu) ili umjetna (konstruirana u računalima, digitalnim ili analognim sklopovima). Umjetna neuralna mreža je stvorena po uzoru na biološku stoga se uz sličnosti u principu rada koriste i slični termini (neuron, učiti, akson, ...). Na slici 4.9 je prikazana jedna živčana stanica, neuron.



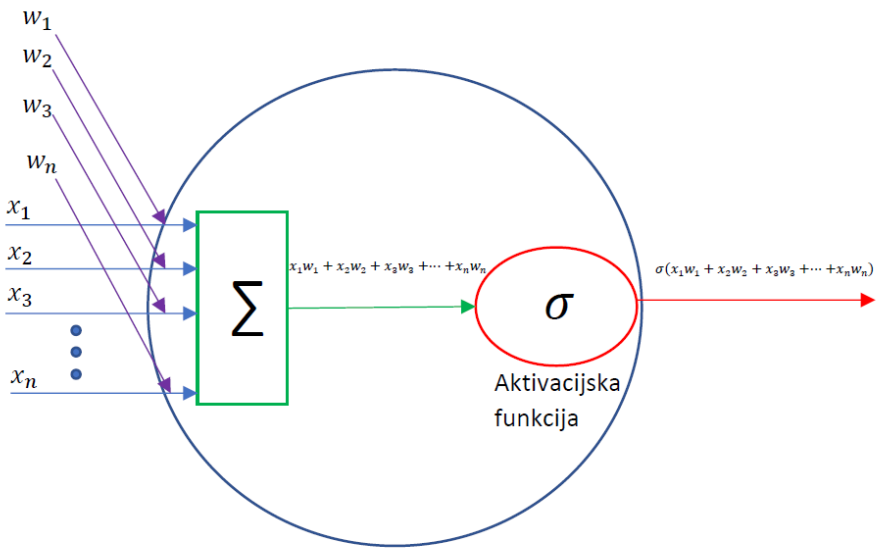
Slika 4.9: Ilustracija neurona. (slika preuzeta s [14])

Neuron kroz dendrite, na koje se spajaju ostali neuroni preko telodendrona, prima informacije koje procesira u jezgri. Kroz akson, odnosno telodendrone, procesiranu informaciju šalje dalje ostalim neuronima u mreži. Simulaciju takve biološke mreže moguće je izvesti kroz niz slojeva koje čine neuroni, gdje ulaze neurona $(n+1)-vog$ sloja čine izlazi neurona $n-tog$ sloja. Prema lokaciji u mreži razlikuju se ulazni, izlazni i skriveni slojevi. Svaka umjetna neuralna mreža mora imati jedan ulazni i jedan izlazni sloj dok broj skrivenih slojeva može biti bilo koji broj iz skupa $\mathbb{N}_0$ i ovisi o problemu na kojeg se neuralna mreža primjenjuje. Primjer neuralne mreže je prikazan na slici 4.10. Neuralna mreža je potpuno povezana ako je svaki neuron iz $n-tog$ sloja povezan sa svakim neuronom iz $(n+1)-vog$ sloja, u suprotnom mreža je djelomično povezana.



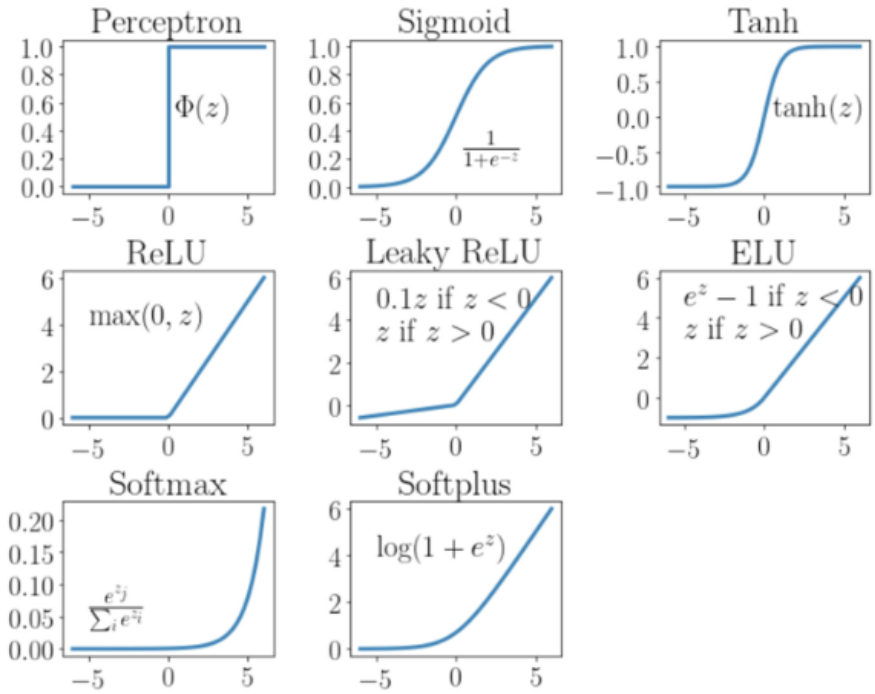
Slika 4.10: Neuralna mreža.

Slika 4.11 prikazuje model neurona. Na ulazu neuron prima skup značajki x_i čija je vrijednost otežana težinom (engl. weight) w_i. Ta težina govori o važnosti pojedine značajke. Te vrijednosti se sumiraju te se tako dobiva linearna kombinacija otežanih značajki koja se kao argument predaje aktivacijskoj funkciji σ.



Slika 4.11: Model neurona.

Aktivacijska funkcija je ne linearna funkcija koja ograničava izlaznu vrijednost neurona na određeni interval. Postoje više aktivacijskih funkcija te odabir iste ovisi o problemu. Neke od aktivacijskih funkcija su dane kao primjer na slici 4.12.



Slika 4.12: Aktivacijske funkcije. (slika preuzeta iz [17])

Pojam učenja neuralne mreže podrazumijeva proces odabira težine tako da funkcija gubitka (engl. loss function) $L(w)$ poprimi najmanju moguću vrijednost. Trening primjeri prolaze kroz neuralnu mrežu pomoću algoritma zvanog "feedforward" koji na izlazu neuralne mreže predviđa oznaku danog primjera. Predviđena oznaka se uspoređuje sa stvarnom te se kroz proces zvan "backpropagation" parametri modela (težine) namještaju tako da razlika bude što manja. Upravo je funkcija gubitka ta koja uspoređuje predviđenu i stvarnu oznaku. Neke od često korištenih funkcija gubitka su:

- Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2,$$

- Mean Apsolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|,$$

- Binary Cross-Entropy (BCE)

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} \cdot log(y^{(i)}) + (1 - \hat{y}^{(i)}) \cdot log(1 - y^{(i)}),$$

- Categorical Cross-Entropy (CCE)

$$CCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\hat{y}^{(ij)} \cdot log(y^{(ij)}).$$

Kao što je već navedeno vrijednosti težina se moraju mijenjati kako bi njihova vrijednost na kraju bila najoptimalnija. Funkcija koja određuje kako će se vrijednost utega mijenjati se zove optimizacijska funkcija. Postoji više optimizacijskih funkcija, a neke od njih su Gradijent spust, Stohastički gradijent spust, RMSProp, Adagrad i trenutno najpopularniji Adam.

Za gradnju modela neuralne mreže u Pythonu korištena je Keras biblioteka. Struktura korištenog modela neuralne mreže je prikazana na slici 4.13.

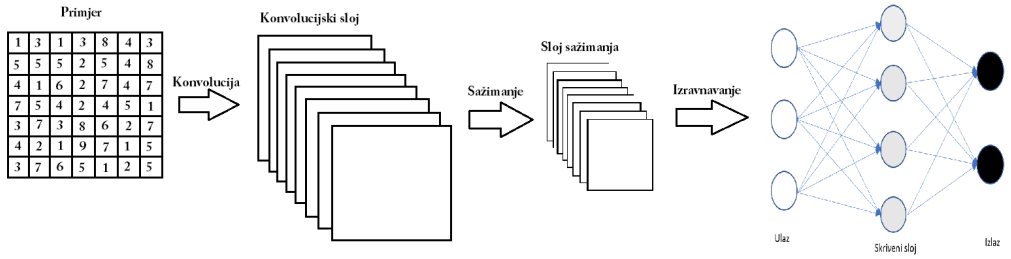
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 120)	3720
dense_1 (Dense)	(None, 60)	7260
dense_2 (Dense)	(None, 60)	3660
dense_3 (Dense)	(None, 10)	610

Slika 4.13: Prikaz strukture neuralne mreže.

Implementacija algoritma u Pythonu se može naći u dodatku E.

4.2.5 Konvolucijska neuralna mreža

Konvolucijska neuralna mreža je tip neuralne mreže koja uz potpuno povezane slojeve (neuralna mreža iz prethodnog primjera) ima barem jedan konvolucijski sloj. Uglavnom se uz navedene slojeve dodaju još i slojevi sažimanja (engl. *pooling layer*) i izravnavanja (engl. *flatten layer*). Skica jedne konvolucijske neuralne mreže je prikazana na slici 4.14.



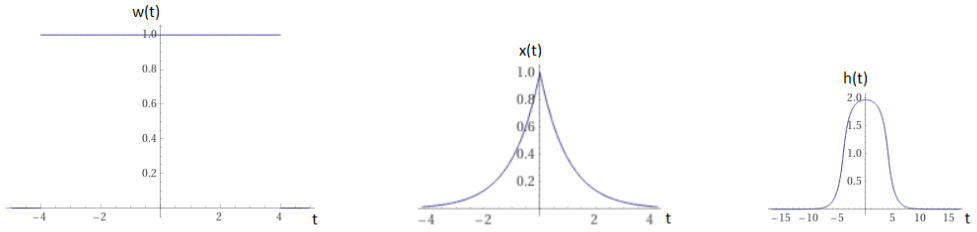
Slika 4.14: Skica konvolucijske neuralne mreže.

- **Konvolucijski sloj**

  Konvolucija je matematički operator koji djelovanjem na dvije funkcije stvara treću funkciju koja izražava količinu preklapanja te dvije funkcije (4.3).

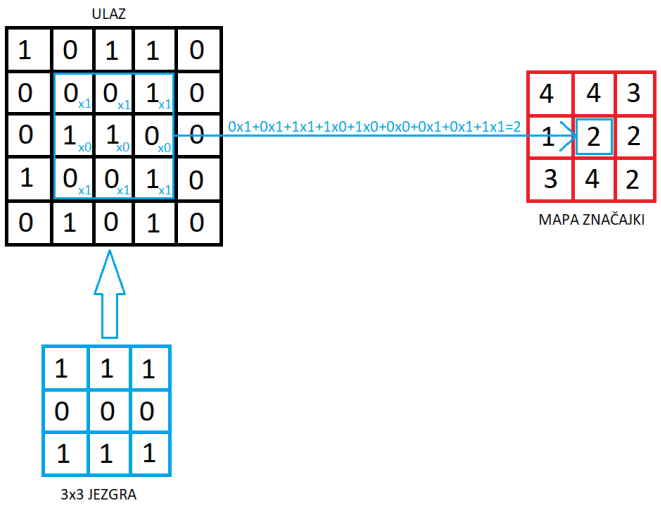
$$h(t) = (w * x)(t) = \sum_{\tau=\tau_{min}}^{\tau_{max}} w(\tau)x(t+\tau) \tag{4.3}$$

  Primjer rada konvolucije je prikazan na slici 4.15.



Slika 4.15: Primjer rada konvolucije.

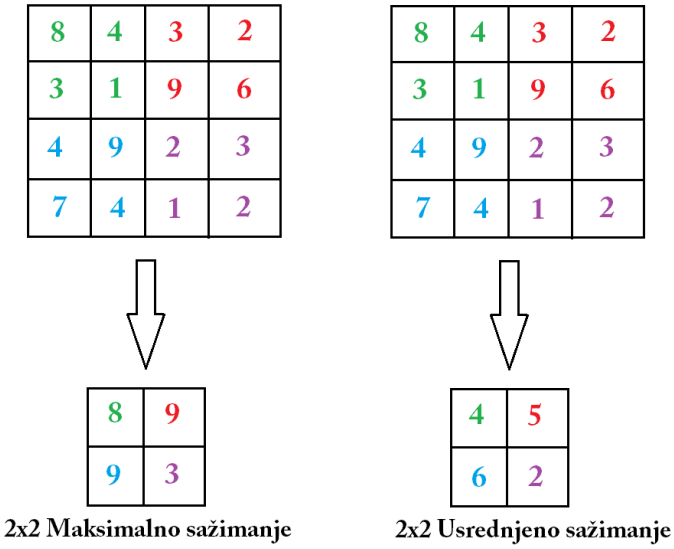
U kontekstu strojnog učenja, odnosno konvolucijske neuralne mreže funkcija x predstavlja ulaz, w jezgru, a h mapu značajki. Na slici 4.16 je ilustriran primjer prolaska jezgre kroz ulazni 2D primjer te izlaznu mapu značajki koja je rezultat konvolucije. Tako slojevi blizu ulaza mogu naučiti neke jednostavne značajke npr. linije, a dublji slojevi mogu učiti neke kompliciranije značajke kao npr. obrise.



Slika 4.16: Primjer prolaska jezgre.

- **Sloj sažimanja**

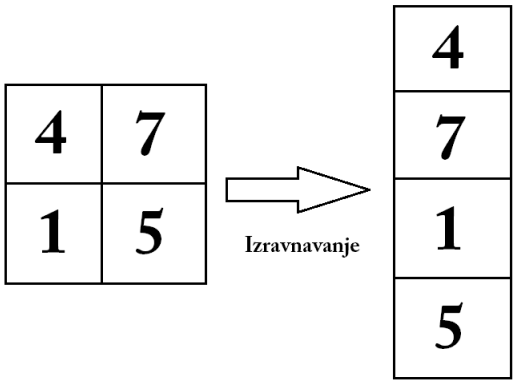
  Problem s mapom značajki je taj što pamti točni položaj značajke iz ulaznog podatka, te time postaje jako osjetljiva na promjenu lokacije značajke. Pa tako ako se npr. slika zarotira, mapa značajka će biti drukčija. Jedan od rješenja tog problema je dodavanje sloja sažimanja koji u suštini smanjuje razlučivost signala. Najčešći način na koji se to radi je pomoću filtera dimenzije 2×2 te uobičajeno uzima aritmetički prosjek (engl. average pooling) vrijednosti ulaznog signala unutar dane dimenzije ili uzimanjem najveće vrijednosti (engl. max pooling). Prikaz rada spomenutih filtera je prikazan na slici 4.17.



Slika 4.17: Primjer rada sloja sažimanja.

- **Sloj izravnjavanja**

  Sloj izravnavanja uzima 2d podatak iz sloja sažimanja i pretvara ga u vektor koji se predaje potpuno povezanoj neuralnoj mreži. Skica tog procesa je prikazana na slici 4.18.



Slika 4.18: Primjer rada sloja izravnavanja.

Za gradnju modela konvolucijske neuralne mreže u Pythonu korištena je Keras biblioteka. Struktura korištenog modela konvolucijske neuralne mreže je prikazana na slici 4.19.

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 30, 32)	192
conv1d_2 (Conv1D)	(None, 30, 64)	6208
conv1d_3 (Conv1D)	(None, 30, 128)	24704
max_pooling1d_2 (MaxPooling1	(None, 15, 128)	0
flatten_1 (Flatten)	(None, 1920)	0
dense_1 (Dense)	(None, 120)	230520
dense_2 (Dense)	(None, 60)	7260
dense_3 (Dense)	(None, 60)	3660
dense_4 (Dense)	(None, 10)	610

Slika 4.19: Prikaz strukture konvolucijske neuralne mreže.

Implementacija algoritma u Pythonu se može naći u dodatku F.

5 Rezultati

Navedeni algoritmi korišteni su i na simulacijskim i na eksperimentalnim podacima. Točnosti predviđanja algoritama s pripadajućom pogreškom prikazane su u tablici 5.1.

Algoritam	Točnost predviđanja (simulacija)	Točnost predviđanja (eksperiment)
K najbližih susjeda	$66.00\% \pm 0.47\%$	$26.52\% \pm 1.42\%$
Stablo odlučivanja	$66.52\% \pm 6.94\%$	$28.41\% \pm 4.04\%$
Neuralna mreža	$65.51\% \pm 4.34\%$	$12.50\% \pm 1.86\%$
Konvolucijska neuralna mreža	$69.51\% \pm 1.20\%$	$14.77\% \pm 4.25\%$

Tablica 5.1: Rezultati predviđanja algoritma.

6 Zaključak

U daljnjem razmatranju ovog problema predlaže se korištenje 1D konvolucijske neuralne mreže s obzirom na to da daje najkonzistentniju visoku točnost predviđanja (na simulacijskim podacima) kao što se može zaključiti iz tablice 5.1. Kod eksperimentalnih podataka niti jedan algoritam ne daje zadovoljavajuću točnost predviđanja. Uspoređujući eksperiment sa simulacijom mogu se prepoznati neke bitne razlike u geometriji prostora i broju detektora koje bi mogle biti uzrok ovakvom ishodu. Konkretno, za očekivati je da bi rezultati na mjerenjima bili znatno bolji kada bi se umjesto jednog detektora u mjerenjima koristilo 15 detektora kao u numeričkim simulacijama. Predlaže se nastavak istraživanja s većim brojem detektora te većim brojem primjera za učenje i testiranje algoritma. Nakon provedenih mjerenja RSSI vrijednosti predlaže se mjerenje CSI vrijednosti s obzirom da sadrži vrijednost signala svakog podnosioca te samim time daje više informacija.

Dodaci

Dodatak A Priprema podatka (simulacija)

```
def cleanit(path, z):
    df = pd.read_csv(path, skiprows=4)
    df.drop(df.columns[0:((z*2)+1)], axis=1, inplace=True)
    data_s = df.to_numpy()                                      # 15*10 matrica s stringovima
    data_c = np.zeros([10, 15], dtype=complex)                  # 15*10 matrica s 0 u kompleksnom zapisu
    for m in range(0, 15):
        for n in range(0, 10):
            data_c[n, m] = complex(data_s[n, m])                # 15*10 matrica u kompleksnom zapisu

    data_real = data_c.real                                     # 15*10 matrica - realni dio
    data_imaginary = data_c.imag                                # 15*10 matrica - imaginarni dio

    dataf = np.hstack((data_real, data_imaginary))              # 30*10 matrica [[realni] [imaginarni]]
    yf = np.full((1, len(dataf)), z)
    return dataf, yf

(data1_train, y1_train) = cleanit('concrete_room_1p_set1.csv', 1)
(data2_train, y2_train) = cleanit('concrete_room_3p_set1.csv', 3)
(data3_train, y3_train) = cleanit('concrete_room_5p_set1.csv', 5)
(data4_train, y4_train) = cleanit('concrete_room_7p_set1.csv', 7)
(data5_train, y5_train) = cleanit('concrete_room_9p_set1.csv', 9)

(data12_train, y12_train) = cleanit('concrete_room_1p_set2.csv', 1)
(data22_train, y22_train) = cleanit('concrete_room_3p_set2.csv', 3)
(data32_train, y32_train) = cleanit('concrete_room_5p_set2.csv', 5)
(data42_train, y42_train) = cleanit('concrete_room_7p_set1.csv', 7)
(data52_train, y52_train) = cleanit('concrete_room_9p_set1.csv', 9)

(data13_train, y13_train) = cleanit('concrete_room_1p_set3.csv', 1)
(data23_train, y23_train) = cleanit('concrete_room_3p_set3.csv', 3)
(data33_train, y33_train) = cleanit('concrete_room_5p_set3.csv', 5)
(data43_train, y43_train) = cleanit('concrete_room_7p_set3.csv', 7)
(data53_train, y53_train) = cleanit('concrete_room_9p_set3.csv', 9)

(data14_train, y14_train) = cleanit('concrete_room_1p_set4.csv', 1)
(data24_train, y24_train) = cleanit('concrete_room_3p_set4.csv', 3)
(data34_train, y34_train) = cleanit('concrete_room_5p_set4.csv', 5)
(data44_train, y44_train) = cleanit('concrete_room_7p_set4.csv', 7)
(data54_train, y54_train) = cleanit('concrete_room_9p_set4.csv', 9)


data = np.vstack((data1_train, data2_train, data3_train, data4_train, data5_train,
                  data12_train, data22_train, data32_train, data42_train, data52_train,
                  data13_train, data23_train, data33_train, data43_train, data53_train,
                  data14_train, data24_train, data34_train, data44_train, data54_train))

label = np.transpose(np.hstack((y1_train, y2_train, y3_train, y4_train, y5_train,
                            y12_train, y22_train, y32_train, y42_train, y52_train,
                            y13_train, y23_train, y33_train, y43_train, y53_train,
                            y14_train, y24_train, y34_train, y44_train, y54_train)))
```

Dodatak B Priprema podatka (eksperiment)

```
def cleanit(path, num_people):
    df = pd.read_csv(path)
    df=df.drop(['time'], axis=1)
    #print(df)
    whole_table=df.to_numpy()
    table = np.zeros( (11, 100) )
    k=0
    for j in range(0,11):
        for i in range(0,100):
            table[j,i]=whole_table[k]
            k=k+1

    print(table)
    label = np.full((1, len(table)), num_people)
    return table, label

(data11_train, y11_train) = cleanit('1_1.csv', 1)
(data12_train, y12_train) = cleanit('1_2.csv', 1)
(data13_train, y13_train) = cleanit('1_3.csv', 1)

(data21_train, y21_train) = cleanit('2_1.csv', 2)
(data22_train, y22_train) = cleanit('2_2.csv', 2)
(data23_train, y23_train) = cleanit('2_3.csv', 2)

(data31_train, y31_train) = cleanit('3_1.csv', 3)
(data32_train, y32_train) = cleanit('3_2.csv', 3)
(data33_train, y33_train) = cleanit('3_3.csv', 3)

(data41_train, y41_train) = cleanit('4_1.csv', 4)
(data42_train, y42_train) = cleanit('4_2.csv', 4)
(data43_train, y43_train) = cleanit('4_3.csv', 4)

(data51_train, y51_train) = cleanit('5_1.csv', 5)
(data52_train, y52_train) = cleanit('5_2.csv', 5)
(data53_train, y53_train) = cleanit('5_3.csv', 5)

(data61_train, y61_train) = cleanit('6_1.csv', 6)
(data62_train, y62_train) = cleanit('6_2.csv', 6)
(data63_train, y63_train) = cleanit('6_3.csv', 6)

(data71_train, y71_train) = cleanit('7_1.csv', 7)
(data72_train, y72_train) = cleanit('7_2.csv', 7)
(data73_train, y73_train) = cleanit('7_3.csv', 7)

(data81_train, y81_train) = cleanit('8_1.csv', 8)
(data82_train, y82_train) = cleanit('8_2.csv', 8)
(data83_train, y83_train) = cleanit('8_3.csv', 8)

data = np.vstack((data11_train, data12_train, data13_train, data21_train, data22_train,
                  data23_train, data31_train, data32_train, data33_train, data41_train,
                  data42_train, data43_train, data51_train, data52_train, data53_train,
                  data61_train, data62_train, data63_train, data71_train, data72_train,
                  data73_train, data81_train, data82_train, data83_train))

label = np.transpose(np.hstack((y11_train, y12_train, y13_train, y21_train, y22_train,
                            y23_train, y31_train, y32_train, y33_train, y41_train,
                            y42_train, y43_train, y51_train, y52_train, y53_train,
                            y61_train, y62_train, y63_train, y71_train, y72_train,
                            y73_train, y81_train, y82_train, y83_train)))
```

Dodatak C Implementacija algoritma K najbližih susjeda

```
label=np.ravel(label)
kfold2 = StratifiedKFold(n_splits=3, shuffle=True, random_state=15)
cvscores2 = []

for train, test in kfold.split(data,label):


    neigh = KNeighborsClassifier(n_neighbors=1)
    neigh.fit(data[train], label[train])

    scores2 = neigh.score(data[test],label[test])

    print("%.2f%%" % (scores2*100))
    cvscores2.append(scores2*100)

print("(%.2f%% +/- %.2f%%)" % (np.mean(cvscores2),np.std(cvscores2)))
```

Dodatak D Implementacija algoritma Stablo odlučivanja

```
kfold3 = StratifiedKFold(n_splits=3, shuffle=True, random_state=15)
cvscores3 = []

for train, test in kfold3.split(data,label):


        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(data[train], label[train])

        scores3 = clf.score(data[test],label[test])

        print("%.2f%%" % (scores3*100))
        cvscores3.append(scores3*100)

print("(%.2f%% +/- %.2f%%)" % (np.mean(cvscores3),np.std(cvscores3)))
```

Dodatak E Implementacija algoritma Neuralna mreža

```
kfold = StratifiedKFold(n_splits=3, shuffle=True, random_state=15)
cvscores = []

for train, test in kfold.split(data,label):
    tf.keras.backend.clear_session()
    model = tf.keras.models.Sequential([
                                    tf.keras.layers.Dense(120,activation=tf.nn.relu),
                                    tf.keras.layers.Dense(60, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(60, activation=tf.nn.relu),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])

    model.compile(loss='sparse_categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
    model.fit(data[train], label[train], epochs=50, validation_data=(data[test], label[test]))
    scores = model.evaluate(data[test], label[test])
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1]*100)
print("(%.2f%% +/- %.2f%%)" % (np.mean(cvscores),np.std(cvscores)))
```

Dodatak F Implementacija algoritma Konvolucijska neuralna mreža

```
kfold = StratifiedKFold(n_splits=3, shuffle=True, random_state=15)
cvscores = []

for train, test in kfold.split(data,label):


    data_train = np.array(data[train]).reshape(data[train].shape[0], data[train].shape[1], 1)
    data_test = np.array(data[test]).reshape(data[test].shape[0], data[test].shape[1], 1)



    cnn_model = tf.keras.models.Sequential()
    cnn_model.add(Conv1D(filters=32, kernel_size=(5,), padding='same',
    activation=tf.keras.layers.LeakyReLU(alpha=0.001), input_shape = (data_train.shape[1],1)))
    cnn_model.add(Conv1D(filters=64, kernel_size=(3,), padding='same',
    activation=tf.keras.layers.LeakyReLU(alpha=0.001)))
    cnn_model.add(Conv1D(filters=128, kernel_size=(3,), padding='same',
    activation=tf.keras.layers.LeakyReLU(alpha=0.001)))
    cnn_model.add(MaxPool1D(pool_size=(3,), strides=2, padding='same'))
    cnn_model.add(Flatten())
    cnn_model.add(Dense(units = 120, activation=tf.keras.layers.LeakyReLU(alpha=0.001)))
    cnn_model.add(Dense(units = 60, activation=tf.keras.layers.LeakyReLU(alpha=0.001)))
    cnn_model.add(Dense(units = 60, activation=tf.keras.layers.LeakyReLU(alpha=0.001)))
    cnn_model.add(Dense(units = 10, activation='softmax'))

    cnn_model.compile(loss='sparse_categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
    cnn_model.fit(data_train, label[train], epochs=20, validation_data=(data_test, label[test]))
    scores = cnn_model.evaluate(data_test, label[test])
    print("%s: %.2f%%" % (cnn_model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1]*100)
print("(%.2f%% +/- %.2f%%)" % (np.mean(cvscores),np.std(cvscores)))
```

Bibliography

[1] John Vetelino, Aravind Reghu, Introduction to Sensors, 1st Edition, 2011.

[2] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In Proceedings of the 19th annual international conference on Mobile computing & networking (MobiCom '13). Association for Computing Machinery, New York, NY, USA, 27–38. https://doi.org/10.1145/2500423.2500436

[3] Human Positioning Estimation Method Using Received Signal Strength Indicator (RSSI) in a Wireless Sensor Network Kimio Oguchia,b*, Shou Marutab, Dai Hanawab,c

[4] Seymour, Tom, and Ali Shaheen. "History of Wireless Communication." (2011).

[5] https://www.britannica.com/biography/Samuel-F-B-Morse (14.08.2022.)

[6] https://www.britannica.com/biography/Alexander-Graham-Bell (14.08.2022.)

[7] https://eyenetworks.no/en/wi-fi-history/ (14.08.2022.)

[8] https://onlinelibrary.wiley.com/doi/full/10.1002/jmri.20002 (15.08.2022.)

[9] Y. Chapre, P. Mohapatra, S. Jha and A. Seneviratne, "Received signal strength indicator and its analysis in a typical WLAN system (short paper)," 38th Annual IEEE Conference on Local Computer Networks, 2013, pp. 304-307, doi: 10.1109/LCN.2013.6761255.

[10] Zheng Yang, Zimu Zhou, and Yunhao Liu. 2013. From RSSI to CSI: Indoor localization via channel response. ACM Comput. Surv. 46, 2, Article 25 (November 2013), 32 pages. https://doi.org/10.1145/2543581.2543592

[11] https://www.holehouse.org/mlclass/01_02_Introduction_regression_analysis_and_gr.html (15.08.2022.)

[12] Rebala, G., Ravi, A., & Churiwala, S. (2019). Machine Learning Definition and Basics. An Introduction to Machine Learning, 1–17. doi:10.1007/978-3-030-15729-6_1

[13] https://www.usgs.gov/special-topics/water-science-school/science/water-you-water-and-human-body (11.09.2022.)

[14] https://upload.wikimedia.org/wikipedia/commons/b/bc/Neuron_Hand-tuned.svg

[15] https://upload.wikimedia.org/wikipedia/commons/e/ed/VFPt_charges_plus_minus_thumb.s

[16] https://www.flickr.com/photos/oskay/4580563691/in/photostream/

[17] Johnson, N. S., Vulimiri, P. S., To, A. C., Zhang, X., Brice, C. A., Kappes, B. B., & Stebner, A. P. (2020). Invited Review: Machine Learning for Materials Developments in Metals Additive Manufacturing. Additive Manufacturing, 101641. doi:10.1016/j.addma.2020.101641