Contents

[1. Uvod 1](#_Toc73017054)

[1.1. Zadatak završnog rada 1](#_Toc73017055)

[2. Strojno učenje 2](#_Toc73017056)

[2.1. Nadzirano učenje 2](#_Toc73017057)

[2.2. Nenadzirano učenje 3](#_Toc73017058)

[2.3. Nadzirano učenje – Problem klasifikacije 4](#_Toc73017059)

[2.4. Neuronske mreže 4](#_Toc73017060)

[2.4.1. Aktivacijske funkcije 6](#_Toc73017061)

* UVOD
  + Zadatak
* [[KORIŠTENE TEHNOLOGIJE]]
  + Strojno učenje
    - Vrste algoritama (neke navesti, te dovesti u ciljano nadgledano učenje -> klasifikacija)
  + Nadgledano učenje
    - Klasifikacija
  + Modeli strojnog učenja
    - Umjetne neuronske mreže
  + Python
    - Tensorflow  
      [[i njeni slojevi ili da to objasnim kako nailazim na slojeve u "izrada modela"?]]
  + Blender
    - [[Postupak za umjetne podatke]]
* [[ISTRAŽIVANJE PODATAKA]]
  + Skup podataka
    - Pravi i umjetni podatci
  + Istraživanje odabranog skupa podataka koristeći Python
* [[IZRADA MODELA]]
  + [[Primjer jednostavne mreže]]  
      [[Analiza grešaka jednostavne mreže, npr ako ima overfitting]]
  + [[? kontam ići dalje sa kompleksnijim mrežama koje bi riješile po mogućnosti npr. overfitting i slične probleme]]
  + [[?]]
* ZAKLJUČAK
* ...

# Uvod

Cilj ovog završnog rada je istražiti kreiranje modela koji su trenirani za prepoznavanje nekog predodređenog skupa LEGO kocaka. To se može postići na više načina, te većina rada će se vrtjeti u Python programskoj okolini.

TODO

## Zadatak završnog rada

Proučiti i opisati problem klasificiranja LEGO kocaka. Predložiti model i dizajnirati sustav za klasificiranje te ga ispitati na odgovarajućem podatkovnom skupu.

# Strojno učenje

Strojno učenje je disciplina proučavanja korištenja neke vrste algoritma koja može sebe modificirati kroz neke određene akcije da daje traženi izlaz na neki dani ulaz. Te algoritme je potrebno trenirati tj. potrebno ih je učiti prije nego se mogu koristiti za svoje svrhe, te po tome ih dijelimo na dvije većinske grupe: nadzirano učenje i nenadzirano učenje.

## Nadzirano učenje

Algoritmi koji pripadaju nadziranom učenju se moraju učiti nad određenim skupom podataka, svaki komad informacije u podatcima mora imati svoju oznaku na kojoj piše šta je točno. Kad se algoritam trenira, on se uči nad ovim podatcima sa oznakama, te time može namještati svoje unutrašnje parametre na takav način da određeni podatci utječu na određen način na parametre, te algoritam to pamti preko danih oznaka, pa nakon treniranja kada se pokušava validirati ispravnost algoritma daju mu se podatci bez oznaka, te algoritam mora odrediti pripadajuću oznaku za dane ulazne podatke.

Neki od problema koji spadaju u nadzirano učenje su:

* Klasifikacija – za određen broj ulaznih podataka algoritam mora moći odrediti njihove pripadnosti u kategorija, npr. na temelju dane slike životinje pripadno trenirani algoritam mora moći odrediti kojoj klasi životinja pripada, tipa ako je dana slika psa algoritam određuje koja je sorta pasmine i slično. Ovaj rad za Lego kocke je tipa klasifikacije, za određene slike Lego kocaka određuje se koje su točno model
* Regresija – na temelju ulaznih podataka pokušava se dobiti pripadajući realni broj koji opisuje najbolje ulazne podatke (za razliku od klasifikacije, regresija daje realan broj dok klasifikacija daje jedan od mogućih izlaznih oznaka), npr. za dane podatke o rabljenom automobilu kao što su kilometraža, marka, model, postotak štete na autu i slično, algoritam može nam dati procjenu vrijednosti automobila
* Prijevod jezika – algoritam prima riječi ili tekst, te njihove prijevode za učenje, te kad se nakon treniranja da neki novi tekst ono mora moći odrediti prijevod danog teksta
* Prepoznavanje grešaka – algoritmu se daju podatci normalnog ponašanja za treniranje, te pri validaciji nad abnormalnim podatcima algoritam mora moći prepoznati visoko odstupanje. Npr. algoritmu se daju slike ispravnih mehaničkih komponenti za trening, te na novim slikama mora uočiti razlike kao što su pukotine ili iskrivljenje materijala[<https://global.canon/en/technology/crack2019.html>]
* Uklanjanje smetni – algoritmu se daju podatci koji ga upućuju na ciljane smetnje te trenira se tako da izmjenjuje podatke dok se smetnje ne uklone. Npr. algoritam se uči na slikama sa pomućenjem, npr. od kamera koje su se kretale pri snimanju, te trenira se tako da smanji utjecaj pomućenja mijenjanjem slike

## Nenadzirano učenje

Algoritmi za nenadzirano učenje se treniraju na drugačiji način naspram algoritama za nadzirano. Oni za trening primaju podatke bez oznaka, te moraju onda pronaći neku vrstu strukture u danim podatcima, kao što su grozdovi ili skupovi podataka ovisno o informacijama u podatcima, te za validaciju onda moraju točno svrstati nove podatke u dane skupove, iako točnost je ovisna o ljudskoj osobi koja procjenjuje kvalitetu kreiranih grupa.

Glavna korist nenadziranog učenja nad skupovima podatka je procjena gustoće (engl. *density estimation*), algoritam prema podatcima kreira polja koja sadrže neki postotak vjerojatnosti da podatak pripada tom skupu, što je bliže „centru“ polja to je veća vjerojatnost da pripada tom skupu.

Neki od problema koji spadaju u nenadzirano učenje su:

* Grupiranje – algoritmu se daju podatci sa određenim informacijama, algoritam pokušava te podatke grupirati u svoje grupe ovisno o njihovim informacijama, postoji više metoda za grupiranje
* K-means – metoda grupiranja gdje algoritam koristi informacije u podatcima da odredi skupova podataka, isprva odabere nasumičnih podataka te onda iterativno traži prosječnu udaljenost prema svim podatcima koji se smatraju pod tim skupom, nakon toga ponovno izračuna središnje točke svih skupova i ponavlja algoritam dok se ne konvergira u mirno stanje (konačno konvergirano stanje ne mora biti optimalno stanje)
* [[TODO plus i primjeri za potkrijepiti]]

## Nadzirano učenje – Problem klasifikacije

Klasifikacija je problem iz područja statistike, u problemu se pokušava doći do strukturiranog rješenja gdje nekoj promatranoj jedinici damo jednu oznaku iz skupa oznaka. Jedan jednostavan primjer bi bila klasifikacija elektronske pošte u ulazni sandučić ili u neželjenu poštu (engl. *spam*).Takvi ulazi moraju u sebi sadržavati neku informaciju (engl. *feature*) koja algoritmu pomaže odrediti kojoj od klasa ulazni podatak pripada, takve informacije mogu biti same po sebi kategorične (npr. osoba može imati krv tipa „A“, „AB“, „A+“, itd.), mogu biti bazirane na brojevima (npr. slike imaju piksele koje sadrže tri kanala za boje, najčešće sa vrijednostima od 0 do 255) ili jednostavno uspoređivati dolazeće podatke sa prijašnje treniranim podatcima tako da uspoređuje udaljenost ili sličnost novih podataka od treniranih.

Algoritam koji obavlja klasifikaciju se naziva klasifikator, ako postoje samo dvije klase za klasificirati onda se metoda zove „binarna klasifikacija“, ako imamo više klasa metoda se zove „klasifikacija s više klasa“ a ako imamo metodu gdje neki izlaz može imati više klasa pridruženo onda se radi o „klasifikaciji s više oznaka“. Ovaj rad će se baviti sa „klasifikacijom s više klasa“ te morat će pridružiti ulaznim slikama izlazne oznake.

Zato što se koriste slike kao ulazni podatci u algoritam klasifikacije s više oznaka, potrebno je odabrati pripadajuću strategiju s kojom se može problem riješiti, te odabrano rješenje ovog rada su umjetne neuronske mreže (engl. *Artificial neural networks*).

## Neuronske mreže

Umjetne neuronske mreže (engl. *Artificial neural networks – ANNs*) su algoritmi koji se baziraju na neuronskim mrežama životinja iz pravog svijeta, neuronska mreža se sastoji od kolekcije spojenih jedinica koje se nazivaju neuroni. Spoj među neuronima je sličan spoju kojeg prave pravi živci u obliku sinapsa, samo za olakšano i sigurnije programiranje tih spojeva modelira se sve u obliku slojeva neurona, slojevi su međusobno spojeni i tako se signal na ulazu propagira prema izlazu, kao što je prikazano na slici 2.4.3.



Slika 2.4.1 – Uzor neurona je prava biološka živčana stanica [[izvor](https://commons.wikimedia.org/wiki/File:Neuron3.png) slike]

Na slici se vidi da svaki neuron po uzoru akson – sinapsa – dendrit (izlaz – prijenos – ulaz) ima svoje ulaze () na dendritima, neka funkcija stanice u tijelu, te onda preko aksona ima izlaz() kojim se dalje propagira do kraja sustava. Prethodno spomenuta funkcija stanice neurona određuje kakav će izlaz biti na svakom izlazu ovisno o ulazu, funkcija sadrži u sebi težine koje se pri treningu mijenjaju ovisno o stimulusu te onda odlazi u aktivacijsku funkciju. Najjednostavnije objašnjenje je da svaki izlaz neurona je realna funkcija, te težina pojačava ili oslabljuje „signal“ na danom izlazu.



Slika 2.4.2 – Matematički model neurona[[izvor slike](https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc)]

Osnovni matematički model neurona je da svaki ulaz ima svoju težinu [[izvor za ovu sekciju](http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf)], težine se postavljaju tijekom treninga ovisno o važnosti ulaza. Na kraju neuron izvršava neku funkciju (često zvana aktivacijska funkcija) nad sumom svih ulaza množeni svojim težinama te to predstavlja kao izlaz svim sljedećim spojenim neuronima. [[Ovaj dio bilo bi potrebno uklopiti sa ostatkom teksta]]



Slika 2.4.3 – Primjer neuronske mreže sa jednim skrivenim slojem

Kao što je na početku poglavlja spomenuto, neuroni su često poslagani u slojevima zbog jednostavnosti kreiranja modela, to ne znači da signal putuje jednom kroz model i završava odmah na izlazu, nego u modelu se mogu nalaziti razne funkcionalnosti između slojeva koje pomažu modelu postići veći uspjeh te glavni cilj tih slojeva je izbjegavanje mrtvih neurona. [[Prvo moramo pregledati kakve aktivacijske funkcije postoje i kako mogu pomoći. OVO možda pomaketi, pokazati treniranje prvo i zapisati tekst za taj dio pa micati okolo stvari]]

### Aktivacijske funkcije

Aktivacijske funkcije korištene u umjetnim neuronima djeluju nad sumom ulaza množeni sa njihovim pripadajućim težinama. Te funkcije daju drugačije izlaze ovisno o treniranim parametrima težine u neuronima.[[izvor za ovu sekciju](https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/)]

Najčešće korišteni u praksi su:

* Sigmoid(logistička) funkcija:
* Tanh aktivacijska funkcija:
* ReLu aktivacija (ispravljena linearna jedinica):



Slika 2.4.1.1 – Prikaz aktivacijskih funkcija ReLu, sigmoid i Tanh

### Treniranje neuronskih mreža

[[namjesitti ovo i aktivacijske funkcije da ima smisla]]Treniranje neuronskih mreža se izvodi preko stohastičkog gradijentnog spusta (engl. *stochastic gradient descend*), na kraju kad model završi predviđanje algoritam treniranja izračunava pogrešku na svakom izlazu izlaznog sloja mreže, te greške onda prema ulazu propagira kroz sve slojeve unatrag. To zahtjeva da su aktivacijske funkcije derivabilne da bi se mogao proces nastaviti u tom smjeru. Kako algoritam treniranja prolazi kroz neurone ono namješta njihove težina na ulazima prema vrijednostima koje misli da će dovesti do boljih rezultata.

Ovdje se pojavljuje problem u treniranju neuronskih mreža, naime problemi nestajućeg i eksplodirajućeg gradijenta [[izvor za ovu sekciju](https://books.google.hr/books/about/Deep_Learning.html?id=Np9SDQAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false)]. Kako se trenira vrlo duboka mreža sa mnogo slojeva, dolazi problem nestajućeg gradijenta jer kako se greška i ispravci propagiraju prema nazad, kako dolazi prema kraju taj gradijent se toliko smanjio da ima malen ili neznatan utjecaj. Na suprotnom kraju, kako se gradijent propagira kroz slojeve može doći do toga da eksplodira, te da svi neuroni onda usklađuju težine sa tim gradijentom što dovodi do problema pri sljedećem ciklusu treniranja.

Ciklusi treniranja nad skupom podataka se nazivaju epohe, svaka epoha ima

[[Ovaj dio možda pomaketi kod aktivacijskih funkcija]]U dijelu nestajućeg gradijenta važnu ulogu igra aktivacijska funkcija tih neurona.