SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

NASLOV DIPLOMSKOG RADA

Dora Katić

Split, srpanj 2021.

SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Diplomski studij: **Računarstvo**

Smjer/Usmjerenje: **Računarstvo**

Oznaka programa: 250

Akademska godina: 2020./2021.

Ime i prezime: **Dora Katić**

Broj indeksa: 512-2019

**ZADATAK DIPLOMSKOG RADA**

Naslov:

Zadatak: Napraviti pregled različitih područja primjene strojnog učenja (ML, Machine Learning), obraditi osnovne korake u strojnom učenju, od odabira i pripreme podataka do interpretacije dobivenih rezultata. Usporediti gotova programska okruženja za aplikacije strojnog učenja i napraviti pregled dostupnih javnih skupova podataka. Odabrati otvoreni skup podataka, gotovo okruženje te zadatak koji će se riješiti jednim od pristupa strojnog učenja. Prezentirati dobivene rezultate.

Prijava rada: 23.02.2021.

Rok za predaju rada:

Rad predan:

Predsjednik

Odbora za diplomski rad: Mentor:

prof. dr. sc. Maja Štula

**IZJAVA**

Ovom izjavom potvrđujem da sam diplomski rad s naslovom \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pod mentorstvom prof. dr. sc. Maje Štule pisala samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući navela u diplomskom radu citirala sam i povezala s korištenim bibliografskim jedinicama.

Studentica

Dora Katić

**SADRŽAJ**

[1. UVOD 1](#_Toc75035215)

[2. STROJNO UČENJE 2](#_Toc75035216)

[2.1 Umjetna inteligencija 2](#_Toc75035217)

[2.2 Strojno učenje 3](#_Toc75035218)

[2.3 Odabir modela i funkcije za strojno učenje 3](#_Toc75035219)

[2.4 Nadzirano učenje 7](#_Toc75035220)

[2.5 Nenadzirano učenje 8](#_Toc75035221)

[2.6 Polu-nadzirano učenje 9](#_Toc75035222)

[2.7 Podržano učenje 9](#_Toc75035223)

[3. KLASIFIKACIJA 11](#_Toc75035224)

[3.1 Binarna klasifikacija 11](#_Toc75035225)

[3.2 Višeklasna klasifikacija 13](#_Toc75035226)

[4. LOGISTIČKA REGRESIJA 16](#_Toc75035227)

[4.1 Gradijentni spust 18](#_Toc75035228)

[4.2 Višeklasna logistička regresija 20](#_Toc75035229)

[5. NEURONSKE MREŽE 21](#_Toc75035230)

[5.1 Osnovni pojmovi 21](#_Toc75035231)

[5.2 Backpropagation i forward propagation algoritmi 23](#_Toc75035232)

[5.3 Deep learning ??????????????????????????????? 24](#_Toc75035233)

[6. VIŠEKLASNA KLASIFIKACIJA CYBER-NAPADA 25](#_Toc75035234)

[7. ZAKLJUČAK 26](#_Toc75035235)

[LITERATURA 27](#_Toc75035236)

[POPIS OZNAKA I KRATICA 28](#_Toc75035237)

[SAŽETAK 29](#_Toc75035238)

[SUMMARY 30](#_Toc75035239)

# 

# UVOD

Svakim danom sve više raste količina podataka koji se koriste za analize i rad u različitim područjima u svim aspektima života. Događa se da potrebe nadilaze mogućnost čovjeka da samostalno proučava i pohranjuje te podatke zbog čega dolazi do razvoja strojnog učenja.

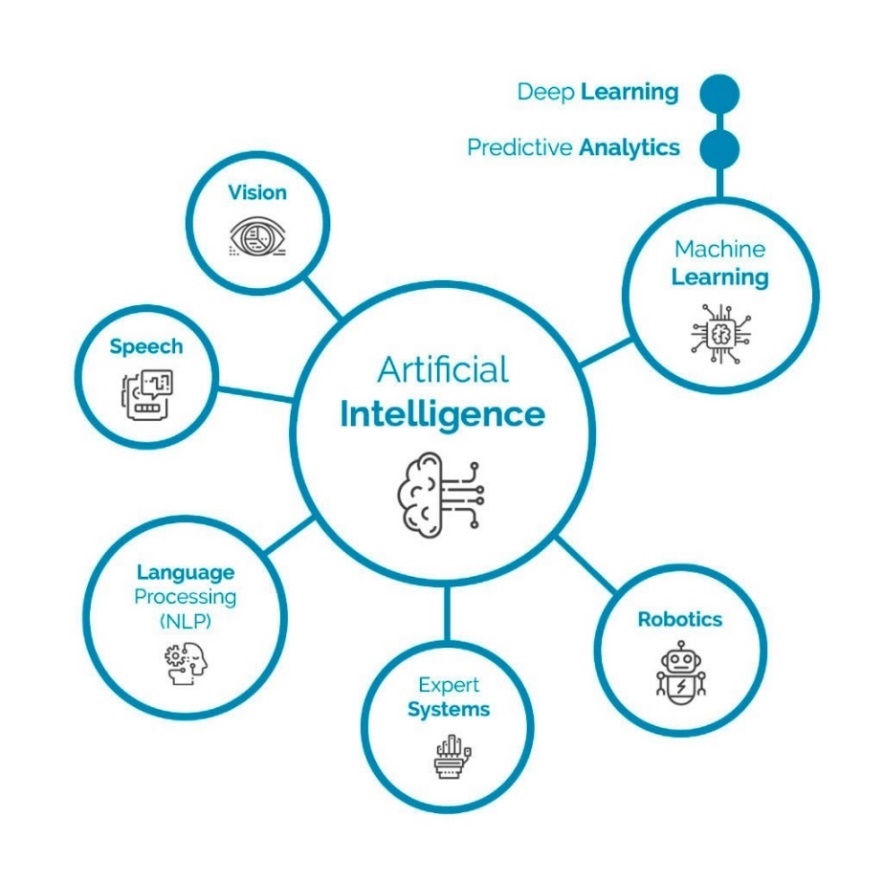
# STROJNO UČENJE

Prvenstveno, strojno učenje (engl. *Machine learning*, ML) grana je umjetne inteligencije čiji je cilj da računalo ili stroj oponaša ljudsko ponašanje odnosno da donosi zaključke o danom problemu i uči iz iskustva.

## Umjetna inteligencija

Umjetna inteligencija (UI) područje je računalne znanosti koje razvija inteligentne strojeve ili alate koji se ponašaju jako slično kao čovjek. Cilj umjetne inteligencije je razviti takav sustav koji će se samostalno snalaziti u novim situacijama, koristeći vlastitu „inteligenciju“ što znači bez intervencije korisnika. [14] Podjela umjetne inteligencije prikazana je nA SLICI

https://cdn.auraquantic.com/wp-content/uploads/2020/04/Artificial-Intelligence-img-1024x1024-1.jpg



UI temelji se na algoritmima. Oni predstavljaju upute koje stroj može izvoditi. Često se složeniji algoritmi stvaraju tako da se kombinira više jednostavnijih. Mnogi UI algoritmi imaju mogućnost da se samostalno poboljšavaju tako što uče neke nove strategije, koje su se pokazale kao dobre u prethodnim iskustvima, ili mogu stvarati i neke nove algoritme.

Izazov s kojim se susreće svaka vrsta umjetne inteligencije svakako je ograničenost kognitivnih sposobnosti strojeva i njihove arhitekture. Zbog toga postoje dva tipa modela: strukturalni i funkcionalni. Strukturalnim modelom nastoji se imitirati operacije uma kao što su rasuđivanje i logika. Funkcionalni model odnosi se na povezivanje podataka s onim što je izračunato.

## Strojno učenje

Samo strojno učenje odnosi se na programiranje računala da koristi već gotove podatke za poboljšavanje svog rada tako što uči iz njih te prepoznaje nekakve obrasce ili uzorke. Računalo koristi algoritme pomoću kojih uči što rezultira time da računalo samo, automatski može obraditi nove podatke i dati rezultate obrade preko stvorenih modela.

Naziv strojno učenje prvi je počeo koristiti Arthur Samuel koji je 1952. osmislio mali program za igranje šaha koji strojno učenje opisuje kao „područje koje računalu omogućuje učenje bez da se eksplicitno programira“. Tom Mitchell daje i moderniju definiciju: „Kaže se da računalni program uči iz iskustva E s obzirom na neku klasu zadataka T i mjeru uspješnosti P, ako se njegova uspješnost na zadacima iz T, mjerena mjerom P, poboljšava s iskustvom E.“. Zatim se tijekom 60-ih i 70-ih počinje javljati interes za klasifikaciju i prepoznavanje uzoraka kroz razne knjige i radove, a tako se sve više razvija i strojno učenje kakvo poznajemo danas.

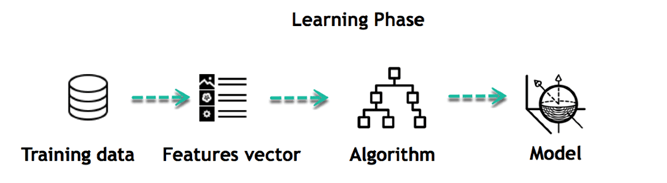
## Odabir modela i funkcije za strojno učenje

Proces strojnog učenja temelji se na nekoliko koraka:

1. Odluka o tome kakvi podaci se koriste za treniranje računala
2. Prikupljanje podataka za treniranje – podaci moraju imati ulazne (i eventualno pripadne izlazne vrijednosti)
3. Podaci se upisuju u *feature* vektore, pripremaju se za korištenje tako da se prilagođavaju potrebama problema koji se rješava, set podataka dijeli se na dio za treniranje i dio za testiranje
4. Odabir algoritama i definiranje funkcija
5. Pokretanje algoritma i stvaranje modela – optimiziranje parametara potrebnih za preciznost
6. Testiranje dobivene funkcije nad testnim podacima – računa se preciznost dobivenih rezultata
7. Ispravljanje mogućih pogrešaka i poboljšanje modela ukoliko je potrebno (u slučaju male preciznosti)

Kako bi strojno učenje bilo uspješno provedeno na nekom skupu podataka, potrebno je odabrati dobar algoritam koji uči iz podataka te onda stvoriti dobar model.[8] Model predstavlja ono što je naučeno algoritmom kroz dobro postavljenu hipotezu koja se označava s i ona se opisuje polinomom određenog stupnja.

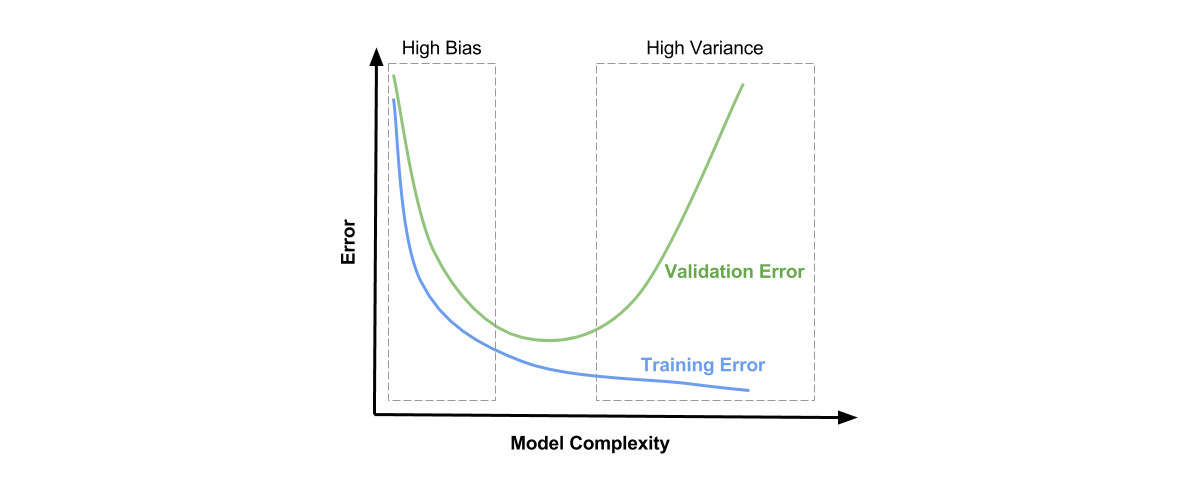
https://www.guru99.com/images/1/030819\_1026\_SupervisedM2.png

  
Podaci koji se koriste opisani su vektorom značajki (engl. *feature vector*) koji je definiran kao gdje je X oznaka za skup ulaznih vrijednosti, a pojedina ulazna vrijednost odnosno značajka i ti se podaci prikupljaju mjerenjem ili praćenjem onoga što želimo obrađivati algoritmom. Oznakom *m* označava se broj primjera za treniranje, a *n* je dimenzija vektora.

Strojno učenje usko je vezano i uz druga polja kao što su npr. statistika i optimizacija. Kako bi provjerili koliko je dobro postavljena hipoteza, moramo imat funkciju koštanja (engl. *cost function*) koja se označava s , gdje su parametri. Ona nam daje razliku između predviđene izlazne vrijednosti i stvarnog izlaza te se još označava i kao *error*. Cilj je minimizirati tu funkciju i zapravo na taj način optimizirati povezanost modela i sustava koji taj model predstavlja. Strojno učenje tako zapravo teži generalizaciji, odnosno želi se minimizirati gubitak na novim, nepoznatim primjerima, a blisko statistici zaključke donosi na temelju uzoraka te se opisuje koliko dobro je postavljen model.

Postupak odabira najboljeg modela nazivamo odabir modela (engl. *model selection*). [9] Budući da je složenost modela određena hiperparametrima, o njima i njihovoj optimizaciji ovisi sam odabir modela. [5] Bolja je odluka korištenje jednostavnijih modela koji su lakši za razumijevanje, a i bolje obavljaju već spomenutu generalizaciju. Također, takve je modele lakše naučiti jer imaju manje parametara za optimizaciju, a uz to je lakše iz njih izvesti pravila. Kako bi uspješno odabrali model, podatke je potrebno podijeliti na tri dijela: trening (~60% podataka), *cross validation*(~20% podataka) i testni dio(~20% podataka) te odaberemo stupanj polinoma kojim želimo definirati hipotezu. Za svaki stupanj polinoma računamo pri čemu koristimo *cross validation* dio. Tražimo onaj s minimalnom funkcijom i njega odabiremo kao stupanj koji ćemo koristiti za hipotezu, odnosno model. Problemi koji se mogu javljati su prenaučenost (engl. *overfitting* ili *high variance*) te podnaučenost (engl. *underfitting* ili *high bias*). Prenaučenost se javlja kada model u obzir uzima i detalji i šum u setu podataka za treniranje tako da oni negativno utječu na ponašanje modela za nove podatke i njegovu mogućnost generalizacije, odnosno stupanj polinoma prevelik je za podatke koje imamo. S druge strane, podnaučenost znači da model ne može modelirati ni trening podatke niti generalizirati nove podatke što rezultira lošim performansama . To znači da podatke opisujemo koristeći polinom malog stupnja, a zapravo trebamo veći stupanj da bi ih točno opisali. Ovaj problem je lakše rješiv jer ga je lako detektirati raznim metrikama za izvođenje. [7] Kako bi spriječili da dođe do prenaučenosti ili podnaučenosti, koriste se razne krivulje učenja (engl. *learning curves*) iz kojih se jasno vidi o kojem se problemu radi. Takve krivulje dobijemo kada iscrtamo funkciju koštanja (*error*) za trening set – i *cross validation* set – kao funkciju u ovisnosti o broju primjera *m*.  
Općenito vrijedi:

* Ako se radi o podnaučenosti (*high bias*) – i i imaju velike vrijednosti, također
* Ako se radi o prenaučenosti (*high variance*) – mala vrijednost, ima puno veću vrijednost od



Ukoliko imamo velike greške za novi set podataka, neka od rješenja su:

* Ispravljanje *high variance* – dodavanje novih primjera, smanjiti broj značajki
* Ispravljanje *high bias* – dodavanje dodatnih značajki, dodavanje polinomnih značajki

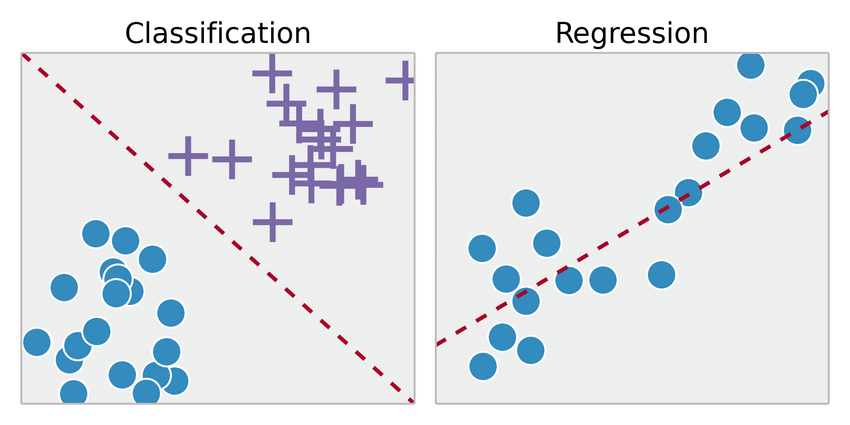
Važno je naglasiti da se pri dizajniranju cijelog sustava mora paziti i na veličinu pogreške. Za strojno učenje postoje različite metrike za analizu rada modela. Uvijek se kreće od nekog osnovnog, malog algoritma koji se brzo implementira. Njega se testira na *cv* setu podataka te se provjeravaju krivulje učenja. Prvo se ručno provjeravaju primjeri s potencijalnom greškom, ukoliko je to moguće, a nakon toga kreće se na korištenje metrika. Osnovne metrike su točnost (engl. *accuracy*)*,* preciznost (engl. *precision*) i odziv (engl. *recall*) koje se računaju različito, ovisno o tome o kojoj se vrsti strojnog učenja radi, a i o samim potrebama i zadacima.

Za strojno učenje jako je važno na koji se način prikupljaju podatci i u kakvom su oni obliku. Potrebno je prilagoditi format podataka kako bi postojalo nekakvo općenito pravilo po kojem ih računalo analizira. Tako prilagođeni podaci koriste se zatim u algoritmima koji su definirani pomoću modela za strojno učenje, kriterija za evaluaciju te postupka za optimizaciju svih korištenih parametara. [1] Ovisno o algoritmima i načinu provedbe strojnog učenja, definirane su dvije osnovne vrste strojnog učenja koje su opisane u nastavku.

## Nadzirano učenje

Nadzirano učenje (engl. *supervised learning)* odnosi se na strojno učenje pri kojem računalo uči iz već postojećih primjera koji imaju oznake. To znači da se analiziraju parovi ulaz-izlaz iz skupa podataka, računalo uočava uzorke i poveznice te se definira funkcija koja mu omogućava da za neku ulaznu vrijednost daje ispravnu izlaznu vrijednost. [3]  
Hipoteza se definira kao funkcija koja primjerima dodjeljuje oznake (engl. *labels*): . X je oznaka za skup ulaznih vrijednosti, a pojedina ulazna vrijednost, dok je Y skup izlaznih vrijednosti gdje se s označava pojedina izlazna vrijednost. Par () naziva se primjer za treniranje (engl. *training example*), a lista od *i=1,…,m* primjera je skup za treniranje (engl. *training set*). Važno je naglasiti da se *i* odnosi na indeks, a ne na eksponent. Hipoteza *h(x)* mora omogućavati da se za dani *x* dobije dobra procjena odgovarajuće vrijednosti *y*. Kažemo da je hipoteza *h* konzistentna s primjerom za učenje *(x,y)* ako i samo ako vrijedi: *h(x)=y*.

Postoje dva osnovna tipa nadziranog učenja, a to su klasifikacija i regresija. Klasifikacijom se određuje pripadnost nekoj klasi, odnosno *y* može poprimiti samo nekoliko diskretnih vrijednosti i ona je detaljno opisana u POGLAVLJU, dok se regresijom radi predviđanje neke kontinuirane izlazne vrijednosti. Njihova razlika vizualno je prikazana na SLICI

https://www.researchgate.net/profile/Yves-Matanga/publication/326175998/figure/fig9/AS:644582983352328@1530691967314/Classification-vs-Regression.png  


LINEARNA REGRESIJA MOZDA????

https://www.gatevidyalay.com/wp-content/uploads/2020/01/Representing-Linear-Regression-Model.png

## Nenadzirano učenje

Za razliku od nadziranog učenja, kod nenadziranog učenja (engl. *unsupervised learning*) problemu se može pristupiti tako da ne znamo točno kakav rezultat očekujemo nego tražimo nekakvu strukturu u podacima. Skup podataka sastoji se samo od ulaznih vrijednosti X te se traži veza među njima bez da postoje izlazne varijable Y. Nenadzirano učenje omogućava obavljanje složenijih zadataka u odnosu na nadzirano, ali može biti više nepredvidljivo.   
Najpoznatije vrste nenadziranog učenja su PCA (engl. *Principal Component Analysis*) te grupiranje (engl. *clustering*). Grupiranje se koristi na način da se podaci dijele u grupe, odnosno *cluster*e na način da se zajedno grupiraju oni podaci koji imaju zajedničke atribute te između kojih postoji nekakva poveznica. [6] Grupiranje samo po sebi nije jedan algoritam, ali svaki algoritam koji se koristi za grupiranje omogućava pronalaženje strukture u podatcima. Najčešće korišten *clustering* algoritam je *k-means* algoritam, gdje *k* označava broj korištenih *clustera*.

Ovakav pristup također omogućava i detekciju anomalija (engl. *anomaly detection*), odnosno podataka koji ne pripadaju nijednoj grupi. Ponekad je teško odabrati treba li se koristiti detekcija anomalija ili nadzirano učenje pa postoji nekoliko smjernica po kojima se dobro voditi pri tom odabiru:

|  |  |
| --- | --- |
| **Detekcija anomalija** | **Nadzirano učenje** |
| * Mali broj pozitivnih primjera (y=1 označava anomaliju) * S obzirom na mali broj pozitivnih primjera teško je naučiti iz njih jer buduća anomalija neće ličiti na prethodnu pa je lakše modelirati negativne primjere * Veliki broj negativnih primjera (y=0) koje onda možemo koristiti za stvaranje modela | * Velik broj i pozitivnih i negativnih primjera * Koristi puno pozitivnih i negativnih primjera za treniranje jer postoji dovoljan broj pozitivnih za razliku od detekcije anomalija |
| **Primjeri korištenja:** detekcija prevara, greške u proizvodnji, nadzor strojeva | klasifikacija emailova (*spam* poruka ili ne), predviđanje vremena |

## Polu-nadzirano učenje

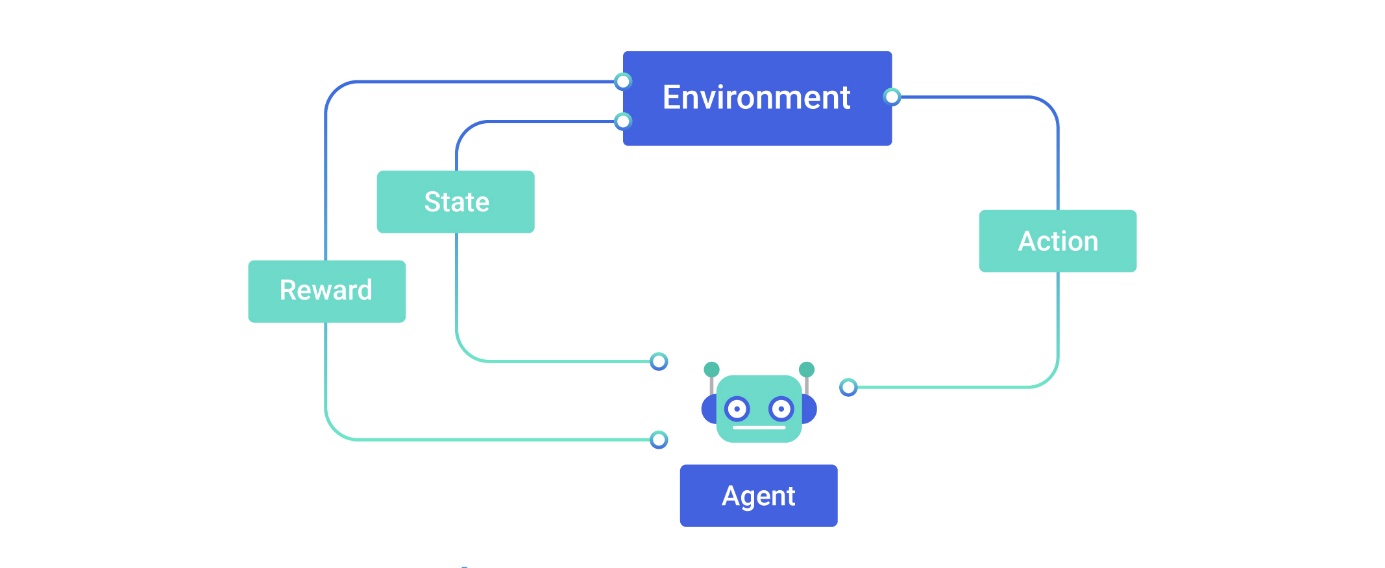
Polu-nadzirano učenje (engl. *semi-supervised learning*) vrsta je strojnog učenja koje za treniranje kombinira malu količinu označenih podataka i veliku količinu neoznačenih podataka. [10] Koristi se za setove podataka nad kojima se radi klasifikacija što znači da trebamo algoritme nadziranog učenja, ali želimo trenirati model bez da označavamo sve one primjere koji su bez oznake za što nam onda trebaju algoritmi nenadziranog učenja. Jedan od načina koji se često koristi je korištenje grupiranja i klasifikacije. Grupiranjem se pronalaze najvažniji podaci te ih onda možemo označiti i koristiti u klasifikaciji. Na ovaj se način može postići poboljšanje što se tiče točnosti modela.

## Podržano učenje

Podržano učenje je nešto drugačiji tip učenja od prethodno spomenutih. Odnosi se na korištenje inteligentnih agenata koje se trenira da poduzimaju akcije, u svom okruženju, s ciljem da se dođe do onoga što korisnik želi. Koristi u teorijama igara, robotici, statistici, navigaciji itd.

NA SLICI je prikazano kako se odvija podržano učenje, a da bi ono bilo uspješno provedeno potrebne su tri glavne komponente: agent (donosi odluke), okruženje (sve što je u interakciji s agentom) i akcije (radnje koje agent radi i o kojima donosi odluke). Samo učenje odvija se kroz sekvence odluka koje donosi model, na način da koristi pokušaje i pogreške da bi došao do rješenja tako što se nagrađuju ili kažnjavaju odluke koje su donesene. [11] Svaki agent nastoji maksimizirati ukupne nagrade tijekom određenog vremena. Cilj će dostići mnogo efikasnije ako pronađe dovoljno dobru strategiju pri donošenju odluka, a pri tome ne dobiva nikakve sugestije ili savjete vezano za odluke.

https://perfectial.com/wp-content/uploads/2018/07/img2-7.jpg



Ono što je izazov kod ovakve vrste strojnog učenja prvenstveno je postavljanje okruženja za simulacije jer ono jako ovisi o zadatku na kojem se radi. Također, skaliranje i popravljanje modela još je jedan izazov jer za komunikaciju ne postoji ništa osim nagrade i kazne. Uz to, agent može odraditi zadatak, ali ne na optimalan način ili da optimizira nagradu, a ne odradi onaj zadatak za koji je dizajniran. Svakako, podržano učenje jako je korisno i sve se više koristi u raznim područjima, ali je potreban trud da bi se sve uspješno odradilo.

# KLASIFIKACIJA

Klasifikacija je vrsta nadziranog strojnog učenja kod koje se podacima odnosno primjerima pridjeljuje klasa kojoj oni pripadaju. Klase su zapravo već spomenute oznake koje pripadaju skupu Y, a s obzirom da se radi o nadziranom učenju oznake klasa unaprijed su poznate. One se povezuju s pripadnim podacima preko funkcija za mapiranje, ali u ovom slučaju te izlazne vrijednosti poprimaju diskretne vrijednosti. Cilj klasifikacije razviti je dovoljno dobar model koji će točno odrediti kojoj klasi pripadaju novi podaci koji nisu bili korišteni pri treniranju algoritma.

Postoje dvije vrste klasifikacija koje su opisane u nastavku.

## Binarna klasifikacija

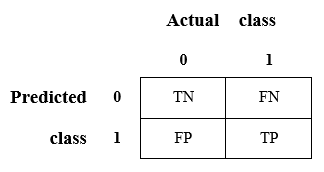
Ako se klasifikacija ograniči na samo dvije klase, govorimo o binarnoj klasifikaciji. Tada vrijedi: , gdje y=1 označava da je primjer pozitivan za jednu od dvije klase (pripada toj klasi), dok y=0 označava da je primjer negativan za tu klasu, odnosno primjer ne pripada toj klasi. [9] Takav se klasifikator nazivan binarni klasifikator. Hipoteza za binarnu klasifikaciju ima oblik: i ona nam daje vjerojatnost da je izlazna vrijednost 1. Na primjer, uvjeti mogu biti postavljeni na slijedeći način:

Vrijednost 0.5 naziva se vrijednosni prag (engl. *threshold*), a mjesto gdje je naziva se granica odluke (engl. *decision boundary*), koja ne mora biti samo linearna nego može poprimiti i druge oblike.

Evalucija binarne klasifikacije provodi se korištenjem već spomenutih metrika: točnost, preciznost i odziv. Kako bi se metrike koristile, prvenstveno je potrebno definirati matricu zabune (engl. *confusion matrix*). Ona se sastoji od 4 kategorije u koje se dijele dobiveni rezultati:

* TP (true positive) – primjeri koji su točno klasificirani kao pozitivni
* FP (false positive) – primjeri koji su klasificirani kao pozitivni, a nisu pozitivni
* FN (false negative) – primjeri koji su klasificirani kao negativni, a nisu negativni
* TN (true negative) – primjeri koji su točno klasificirani kao negativni

*Confusion matrix* prikazana je nA SLICI



Ukupan broj primjera označen je s N. Na osnovu matrice zabune, metrike se definiraju na sljedeći način. [12]

**TOČNOST**  
 je omjer ispravno klasificiranih primjera i ukupnog broja primjera. Problem s točnošću javlja se kada imamo disbalans u podacima. Takva pojava naziva se *skew* u podacima i znači da je distribucija podataka nagnuta na lijevu ili desnu stranu, odnosno nije simetrična. Tada se može dogoditi da imamo jako visok udio primjera jedne klase (npr. pozitivne) pa će klasifikator, koji svaki novi primjer klasificira kao pozitivan, imati visoku točnost, a zapravo se radi o jako lošem modelu.

#### **PRECIZNOST**

je omjer ispravno klasificiranih pozitivnih primjera i ukupnog broja primjera koji su klasificirani kao pozitivni. Odgovara na pitanje koliko je primjera zapravo točno klasificirano od svih koje je model klasificirao kao pozitivne.

#### **ODZIV**

je omjer ispravno klasificiranih pozitivnih primjera i ukupnog broja primjera koji su zapravo pozitivni. Odgovara na pitanje koliko je primjera model točno klasificirao za pozitivnu klasu. Ako, recimo, model stalno predviđa *y*=0, odziv će biti jednak nuli.

U praksi je teško postići i visoku preciznost i visok odziv pa se radi tzv. *trade off*. To znači da ako, primjerice, želimo izbjeći *false negative* rezultate, moramo imati viši odziv, a manju preciznost.

Posebna metrika koja je izvedena iz prethodno navedenih osnovih metrika je F1 *score*. On je težinski prosjek preciznosti i odziva. F1 u obzir uzima i FP i FN i često je dosta korisniji od točnosti, posebno kod nejednake distribucije klasa. Definira se kao:

Metode koje se najčešće koriste za binarnu klasifikaciju su:

* Stablo odlučivanja
* Neuralne mreže
* Bayesian klasifikacija
* Support Vector Machines

## Višeklasna klasifikacija

Često se javlja potreba da se primjeri klasificiraju u više od dvije klase i to nazivamo višeklasnom klasifikacijom (engl. *multiclass classification*). Tada se pripadna oznaka klase za neki primjer prikazuje kao *K* – dimenzijski vektor, . Općenito, postoji K klasa koje se označavaju s te vrijedi:

Npr. znači da primjer pripada klasi .

Metode koje se najčešće koriste za višeklasnu klasifikaciju su:

* Naive Bayes
* KNN (*K-nearest neighbours*)
* Neuralne mreže
* Logistička regresija (s jednom od slijedećih metoda)

Najčešće se problemi višeklasne klasifikacije rješavaju tako da se problem svodi na binarnu klasifikaciju, a za to se koriste *one-vs-all* (još se naziva i *one-vs-rest*) i *one-vs-one* metode. [13]

#### ***One-vs-all***

Kod ove metode odaberemo jednu klasu koju ćemo trenirati, dok sve ostale klase okupimo u drugu klasu. Tako smo dobili problem binarne klasifikacije i postupak ponavljamo za sve postojeće klase, odnosno imamo onoliko binarnih klasifikacija koliko imamo klasa.

Primjer: imamo 3 klase (red, blue, green) što znači da imamo 3 binarne klasifikacije:

* Binarna klasifikacija 1: red – [blue, green]
* Binarna klasifikacija 2: blue – [red, green]
* Binarna klasifikacija 3: green – [red, blue]

Kod ovakve se metode za svaki binarni klasifikator postavlja hipoteza:

=, *i=1,2,3,…,k,* gdje je *k* broj klasa.

Klasifikator se trenira za svaku klasu *i* da predviđa vjerojatnost da je *y*=*i*. Za svaki novi ulaz *x* odabire se klasa *i* koja maksimizira kako bi se predvidjelo kojoj klasi pripada taj ulaz.

Ova je strategija popularna, ali ima određene probleme. Glavni problem je da klase mogu biti jednoliko raspoređene u trening setu, ali s obzirom da treniramo jednu po jednu klasu, set negativnih će biti puno veći od pozitivnih što može loše utjecati na model i samu klasifikaciju. Isto tako proces može dugo trajati za velike setove podataka, spore modele ili veliki broj klasa.

#### ***One-vs-one***

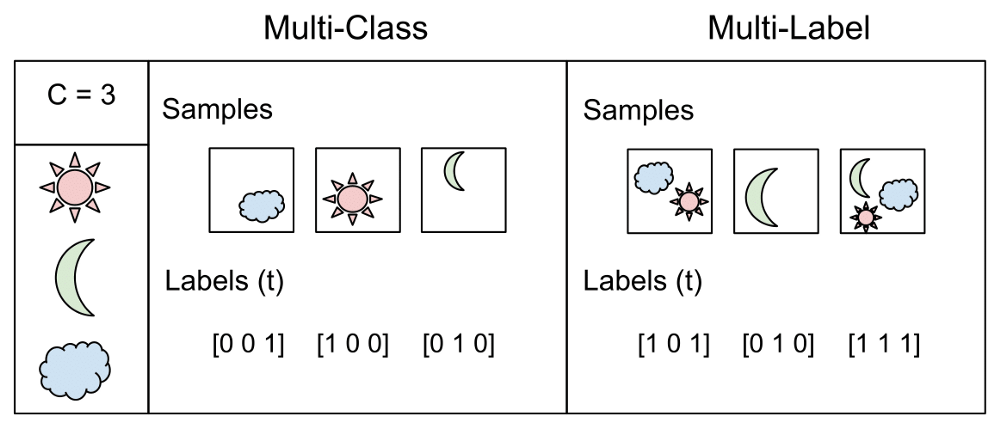
Kod ove metode imamo binarnih klasifikacija, gdje je *K* broj klasa. [14] To znači da svaki par klasa treniramo jednom.

Primjer: imamo 4 klase (red, blue, green, yellow) što znači da imamo 6 binarnih klasifikacija:

* Binarna klasifikacija 1: red – blue
* Binarna klasifikacija 2: red – green
* Binarna klasifikacija 3: red – yellow
* Binarna klasifikacija 4: blue – green
* Binarna klasifikacija 5: blue – yellow
* Binarna klasifikacija 6: green – yellow

Često dolazi do zabune kada se govori o *multi-class* i o *multi-label* klasifikaciji. Ove dvije vrste se jako razlikuju i ne smije ih se miješati. *Multi-label* odnosi se na primjere koji mogu pripadati u više klasa, dok kod *multi-class* svaki primjer pripada točno određenoj klasi.

https://4.bp.blogspot.com/-sCcOrQsTH9Q/XG1yv7mhERI/AAAAAAAAAJI/aEj6Jf1lookERHqPQS\_Y6Q9bxBcTV7TIwCLcBGAs/s1600/multiclass-multilabel.png



Postoji nekoliko metoda za provođenje *multilabel* klasifikacije:

* Transformacija u problem binarne klasifikacije – korištenje *binary relevance*metode (treniranje jednog binarnog klasifikatora za svaku klasu)
* Transformacija u problem višeklasne klasifikacije – korištenje *label powerset* transformacije (koristi se jedan binarni klasifikator za svaku kombinaciju klasa koje postoje)
* Korištenje više *multiclass* klasifikatora za stvaranje *multilabel* klasifikatora

S obzirom na paradigme učenja, tehnike koje se koriste za *multilabel* klasifikaciju mogu se podijeliti na *batch* ili *online* ML. Kod *batch* algoritama, svi podaci moraju biti dostupni prije samog korištenja te se koristi cijeli trening set. S druge strane, *online* algoritmi inkrementalno sastavljaju svoje modele u sekvencijalnim iteracijama.

# LOGISTIČKA REGRESIJA

Logistička regresija klasifikacijski je algoritam, ali se naziva regresija jer se koriste jako slične tehnike kao kod linearne regresije. Jedan je od najšire korištenih algoritama u strojnom učenju. Pri takvoj klasifikaciji, izlazne su vrijednosti diskretne (*y*=0 ili *y*=1), a vrijednost hipoteze . Logistička regresija ima svoju funkciju koštanja te se može koristiti i za višeklasnu klasifikaciju što će biti opisano u nastavku.

Kako bi definirali hipotezu, prvo je potrebno definirati aktivacijsku funkciju koja se naziva *sigmoid* (*logistic*) funkcija:

,

, gdje je *z* neki realan broj.

https://miro.medium.com/max/970/1\*Xu7B5y9gp0iL5ooBj7LtWw.pn

Kao što je prikazano na SLICI, *sigmoid* se približava 0 kada *z* ide u -∞ te se približava 1 kada *z* ide u +∞, odnosno g(z) poprima vrijednosti između 0 i 1. Hipoteza se definira kao:

Na taj se način ostvaruje traženi uvjet da . Kao što je već spomenuto, dobivena vrijednost odnosi se na vjerojatnost da je *y*=1, što se označava kao:

🡪 „vjerojatnost da je *y*=1 za dane značajke *x* s parametrima “

Funkcija koštanja za pojedini primjer definirana je kao:

Ovako definirana funkcija ima nekoliko dobrih karakteristika. Iz definicije zaključujemo da je *Cost*=0 ako je , odnosno ako hipoteza točno predviđa izlaznu vrijednost, nemamo nikakav gubitak. S druge strane, kada npr. za y=1 , tada . Takav je zaključak intuitivan: ako je , odnosno , a zapravo vrijedi *y*=1, algoritam će biti penaliziran s funkcijom koštanja velike vrijednosti. Definicija se može zapisati i na jednostavniji način (zbog toga što imamo samo dvije vrijednosti koje *y* poprima):

Ukupna funkcija koštanja za cijeli trening set definira se kao:

Ovako definirana funkcija koštanja je konveksna i koristi se uvijek za postavljanje modela logističke regresije.

Kod logističke regresije može se pojaviti i problem prenaučenosti pa se uz već spomenuta rješenja koristi i regularizacija. Ideja iza same regularizacije je korištenje manjih parametara te se tako stvara jednostavnija hipoteza i manja je vjerojatnost da će doći do prenaučenosti. Regularizacija se provodi dodavanjem regularizacijskog izraza s parametrom α na funkciju koštanja.

Taj se parametar naziva regularizacijski parametar i on kontrolira odnos između dva cilja: uklapanje podataka u hipotezu i zadržavanje parametara u malim vrijednostima. α ne smije biti prevelike vrijednosti jer algoritam može postati podnaučen i neće dobro raditi čak ni za trening set. Ovaj izraz ne penalizira , ali to ne predstavlja značajnu razliku.

## Gradijentni spust

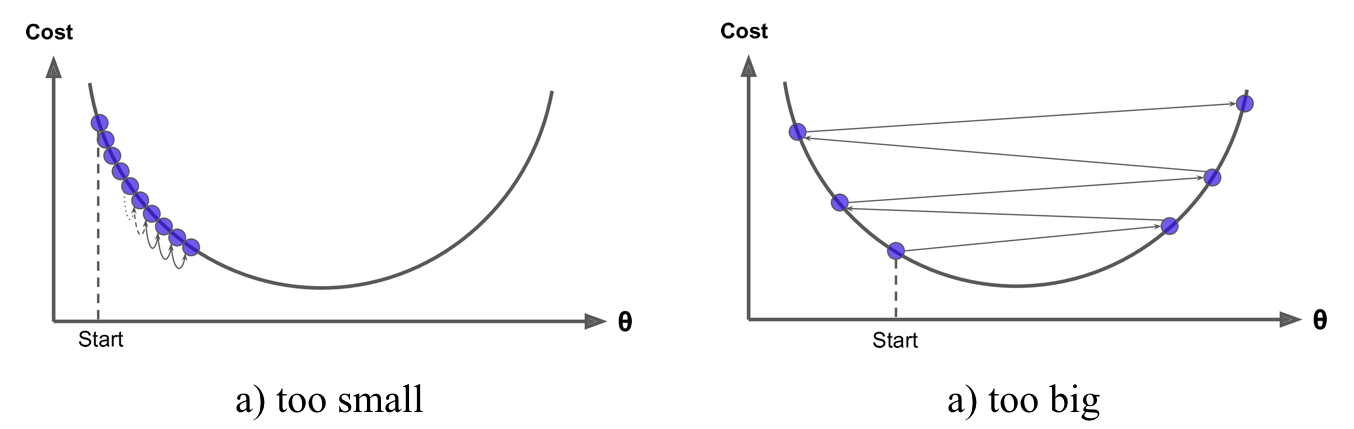
Osim hipoteze i funkcije koštanja, potrebno je odabrati parametre , nakon čega je moguće raditi predviđanja korištenjem dane hipoteze. Kako bi parametri bili dobro postavljeni, potrebno je pronaći takve parametre za koje je vrijednost minimalna. Tada za neki novi primjer *x* možemo raditi predviđanja kako bi dobili izlaznu vrijednost koristeći hipotezu (imamo mjere koje govore koliko dobro ona zadovoljava) te dobivene parametre. Za minimizaciju funkcije koštanja koristi se gradijentni spust (engl. *gradient decent*). Općenito, gradijentni spust odvija se na način da se uzmu neki početni parametri te se računa . Postupak se ponavlja tako što se parametri mijenjaju kako bi se došlo do minimalne vrijednosti .

https://miro.medium.com/max/600/1\*iNPHcCxIvcm7RwkRaMTx1g.jpeg



Za gradijentni spust posebno je važna stopa učenja (engl. *learning rate*) koja se označava s . Ona kontrolira koliki je korak pri gradijentnom spustu. Pri izboru stope učenja treba voditi računa o eventualnim problemima koji mogu nastati. Ako koeficijent premali, gradijentni spust bit će spor i previše će koraka trebati do minimuma. S druge strane, ako je prevelik, gradijentni spust može jednostavno preskočiti minimum te se može dogoditi da ne konvergira ili da čak divergira. Smjer u kojem idemo prema minimumu određen je parcijalnom derivacijom .

<http://uc-r.github.io/public/images/analytics/gbm/learning_rate_comparison.png>



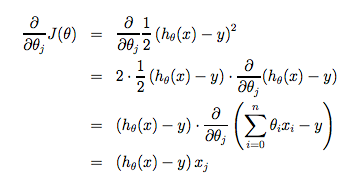
Vrijednost može biti fiksna jer gradijentni spust, kada se približava minimumu, automatski radi manje korake (jer se derivacija približava 0) što znači da ne treba mijenjati kroz vrijeme.

Općenita definicija gradijentnog spusta:

Ponavljaj {

}

Parcijalna derivacija za jedan primjer računa se na način:



Za logističku regresiju, gradijentni spust zato se definira na sljedeći način:

Ponavljaj {

(simultano ažuriraj sve vrijednosti )

}

mora se smanjivati pri svakoj iteraciji.

Za gradijentni spust također postoji regularizacijski izraz. Kako je već spomenuto, u regularizaciju s ne ulazi pa s njim radimo na drukčiji način:

Ponavljaj {

}

To znači da se parcijalna derivacija sada definira kao:

## Višeklasna logistička regresija

Višeklasna logistička regresija naziva se još i *multinomial* *logistic regression* i *softmax regression* (SMR). Ona je zapravo generalizacija logističke regresije koja sažima *k* dimenzionalni vektor proizvoljnih vrijednosti u *k* dimenzionalni vektor vrijednosti u rasponu (0, 1). Možemo je koristiti za višeklasnu klasifikaciju, uz uvjet da neki primjer može pripadati samo jednoj klasi.

??????????????????????????????????????????????????????????????'

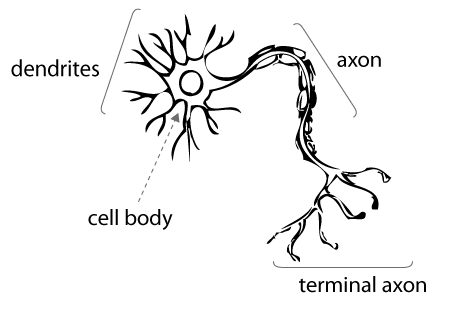
# NEURONSKE MREŽE

Umjetna neuronska mreža (engl. *neural network*, NN) skup je algoritama čiji je cilj prepoznavanje osnovnih veza među podacima kroz proces imitacije operacija ljudskog mozga, odnosno kroz simulaciju rada neurona u mozgu.

## Osnovni pojmovi

Neuroni su osnovne jedinice živčanog sustava i najsloženije su u ljudskom organizmu. [16] Sastavni dijelovi neurona su:

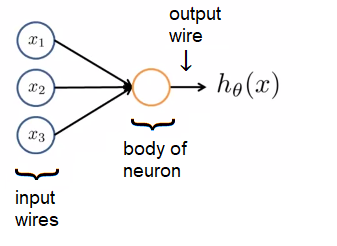
* Dendriti – kraći produžeci koji služe kao „ulazna žica“ (engl. *input wire*) na način da s osjetilnih organa dovode pobudu na tijelo stanice
* Tijelo stanice (engl. *cell body*) – u njemu se nalazi jezgra i kromosomi
* Aksoni – duži produžetak koji služi kao „izlazna žica“ (engl. *output wire*), odnosno prenosi živčane impulse s tijela stanice na druge živčane stanice ili izvršne organe



<https://s18798.pcdn.co/yungjurick/wp-content/uploads/sites/12997/2020/03/1_eBMwpBBboAXgqsawwOKkPw.png>

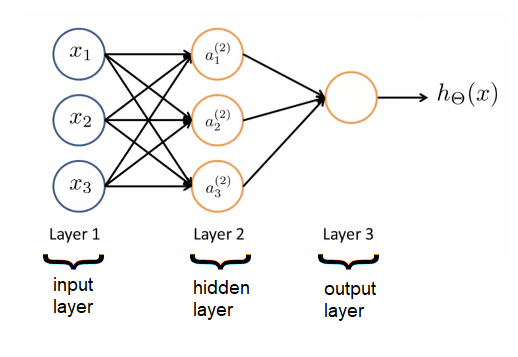
Kod umjetne neuronske mreže, koristi se zapravo jednostavni model onoga što neuron radi, odnosno neuron se modelira kao logistička jedinica, gdje je:

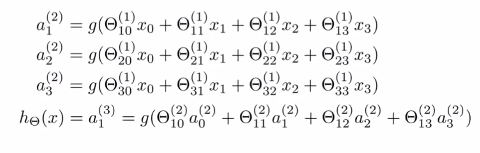
, odnosno ponovo imamo *sigmoid* aktivacijsku funkciju koja se označava i s *g*(*x*).



Uz *x1, x2* i *x3* ponekad se dodaje i *x0* koji se naziva *bias unit* za koji uvijek vrijedi *x*0=1.

Umjetne neuronske mreže su slojevite i sastoje se od tri glavna sloja: ulazni sloj (engl. *input layer*), skriveni sloj (engl. *hidden layer*) i izlazni sloj (engl. *output layer*). Ulazni sloj sadrži značajke *x* iz skupa podataka, izlazni sloj daje finalnu vrijednost *y* koju računa hipoteza , a skriveni sloj sadržava vrijednosti koje ne vidimo u trening setu pa stoga i naziv skriveni. Neuronska mreža može imati i više skrivenih slojeva.





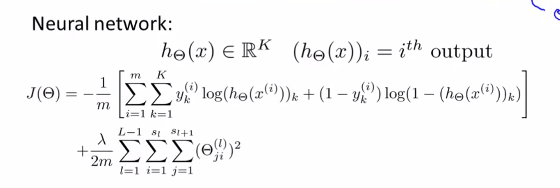
– aktivacija jedinice (neurona) *i* u sloju *j* (aktivacija se odnosi na vrijednost koja je izračunata i dana kao izlaz)

– matrica težina (težinskih faktora ili parametara) koja kontrolira mapiranje iz sloja *j* u sloj *j*+1, ako mreža ima jedinica u sloju *j*, a jedinica u sloju *j*+1, onda će imati dimenzije .

Ako za primjer uzmemo neuronsku mrežu koja rješava problem klasifikacije, podaci za treniranje su u obliku: .  
*L* – ukupan broj slojeva u neuronskoj mreži  
 – broj jedinica (bez *bias* jedinice) u sloju *l*Ako se radi o binarnoj klasifikaciji na kraju se dobije samo 1 izlazna jedinica, dok za višeklasnu klasifikaciju (*K* klasa) imamo *K* izlaznih jedinica.

Neuronske mreže mogu imati različite dijagrame i to se naziva arhitektura neuronskih mreža. Pojam arhitektura odnosi se na način na koji su povezani različiti neuroni.

Za neuronske mreže funkcija koštanja zapravo je generalizacija funkcije koštanja za logističku regresiju. Tako umjesto jedne izlazne jedinice, imamo njih *K* (s tim da *K* može biti i 1 ako se radi o binarnoj klasifikaciji). Razlika u funkciji koštanja je u sumaciji po izlaznim jedinicama, odnosno imamo funkciju koštanja logističke regresije za svaku izlaznu jedinicu i njih sumiramo.



## *Backpropagation* algoritam

Uz prethodno definirani način koji se odnosi na *forward propagation* algoritam, drugi algoritam za minimizaciju funkcije koštanja naziva se *backpropagation* (BP)algoritam. Odnosi se na to kako se računa gradijent i cilj mu je, ponovo, pronaći parametar za kojeg je minimalan. [17] Za računanje derivacija koristi se *backpropagation* algoritam. Prvenstveno, definiramo pojmove važne za taj postupak:

* – aktivacija jedinice (čvora) *j* u sloju *l*
* za svaki čvor definiramo izraz – „greška“ čvora *j* u sloju *l*, odnosno greška u aktivaciji čvora

Za svaku izlaznu jedinicu definiramo:

,   
Iz ove definicije jasno je da je razlika između onoga što izračuna hipoteza i stvarne vrijednosti u primjeru.

Da bi prikazali rad *backpropagation* algoritma, uzimamo trening set oblika: .

1. Postavi , za sve *i*, *j*, *l*

koristimo za računanje parcijalne derivacije

1. For *i=1* to *m* {

Postavi

Radi *forward propagation* kako bi izračunali za *l=*2, 3,…, *L*

Koristeći , izračunaj

Izračunaj

}

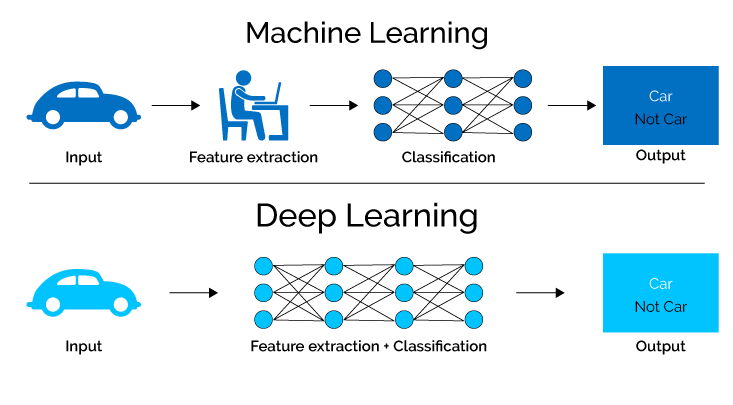
1. Nakon što se završi *for* petlja, računa se sljedeće:

Kada je *j*=0 to odgovara *bias* jedinici pa nema ni regularizacijskog izraza.

Na kraju možemo zaključiti da vrijedi:

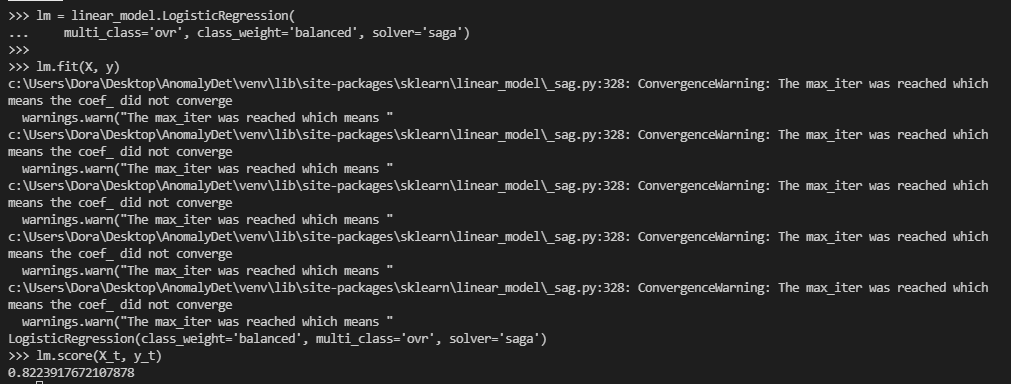
## Deep learning ???????????????????????????????

<https://i2.wp.com/semiengineering.com/wp-content/uploads/2018/01/MLvsDL.png?ssl=1>



# VIŠEKLASNA KLASIFIKACIJA CYBER-NAPADA

First run – logistic regression



Second run - 0.8398687012065295

Third run - 0.8535752306600426

NEURAL

1. The **number of hidden** neurons should be between the size of the input **layer** and the size of the output **layer**.
2. The **number of hidden** neurons should be 2/3 the size of the input **layer**, plus the size of the output **layer**.
3. The **number of hidden** neurons should be less than twice the size of the input **layer**.

# ZAKLJUČAK

# LITERATURA

1. Ethem Alpaydin, „Introduction to Machine Learning, Second Edition“, 2010 Massachusetts Institute of Technology
2. Ethem Alpaydin, „Strojno učenje“, 2016 Massachusetts Institute of Technology, za hrvatsko izdanje 2021 MATE d.o.o Zagreb
3. <https://en.wikipedia.org/wiki/Supervised_learning> (12.6.2021.)
4. <https://www.coursera.org/learn/machine-learning/home/welcome>, by Stanford University (12.6.2021.)
5. <https://www.fer.unizg.hr/_download/repository/SU-2017-02-OsnovniKoncepti.pdf> (12.6.2021.)
6. <https://en.wikipedia.org/wiki/Cluster_analysis> (13.6.2021.)
7. <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/> (13.6.2021.)
8. <https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/> (14.6.2021.)
9. <https://www.fer.unizg.hr/_download/repository/StrojnoUcenje.pdf> (14.6.2021.)
10. <https://en.wikipedia.org/wiki/Semi-supervised_learning> (15.6.2021.)
11. <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/> (15.6.2021.)
12. <https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/> (15.6.2021.)
13. <https://en.wikipedia.org/wiki/Multiclass_classification> (15.6.2021.)
14. <https://en.wikipedia.org/wiki/Artificial_intelligence> (18.6.2021.)
15. <https://ec.europa.eu/croatia/basic/what_is_artificial_intelligence_hr> (18.6.2021.)
16. <https://hr.wikipedia.org/wiki/Neuron> (19.6.2021.)
17. <https://en.wikipedia.org/wiki/Backpropagation> (20.6.2021.)

# POPIS OZNAKA I KRATICA

# SAŽETAK

# SUMMARY