SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 689

**Predviđanje kretanja tržišta dionica neuronskim mrežama**

Dario Pavlović

Zagreb, lipanj 2022.

Ova stranica treba biti:

ILI prazna stranica

ILI stranica sa zahvalom (po želji studenta; zahvala nije obvezna).

Sadržaj

[Uvod 1](#_Toc105080894)

[1. Strojno učenje 2](#_Toc105080895)

[1.1. Motivacija i definicija 2](#_Toc105080896)

[1.2. Vrste strojnog učenja 3](#_Toc105080897)

[1.2.1. Nadzirano učenje 3](#_Toc105080898)

[1.2.2. Nenadzirano učenje 3](#_Toc105080899)

[1.2.3. Podržano učenje 3](#_Toc105080900)

[2. Umjetne neuronske mreže 4](#_Toc105080901)

[2.1. Motivacija i definicija 4](#_Toc105080902)

[2.1.1. Biološki neuron 4](#_Toc105080903)

[2.2. Umjetni neuron 4](#_Toc105080904)

[2.2.1. Aktivacijske funkcije 5](#_Toc105080905)

[2.2.2. Vrste neuronskih mreža 6](#_Toc105080906)

[2.3. Višeslojne mreže bez povratnih veza 6](#_Toc105080907)

[2.4. Učenje 7](#_Toc105080908)

[2.4.1. Unakrsna provjera 7](#_Toc105080909)

[2.4.2. Funkcija gubitka 8](#_Toc105080910)

[2.4.3. Optimizacija neuronske mreže 9](#_Toc105080911)

[2.5. Povratne neuronske mreže 10](#_Toc105080912)

[2.5.1. Definicija 10](#_Toc105080913)

[2.5.2. Vrste povratnih neuronskih mreža 12](#_Toc105080914)

[2.5.3. Problemi običnih rekurentnih neuronskih mreža 12](#_Toc105080915)

[2.5.4. LSTM mreže 13](#_Toc105080916)

[3. Programsko ostvarenje modela za predviđanje 16](#_Toc105080917)

[3.1. Opis problema predviđanja cijena dionica 16](#_Toc105080918)

[3.2. Različite metode pokušaja predviđanja dionica 16](#_Toc105080919)

[3.3. Izgradnja modela neuronske mreže 16](#_Toc105080920)

[3.3.1. Alati i tehnologije 16](#_Toc105080921)

[3.3.2. Pregled podataka cijena dionica 17](#_Toc105080922)

[3.3.3. Pregled bitnih dijelova programske realizacije 17](#_Toc105080923)

[4. Rezultati izgrađenog modela 20](#_Toc105080924)

[Zaključak 21](#_Toc105080925)

[Literatura 22](#_Toc105080926)

[Sažetak 23](#_Toc105080927)

[Summary 24](#_Toc105080928)

[Skraćenice 25](#_Toc105080929)

[Privitak 26](#_Toc105080930)

# Uvod

U današnje vrijeme dosta se priča o razvoju umjetne inteligencije, no što je umjetna inteligencija točno? Sljedeći citat nam može pobliže objasniti šta bi trebala biti umjetna inteligencija.  
„Umjetna inteligencija – naziv za znanstvenu disciplinu koja se bavi izgradnjom računalnih sustava čije se ponašanje može tumačiti kao inteligentno“ - John McCarthy, (1956.). Strojno učenje možemo nazvati primjenom umjetne inteligencije. Cilj nekog modela strojnog učenja je da uči i poboljšava se iz prethodno stečenih znanja i iskustava. Kao što smo strojno učenje nazvali primjenom umjetne inteligencije, tako možemo duboko učenje nazvati primjenom strojnog učenja. Duboko učenje pokušava imitirati način na koji ljudska inteligencija funkcionira da bi stekao određena znanja. Napretkom metoda strojnog učenja, u slučaju ovog završnog rada metoda dubokog učenja, pronalazimo njihove razne primjene u životu, primjerice kod autonomnih automobila, prepoznavanja objekata, prepoznavanja izraza lica, traženje uzorka korisnikovog ponašanja u svrhu plasiranja ciljanih reklami, i tako dalje. Tako će svrha i cilj ovog rada biti traženje uzorka kretanja i pokušaj predviđanja tržišta dionica koristeći model dubokog učenja, preciznije rekurentnu neuronsku mrežu.

# Strojno učenje

## Motivacija i definicija

Strojno učenje definiramo kao primjenu umjetne inteligencije, to jest strojnim učenjem želimo postići da naš sustav preko prethodno stečenih znanja i iskustva odlučuje. Sljedećim citatom možemo dobro predstaviti strojno učenje.

„Ono što je zajedničko mnogim definicijama jest spoznaja da danas živimo u svijetu u kojem smo okruženi obiljem podataka, te da nam je interesantno razvijati programske sustave koji su sposobni iskorištavati te podatke, učiti iz njih i na temelju toga nuditi korisna ponašanja.“ – [1]

Kako je i u citatu navedeno, za provedbu odlučivanja i stjecanja znanja trebamo imati podatke ili iz samog kompleta podataka želimo uočiti uzorak i saznati nešto više o tim podacima. Raspoloživi podaci mogu biti numerički ili kategorički. Primjer numeričkih podataka su godine starosti čovjeka, cijene, broj ljudi na koncertu, dok su primjeri kategoričkih podataka brand odjeće, marka automobila, boja očiju, itd.

Diagram

Description automatically generated

Slika 1.1 Odnos umjetne inteligencije, strojnog učenja i dubokog učenja [13]

## Vrste strojnog učenja

Strojno učenje dijelimo na tri glavna područja, a to su:

* Nadzirano učenje
* Nenadzirano učenje
* Podržano učenje

### Nadzirano učenje

Nadzirano učenje se definira setom ulaznih podataka x koji se preslikavaju na izlaz y, to jest naš skup za učenje je oblika , gdje xi predstavlja podatke o primjerku, a yi predstavlja ciljnu vrijednost koju pridružuje tom primjerku (engl. *target value*). Zapravo tražimo presliku , gdje ako je yi diskretna/nebrojčana vrijednost radimo klasifikaciju, a ako je yi kontinuirana/brojčana vrijednost radimo regresiju. Umjetne neuronske mreže generalno spadaju u ovu vrstu strojnog učenja, iako postoje i nenadzirane vrste neuronskih mreža.

### Nenadzirano učenje

Nenadzirano učenje definiramo samo setom ulaznih podataka, skup učenja je onda oblika , vidimo da naš skup onda ne prima yi, odnosno ciljne vrijednosti uz ulazne. U postupke nenadziranog učenja spadaju postupci grupiranja (engl. *clustering*), postupci smanjena dimenzionalnost (engl. *dimensionality reduction*), postupak otkrivanja stršećih/novih vrijednosti (engl. *outlier/novelty detection*).

### Podržano učenje

Podržano učenje se bavi optimizacijom ponašanja. Odnosno zadaća podržanog učenja je razviti optimalnu strategiju ponašanja na temelju pokušaja s odgođenom nagradom.

# Umjetne neuronske mreže

## Motivacija i definicija

Umjetne neuronske mreže ili skraćeno neuronske mreže, su model dubokog učenja inspiriran biološkim neuronima koji su sastavni dio mozga i živčanog sustava. U sljedećoj usporedbi možemo vidjeti biološku inspiraciju za model umjetnog neurona.

### Biološki neuron

Diagram

Description automatically generated

Slika 2.1 Biološki neuron [6]

Na slici 2.1 se nalazi biološki neuron, koji se sastoji od tijela u kojem se u središtu nalazi jezgra, a na rubovima tijela se nalaze dendriti koji primaju impulse od susjednih neurona. Prema desno se širi akson, na kraju kojeg se nalaze teledendroni sa završnim kvržicama koji šalju impulse dalje u susjedne neurone.

## Umjetni neuron

Inspirirani biološkim modelom, 1943. godine Warren McCulloh i Walter Pitts definiraju matematički model umjetnog neurona prikazanog na slici 2.2.

Diagram

Description automatically generated

Slika 2.2 Model umjetnog neurona (engl. perceptron)

Ulazi x1, x2, …, xn predstavljaju dendrite biološkog neurona, te oni sadrže određene vrijednosti pobude prethodnog neurona. Težine w1, w2, …, wn predstavljaju u kojoj jačini signal primljen kroz ulaz xi utječe na neuron. Uz to postoji i težina koju predstavlja konstantan pomak (engl. *bias*) w0.

|  |  |
| --- | --- |
|  | (2.1) |

Tijelo stanice umjetnog neurona prema svojem izlazu generira sumu označenu sa *net* , te se ona računa prema izrazu 2.1. Konačni izlaz *o* ovisi o aktivacijskoj funkciji *o* = *f* (*net*).

### Aktivacijske funkcije

Prilikom konstrukcije modela neuronske mreže možemo koristiti različite aktivacijske funkcije kao što su to funkcija skoka, funkcija identiteta, sigmoidna funkcija, hiperbolički tangens, *ReLU* funkcija. Kao primjer aktivacijske funkcije možemo uzeti *funkciju skoka* , neuron koji onda dobivamo poznat je kao *TLU-perceptron* , te funkciju onda definiramo kao

|  |  |
| --- | --- |
|  | (2.2) |

Aktivacijska funkcija koju većinski koristimo u rekurentnim neuronskim mrežama je hiperbolička tangens funkcija (*tanh*):

|  |  |
| --- | --- |
|  | (2.3) |

, *tanh* aktivacijska funkcija vraća vrijednosti između -1 i 1. Osim *tanh* funkcije, LSTM model interno koristi i *sigmoidnu* funkciju oblika:

|  |  |
| --- | --- |
|  | (2.4) |

, *σ* funkcija vraća vrijednosti između 0 i 1. Aktivacijske funkcije *tanh*, *σ* i *ReLU* će nam biti bitne kod modela povratnih neuronskih mreža .

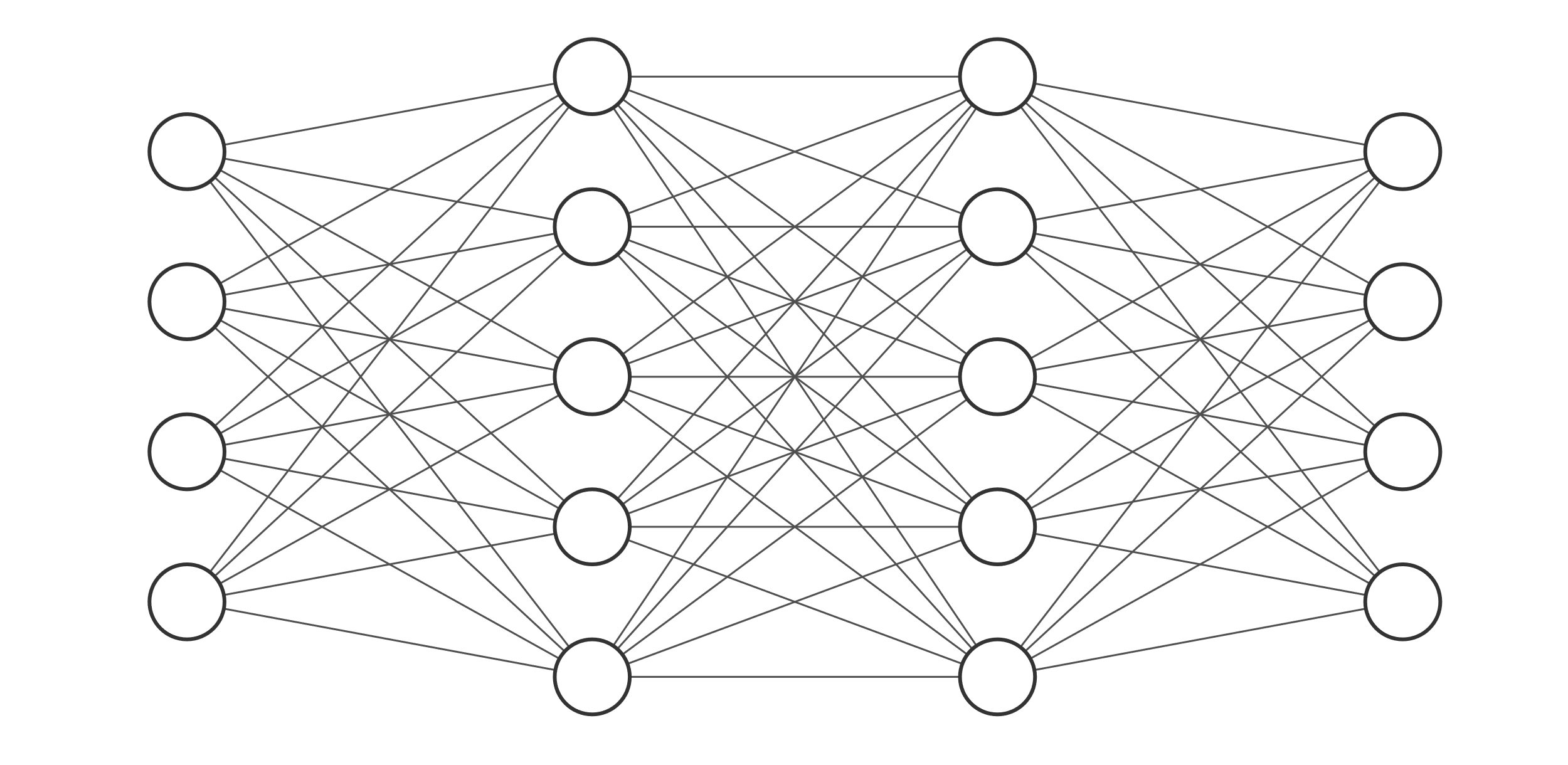
### Vrste neuronskih mreža

Iako neuronske mreže možemo podijelite na dosta načina, ugrubo ih dijelimo na:

* Jednoslojne mreže bez povratnih veza (engl. *single-layer feedforward networks*)
* Višeslojne mreže bez povratnih veza (engl. *multi-layer feedforward networks*)
* Rekurentne neuronske mreže (engl. *recurrent neural networks - RNN*)
* Konvolucijske neuronske mreže (engl. *convolutional neural networks - CNN*)

## Višeslojne mreže bez povratnih veza

Nakon definicije umjetnog neurona, samim time i jednoslojne mreže bez povratnih veza, možemo pričati o višeslojnom modelu umjetnih neuronskih mreža bez povratnih veza čija su oni glavna sastavnica.



Slika 2.3 Višeslojna neuronska mreža

Na slici 2.3 se nalazi jedna neuronska mreža sa četiri sloja. Ulazni sloj čine četiri neurona i kroz njih mreži predajemo ulazne podatke. Zatim ga slijede dva skrivena sloja, svaki po pet neurona. Te kao zadnji sloj imamo izlazni sloj sa četiri neurona iz kojih izvlačimo rezultate.  
Između svakog sloja mreže imamo težine s kojima su čvorovi povezani s onima iz prethodnog i sljedećeg sloja. Skriveni slojevi nisu vidljivi korisniku, jedini koji su u interakciji sa korisnikom su ulazni i izlazni.

Kao ulaz, neuronskoj mreži predočavamo podatke iz naših skupova podataka, te iz njih pokušavamo dobiti izlaze koji će odgovarati ciljanoj regresiji ili klasifikaciji.

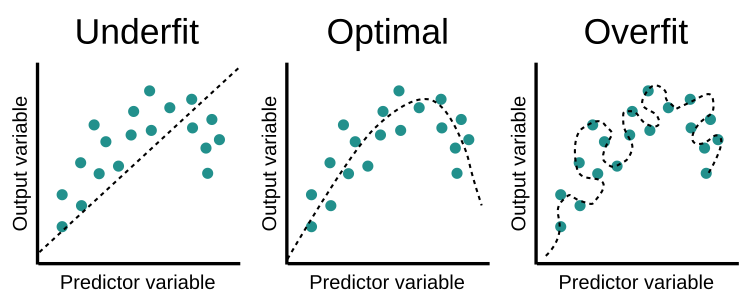
Generalno rad s umjetnim neuronskih mrežama dijelimo u dvije faze: fazu učenja i fazu iskorištavanja. U fazi učenja neuronskoj mreži predočavamo uzorke iz skupa za učenje, gdje je jedna *iteracija* predočavanje jednog uzorka skupa, a jedna *epoha* predočavanje čitavog skupa uzoraka. Uslijed učenja dolazi do promjena u jakosti veza između neurona, time se mreža onda prilagođava viđenim podacima, zatim ju u fazi iskorištavanja koristimo.

## Učenje

Cilj faze učenja neuronske mreže je postići najbolju raspodjelu težina između neurona. Kao što je i navedeno, postoji više vrsti učenja neuronske mreže, dok ćemo u implementaciji koristiti nadzirano učenje. Ulaz nadziranog učenja smo prethodno definirali kao par (xi, yi), gdje xi predstavlja podatke o primjerku, a yi predstavlja ciljnu vrijednost koju pridružuje tom primjerku.

### Unakrsna provjera

Unakrsna provjera (engl. *cross-validation*) je način pristupa učenju modela strojnog učenja. Prilikom učenja modelu predajemo skup podataka koji nazivamo skupom za učenje (engl. *train set*), te pomoću tog skupa, sa funkcijom gubitka i optimizacijom neuronske mreže prilagođavamo težine između čvorova neuronske mreže prema zadanom skupu. Osim skupa za učenje, imamo skup za provjeru (engl. *test set*) koji nam služi za provjeru koliko dobro naš model radi. Za veći set podatak generalno uzimamo omjer 80% podataka za skup za učenje, a 20% podataka kao skup za provjeru. Za manji set podataka bolja podjela bi bila 70/30 ili 60/40. Važnost unakrsne provjere je ta da izbjegnemo prekomjerno prilagođavanje modela (engl. *overfitting*). Ako dođe do prekomjernog prilagođavanja našeg modela onda on loše generalizira, to jest na još neviđenim podacima će predviđati loše.



Slika 2.4 Nedovoljno, optimalno i prekomjerno prilagođavanja regresijske funkcije [2]

Osim problema prekomjernog prilagođavanja možemo i susresti problem prevelike generalizacije modela. (TODO objasnit preveliku generalizaciju ako potrebno)

### Funkcija gubitka

Prilikom učenja neuronske mreže, u svakoj iteraciji izlaz uspoređujemo sa ciljnim izlazom, te kako bismo „ocijenili“ koliko dobro mreža procjenjuje ciljnu vrijednost *y*, koristimo funkciju gubitka (engl. *loss function*) označenu sa *L*. Za odabir funkcije gubitka imamo više različitih opcija poput:

* Regresijske funkcije gubitka
* Srednja kvadratna pogreška (engl. *Mean squared error loss*)
* Srednja kvadratna logaritamska pogreška (engl. *Mean squared logarithmic error loss*)
* Srednja apsolutna pogreška (engl. *Mean absolute error loss*)
* Binarne klasifikacijske funkcije gubitka
* Gubitak zglobnice (engl. *Hinge loss*)
* Kvadrirani gubitak zglobnice (engl. *Squared hinge loss*)
* Binarna unakrsna entropija (engl. *Binary cross-entropy*)
* Višeklasne klasifikacije funkcije gubitka

Pošto se u ovom radu implementirani model bavi regresijom i kontinuiranim vrijednostima cijena dionica, koristimo funkciju srednje kvadratne pogreške (engl. *mean squared error - MSE*).

|  |  |
| --- | --- |
|  | (2.5) |

Formula (2.5) prikazuje izračun standardne pogreške kvadrata, gdje su vrijednosti koje nam daje neuronska mreža, a ciljne vrijednosti.

Nakon izračuna pogreške u svakoj iteraciji, cilj nam je prilagoditi neuronsku mrežu tako što mijenjamo težine veza između umjetnih neurona. Ovaj postupak nazivamo algoritmom propagacije pogreške unatrag (engl. *backpropagation algorithm*), odnosno postupak optimizacije neuronske mreže tako da minimiziramo funkciju gubitka.

### Optimizacija neuronske mreže

Prilikom optimizacije mreže, odnosno smanjenja funkcije gubitka, imamo dvije mogućnosti optimizacije:

* Računamo gradijent na temelju svih primjera i zatim korigiramo težine veza
* Za određeni broj primjera (engl. *batch size*) računamo gradijent i odmah korigiramo težine veza

U implementaciji našega modela koristimo drugu navedenu metodu, odnosno za određeni broj primjera računamo prilagođeni gradijent i odmah korigiramo.

Za optimizaciju mreže postoje različiti algoritmi kao što su:

* Standardni gradijentni spust (engl. *gradient descent*)
* Stohastički gradijentni spust (engl. *stohastic gradient descent*)
* Adaptivni gradijent (engl. *adaptive gradient – AdaGrad*)
* AdaDelta
* RMSprop
* Adam

Prema provedenim eksperimentima [3], *Adam* nadmašuje ostale algoritme. Iako postoje argumenti da standardni gradijentni spust bolje generalizira [4], generalno najbolji za našu implementaciju je adaptivni optimizacijski algoritam *Adam* zbog najbrže konvergencije, te je memorijski efektivan za veliki set podataka, zbog toga ćemo i prilikom implementacije modela koristiti njega. Izgled formule za standardni gradijentni spust je sljedeći:

|  |  |
| --- | --- |
|  | (2.6) |

, gdje predstavlja *stopu učenja*, a predstavlja gradijent funkcije u .

Iako ga koristimo kasnije u samoj implementaciji, matematičku pozadinu *Adam* optimizacijskog algoritma nećemo navoditi u ovom radu zbog njene kompleksnosti. Za više o *Adam* optimizacijskom algoritmu i njegovoj matematičkojpozadini pogledati [3].

## Povratne neuronske mreže

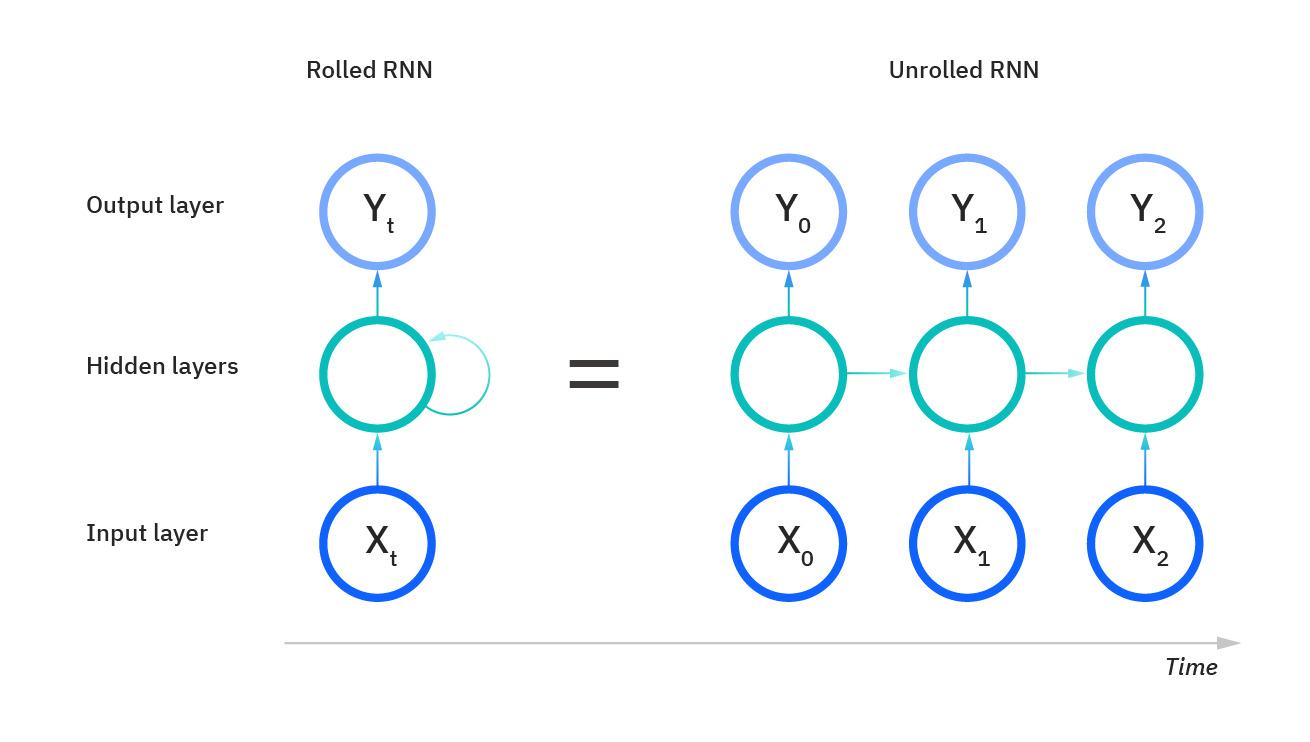
### Definicija

Povratne ili rekurentne neuronske mreže su vrsta umjetnih neuronskih mreža koje sadrže povratne veze između neurona, što im omogućava privremeno dinamičko ponašanje. Povratne neuronske mreže inspiraciju dobivaju iz dinamičkih sustava. Koristimo ih jer su idealne kod modeliranja slijednih podataka, kao što su to naprimjer tekst, zvuk, cijene dionica itd.

Background pattern

Description automatically generated

Slika 2.5 Rekurentna neuronska mreža



Slika 2.6 „Spakirana“ i „razmotana“ rekurentna neuronska mreža [5]

Osim slike 2.5, rekurentne neuronske mreže možemo prikazati kao i na slici 2.6, u njihovom „spakiranom“ stanju i „razmotanom“ stanju koji prikazuje neuronsku mrežu kroz vremenske korake *t*. Na slici 2.6 plavim čvorovima su označeni ulazi xt, zelenim čvorovima skriveni slojevi ht, te svjetloplave boje izlazi yt.

Definirajmo kao vektor skrivenog sloja u vremenskom koraku (engl. *timestep*) *t*,  
 vrijednosti onda računamo idućom formulom:

|  |  |
| --- | --- |
|  | (2.7) |

Iz formule vidimo da stanje ovisi o vrijednost stanja u prethodnom koraku i o trenutačnoj vrijednosti vektora ulaza . U jednostavnoj povratnoj neuronskoj mreži funkcija *f* je jednoslojna neuronska mreža.

|  |  |
| --- | --- |
|  | (2.8) |
|  | (2.9) |

Vrijednost *net* sloja računamo prema formuli 2.8, gdje su i pripadajuće matrice težina između neurona, a je vektor pristranosti, odnosno konstantni pomak. Za računanje projekcije u izlazni sloj koristimo sljedeću formulu:

|  |  |
| --- | --- |
|  | (2.10) |

Kod povratnih mreža za funkciju *g* uzimamo aktivacijske funkcije koje su prethodno definirane u poglavlju *2.2.1. Aktivacijske funkcije*, a to su *tanh* (2.3),(2.4) i ReLU koju jednostavno definiramo kao .

### Vrste povratnih neuronskih mreža

Box and whisker chart

Description automatically generated

Slika 2.7 Vrste povratnih mreža [7]

Na slici 2.7 pravokutnici predstavljaju vektore, a strelice predstavljaju funkcije (npr. matrično množenje). Ulazni vektori su crvene boje, izlazni plave boje, a zeleni predstavljaju RNN stanja. Slika označena sa **(1)** nam predstavlja fiksiran ulaz i izlaz bez povratnih veza, kao primjer problema koji u to spada je klasifikacija slika. Nadalje slika **(2)** prikazuje slijedni izlaz, u što spada opis slike riječima. Brojem **(3)** označavamo slijedni ulaz i fiksiran izlaz, npr. riječ moramo klasificirat kao „pozitivna“ ili „negativna“. Na slici **(4)** imamo slijedni ulaz i izlaz, npr. prijevod rečenice iz jednog jezika u drugi. I na zadnjoj slici **(5)** imamo usklađeni slijedni ulaz i izlaz, npr. klasifikacija svake sličice u videu. U svakom od navedenih slučaja uviđamo da nema ograničenja na količinu zelenih pravokutnika koji predstavljaju povratnu transformaciju. Detaljnije u [7].

### Problemi običnih rekurentnih neuronskih mreža

U nedostatke i probleme povratnih neuronskih mreža spada zahtjevno i sporo računanje, učenje može biti kompleksno, mreža ne uzima buduće ulaze u svrhu odlučivanja, itd. No glavni problem običnih povratnih mreža je da pati od problema kratke memorije, ako je slijedni ulaz dug, onda će mreža imati problem „prisjećanja“ ranijih ulaznih podataka. Konkretnije, za vrijeme propagacije pogreške unazad , dolazi do problema eksplodirajućeg i nestajućeg gradijent.

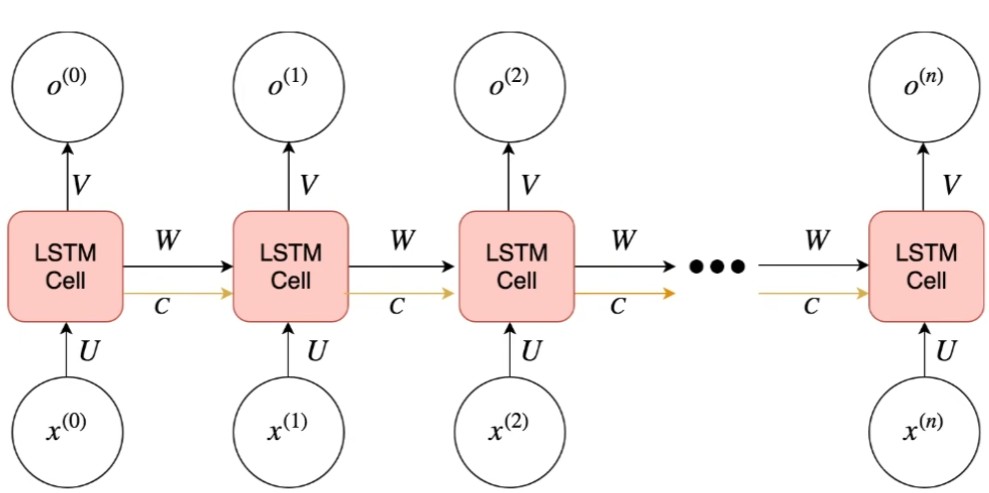
|  |  |
| --- | --- |
|  | (2.11) |
|  | (2.12) |

Problem nestajućeg gradijenta (2.11) je taj da kada želimo optimizirati mrežu nakon nekog ulaza i izračuna pogreške, gradijent između dva uzastopna skrivena stanje se eksponencijalno brzo smanji u vrijednosti i tako ne prilagodi ostatak mreže, tako onda kod tog dijela mreže ne dođe do prilagođavanja težina. Kod problema eksplodirajućeg gradijenta (2.12) naš gradijent eksponencijalno postane ogroman ili čak toliko velik da se ne može zapisati u memoriji računala (*NaN*). Više o problemima povratnih mreža i o matematičkoj pozadini problema nestajućeg i eksplodirajućeg gradijenta u [8].

Kao rješenje problema s kratkom memorijom i nestajućim/eksplodirajućim gradijentom koristimo podvrste povratnih mreža s propusnicama (engl. *gated recurrent neural networks*), kao što su *Long Short-Term Memory* (LSTM) i *Gated Recurrent Units* (GRU). Pošto u implementaciji koristimo LSTM model, on će biti detaljnije objašnjen u nastavku. Za detaljnije o povratnim mrežama s propusnicama pogledati [9].

### LSTM mreže

LSTM ili *Long Short-Term Memory* spada u podvrstu povratnih neuronskih mreža s propusnicama. Koristi se za prepoznavanje slijeda riječi, opis slike, video-tekst konverziju i ostale slične probleme. LSTM je napravljen u svrhu rješavanja problema kratkotrajne memorije koju obična povratna neuronska mreža ima. Da bismo bolje razumjeli kako se LSTM „bori“ protiv kratkotrajne memorije, to jest nestajućeg gradijenta, u nastavku objašnjavamo njegovu strukturu.



Slika 2.8 Prikaz povratne neuronske mreže sa LSTM ćelijama u skrivenom sloju

Na slici 2.8 vidimo izgled povratne neuronske mreže sa LSTM ćelijama, uz obične slojeve i već definirane parametre uočavamo i dodatnu povezanost između ćelija označenu sa *C* koja označava stanje prethodne ćelije.

Diagram

Description automatically generated

Slika 2.9 Interna struktura LSTM ćelije (napomena: ulazi i izlazi W i C na slici 2.8 su zamijenjeni u odnosu na ovu sliku) [10]

LSTM ćelija sadrži tri propusnice. Ulaznu propusnicu (engl. *input gate*), označenu sa na slici 2.9, koja provjerava je li ćelija ažurirana, računamo ju formulom:

|  |  |
| --- | --- |
|  | (2.13) |

Zatim propusnicu zaborava (engl. *forget gate*), označenu sa na slici, koja provjerava je li memorija postavljena na nula, nju računamo:

|  |  |
| --- | --- |
|  | (2.14) |

Te još definiramo i izlaznu propusnicu (engl. *output gate*), označenu sa na slici, koja kontrolira je li informacija trenutne ćelije vidljiva, njena formula je:

|  |  |
| --- | --- |
|  | (2.15) |

Uočavamo da sve funkcije za svoju aktivacijsku funkciju koriste *sigmoidnu* funkciju, nju koristimo iz razloga da naše funkcije sačinjavaju „glatke“ krivulje u rangu od 0 do 1, time model ostaje diferencijabilan. Osim navedenih funkcija propusnica imamo i vektor koji ažurira stanje ćelije, njega računamo kao:

|  |  |
| --- | --- |
|  | (2.16) |

Ovdje pak koristimo *tanh* aktivacijsku funkciju, zbog njenog ranga koji je centriran oko 0, odnosno ide od -1 do 1, ona distribuira gradijente poprilično dobro, što dozvoljava da informacija o stanja ćelije traju dulje bez nestajanja ili eksplodiranja u vrijednosti. Vrijednost stanja modeliramo formulom:

|  |  |
| --- | --- |
|  | (2.17) |

Direktno iz formule 2.17 vidimo da stanje ovisi o *input gate* funkciji koju primjenjujemo na , jer ona odlučuje hoćemo li ažurirati ćeliju. Osim toga, stanje ovisi i o *forget gate* funkciji koju primjenjujemo na prethodno stanje , jer ona određuje koliko prošlog stanja zaboravljamo. Konačnu definiramo formulu s kojom računamo vrijednost skrivenog vektora :

|  |  |
| --- | --- |
|  | (2.18) |

Za dobivanje vrijednosti , na vrijednost stanja ćelije primjenjujemo *output gate* funkciju .

Za još detaljnije shvaćanje svakog koraka i matematičkih definicija LSTM ćelije pogledati [7], [11] i [12].

# Programsko ostvarenje modela za predviđanje

## Opis problema predviđanja cijena dionica

Predviđanje kretanja tržišta dionica je pokušaj određivanja budućih cijena dionica. Jedan od razloga zašto predviđamo može biti ostvarivanje profita prilikom kupovanja i prodaje dionica. Usprkos volatilnosti, cijene dionica nisu nasumične vrijednosti, odnosno iz njih možemo pokušati uočiti uzorak. *Učinkovita hipoteza na tržištu* (engl. *efficient-market hypothesis*) je bitan ekonomski pojam za naglasiti. On tvrdi da trenutna vrijednosti imovine na tržištu odražava sve raspoložive podatke (povijesne, javne i privatne). Odnosno teorija smatra da svaka buduća vijest ili događaj koji može utjecati na cijenu imovine će prouzročiti prilagodbu cijene tako brzo da je nemoguće dobiti ekonomsku korist od nje. [14]

## Različite metode pokušaja predviđanja dionica

Kao najuspješnije metode za predviđanje dionica, koristeći samo podatke o cijenama, su se pokazale neuronske mreže i metode potpornih vektora (engl. *Support-vector machine*). Drugi način na koji bi mogli predviđati cijene su da gledamo neke druge parametre i uvjete, te zbog njih očekujemo određeni obrazac ponašanja. Možemo navesti četiri primjera analitičkog ili vanjsko faktorskog predviđanja:

* + - * Momentum
      * Povrat srednje vrijednosti (engl. *Mean Inversion*)
      * *Martingale*
      * Potraga za vrijednošću

Momentum je kada očekujemo da će cijene pratiti trend trenutačnog gibanja bio to porast ili pad, pa sukladno tome reagiramo na tržište ulaganjem ili prodavanjem. Povrat srednje vrijednosti je teorija koja sugerira da kroz vrijeme vrijednost neke imovine dugoročno teži srednjoj vrijednosti skupa ukupnih vrijednosti. Martingale je investicijska strategija koja tvrdi da svaki put kada izgubimo „okladu“ udvostručujemo uloženi iznos, te time nastojimo vratiti izgubljeni kapital. Potraga za vrijednošću znači da pokušavamo što ranije pronaći dionice nečega u čemu vidimo kvalitetu dok su još jeftine, te ulažemo u njih jer znamo da će postati skuplje. Više o ovim metodama u [15].

## Izgradnja modela neuronske mreže

### Alati i tehnologije

Realizaciju našega modela za predviđanje ostvarujemo u programskom jeziku *Python* zbog raznih biblioteka i programske podrške koja nam olakšava pri implementaciji kompleksnih modela strojnog učenja.

Pythonovebiblioteke koje se koriste u radu su:

* + - * *numpy*
      * *pandas*
      * *sklearn*
      * *tensorflow (keras)*

Biblioteku *numpy* koristimo u svrhu pretvaranja neobrađenih podataka u vektore ulaza i izlaza u naš model neuronske mreže. *Pandas* koristimo za pretvaranje podataka u *dataframeove* i za vađenje podataka o stanju cijena dionica, u našem slučaju ćemo ih izvlačiti preko *yahoo finance* API. *Sklearn* ili *scikit-learn* je besplatna pythonova biblioteka strojnog učenja, konkretnije mi ćemo koristiti njenu funkciju skaliranja (engl. *MinMaxScaler*) potrebnu za skaliranja ulaznih podataka prije unosa u neuronsku mrežu. Još koristimo i *tensorflow* biblioteku koja nam je najbitnija za izgradnju našega modela, konkretnije *keras* ima implementirane metode i objekte koje iskorištavamo da bismo složili našu željenu neuronsku mrežu.

### Pregled podataka cijena dionica

Kao što je i prethodno navedeno, podatke o cijenama dionica uzimamo iz *yahoo finance* API-ja koristeći biblioteku *pandas*. Izgled podataka o dionica vidimo na slici 3.1.

A picture containing calendar

Description automatically generated

Slika 3.1 Izgled učitanih podataka

*Date* označava datum (subotom i nedjeljom tržište dionica je zatvoreno, pa tih datuma nemamo). *High* označava najveću vrijednost dionice u tom danu, *Low* označava najmanju vrijednost u dan. *Open* označava vrijednost dionice nakon otvaranja tržišta, *Close* označava zadnju vrijednost prije zatvaranja tržišta. *Volume* označava količinu razmijenjenih dionica u tom danu. *Adj* *Close* je *Close*, ali prilagođen tako da uzimamo dodatne faktore u račun kao što su dividendi, podjelu dionica (engl. *stock split*), itd. Iako nema velike razlike između vrijednosti *Close* i *Adj Close*, u implementaciji koristimo *Adj Close*.

### Pregled bitnih dijelova programske realizacije

(1)data = pdr.DataReader(company, 'yahoo', start=start\_date, end=end\_date)

(2)scaler = MinMaxScaler(feature\_range=(0,1))

(3)scaled\_data = scaler.fit\_transform(data['AdjClose'].values.reshape(-1, 1))

(4)for i in range(timestep, len(scaled\_data)):

(5) x\_train.append(scaled\_data[i-timestep:i, 0])

(6) y\_train.append(scaled\_data[i, 0])

(7)x\_train, y\_train = np.array(x\_train), np.array(y\_train)

(8)x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

Kod 3.1 Dobavljanje i pripremanje podataka za treniranje

Redom objašnjavamo kod 3.1: (1) Učitavamo podatke s *yahoo finance* API, (2)(3) Skaliramo podatke u vrijednosti između 0 i 1 (glavni razlog je taj što nam mreža onda brže konvergira), (4)(5)(6) ovisno o *timestepu* punimo ulazne i odgovarajuće ciljne vrijednosti u varijable *x\_train* i *y\_train­*, (7)(8) pretvaramo ulazne i ciljne vrijednosti u vektore.

(1)model = Sequential([

(2)LSTM(units=50, return\_sequences=True, input\_shape=(x\_train.shape[1], 1)),

(3)Dropout(0.2),

(4)LSTM(units=50, return\_sequences=True),

(5)Dropout(0.2),

(6)LSTM(units=50),

(7)Dropout(0.2),

(8)Dense(units=1)

])

(9)model.compile(optimizer='adam', loss='mean\_squared\_error')

(10)model.fit(x\_train, y\_train, epochs=epochs, batch\_size=batch\_size)

Kod 3.2 Izgradnja neuronske mreže i treniranje

Prilikom realizacije modela (1) koristimo *Sequential* klasu koja nam dozvoljava kreiranje neuronske mreže po slojevima. U liniji (2) definiramo LSTM sloj kojemu pridodajemo izgled ulaznog vektora preko parametra *input­\_shape*. *Units* parametar određuje dimenzije izlaza. Boolean parametar *return­\_sequence* određuje vraćamo li zadnji izlaz u izlaznom slijedu ili pak cijeli slijed. Kao zadnji izlazni sloj (8) koristimo obični NN sloj veličine 1 koji izbacuje vrijednost predviđanja. Između „gradivnih“ slojeva naše mreže koristimo i *Dropout* slojeve (3)(5)(7) koji su još jedan način obrane od prekomjernog prilagođavanja naše mreže, tako što po određenoj stopi (u našem slučaju 20%) nasumično onemogućava neurone i njima odgovarajuće veze. Na liniji (9) konfiguriramo koji optimizacijski algoritam (*optimizer*) i koju funkciju gubitka (­*loss*) koristiti prilikom učenja mreže, detaljnije o optimizacijskom algoritmu u poglavlju *2.4.3 Optimizacija neuronske mreže*, a detaljnije o funkciji gubitka u poglavlju *2.4.2 Funkcija gubitka*. Metoda *fit* pokreće učenje našega modela mreže, u koji predajemo ulazne vrijednosti, ciljne vrijednosti, *epochs* i *batch\_size*. *Epochs* parametar određuje koliko puta ćemo provesti naš set ulaznih podataka ponovo kroz model za vrijeme učenja. *Batch­\_size* predstavlja broj primjeraka nakon kojih će gradijent ažurirati mrežu.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 3.2 Sažetak našeg modela (model.summary())

Osim metode sažetka modela na slici 3.2, koristimo *keras* biblioteku *vis­\_utils* za dobivanje izgleda našega modela kao na slici 3.3.

Diagram

Description automatically generated

Slika 3.3 Izgled slojeva našega modela

Nakon učenja, spremni smo unositi naše vrijednosti skupa za provjeru. Skup za provjeru učitavamo na isti način kao i skup za test u kodu 3.1, no u ovom slučaju nam ne trebaju ciljne vrijednosti jer njih pokušavamo sami dobiti.

(1)predicted\_prices = model.predict(x\_test)

(2)predicted\_prices = scaler.inverse\_transform(predicted\_prices)

Kod 3.3 Dobivanje predviđenih cijena iz modela

*Predict* metodi (1) predajemo podatke, a model nam vraća neskalirane vrijednosti. U liniji (2) nazad skaliramo vrijednosti u one prave i spremni smo za prikazivanje rezultata.

// TODO AKO JE POTREBNO OBJ KOD ISPISIVANJA GRAFOVA

# Rezultati izgrađenog modela

U rezultatima, generalno za skup učenja koristimo vrijednosti dionica određene tvrtke u razdoblju od 1.1.2012 do 1.1.2020, dok za skup za provjeru koristimo vrijednosti od 1.1.2020 do trenutnog datuma pokretanja programa.

Parametri koji variraju u sljedećim rezultatima su *epoch* (određuje koliko puta ćemo provesti naš set ulaznih podataka ponovo kroz model za vrijeme učenja), *timestep* (koliko dana unazad gledamo da bi predvidjeli idući dan) i *batch\_size* (predstavlja broj primjeraka nakon kojih će gradijent ažurirati mrežu).

# Zaključak

# Literatura

1. Čupić Marko, *Uvod u strojno učenje*, Zagreb, 2020
2. Poveznica: <https://www.educative.io/edpresso/overfitting-and-underfitting>; pristupano 15.5.2022
3. Kingma, D. P., Ba, J., *Adam: A method for stochastic optimization*., 2014
4. Hardt, M., Recht, B., Singer, Y., *Train faster, generalize better: Stability of stochastic gradient descent*, International Conference on Machine Learning, 2016
5. Poveznica: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>; pristupano 20.5.2022
6. Čupić Marko, *Umjetne neuronske mreže*, Zagreb, 2016
7. Andrej Karptahy, poveznica: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>; pristupano 22.5.2022
8. Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, *On the difficulty of training Recurrent Neural Networks*, 2012
9. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, 2014
10. Le, Xuan Hien & Ho, Hung & Lee, Giha & Jung, Sungho, *Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting*, 2019
11. Poveznica: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>; pristupano 24.5.2022
12. Poveznica: <https://jaketae.github.io/study/dissecting-lstm/>; pristupano 24.5.2022
13. Poveznica: <https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960>; pristupano 28.5.2022
14. Poveznica: <https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp>; pristupano 28.5.2022
15. Poveznica: <https://www.investopedia.com/articles/07/mean_reversion_martingale.asp>; pristupano 28.5.2022

# Sažetak

Naslov, sažetak, ključne riječi (na hrvatskom jeziku)

Sažetak opisuje sadržaj rada, prepričan u stotinjak riječi.

# Summary

Title, summary, keywords (na engleskom jeziku)

# Skraćenice

LSTM *Long short-term memory* naziv podvrste povratnih mreža

RNN *Recurrent neural networks* povratne neuronske mreže

CNN *Convolutional neural networks* konvolucijske neuronske mreže

SVM *Support-vector machine* metode potpornih vektora

# Privitak