Neuronske mreže i predikcija cijena dionica

Žuro, Matko

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike

Studij

Neuronske mreže i predikcija cijena dionica

Diplomski rad

Mentor:
Danijel Grahovac

Student:
Matko Žuro

Osijek, 2024

Sadržaj

1 | Uvod u duboko učenje

Duboko učenje, kao napredni dio strojnog učenja, omogućuje računalima da uče iz iskustva i razumiju svijet na način sličan ljudima. Umjesto da se oslanja na ručno programiranje pravila i značajki, duboko učenje omogućuje modelima da sami otkrivaju relevantne obrasce i informacije iz velikih količina podataka. Osnovni elementi dubokog učenja uključuju umjetne neurone, koji oponašaju funkcije bioloških neurona. Ti neuroni su organizirani u slojeve unutar mreže, gdje svaki sloj obrađuje podatke na sve složenijoj razini. Primjerice, tradicionalni algoritmi strojnog učenja zahtijevali bi ručno određivanje značajki poput veličine i boje prilikom razvrstavanja objekata, dok modeli dubokog učenja koriste neuronske mreže koje automatski prepoznaju te značajke iz podataka, učeći iz primjera [4].

Duboko učenje je srž mnogih napredaka u umjetnoj inteligenciji (AI), s primjenama u područjima poput prepoznavanja govora, računalnog vida, obrade prirodnog jezika i autonomnih vozila [9]. Ključna prednost dubokog učenja leži u sposobnosti automatskog učenja složenih reprezentacija podataka kroz hijerarhiju slojeva Na primjer, u modelima za prepoznavanje lica, prvi sloj prepoznaje rubove, drugi sloj kombinira te rubove u oblike, a treći sloj prepoznaje specifične crte lica poput očiju, nosa i usta. Perceptron, razvijen 1962. godine, bio je jedan od prvih modela umjetnog neurona. Međutim, prave prekretnice u razvoju dubokog učenja dogodile su se 1980-ih s uvođenjem algoritama poput povratne propagacije, koji su omogućili efikasnije treniranje višeslojnih perceptrona. Ovi napredci doveli su do razvoja složenijih i učinkovitijih dubokih modela, koji su rezultirali značajnim uspjesima u mnogim područjima [4].

Iako duboko učenje donosi veliki napredak, suočava se s nekoliko značajnih izazova. Prvo, potreba za velikim količinama označenih podataka može biti ograničavajući faktor jer takvi podaci često nisu lako dostupni. Drugo, treniranje dubokih mreža zahtijeva veliku računalnu snagu, što može biti skupo i energetski zahtjevno. Treće, duboki modeli često djeluju kao „crne kutije“, što otežava interpretaciju i razumijevanje njihovih odluka. To može biti posebno problematično u kritičnim aplikacijama, poput medicinske dijagnostike, gdje je ključno razumjeti razloge iza određenih odluka. Unatoč tim izazovima, duboko učenje rješava probleme koji su previše složeni za tradicionalne metode strojnog učenja. Njegova sposobnost analize velikih količina podataka i otkrivanja skrivenih obrazaca čini ga izuzetno moćnim alatom [9].

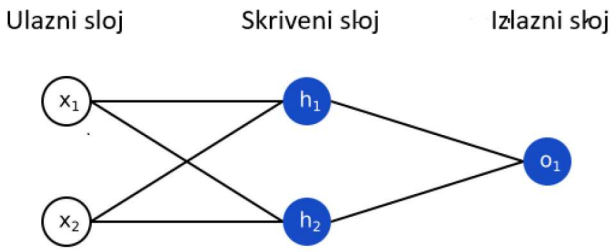
U financijama, duboko učenje ima značajnu primjenu, posebno u predikciji ci-

jena dionica. Modeli poput rekurentnih neuronskih mreža i njihovih varijanti, poput LSTM (Long Short-Term Memory) mreža, omogućuju analizu vremenskih nizova i predikciju budućih cijena dionica na temelju povijesnih podataka [7]. Time se investitorima i analitičarima omogućuje donošenje informiranih odluka i optimizacija strategija ulaganja.

Nakon ovog uvoda u osnove i značaj dubokog učenja, u sljedećem poglavlju ćemo detaljnije istražiti neuronske mreže, temeljnu komponentu dubokih modela, koje omogućuju prepoznavanje složenih obrazaca u podacima.

2 | Neuronske mreže

Jednostavne neuronske mreže sastoje se od ulaznog sloja, jednog ili više skrivenih slojeva i izlaznog sloja. Ulazni sloj sadrži neurona koliko ima i prediktora, skriveni sloj ima proizvoljan broj neurona, a izlazni sloj sadrži jedan neuron za regresijske probleme, a za klasifikaciju sadrži onoliko koliko ima kategorija izlazne varijable. Informacije se procesiraju u neuronima te se prenose kroz slojeve s pomoću veza koje se aktiviraju pomoću tzv. aktivacijske funkcije. Aktivacijska funkcija odlučuje hoće li se neuron aktivirati i prenijeti signal dalje. Tijekom faze učenja, mreža se trenira s pomoću skupa ulaznih podataka, a izlaz se uspoređuje s očekivanim izlazom. Težine i pomaci veza između neurona prilagođavaju se na temelju pogreške između očekivanog i stvarnog izlaza, koristeći algoritam nazvan povratna propagacija. Cilj je minimizirati pogrešku i omogućiti mreži da uči na temelju novih podataka [4].

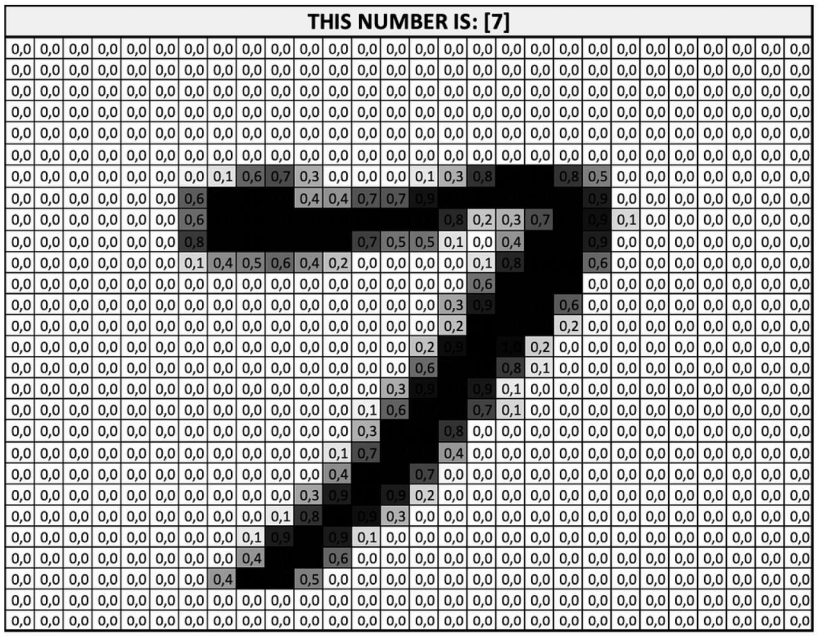


Slika 2.1: Prikaz jednostavne neuronske mreže. Izvor: [8].

Primjer prepoznavanja ručno napisanih brojeva može pomoći u razumijevanju osnovnih funkcija neuronskih mreža. Važno je napomenuti da sljedeći primjer služi kao pojednostavljena ilustracija za razumijevanje osnovnih funkcija neuronskih mreža. U stvarnosti, proces prepoznavanja obrazaca unutar neuronskih mreža može biti mnogo složeniji i uključivati različite dodatne korake i faktore [3].

Primjer 1. *Svaki broj se predstavlja kao matrica piksela, obično veličine 28x28. Neuronska mreža procesira ulaznu sliku kroz nekoliko slojeva, gdje prvi sloj detektira osnovne karakteristike poput rubova, dok kasniji slojevi prepoznaju složenije značajke kao što su oblici i specifične krivulje. Konačni sloj daje izlaz koji predstavlja vjerojatnost pripadnosti broju od 0 do 9. Ovaj proces ilustrira kako mreže mogu izvući složene obrasce iz*

sirovih podataka, prelazeći od jednostavnih vizualnih elemenata do prepoznavanje specifičnih oblika.



Slika 2.2: Prikaz broja pomoću 28×28 piksela. Izvor: [3].

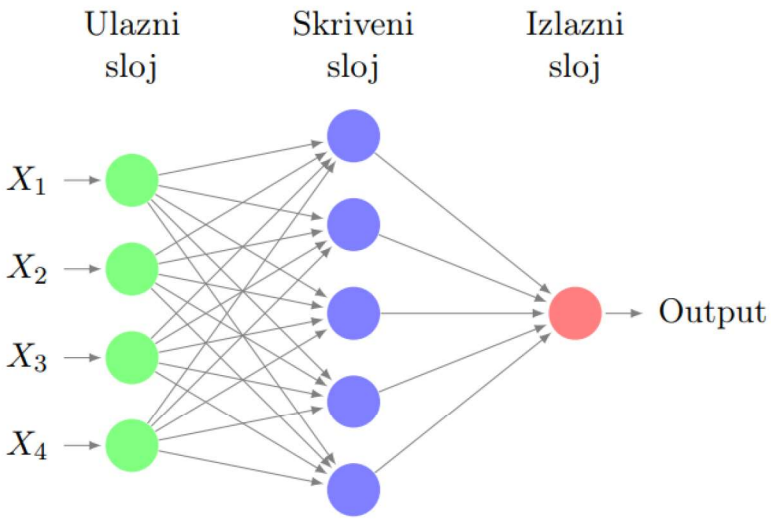
U kontekstu predikcije cijena dionica, neuronske mreže mogu se koristiti za analizu velikih količina povijesnih podataka i otkrivanje skrivenih obrazaca koji bi mogli ukazivati na buduće promjene cijena. Iako se postupak treniranja i primjene može razlikovati, osnovni principi ostaju isti: mreža uči iz povijesnih podataka, prilagođava svoje parametre i koristi ih za predikciju budućih vrijednosti. Korištenje dubokog učenja omogućuje predikcije koje tradicionalni modeli ne mogu postići zbog složenosti financijskih podataka i njihove dinamike.

Uz jednostavne mreže, postoji nekoliko drugih tipova neuronskih mreža koje se koriste za specifične zadatke. Konvolucijske neuronske mreže posebno su učinkovite u obradi vizualnih podataka, dok rekurentne neuronske mreže obrađuju sekvencijalne podatke, što ih čini pogodnima za zadatke poput obrade prirodnog jezika i analize vremenskih nizova. Posebne varijante rekurentnih neuronskih mreža, poput LSTM mreža, rješavaju problem dugoročnog pamćenja, omogućujući mreži učenje iz dugih nizova podataka [9].

Dok smo u prethodnom poglavlju pregledali osnovne komponente i vrste neuronskih mreža, kako bismo u potpunosti razumjeli njihovu sposobnost, potrebno je dublje zaroniti u matematičku strukturu koja omogućuje mrežama da prepoznaju složene obrasce i donose precizne odluke. Sljedeće poglavlje bavit će se upravo tim matematičkim temeljima.

2.1 Matematika iza neuronskih mreža

Neuronske mreže koriste ulazne vektore, koji se sastoje od p komponenti označenih kao $X = (X_1, X_2, \ldots, X_p)$. Svaka komponenta X_j predstavlja slučajnu varijablu, odnosno jednu ulaznu značajku ili prediktor. Ove ulazne značajke mreža koristi za izgradnju nelinearne funkcije $f(X)$, koja zatim služi za predviđanje ciljne varijable Y. Cilj mreže je kroz proces učenja optimizirati funkciju $f(X)$ kako bi što preciznije predviđala izlazne vrijednosti na temelju danih ulaznih podataka.



Slika 2.3: Prikaz jednostavne feed-forward neuronske mreže s $p = 4$ ulazna prediktora. Izvor: [5].

Na slici 2.3 prikazana je osnovna struktura feed-forward neuronske mreže koja se koristi za modeliranje kvantitativnih izlaza na temelju $p = 4$ ulaznih značajki. U ovoj mreži, četiri varijable $X_1, \ldots, X_4$ čine jedinice ulaznog sloja. Svaka ulazna varijabla povezana je sa svim skrivenim jedinicama; u ovom primjeru, mreža sadrži K skrivenih jedinica, pri čemu smo odabrali $K = 5$. Matematički opis ove mreže, koja uključuje p ulaznih varijabli i K skrivenih jedinica, izražava se formulom:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) = \beta_0 + \sum_{k=1}^{K} \beta_k g\left(b_k + \sum_{j=1}^{p} w_{kj} X_j\right). \tag{2.1}$$

Ovdje $\beta_0, \cdots, \beta_k$ predstavljaju koeficijente modela, dok su w_{kj} težine koje određuju doprinos svakog ulaza X_j svakom skrivenom sloju. Parametri $w_{k0} = b_k$, poznati kao pristranosti (bias), omogućuju modelu da pomakne ulazne vrijednosti aktivacijskih funkcija neurona, što omogućuje mreži da generira nenulti izlaz čak i kada su svi ulazni podaci nula. Ovo povećava fleksibilnost modela, omogućujući mu da bolje prilagodi funkciju $f(X)$ različitim skupovima podataka. Funkcije

$h_k(X)$ su nelinearne transformacije ulaznih podataka, dok g predstavlja aktivacijsku funkciju koja obrađuje linearnu kombinaciju ulaza.

2.1.1 Aktivacijske funkcije

U modelu neuronske mreže, funkcija aktivacije g je nelinearna funkcija koja se primjenjuje na težinsku sumu ulaznih vrijednosti plus bias. Bez funkcije aktivacije, izlaz neuronske mreže bio bi jednostavno linearna funkcija ulaznih vrijednosti, a mreža ne bi bila sposobna naučiti kompleksne odnose između ulaza i izlaza [5]. Prvo se izračunavaju aktivacije $a_k, k = 1, \cdots, K$, u skrivenom sloju kao funkcije ulaznih varijabli $X_1, \cdots X_p$:

$$a_k = h_k(X) = g(b_k + \sum_{j=1}^{p} w_{kj} X_j), \tag{2.2}$$

gdje je $g(z)$ nelinearna aktivacijska funkcija koja se specificira unaprijed. U ovom kontekstu, X označava vektor ulaznih varijabli $X = [X_1, X_2, \cdots, X_p]$, koje predstavljaju različite karakteristike ili značajke koje mreža koristi za učenje. Svaku aktivaciju a_k možemo zamisliti kao drugačiju transformaciju $h_k(X)$ originalnih ulaznih varijabli. Ovih K aktivacija iz skrivenog sloja zatim se dovode u izlazni sloj, što rezultira:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k a_k,$$

linearnim regresijskim modelom u K aktivacija. Svi parametri $\beta_0, \ldots, \beta_K$ i $w_{10}, \ldots, w_{Kp}$ trebaju se procijeniti iz podataka.

U ranim primjerima neuronskih mreža preferirana je bila sigmoidna aktivacijska funkcija čiji oblik prikazuje Slika 2.4:

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}.$$

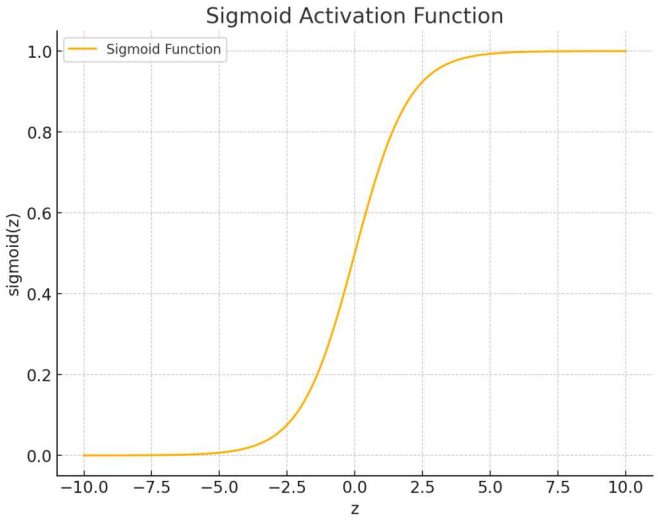
Slika 2.4 prikazuje sigmoidnu funkciju koja $z \in \mathbb{R}$ preslikava u $g(z) \in \langle 0, 1 \rangle$. Često se koristi u problemima binarne klasifikacije gdje izlaz mreže predstavlja vjerojatnost binarnog ishoda.

Preporučeni izbor u modernim neuronskim mrežama je funkcija aktivacije *ReLU* (Rectified Linear Unit):

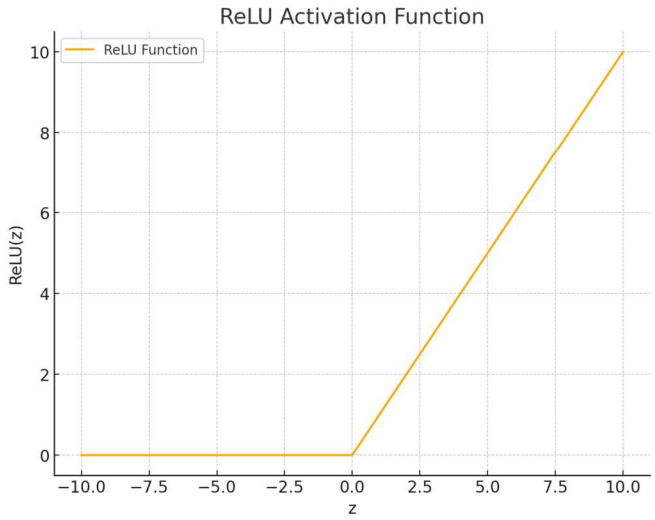
$$g(z) = (z)_+ = \begin{cases} 0, & z < 0 \\ z, & \text{inače.} \end{cases}$$

Preporučuje se iz nekoliko razloga:

1. **Jednostavniji izračun:** *ReLU* aktivacija se može izračunati brže od sigmoidne aktivacije, jer ne zahtijeva računanje eksponencijalne funkcije koja je prisutna u sigmoidnoj funkciji. Umjesto toga, *ReLU* aktivacija se sastoji od jednostavnog



Slika 2.4: Sigmoid aktivacijska funkcija.



Slika 2.5: ReLU aktivacijska funkcija.

elementarnog koraka koji se primjenjuje na ulazni podatak. Ovo značajno smanjuje vrijeme potrebno za izračun aktivacije.

2. **Jednostavnije derivacije:** Tijekom treniranja neuronskih mreža, koristi se gradijentni spust kako bi se prilagodile težine neuronskih veza. Derivacije *ReLU* aktivacije su također jednostavnije od sigmoidne aktivacije, što pojednostavljuje postupak računanja gradijenata.

3. **Efikasnija pohrana:** Tijekom izvođenja neuronske mreže, potrebno je pohraniti vrijednosti aktivacija za svaki neuron u svakom sloju. *ReLU* aktivacija ima diskretnu vrijednost (0 ili ulazni podatak) za negativne ulaze, što značajno smanjuje količinu potrebne memorije za pohranu aktivacija.

Sve ove karakteristike čine ReLU aktivaciju učinkovitijom od sigmoidne aktiva-

cije u praksi, posebno za velike neuronske mreže koje zahtijevaju brzo izvođenje i pohranu velikog broja vrijednosti aktivacije.

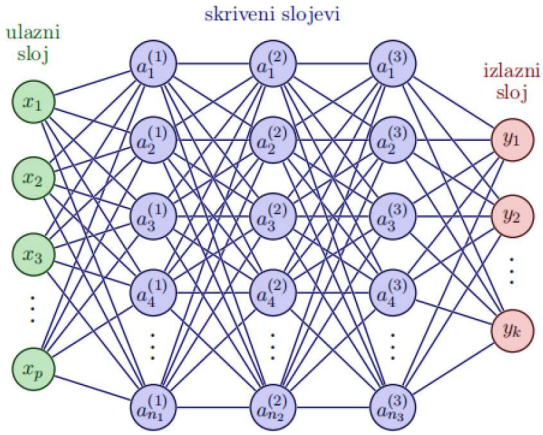
Izbor aktivacijske funkcije direktno utječe na oblik i ponašanje funkcije $f(X)$. Dok sigmoidna funkcija pruža glatku nelinearnost korisnu u specifičnim kontekstima kao što su binarne klasifikacije, ReLU funkcija omogućava brže i efikasnije učenje, time čineći modeliranje složenih obrazaca u podacima mnogo pristupačnijim. Stoga, odabir aktivacijske funkcije treba biti pažljivo razmotren u kontekstu specifičnih zahtjeva modela i prirode podataka [4] .

2.2 Neuronske mreže s više skrivenih slojeva

Nakon što smo razmotrili osnovne koncepte neuronskih mreža, uključujući njihove matematičke osnove i ulogu aktivacijskih funkcija, sada se fokusiramo na neuronske mreže koje sadrže više skrivenih slojeva. Ovi dodatni slojevi omogućuju modelima da nauče složene, nelinearne odnose u podacima, što je temeljna prednost dubokog učenja.

Uvest ćemo sljedeće oznake:

- Eksponent označava broj sloja u mreži,
- L predstavlja broj skrivenih slojeva,
- K_l označava broj neurona u sloju $l = 0, 1, \ldots, L+1$, pri čemu je $K_0 = p$ broj ulaznih varijabli, a K_{L+1} broj izlaznih varijabli.



Slika 2.6: Neuronska mreža s više skrivenih slojeva. Izvor: [5].

Kako bi se razumio način na koji neuronska mreža obrađuje podatke, mreža se može opisati rekurzivnim jednadžbama koje definiraju predaktivacije i aktivacije u svakom sloju.

Za prvi skriveni sloj ($l = 1$), predaktivacije $z_i^{(1)}(x)$ računaju se na sljedeći način:

$$z_i^{(1)}(x) = w_{i0}^{(1)} + \sum_{j=1}^{K_0} w_{ij}^{(1)} x_j, \quad i = 1, \ldots, K_1,$$

dok se za sve sljedeće slojeve ($l = 2, \ldots, L+1$), predaktivacije $z_i^{(l)}(x)$ računaju prema sljedećoj formuli:

$$z_i^{(l)}(x) = w_{i0}^{(l)} + \sum_{j=1}^{K_{l-1}} w_{ij}^{(l)} g_{l-1}\left(z_j^{(l-1)}(x)\right), \quad i = 1, \ldots, K_l$$

gdje $g(\cdot)$ označava aktivacijsku funkciju, koja se primjenjuje na predaktivacije. $z_i^{(l)}(x)$ predstavljaju linearnu kombinaciju težina i aktivacija iz prethodnog sloja, uz dodatak pristranosti. Te predaktivacije služe kao ulaz za aktivacijsku funkciju u sloju l, definiranu kao:

$$a_i^{(l)}(x) = g_l\left(z_i^{(l)}(x)\right), \quad i = 1, \ldots, K_l, \quad l = 1, \ldots, L+1.$$

Aktivacijska funkcija $g_l(z)$ uvodi nelinearnost u model, što omogućuje mreži da uči složene obrasce i odnose u podacima.

U matematičkom zapisu, označimo težine slojeva matricama $W^{(l)}$, gdje $W^{(l)} = \left[w_{ij}^{(l)}\right]$ za $i = 1, \ldots, K_l$ i $j = 1, \ldots, K_{l-1}$, te vektore pristranosti $w_0^{(l)}$. Na taj način možemo opisati mrežu kao:

$$z^{(1)}(x) = w_0^{(1)} + W^{(1)} x,$$
$$z^{(l)}(x) = w_0^{(l)} + W^{(l)} g_{l-1}\left(z^{(l-1)}(x)\right), \quad l = 2, \ldots, L+1,$$
$$a^{(l)}(x) = g_l\left(z^{(l)}(x)\right), \quad l = 1, \ldots, L+1.$$

Ove jednadžbe pokazuju kako svaki sloj u mreži transformira ulazne podatke u složenije reprezentacije, što na kraju omogućava mreži da generira izlazne vrijednosti.

Konačni izlaz neuronske mreže, y_i, može se izraziti kao:

$$y_i = a_i^{(L+1)}(x), \quad i = 1, \ldots, K_{L+1},$$

ili vektorski kao $y = f(x) = a^{(L+1)}(x)$. Ako mreža ima samo jednu izlaznu varijablu, tj. $K_{L+1} = 1$, tada funkcija $f(x)$ postaje skalarna vrijednost. Cijeli proces rada neuronske mreže može se izraziti sljedećom funkcijom:

$$f(x) = g_{L+1}\left(W^{(L)} g_L\left(W^{(L-1)} g_{L-1}\left(\cdots g_1\left(W^{(1)} x + w_0^{(1)}\right)\right)\right) + w_0^{(L)}\right).$$

Nepoznati parametri modela $\theta = \{W^{(1)}, W^{(2)}, \ldots, W^{(L)}, w_0^{(1)}, w_0^{(2)}, \ldots, w_0^{(L)}\}$ uključuju sve težine i pristranosti u mreži, te je ukupan broj parametara u modelu dan izrazom:

$$\sum_{l=1}^{L+1}(K_l K_{l-1} + K_l).$$

Ovakva arhitektura omogućuje mreži da postupno stvara složene transformacije ulaznih podataka, koje se na kraju koriste za generiranje izlaznih predikcija. Zbog toga se ovakva mrežna arhitektura naziva feedforward neuronskom mrežom [5].

2.3 Teoremi o univerzalnoj aproksimaciji

Nakon što smo detaljno razmotrili strukturu neuronskih mreža i njihove sposobnosti da modeliraju složene odnose između ulaza i izlaza, prirodno je zapitati se koliko su te mreže zaista moćne u aproksimaciji funkcija. Teoremi o univerzalnoj aproksimaciji pružaju matematičku potvrdu da neuronske mreže mogu aproksimirati gotovo bilo koju funkciju, što je od ključnog značaja za razumijevanje njihove primjenjivosti u stvarnim problemima.

Jedan od najvažnijih rezultata u ovom području odnosi se na neuronske mreže s jednim skrivenim slojem. Ovaj rezultat pokazuje da takve mreže mogu aproksimirati bilo koju neprekidnu funkciju na kompaktu, pod uvjetom da imaju dovoljan broj neurona u skrivenom sloju. Drugim riječima, iako mreža ima samo jedan skriveni sloj, ako je broj neurona u tom sloju dovoljno velik, mreža može postići željenu točnost aproksimacije.

Teorem 1 (vidjeti [2][Teorem o univerzalnoj aproksimaciji]). *Neka je* $f : [0,1]^n \to \mathbb{R}$ *neprekidna funkcija. Tada za svaki* $\epsilon > 0$ *postoji funkcija oblika:*

$$\tilde{f}(x) = \sum_{i=1}^{B} \beta_i g(w^T x + b_i),$$

takva da za sve $x \in [0,1]^n$ *vrijedi:*

$$|f(x) - \tilde{f}(x)| < \epsilon.$$

Ovaj teorem potvrđuje da neuronske mreže mogu aproksimirati bilo koju neprekidnu funkciju s proizvoljnom točnošću, što je od ključnog značaja za razumijevanje njihove primjenjivosti u stvarnim problemima. Na primjer, u kontekstu predikcije cijena, ovaj teorem podržava ideju da neuronske mreže mogu modelirati složene odnose između povijesnih i budućih cijena, omogućujući tako precizne prognoze. Međutim, ovaj rezultat vrijedi pod pretpostavkom da je budućnost funkcija prošlosti, odnosno da postoji dovoljno informacija u povijesnim podacima za precizno modeliranje budućih vrijednosti.

U praktičnom smislu, teorem o univerzalnoj aproksimaciji pruža matematičku osnovu za upotrebu neuronskih mreža u raznim aplikacijama. Iako teoretski nije moguće odrediti točan broj neurona potreban za postizanje željene preciznosti, ovaj rezultat nam ipak daje sigurnost da, uz dovoljno složen model, možemo aproksimirati funkcije od interesa. Iako je teorem o univerzalnoj aproksimaciji

formuliran za mreže s jednim slojem, njegove implikacije su primjenjive i na mreže s više slojeva, koje su danas standard u mnogim područjima primjene dubokog učenja.

2.4 Prilagodba neuronskih mreža

Nakon što smo ustanovili teoretske temelje koji potvrđuju sposobnost neuronskih mreža da aproksimiraju gotovo bilo koju funkciju, prelazimo na ključni aspekt treniranja mreža. Proces prilagođavanja parametara mreže omogućava da ove teorijske sposobnosti postanu stvarnost, čineći neuronske mreže učinkovitima u praktičnim primjenama.

Prilikom treniranja neuronskih mreža, cilj je pronaći takve parametre (težine w_{kj} i pristranosti β_k) koji minimiziraju razliku između predviđenih vrijednosti modela $f(x_i)$ i stvarnih izlaznih vrijednosti y_i. Optimizacija tih parametara ključna je za uspješno treniranje modela, što se obično postiže minimizacijom funkcije troška, još poznata kao funkcija cilja (*eng. cost function*) [4].

Počinjemo s jednostavnom mrežom koju prikazuje Slika 2.3. Koristeći skup podataka s vrijednostima (x_i, y_i), $i = 1, \ldots, n$, model bismo mogli prilagoditi rješavanjem problema minimizacije kvadratne pogreške.

$$\min_{w_k, \beta} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

gdje je

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij})$$

Ovaj proces nije trivijalan zbog nekonveksne prirode problema. Nekonveksnost znači da funkcija troška može imati više lokalnih minimuma, što otežava pronalaženje globalno optimalnog rješenja. U praksi, optimizacija se provodi iterativnim algoritmima kao što su gradijentni spust i njegove varijante.

Jedan od glavnih izazova u prilagođavanju neuronskih mreža je upravo upravljanje ovom nekonveksnošću. Postoje razne tehnike koje se koriste kako bi se optimizacija olakšala, uključujući korištenje različitih algoritama za optimizaciju, pravilno postavljanje stope učenja, uvođenje regularizacije (koja pomaže u sprječavanju overfittinga), te korištenje tehnika kao što je rano zaustavljanje. Također, proces prilagođavanja mreže može biti vrlo računalno intenzivan, pogotovo kada se radi o dubokim mrežama s velikim brojem slojeva i neurona. Zbog toga se koriste napredni alati i okviri, koji omogućuju relativno automatiziran način treniranja modela bez potrebe za detaljnim poznavanjem svih tehničkih aspekata.

U financijskim modelima, standardne funkcije gubitka kao što su Mean Squared Error (MSE) i Mean Absolute Error (MAE) često se koriste za minimizaciju

pogreške predikcije. MSE kažnjava velike pogreške više od malih, dok MAE daje ravnomjernu kaznu za sve pogreške. Ove funkcije pomažu modelu da se fokusira na minimizaciju ukupne predikcijske pogreške, što je ključno za točnost predikcija [1].

2.4.1 Gradijentni spust

Kako bismo učinkovito prilagodili neuronske mreže i optimizirali njihove performanse, potrebno je odabrati prikladan algoritam za minimizaciju funkcije troška. Gradijentni spust je klasična metoda optimizacije koja iterativno računa niz vektora u prostoru parametara s ciljem konvergiranja prema vektoru koji minimizira funkciju troška. Ako trenutno imamo vektor parametara θ, cilj je pronaći malu promjenu $\Delta\theta$ koja će smanjiti vrijednost funkcije troška u sljedećem koraku.

Taylorov red koristi se za aproksimaciju funkcije u blizini neke točke, što je ključno za razumijevanje kako se funkcija troška mijenja u odnosu na male promjene u parametrima modela. U kontekstu gradijentnog spusta, cilj nam je pronaći smjer i veličinu promjene parametara koja će rezultirati smanjenjem funkcije troška. Prvi red Taylorovog razvoja daje linearnu aproksimaciju funkcije troška, što je dovoljno za određivanje smjera u kojem funkcija najbrže opada. Ova linearna aproksimacija temelji se na gradijentu funkcije troška, koji opisuje smjer najvećeg porasta funkcije. Stoga, suprotno od tog smjera, odnosno negativni gradijent, ukazuje na smjer u kojem funkcija najbrže opada.

Upravo zbog toga koristimo Taylorov red u gradijentnom spustu da bismo dobili jednostavnu, ali korisnu aproksimaciju promjene funkcije troška uzrokovane malim promjenama parametara. Ova aproksimacija omogućava nam da iterativno prilagođavamo parametre kako bismo smanjili funkciju troška i postigli bolju prilagodbu modela podacima.

Za male $\Delta\theta$, Taylorov red daje:

$$f(\theta + \Delta\theta) \approx f(\theta) + \nabla f(\theta)^T \Delta\theta \tag{2.3}$$

gdje je $\nabla f(\theta)$ gradijent funkcije troška u točki θ.

Gradijent funkcije troška $\nabla f(\theta)$ predstavlja vektor parcijalnih derivacija funkcije troška $f(\theta)$ u odnosu na svaki parametar θ. Točnije, ako je $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$ vektor svih parametara modela, tada je gradijent definiran kao:

$$\nabla f(\theta) = \left(\frac{\partial f(\theta)}{\partial \theta_1}, \frac{\partial f(\theta)}{\partial \theta_2}, \ldots, \frac{\partial f(\theta)}{\partial \theta_m} \right).$$

Gradijent funkcije troška možemo izraziti kao sumu parcijalnih derivacija preko svih podataka za treniranje. Pretpostavimo da imamo N točaka za treniranje $\{x^{(i)}\}_{i=1}^N$ sa željenim izlazima $\{y_i\}_{i=1}^N$. Tada je funkcija troška definirana kao:

$$f(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \|y_i - a^{[L]}(x^{(i)})\|_2^2 \tag{2.4}$$

Gradijent funkcije troška, $\nabla f(\theta)$, tada se može zapisati kao:

$$\nabla f(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_{x^{(i)}}(\theta) \tag{2.5}$$

gdje je $f_{x^{(i)}}(\theta)$ parcijalni trošak za točku $x^{(i)}$.

Kako bi se smanjila vrijednost funkcije troška, promjena $\Delta\theta$ se bira u smjeru suprotnom od gradijenta:

$$\theta \to \theta - \eta \nabla f(\theta) \tag{2.6}$$

gdje je η mali korak, poznat kao stopa učenja. Ova jednadžba definira metodu gradijentnog spusta. Odabiremo početni vektor i iteriramo pomoću ove jednadžbe dok se ne postigne neki kriterij zaustavljanja ili dok se broj iteracija ne prekorači zadani proračunski limit [6].

Kada imamo velik broj parametara i velik broj točaka za treniranje, izračunavanje gradijenta u svakoj iteraciji može biti izuzetno skupo. To je zato što svaki izračun gradijenta uključuje sumiranje doprinosa svih točaka za treniranje, što može zahtijevati znatnu računalnu snagu, posebno kod velikih skupova podataka. Puno jeftinija alternativa je zamijeniti srednju vrijednost pojedinačnih gradijenata preko svih točaka za treniranje gradijentom na jednoj, nasumično odabranoj, točki za treniranje. To dovodi do najjednostavnijeg oblika stohastičkog gradijentnog spusta.

2.4.2 Stohastički gradijentni spust

Stohastički gradijentni spust (SGD) je varijanta gradijentnog spusta koja se često koristi za velike skupove podataka. Umjesto izračunavanja gradijenta na cijelom skupu podataka, SGD aproksimira gradijent koristeći jedan nasumično odabrani primjer iz skupa podataka.

Postupak se sastoji od sljedećih koraka:

1. Nasumično odaberemo cijeli broj i iz skupa $\{1, 2, 3, \ldots, N\}$.
2. Ažuriramo parametre:
$$\theta \to \theta - \eta \nabla f_{x^{(i)}}(\theta)$$

Gdje je $f_{x^{(i)}}(\theta)$ funkcija troška za odabrani primjer $x^{(i)}$. Ovaj pristup značajno smanjuje troškove računalne obrade po iteraciji, ali postoji kompromis jer svako ažuriranje ne garantira smanjenje ukupne funkcije troška u svakoj iteraciji.

Još jedan popularan pristup je korištenje mini-batch verzije SGD-a. Ovdje, umjesto jednog primjera, koristi se mala podskupina podataka (mini-batch) za izračunavanje prosječnog gradijenta:

1. Nasumično odaberemo m primjera $k_1, k_2, \ldots, k_m$ iz skupa $\{1, 2, 3, \ldots, N\}$.

2. Ažuriramo parametre:

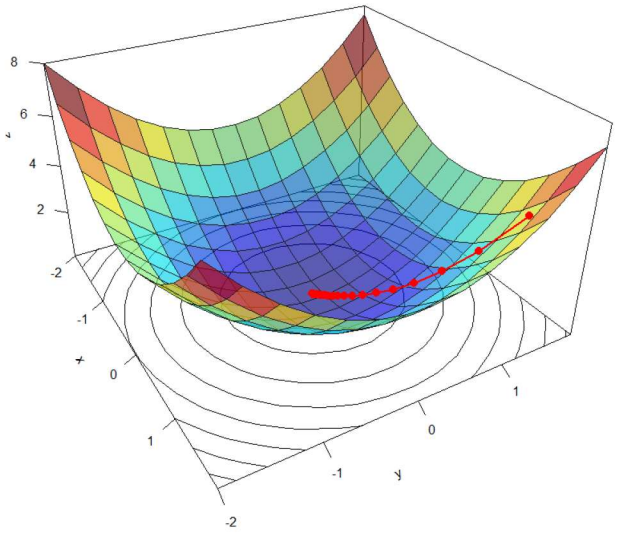
$$\theta \to \theta - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla f_{x^{(k_i)}}(\theta)$$

Upotreba mini-batch-eva u stohastičkom gradijentnom spustu omogućava bržu konvergenciju i smanjenje varijance gradijenta, dok i dalje zadržava prednosti jednostavnijeg algoritma. Ovo je posebno korisno kada se radi o velikim skupovima podataka gdje je klasični gradijentni spust računalno previše zahtjevan.

Stoga, pravilno prilagođavanje neuronskih mreža, uz korištenje varijanti gradijentnog spusta kao što je stohastički gradijentni spust, omogućuje učinkovitije treniranje modela na velikim skupovima podataka, što je ključno za postizanje visoke točnosti predikcija u stvarnom svijetu [6].

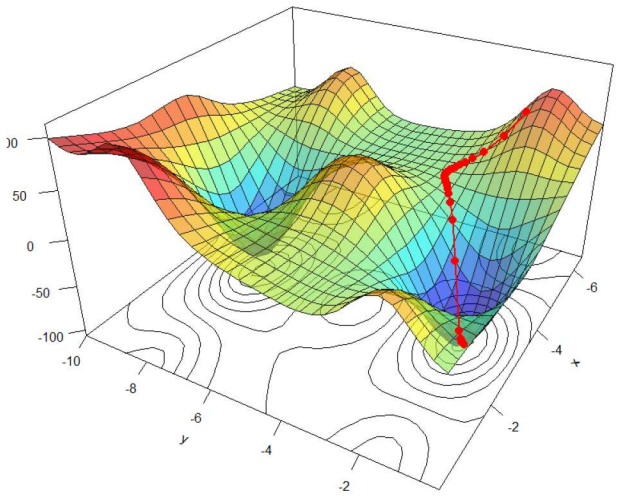
2.4.3 Grafički prikaz gradijentnog spusta

Imamo funkciju $f_1(x_1, x_2) = x_1^2 + x_2^2$, i u algoritmu za traženje minimuma funkcije f_1, kao početnu točku biramo $(x_1, x_2) = (1.5, 1.5)$. Želimo procesom gradijentnog spusta doći do minimuma funkcije f_1. Provedbom procesa gradijentnog spusta dobijemo rezultat prikazan na slici 2.7.



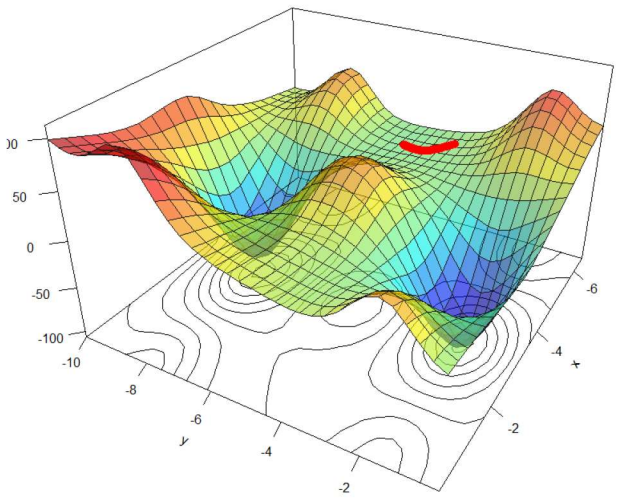
Slika 2.7: Grafički prikaz gradijentnog spusta na primjeru traženja minimuma funkcije f_1

Ako uzmemo funkciju $f_2(x_1, x_2) = \sin(x_2) \cdot \exp(1 - \cos(x_1))^2 + \cos(x_1) \cdot \exp(1 - \sin(x_2))^2 + (x_1 - x_2)^2$, i početne vrijednosti $(x_1, x_2) = (-6, -2)$, korištenjem gradijentnog spusta dobijemo rezultat prikazan na slici 2.8.



Slika 2.8: Grafički prikaz gradijentnog spusta na primjeru traženja minimuma funkcije f_2

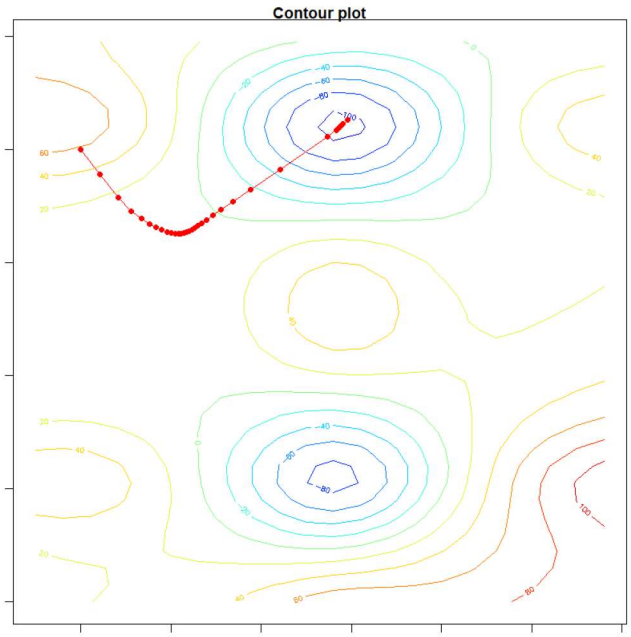
Ako odaberemo istu funkciju, ali promijenimo početne vrijednosti na $(x_1, x_2) = (-6, -4)$ i povećamo broj iteracija, dobit ćemo rezultat prikazan na slici 2.9, što pokazuje kako se gradijentni spust može suočiti s izazovima na složenijim funkcijama koje sadrže više lokalnih minimuma. Ovdje možemo vidjeti kako početna točka i broj iteracija utječu na konačan ishod.



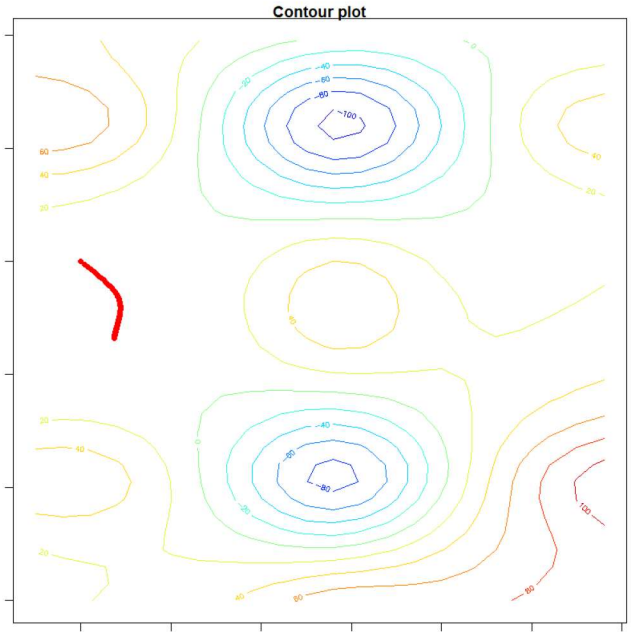
Slika 2.9: Grafički prikaz gradijentnog spusta na funkciji f_2 s drugim početnim vrijednostima i većim brojem iteracija

Slike 2.10 i 2.11 prikazuju konture funkcije zajedno s putanjom gradijentnog

spusta prikazanom crvenim točkama.



Slika 2.10: Prikaz procesa gradijentnog spusta koristeći contour plot



Slika 2.11: Prikaz procesa gradijentnog spusta koristeći contour plot

2.5 Povratna propagacija

Povratna propagacija pogreške jedan je od ključnih algoritama za treniranje dubokih neuronskih mreža. Ovaj algoritam omogućava mreži da uči iz pogrešaka

nastalih tijekom predikcije, prilagođavajući svoje parametre kako bi poboljšala točnost u budućim predikcijama. Kroz proces povratne propagacije, mreža računa koliko svaka težina i pristranost u modelu doprinosi ukupnoj pogrešci, te na temelju tih informacija prilagođava parametre kako bi minimizirala ukupnu pogrešku.

Cilj povratne propagacije je minimizirati funkciju gubitka, koja mjeri razliku između predviđenih i stvarnih vrijednosti. Ovaj proces koristi gradijentni spust za pronalaženje optimalnih težina i pristranosti. Gradijenti pokazuju smjer u kojem funkcija pogreške najbrže opada, a povratna propagacija koristi te gradijente kako bi prilagodila parametre kroz mrežu.

Kako povratna propagacija koristi pravilo deriviranja složenih funkcija

Povratna propagacija koristi matematičko pravilo deriviranja složenih funkcija kako bi izračunala kako promjene težina i pristranosti u svakom sloju utječu na ukupnu pogrešku. To pravilo deriviranja omogućava da se pogreška širi unazad kroz mrežu, od izlaznog sloja prema ulaznom, računajući doprinos svake težine i pristranosti. Na taj način mreža prilagođava parametre u svakom sloju na temelju njihovog utjecaja na konačni ishod.

Forward pass (prolazak unaprijed)

Prvi korak povratne propagacije je *forward pass*, gdje podaci prolaze kroz mrežu, od ulaza do izlaza. Ovaj proces uključuje izračunavanje aktivacija svakog neurona, počevši od ulaznog sloja, kroz skriveni sloj, sve do izlaznog sloja. Na kraju *forward pass*-a, mreža daje predikciju na temelju izlaza iz posljednjeg sloja.

Backward pass (prolazak unatrag)

Nakon što mreža napravi predikciju, uspoređuje se predikcija sa stvarnom vrijednošću, što daje pogrešku ili funkciju gubitka. U drugom koraku, nazvanom *backward pass*, mreža koristi tu pogrešku kako bi prilagodila težine i pristranosti unutar svakog sloja.

Računanje gradijenata

Pomoću pravila deriviranja složenih funkcija, povratna propagacija računa gradijente – parcijalne derivacije funkcije gubitka u odnosu na težine i pristranosti. Gradijenti pokazuju u kojem smjeru i koliko je potrebno prilagoditi parametre kako bi se smanjila ukupna pogreška.

$$\theta \rightarrow \theta - \eta \nabla f(\theta)$$

gdje je θ skup svih težina i pristranosti, η stopa učenja, a $\nabla f(\theta)$ gradijent funkcije gubitka u odnosu na parametre.

Ažuriranje težina

Nakon što su izračunati gradijenti, mreža ažurira težine i pristranosti. Težine koje su pridonijele većim pogreškama bit će značajnije prilagođene, dok će one s manjim utjecajem na pogrešku doživjeti manje promjene. Ovaj proces prilagodbe ponavlja se kroz više iteracija kako bi se postiglo smanjenje pogreške i poboljšala točnost modela. Kroz ovaj ciklus *forward* i *backward* prolaza, mreža uči iz svojih pogrešaka, postupno prilagođavajući svoje parametre kako bi povećala točnost predikcija.

2.5.1 Efikasnost kroz vektorizaciju

Jedna od prednosti propagacije unazad je kompatibilnost s vektoriziranim operacijama, što značajno ubrzava proces treniranja, posebno u dubokim mrežama s velikim brojem slojeva. Korištenjem vektorizacije, izračuni se mogu paralelizirati, što smanjuje ukupno vrijeme treniranja mreže.

Primjerice, izračun gradijenata za cijeli sloj može se izvesti u jednom koraku pomoću operacija matrica, umjesto da se iterativno prolazi kroz svaki neuron posebno. Ova metoda koristi algebarske operacije kao što su matrično množenje i Hadamardov produkt, što omogućuje brzo i učinkovito prilagođavanje težina i pristranosti.

2.5.2 Rješavanje problema nestajanja i eksplodiranja gradijenata

Jedan od izazova u primjeni propagacije unazad na duboke mreže je problem nestajanja (vanishing) i eksplodiranja (exploding) gradijenata. Tijekom propagacije greške unatrag kroz slojeve, gradijenti se mogu smanjiti ili povećati eksponencijalno, što može otežati ili onemogućiti pravilno treniranje mreže. Ovo se posebno odnosi na duboke mreže s velikim brojem slojeva, gdje se gradijenti koji se propagiraju unatrag mogu toliko smanjiti da više nemaju značajan utjecaj na prilagodbu težina. S druge strane, gradijenti mogu eksponencijalno rasti, što može dovesti do vrlo velikih promjena u težinama i destabilizirati proces treniranja.

Rješenja ovih problema uključuju korištenje specifičnih arhitektura mreža kao što su Long Short-Term Memory mreže za rekurentne neuronske mreže ili normalizacija batcha koja pomaže u stabilizaciji distribucije aktivacija u svakom sloju [4].

2.6 Prenaučenost (eng. Overfitting)

Prenaučenost nastaje kada trenirana mreža vrlo precizno izvršava zadatke na danim podacima, ali ne može dobro generalizirati na nove podatke. U širem smislu, to znači da se proces učenja previše usredotočio na nevažne i nereprezentativne „šumove“ u danim podacima. Predloženi su mnogi načini za borbu protiv prenaučenosti, a neki od njih mogu se koristiti zajedno.

Validacija i rano zaustavljanje

Jedna učinkovita metoda za borbu protiv prenaučenosti je korištenje skupa za validaciju, koji je odvojen od podataka za treniranje. Izvedba modela se prati na validacijskom skupu, a treniranje se zaustavlja kada se izvedba na tom skupu počne pogoršavati, čak i ako se izvedba na skupu za treniranje poboljšava. Ova metoda, poznata kao *rano zaustavljanje*, sprječava model da se previše specijalizira na podatke za treniranje.

Tehnike regularizacije

Regularizacijske metode ključne su za smanjenje prenaučenosti dodavanjem kazni za složene modele. Dva uobičajena oblika regularizacije su ℓ_1 i ℓ_2 regularizacija. ℓ_1 regularizacija dodaje apsolutnu vrijednost težina u funkciju gubitka, potičući rješenja sa što više nula. ℓ_2 regularizacija dodaje kvadrat težina, obeshrabrujući prevelike vrijednosti težina i time smanjujući složenost modela.

Dropout

Još jedna tehnika za smanjenje prenaučenosti je *dropout*, koja podrazumijeva nasumično postavljanje izlaza dijela neurona na nulu tijekom treniranja. Ovo sprječava mrežu da postane previše ovisna o specifičnim putevima i potiče učenje robusnijih značajki.

Povećanje podataka

U kontekstu dubokog učenja, osobito u zadacima klasifikacije slika, povećanje podataka (engl. data augmentation) odnosi se na tehniku koja se koristi za proširenje skupa podataka za treniranje putem stvaranja novih primjera iz postojećih podataka. To se postiže primjenom nasumičnih transformacija na originalne slike, kao što su rotacije, translacije, zrcaljenja, promjene veličine, obrezivanje i promjene u boji. Na primjer, slika mačke može se obrnuti ili blago rotirati, čime se dobiva novi primjer koji i dalje predstavlja istu klasu, ali s različitim vizualnim karakteristikama.

Ova tehnika povećava raznolikost podataka za treniranje, što pomaže modelu da bolje generalizira na neviđene podatke. Korištenjem ovih transformacija, model postaje manje osjetljiv na male varijacije u ulaznim podacima, čime se smanjuje rizik od prenaučenosti, odnosno prekomjernog prilagođavanja modela specifičnim primjerima iz skupa za treniranje.

Normalizacija batcha

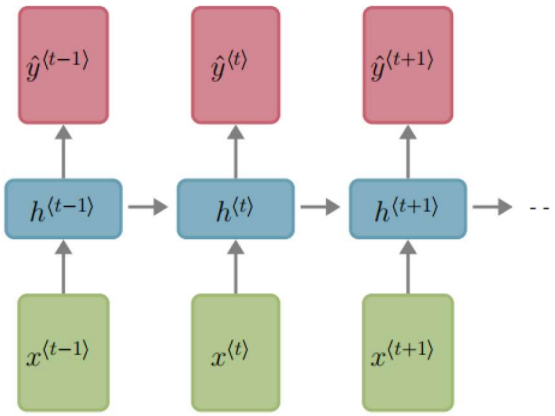
Normalizacija batcha je još jedna tehnika koja pomaže u sprječavanju prenaučenosti i ubrzava treniranje. Normaliziranjem ulaza svakog sloja, stabilizira proces učenja, omogućavajući veće stope učenja i smanjujući ovisnost o tehnikama inicijalizacije. Osim toga, normalizacija batcha ima učinak regulariziranja, smanjujući potrebu za drugim metodama regularizacije poput dropouta [4].

3 | Rekurentne neuronske mreže

Rekurentne neuronske mreže (eng. Recurrent Neural Networks) predstavljaju specifičnu klasu neuronskih mreža dizajniranih za rad sa vremenskim podacima, kao što su prirodni jezik, videozapisi i drugi nizovi podataka. Njihova ključna karakteristika je sposobnost da održavaju unutarnje stanje, što im omogućuje da „pamte“ informacije iz prethodnih ulaza i tako prate ovisnosti između elemenata u nizu. Ova sposobnost čini rekurentne neuronske mreže izuzetno korisnima u zadacima gdje redoslijed i kontekst igraju ključnu ulogu [5].

3.1 Arhitektura i funkcioniranje rekurentnih mreža

U klasičnim feed-forward neuronskim mrežama, informacije teku samo unaprijed kroz slojeve mreže. Međutim, u rekurentnim neuronskim mrežama povratne veze omogućuju da prethodno stanje utječe na trenutni izlaz, čime se stvara mreža koja može „pamtiti“ informacije kroz više vremenskih koraka.



Slika 3.1: Arhitektura rekurentne neuronske mreže. Izvor: [5]

Skriveno stanje $h^{\langle t\rangle}$ u rekurentnim neuronskim mrežama omogućuje da trenutni ulaz $x^{\langle t\rangle}$ utječe na kasnije slojeve mreže. Ovdje $x^{\langle t\rangle}$ predstavlja ulazni podatak u vremenskom koraku t, a njegova priroda ovisi o tipu podataka s kojima mreža radi:

- **U vremenskim nizovima**, $x^{\langle t\rangle}$ može biti numerička vrijednost ili skup vrijednosti u trenutku t (npr. cijena dionice, volumen trgovanja).

- **U tekstualnoj obradi**, $x^{\langle t \rangle}$ može predstavljati jedan token, poput riječi ili znaka, obično kodiranog kao vektor.
- **U videozapisima**, $x^{\langle t \rangle}$ može biti jedan okvir (frame) iz videozapisa ili skup značajki tog okvira.

Matematički, skriveno stanje u vremenskom koraku t može se izračunati kao:

$$h^{\langle t \rangle} = \psi_{\theta_{hx},\theta_{hh}} \left(h^{\langle t-1 \rangle}, x^{\langle t \rangle} \right),$$

gdje je $\psi_{\theta_{hx},\theta_{hh}}$ funkcija koja opisuje kako se iz prethodnog stanja $h^{\langle t-1 \rangle}$ i trenutnog ulaza $x^{\langle t \rangle}$ dobiva trenutno stanje. Funkcija ψ se ne mijenja s vremenom, što znači da su parametri jednaki za svaki vremenski korak.

Ova rekurzivna struktura omogućuje mreži da „pamti“ informacije kroz različite vremenske korake. Naime, mreža koristi skriveno stanje $h^{\langle t \rangle}$, koje se kontinuirano ažurira kako bi zadržalo informacije iz prošlih koraka. Tako, skriveno stanje djeluje kao unutarnja memorija koja omogućuje mreži da koristi informacije iz prošlih ulaza za predikcije u budućim koracima, čime se postiže efekt „pamćenja“.

Izlazni niz definiran je drugom funkcijom $g_{\theta_{yh}}$:

$$\hat{y}^{\langle t \rangle} = g_{\theta_{yh}} \left(h^{\langle t \rangle} \right),$$

gdje skup parametara također ne ovisi o trenutku t.

Funkcije $\psi_{\theta_{hx},\theta_{hh}}$ i $g_{\theta_{yh}}$ često se definiraju po uzoru na standardne neuronske mreže, koristeći aktivacijske funkcije poput hiperbolnog tangensa (tanh) i sigmoidne funkcije (σ_y). Konkretno:

$$h^{\langle t \rangle} = \tanh \left(W_{hh} h^{\langle t-1 \rangle} + W_{hx} x^{\langle t \rangle} + b_h \right),$$

$$\hat{y}^{\langle t \rangle} = \sigma_y \left(W_{yh} h^{\langle t \rangle} + b_y \right),$$

gdje su W_{hx}, W_{hh} i W_{yh} težine, dok su b_h i b_y bias vrijednosti. Dimenzije su sljedeće: p je dimenzija ulaza $x^{\langle t \rangle}$, q dimenzija izlaza $\hat{y}^{\langle t \rangle}$, a m dimenzija skrivenog stanja $h^{\langle t \rangle}$ [4].

Jedan od glavnih izazova treniranja rekurentnih mreža jest problem nestajanja i eksplodiranja gradijenata. Tijekom procesa treniranja, gradijenti koji se propagiraju unatrag kroz vremenske korake mogu postati izuzetno mali (nestajanje gradijenata) ili izuzetno veliki (eksplodiranje gradijenata). Ovo može otežati mreži učenje dugoročnih zavisnosti u podacima, što predstavlja značajnu prepreku za primjenu rekurentnih mreža u zadacima gdje je dugoročna memorija ključna.

3.2 Long short-term memory (LSTM) mreže

LSTM mreže, ili mreže s dugoročnom i kratkoročnom memorijom, razvijene su kao poboljšanje klasičnih rekurentnih neuronskih mreža kako bi se riješili ključni problemi učenja dugoročnih zavisnosti, posebice problem nestajanja gradijenata. Klasične rekurentne neuronske mreže često postaju neučinkovite u učenju dugoročnih zavisnosti jer „zaboravljaju“ informacije koje su se dogodile davno u vremenskom nizu.

LSTM mreže uvode specifične jedinice poznate kao LSTM ćelije, koje su temeljni elementi ove mrežne arhitekture. LSTM ćelije su dizajnirane da se bolje nose s dugoročnim zavisnostima u podacima, omogućujući mreži da „pamti“ relevantne informacije kroz mnoge vremenske korake.

Unutar svake LSTM ćelije nalaze se tri glavna mehanizma, poznata kao „vrata", koja kontroliraju protok informacija kroz ćeliju. Ova vrata pomažu mreži da odluči koje informacije će se zadržati kao dugoročna memorija, a koje će se zaboraviti. Na taj način, LSTM mreže mogu zadržati važne informacije i ignorirati nevažne, što klasične rekurentne neuronske mreže čini manje učinkovitima [9].

3.2.1 Struktura LSTM ćelije

Svaka LSTM ćelija sastoji se od tri osnovna tipa vrata [9]:

Zaboravna vrata (Forget gate, f_t)

Ova vrata odlučuju koje informacije, koje su ranije pohranjene, treba zaboraviti. Drugim riječima, mreža procjenjuje koje informacije više nisu važne za donošenje budućih odluka. Zaboravna vrata se aktiviraju pomoću sigmoidne funkcije, koja prima trenutne ulaze i prethodno skriveno stanje:

$$f_t = \sigma\left(W_f \cdot \left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_f\right).$$

Zaboravna vrata djeluju primarno na dugoročnu memoriju (C_t) i odlučuju koje informacije pohranjene u prethodnim koracima treba zadržati ili zaboraviti. Kada se procijeni da određeni podaci više nisu relevantni, zaboravna vrata smanjuju njihov utjecaj na dugoročnu memoriju.

- Ako je vrijednost f_t blizu 1, informacije pohranjene u dugoročnoj memoriji iz prethodnog koraka ($C^{(t-1)}$) ostaju nepromijenjene i zadržavaju se za daljnju upotrebu.
- Ako je f_t blizu 0, te informacije se „zaboravljaju“, odnosno više ne utječu na stanje dugoročne memorije.

Zaboravna vrata omogućuju mreži da procijeni koje informacije iz prethodnih vremenskih koraka više nisu relevantne za trenutni zadatak. Na taj način, zahvaljujući forget gate-u, mreža može odlučiti koje informacije zadržati, a koje odbaciti,

čime se izbjegava „pamćenje“ nepotrebnih podataka. Ovo je posebno korisno u složenim vremenskim nizovima, gdje je bitno da mreža zadrži samo relevantne obrasce iz prošlosti.

Ulazna vrata (Input gate, i_t)

Ova vrata kontroliraju koliko novih informacija će se dodati u ćelijsko stanje, odnosno dugoročnu memoriju mreže. Ulazna vrata određuju koliko informacija iz trenutnog vremenskog koraka treba pohraniti u dugoročnu memoriju. Na primjer, u predikciji cijena dionica, ulazna vrata koriste podatke iz prethodnih dana i odlučuju koje nove informacije su dovoljno relevantne da ih pohrane za buduće korake. Računa se kandidat za novo ćelijsko stanje:

$$\tilde{C}_t = \tanh\left(W_C \cdot \left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_C\right).$$

Zatim ulazna vrata određuju koliko te nove informacije treba pohraniti:

$$i_t = \sigma\left(W_i \cdot \left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_i\right).$$

Kombiniranjem ove dvije komponente ažurira se dugoročna memorija mreže koja služi za pohranu informacija koje su relevantne tijekom dužih vremenskih razdoblja. Za razliku od kratkoročne memorije, označene kao $h^{\langle t\rangle}$, koja sadrži informacije relevantne samo za tekući vremenski korak i koristi se isključivo unutar tog trenutnog koraka, dugoročna memorija C_t pohranjuje podatke koji su važni kroz dulje vremensko razdoblje. Kratkoročna memorija se kontinuirano mijenja u svakom koraku i ne zadržava informacije za buduću upotrebu, dok dugoročna memorija omogućava mreži da zapamti važne obrasce iz prošlosti koji mogu biti bitni i za buduće korake.

Na primjer, kod predikcije cijena dionica, informacije iz prošlih dana, tjedana ili čak mjeseci mogu biti ključne za razumijevanje trenutnih trendova. LSTM mreža putem dugoročne memorije može zadržati te važne obrasce, čak i ako su se dogodili daleko u prošlosti, i koristiti ih kako bi preciznije predvidjela buduće vrijednosti.

Izlazna vrata (Output gate, o_t)

Izlazna vrata određuju koliko od trenutnog ćelijskog stanja treba koristiti za generiranje izlaza. Izlazna vrata izračunavaju koje informacije treba proslijediti na sljedeći vremenski korak ili koristiti kao trenutni izlaz:

$$o_t = \sigma\left(W_o \cdot \left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_o\right).$$

Konačno skriveno stanje, koje se koristi za izlaz, dobiva se kombiniranjem izlaznih vrata i trenutnog stanja ćelije:

$$h^{\langle t\rangle} = o_t \cdot \tanh(C_t),$$

gdje je C_t trenutna memorija ćelije, definirana kao:

$$C_t = f_t \cdot C^{\langle t-1 \rangle} + i_t \cdot \tilde{C}_t.$$

Ova arhitektura omogućuje LSTM mrežama da zadrže korisne informacije kroz mnoge vremenske korake i ignoriraju one koje nisu relevantne. To omogućuje LSTM-ovima da bolje prepoznaju dugoročne zavisnosti u podacima, što ih čini vrlo korisnim u zadacima kao što su prepoznavanje govora, analiza vremenskih nizova i obrada prirodnog jezika [9].

4 | Model i predviđanje cijena dionica

U ovom dijelu rada analizirani su rezultati dobiveni primjenom dvaju različitih modela za predikciju cijena dionica AAPL i implementaciju jednostavne strategije trgovanja: LSTM modela i ARIMA modela. LSTM model, korišten zbog svoje sposobnosti hvatanja dugoročnih zavisnosti u vremenskim nizovima, služi za prepoznavanje složenih obrazaca u povijesnim podacima. S druge strane, ARIMA model, tradicionalna metoda za analizu vremenskih nizova, korišten je kao referentna točka za usporedbu.

Cilj ove analize je ispitati kako različiti modeli utječu na preciznost predikcija te na krajnje financijske rezultate prilikom trgovanja dionicama. Evaluacija modela provest će se pomoću metričkih pogrešaka (MSE, RMSE, MAE) te analizom uspješnosti implementirane strategije trgovanja.

Oba modela koriste povijesne podatke o cijenama dionica kompanije Apple Inc. (AAPL) prikupljene putem platforme Yahoo Finance za tri različita razdoblja: od 1. siječnja 2020. do 6. lipnja 2024., od 1. siječnja 2016. do 6. lipnja 2020., te od 1. siječnja 2018. do 6. lipnja 2022. Unutar svakog razdoblja, podaci su podijeljeni na skup za treniranje i testni skup tako da prvih 80% podataka predstavlja skup za treniranje, dok posljednjih 20% podataka čini testni skup. Ovakva vremenska podjela omogućuje detaljnu analizu performansi modela, osiguravajući da modeli uče iz ranijih podataka i budu testirani na kasnijim, što reflektira stvarne uvjete predviđanja u vremenskim nizovima.

Prije modeliranja, podaci su normalizirani skaliranjem kako bi se osiguralo učinkovito učenje modela. Ova metoda transformira podatke u raspon od 0 do 1, zadržavajući razmjere među vrijednostima. Za normalizaciju podataka korištena je funkcija `MinMaxScaler` iz biblioteke `sklearn.preprocessing`. Kod modeliranja vremenskih nizova poput cijena dionica, normalizacija pomaže na nekoliko načina:

- **Ujednačavanje različitih skala**: Cijene dionica mogu se mijenjati u širokom rasponu. Normalizacijom smanjujemo vrijednosti na standardni raspon, između 0 i 1. To pomaže modelu da se usredotoči na relativne promjene u podacima umjesto na apsolutne vrijednosti, što omogućuje učinkovitije prepoznavanje obrazaca.

- **Brže i stabilnije učenje**: Modeli dubokog učenja, poput LSTM-a, uče brže i stabilnije kada su podaci normalizirani. Algoritmi optimizacije, poput gradijentnog spusta, mogu brže konvergirati prema optimalnom rješenju kada rade s podacima koji su unutar standardiziranog raspona. Bez normalizacije, veliki raspon vrijednosti može dovesti do sporijeg i nestabilnijeg procesa učenja.

- **Sprječavanje numeričkih problema**: Kod rada s velikim ili jako malim vrijednostima, modeli mogu naići na numeričke probleme poput overflowa ili underflowa. Normalizacijom podataka, ove ekstremne vrijednosti su ublažene, čime se smanjuje rizik od numeričke nestabilnosti tijekom treniranja modela.

- **Optimalno iskorištavanje aktivacijskih funkcija**: Aktivacijske funkcije, kao što su sigmoidna ili hiperbolni tangens, rade najbolje kada su ulazne vrijednosti unutar određenog raspona. Normalizirani podaci omogućuju da se vrijednosti ulaza zadrže unutar tog raspona, osiguravajući učinkovitije učenje modela.

- **Poboljšanja za ARIMA model**: Iako normalizacija nije nužna za ARIMA modeliranje, može imati koristi u stabilizaciji varijance i poboljšanju performansi modela. Normalizacija može pomoći ARIMA modelu da lakše identificira i modelira sezonske obrasce i trendove, te može spriječiti potencijalne numeričke probleme kod rada s velikim vrijednostima podataka.

4.1 Korištenje LSTM modela za predikciju

U ovom istraživanju, LSTM model je korišten za predikciju cijena dionica koristeći vremenski niz cijena kao ulazne podatke. Konkretno, ulaz u LSTM model sastoji se od sekvence duljine 8 dana, koja predstavlja cijene dionica za tih 8 dana. Ovaj niz služi kao temelj za predviđanje cijene dionice za 9. dan. Takav pristup omogućuje modelu da prepozna i iskoristi obrasce u povijesnim podacima, što je ključno za točne predikcije.

Arhitektura korištenog LSTM modela sastoji se od tri LSTM sloja, svaki sa 128 jedinica. Svaka jedinica unutar LSTM sloja sadrži unutarnju strukturu koja uključuje memorijske ćelije te ulazna, izlazna i zaboravna vrata. Ova vrata kontroliraju protok informacija kroz mrežu, omogućujući modelu da uči dugoročne zavisnosti u vremenskim nizovima. Broj jedinica u svakom sloju određuje dimenzionalnost izlaza tog sloja.

Za odabir ove konfiguracije koristila se metoda grid search, koja uključuje sistematsko isprobavanje različitih kombinacija hiperparametara kako bi se pronašla ona koja daje najbolje rezultate. U ovom slučaju, ispitivali smo broj LSTM slojeva i broj jedinica unutar svakog sloja kako bismo postigli optimalnu performansu modela.

Grid search je proces u kojem se testiraju različite kombinacije broja slojeva i neurona u svakom sloju. Tijekom ovog procesa, isprobani su različiti brojevi slojeva (1, 2, 3) i različiti brojevi jedinica u sloju (32, 64, 128, 256) kako bi se utvrdilo koja konfiguracija najbolje odgovara za zadatak predviđanja cijena dionica. Konačna struktura s tri sloja i 128 neurona po sloju pokazala se kao najoptimalnija kombinacija za ovaj problem.

Na kraju mreže nalazi se gusti sloj s jednom jedinicom koji daje konačnu predikciju - cijenu dionice za 9. dan. Gusti sloj, poznat i kao „fully connected" sloj, znači da je svaki neuron u ovom sloju povezan sa svim neuronima u prethodnom sloju. U ovom slučaju, gusti sloj koristi sve naučene obrasce i informacije kako bi donio završnu odluku o predviđanju, pružajući konačnu procjenu zatvarajuče cijene dionice za naredni dan.

Za treniranje modela korištena je funkcija gubitka srednje kvadratne pogreške (Mean Squared Error, MSE). Ova funkcija mjeri prosječnu kvadratnu pogrešku između stvarnih i predviđenih vrijednosti te se koristi za optimizaciju modela tijekom procesa treniranja. MSE je prikladna za ovaj problem jer naglašava veće pogreške, što je korisno pri radu s vremenskim nizovima u kojima su precizne predikcije ključne.

Izlaz modela je predviđena cijena dionice za 9. dan na temelju danih ulaznih podataka. Nakon što se predikcija napravi, izlaz modela se denormalizira kako bi se transformirao natrag u izvornu skalu cijena dionica. Te se predikcije zatim koriste kao ključni input za strategiju trgovanja, omogućujući donošenje informiranih odluka o kupnji i prodaji na temelju očekivanih kretanja tržišta.

4.2 Korištenje ARIMA Modela za Predikciju Cijena Dionica

U ovom istraživanju, ARIMA (Autoregressive Integrated Moving Average) model korišten je za predikciju cijena dionica AAPL. ARIMA je popularan model za analizu i predikciju vremenskih nizova.

Odabir Parametara i Automatska Selekcija

Automatska selekcija parametara za ARIMA model rezultirala je korištenjem funkcije `auto.arima` iz paketa `forecast` u programskom jeziku R, gdje je model ARIMA(0,1,0) s driftom odabran kao najprikladniji za predviđanje cijena dionica AAPL. Automatski odabir ARIMA modela koristi algoritam za testiranje različitih kombinacija parametara i odabir onog koji najbolje opisuje dani vremenski niz prema informacijskom kriteriju, u našem slučaju Akaikeov informacijski kriterij (AIC). Ovo nam omogućuje brz i učinkovit način za modeliranje vremenskih nizova bez potrebe za ručnim podešavanjem svakog parametra.

Konkretno, `auto.arima` isprobava različite kombinacije parametara:

- **p** (autoregresivni red): broj prošlih vrijednosti koje se koriste za predikciju.
- **d** (integrirani red): broj diferenciranja potrebnih za postizanje stacionarnosti niza.
- **q** (red pomičnog prosjeka): broj prošlih reziduala koji se koriste za izradu trenutne predikcije. Reziduali predstavljaju razliku između stvarnih i predviđenih vrijednosti tijekom faze treniranja modela i koriste se za procjenu pogrešaka modela.

U ovom slučaju, model ARIMA(0,1,0) s driftom odabran je kao najprikladniji. Ovaj model je zapravo poznat kao „slučajna šetnja“ s driftom, što znači da se trenutna vrijednost procjenjuje na temelju prethodne vrijednosti uz dodatak konstante (drift). Model ne koristi autoregresivne komponente ($p = 0$) niti pokretni prosjek ($q = 0$), već se oslanja na jedno diferenciranje ($d = 1$) za hvatanje trenda u vremenskom nizu. Drift komponenta omogućuje modelu da uzme u obzir konstantan trend u podacima.

Proces Treniranja i Predikcije

ARIMA model u ovom istraživanju koristi se za predikciju na temelju cijelog skupa podataka za treniranje . Nakon što se ARIMA model trenira na prvom dijelu podataka (80% povijesnih cijena), koristi se za iterativno predviđanje testnog skupa. Predviđanja se vrše jedan korak unaprijed, pri čemu se svaki put nakon predikcije model ažurira s najnovijom stvarnom cijenom. Ovaj pristup omogućuje realističnije predikcije jer model stalno uči iz najnovijih dostupnih podataka. Model može biti ponovno treniran na novim podacima kako bi se prilagodio promjenama u tržištu. Ovo znači da model nije nužno uvijek isti; s novim podacima ili s promjenom tržišnih uvjeta, može se ponovno trenirati i prilagoditi kako bi ostao aktualan.

Proces predikcije uključuje sljedeće korake:

1. Treniranje ARIMA modela na skupu podataka za treniranje.
2. Predikcija cijene za sljedeći dan koristeći trenirani model.
3. Ažuriranje modela s novom stvarnom vrijednošću iz testnog skupa podataka.
4. Ponovno korištenje modela za predikciju sljedećeg dana.

Ovaj proces se ponavlja kroz cijeli testni skup podataka, pružajući iterativne i ažurirane predikcije koje se koriste u strategiji trgovanja.

Primjena u Strategiji Trgovanja

Kao i kod LSTM modela, predikcije ARIMA modela korištene su u implementaciji jednostavne strategije trgovanja.

4.3 Strategija trgovanja

Strategija trgovanja temelji se na predikcijama modela, gdje se donose odluke o kupnji ili prodaji dionica na temelju predviđene promjene cijene. Ova strategija radi na sljedeći način:

- **Inicijalizacija**: Strategija započinje s određenim početnim iznosom gotovine ($10,000$\$) i bez posjedovanih dionica.
- **Kupnja**: Ako model predviđa da će cijena dionice rasti za više od 2 dolara (USD) u odnosu na trenutnu cijenu, strategija sugerira kupnju dionica. Prag od 2 dolara postavljen je kako bi se filtrirale manje, potencijalno nasumične fluktuacije. Na ovaj način, strategija pokušava identificirati značajnije pozitivne promjene koje mogu signalizirati priliku za dobit.
- **Prodaja**: Ako model predviđa pad cijene dionice za više od 2 dolara (USD), strategija sugerira prodaju svih trenutno posjedovanih dionica. Cilj je izbjeći gubitke koji bi mogli nastati uslijed pada vrijednosti dionica.
- **Održavanje pozicije**: Ako promjena cijene prema modelu nije veća od 2 dolara (bilo pozitivna ili negativna), strategija ne poduzima nikakvu akciju. Ovo pomaže u izbjegavanju nepotrebnih transakcija i troškova, te zadržava fokus na značajnijim tržišnim kretanjima.
- **Kraj razdoblja**: Na kraju razdoblja, strategija prodaje sve preostale dionice kako bi se izračunala ukupna konačna vrijednost portfelja.

Prilikom postavljanja praga od 2 dolara, provedena su testiranja s različitim vrijednostima praga. Cilj je bio pronaći optimalnu vrijednost koja pruža ravnotežu između prečestog i prerijetkog trgovanja. Prag manji od 2 dolara doveo bi do povećane osjetljivosti na manje fluktuacije, što bi rezultiralo većim brojem transakcija i potencijalno smanjenim profitom zbog transakcijskih troškova. S druge strane, veći prag bi mogao propustiti važne signale za trgovanje. Eksperimentalno je utvrđeno da je prag od 2 dolara optimalan, jer je pružio najbolji omjer između rizika i potencijalne dobiti, uz održavanje konzistentne strategije trgovanja.

Strategija je osmišljena kako bi omogućila procjenu učinkovitosti modela ne samo u smislu točnosti predikcija, već i u kontekstu stvarnih financijskih odluka. Kombiniranjem predikcija modela s pragom od 2 dolara, strategija pokušava iskoristiti značajnije promjene na tržištu za ostvarivanje potencijalnog profita. Evaluacija uspješnosti strategije uključuje praćenje ukupne vrijednosti portfelja tijekom vremena, uključujući dobitke ili gubitke ostvarene trgovanjem.

4.4 Rezultati analize i usporedba modela

Provedena analiza obuhvatila je testiranje LSTM i ARIMA modela na različitim vremenskim intervalima, uz implementaciju strategije trgovanja koja koristi predikcije ovih modela za donošenje odluka. Kako bi se stekao uvid u performanse

modela, analizirana su tri vremenska intervala: od 1. siječnja 2020. do 6. lipnja 2024., od 1. siječnja 2016. do 6. lipnja 2020., te od 1. siječnja 2018. do 6. lipnja 2022. godine. Uz LSTM i ARIMA modele, u analizu je uključena i strategija „kupi i drži“ (Buy and Hold) kako bi se dobila referentna točka za usporedbu.

4.4.1 Interval od 1.1.2020 do 6.6.2024

Model	MSE	RMSE	MAE	Broj transakcija	Konačna gotovina
LSTM	13.00	3.6	2.78	10	$13,246.88
ARIMA	5.68	2.38	1.78	30	$11,890.45
Kupi i drži	-	-	-	-	$10,139.74

Tablica 4.1: Rezultati predikcije, broj transakcija i konačni financijski rezultati za LSTM i ARIMA modele, te strategiju „kupnje i držanja“ za razdoblje od 2020. do 2024. godine.

Analiza:

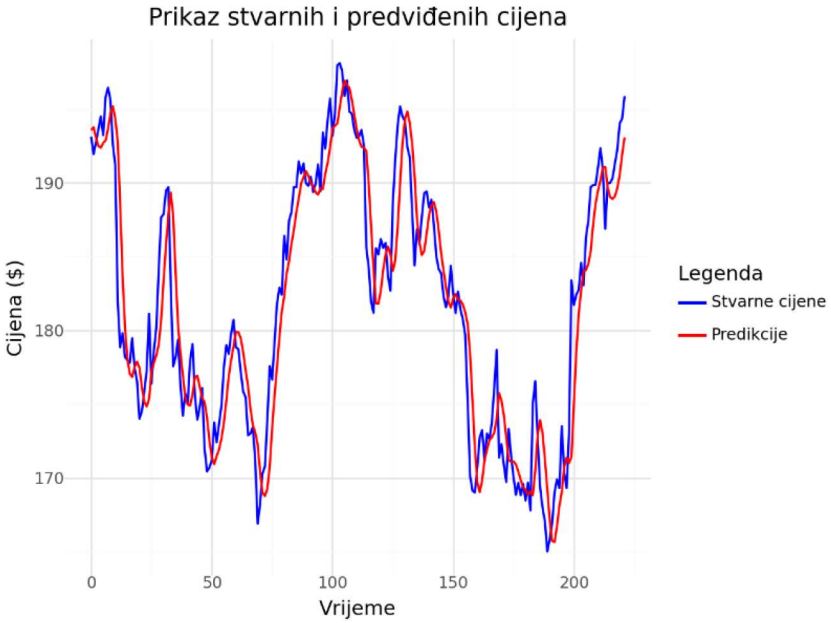
- **LSTM model**: Pokazao je značajan rast portfelja u odnosu na ARIMA model i strategiju „kupi i drži“, sugerirajući da LSTM model može prepoznati značajnije tržišne prilike. Ukupan broj transakcija (10) ukazuje na uravnotežen pristup, gdje LSTM model nije pretjerano reagirao na fluktuacije tržišta, već je birao manje, ali značajnije promjene za trgovanje.

- **ARIMA model**: Postigao je blagi rast u odnosu na „kupi i drži“ strategiju, ali nije bio tako učinkovit kao LSTM model. Međutim, ARIMA model je imao znatno više transakcija (30), što sugerira da je bio osjetljiviji na promjene u tržišnim uvjetima, ali taj veći broj transakcija nije doveo do boljih financijskih rezultata.

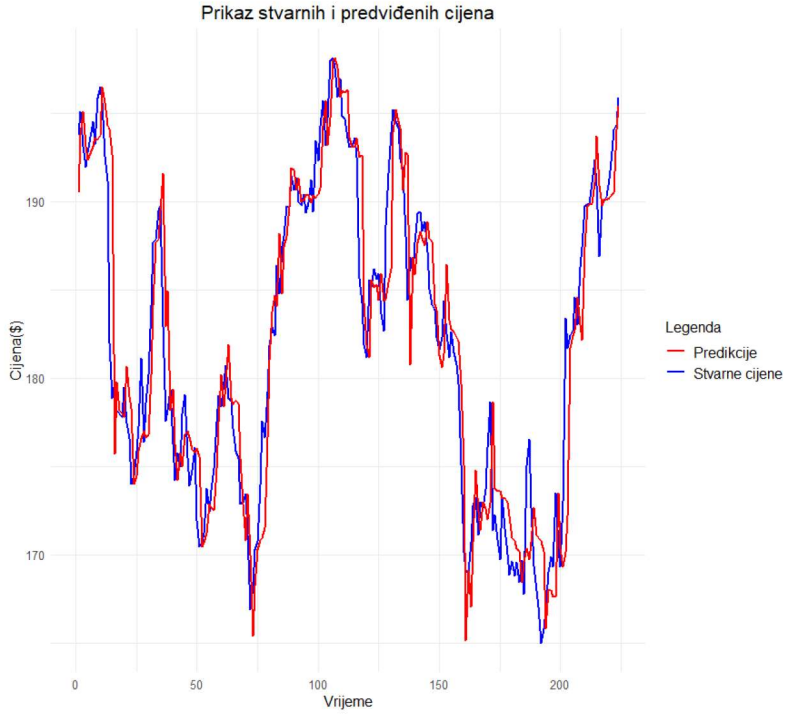
- **Strategija kupnje i držanja**: Zadržala je gotovo početni iznos, što sugerira stabilnost tržišta u ovom razdoblju. Ova strategija nije imala transakcija, jer se temelji na dugoročnom zadržavanju dionica.

Grafički prikazi

Grafički prikazi stvarnih i predviđenih cijena ključni su za vizualizaciju točnosti modela. Oni omogućuju jasan uvid u to kako se predikcije modela slažu sa stvarnim kretanjima cijena dionica, te pomažu u razumijevanju ponašanja modela u odnosu na stvarne tržišne uvjete. Kroz ove prikaze, možemo identificirati obrasce, prepoznati potencijalne greške i procijeniti učinkovitost modela u hvatanju složenih tržišnih trendova.



Slika 4.1: Prikaz stvarnih i predviđenih cijena za Apple (AAPL), koristeći LSTM model.



Slika 4.2: Prikaz stvarnih i predviđenih cijena za Apple (AAPL), koristeći ARIMA model.

4.4.2 Interval od 1.1.2016 do 6.6.2020

Model	MSE	RMSE	MAE	Broj transakcija	Konačna gotovina
LSTM	3.70	1.92	1.33	4	$14,664.71
ARIMA	3.05	1.74	1.13	22	$12,663.32
Kupi i drži	-	-	-	-	$15,996.51

Tablica 4.2: Rezultati predikcije, broj transakcija i konačni financijski rezultati za LSTM i ARIMA modele, te strategiju „kupnje i držanja“ za razdoblje od 2016. do 2020. godine.

Analiza:

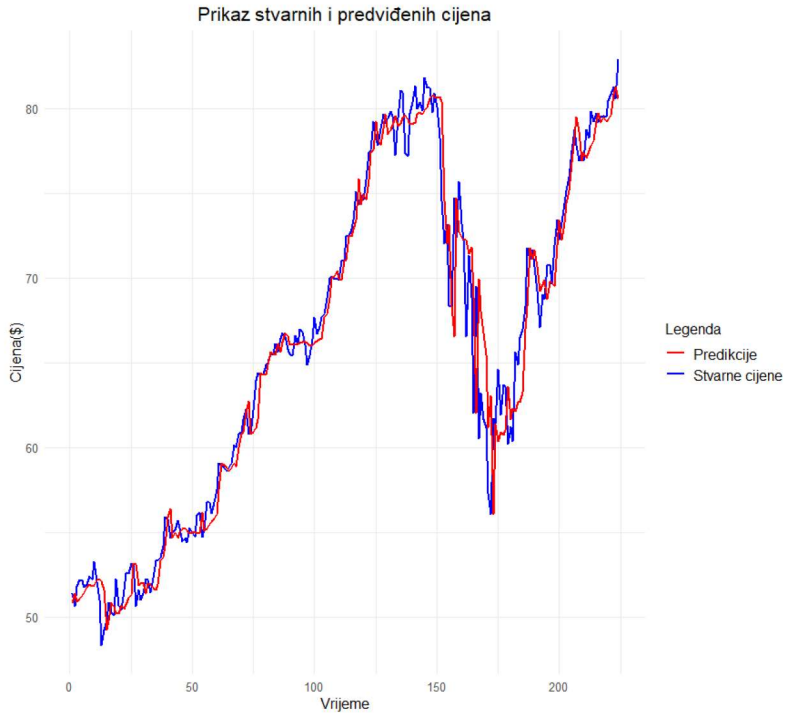
- **LSTM model**: Postigao je značajan rast, ali nije nadmašio strategiju „kupnje i držanja“. Ovo ukazuje na uspješno modeliranje kretanja cijena, ali i na ograničenja modela u odnosu na dugoročni rast cijena dionica. Relativno mali broj transakcija (4) ukazuje na to da LSTM model nije često trgovao, što sugerira konzervativniji pristup koji je prepoznavao samo ključne promjene u tržištu.

- **ARIMA model**: Postigao je manju konačnu vrijednost portfelja i nije nadmašio strategiju „kupnje i držanja“, što ukazuje na poteškoće u modeliranju dugoročnih trendova. Veći broj transakcija (22) ukazuje na osjetljivost modela na tržišne promjene, ali česte transakcije nisu donijele bolje rezultate.

- **Strategija kupnje i držanja**: Rezultirala je najvećom konačnom vrijednošću portfelja, sugerirajući stabilan rast cijena dionica u tom razdoblju.

Grafički prikazi

Slika 4.3: Prikaz stvarnih i predviđenih cijena za Apple (AAPL), koristeći LSTM model.



Slika 4.4: Prikaz stvarnih i predviđenih cijena za Apple (AAPL), koristeći ARIMA model.

4.4.3 Interval od 1.1.2018 do 6.6.2022

Model	MSE	RMSE	MAE	Broj transakcija	Konačna gotovina
LSTM	19.41	4.41	3.34	8	$12,404.66
ARIMA	8.78	2.96	2.25	42	$13,693.59
Kupi i drži	-	-	-	-	$9,947.64

Tablica 4.3: Rezultati predikcije, broj transakcija i konačni financijski rezultati za LSTM i ARIMA modele, te strategiju „kupnje i držanja“ za razdoblje od 2018. do 2022. godine.

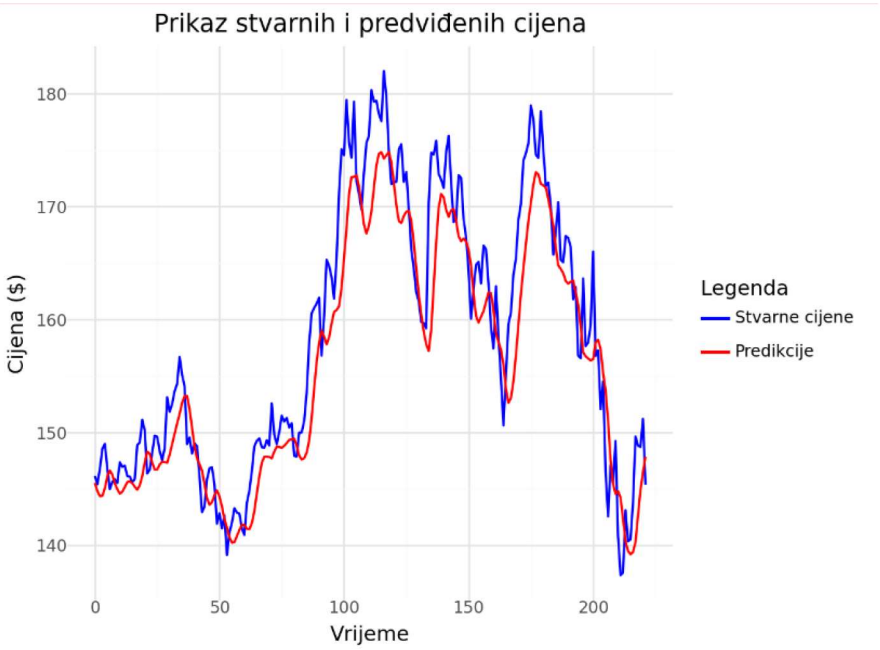
Analiza:

- **LSTM model**: Pokazao je bolji rezultat od strategije „kupnje i držanja“, ali lošiji od ARIMA modela. Ukupan broj transakcija (8) sugerira da je LSTM model imao umjerenu aktivnost, odabirući ključne prilike za trgovanje, što je rezultiralo solidnim rastom portfelja.

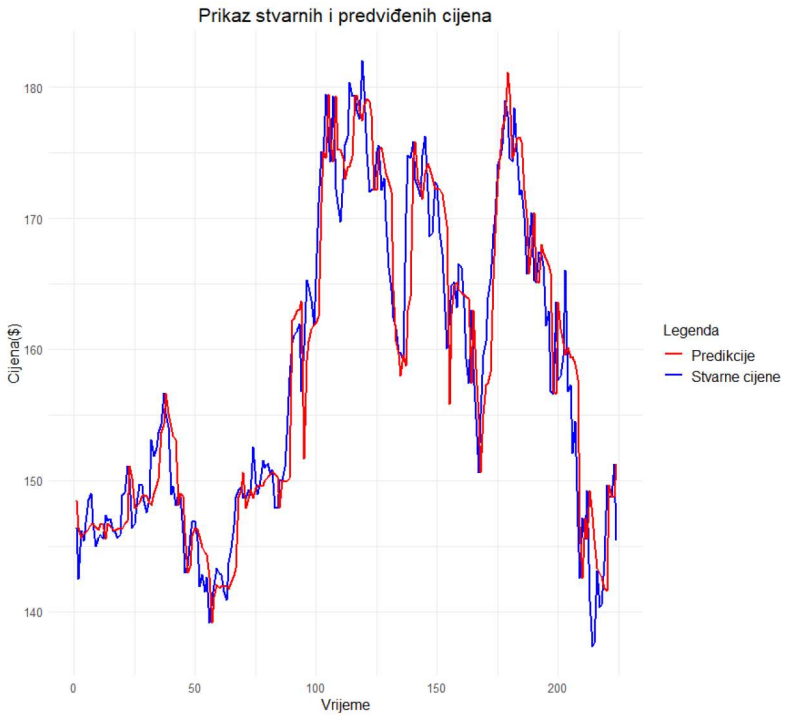
- **ARIMA model**: Završio je s najvišom vrijednošću portfelja te nadmašio i LSTM model i strategiju „kupnje i držanja“. Veći broj transakcija (42) ukazuje na agresivniji pristup.

- **Strategija kupnje i držanja**: Održala je gotovo početni iznos, ali je također doživjela blagi pad, sugerirajući da su cijene dionica bile manje predvidljive ili su padale. Kao i kod drugih razdoblja, nije bilo transakcija.

Grafički prikazi



Slika 4.5: Prikaz stvarnih i predviđenih cijena za Apple (AAPL), koristeći LSTM model.



Slika 4.6: Prikaz stvarnih i predviđenih cijena za Apple (AAPL), koristeći ARIMA model.

4.5 Zaključak

Analiza i usporedba LSTM i ARIMA modela na tri različita vremenska intervala za predviđanje cijena dionica Apple Inc. (AAPL) pružila je značajne uvide u učinkovitost ovih metoda i njihovu primjenjivost u stvarnom trgovanju.

Rezultati su pokazali da LSTM model, iako u nekim razdobljima nadmašuje ARIMA model, nije uvijek bio konzistentan u ostvarivanju najbolje konačne gotovine. Na primjer, u razdoblju od 2018. do 2022. godine ARIMA model je ostvario bolji financijski rezultat s konačnom gotovinom od $13,693.59, dok je LSTM ostvario $12,404.66. Međutim, LSTM model je imao manji broj transakcija (8), što upućuje na konzervativniji pristup u trgovanju u odnosu na ARIMA model, koji je generirao 42 transakcije u istom razdoblju. Ovi rezultati ukazuju na to da LSTM model bira ključne trenutke za trgovanje, izbjegavajući prečeste promjene, dok ARIMA model reagira na više tržišnih oscilacija, ali uz veći rizik i varijabilnost.

U razdoblju od 2020. do 2024. godine, LSTM model je ostvario veću konačnu gotovinu ($13,246.88) u usporedbi s ARIMA modelom ($11,890.45), pri čemu je također imao manji broj transakcija (10 za LSTM, 30 za ARIMA). Ovo sugerira da je LSTM model uspješniji u prepoznavanju značajnih tržišnih prilika i uočavanju dugoročnih obrazaca, dok ARIMA model generira veći broj transakcija s manjim financijskim učinkom.

S druge strane, strategija „kupnje i držanja“ bila je uspješnija u razdoblju od 2016. do 2020., gdje je ostvarila najveću konačnu vrijednost portfelja ($15,996.51), nadmašivši oba modela. Ovaj rezultat ukazuje na to da, u stabilnijim tržišnim uvjetima, jednostavne strategije poput „kupnje i držanja“ mogu biti jednako ili čak više profitabilne u odnosu na složenije modele predikcije poput LSTM i ARIMA.

Važan aspekt ove analize je i broj transakcija koje su oba modela ostvarila. LSTM model generalno je imao manji broj transakcija, odabirući ključne trenutke za trgovanje, što sugerira uravnoteženiji i konzervativniji pristup. S druge strane, ARIMA model bio je osjetljiviji na promjene tržišta, što je rezultiralo većim brojem transakcija, ali to nije uvijek rezultiralo boljim financijskim rezultatima.

Zaključno, rezultati sugeriraju da LSTM model, zbog svoje složenosti i sposobnosti učenja iz nelinearnih i dinamičnih podataka, pruža značajnu prednost u predviđanju cijena dionica, osobito u volatilnim uvjetima. Iako ARIMA model može pružiti korisne predikcije u stabilnijim uvjetima, njegov veći broj transakcija često nije donio bolji financijski rezultat. Kombinacija ovih modela i strategija, uzimajući u obzir individualne ciljeve i toleranciju na rizik, može pružiti sveobuhvatan pristup u trgovanju na financijskim tržištima.

Literatura

[1] J. Brownlee, *How to Choose Loss Functions When Training Deep Learning Neural Networks*, dostupno na: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks.

[2] G. Cybenko, *Approximation by Superpositions of a Sigmoidal Function*, Mathematics of Control, Signals and Systems, **2**(4), 1989, stranice 303-314.

[3] M. Duif, *Exploring How Neural Networks Work and Making Them Interactive*, dostupno na: https://towardsdatascience.com/exploring-how-neural-networks-work-and-making-them-interactive-ed67adbf9283.

[4] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016, stranice 166-317, 373-416.

[5] D. Grahovac, *Statističko učenje: Dijelovi predavanja*, 21. svibnja 2024, stranice 41-62, 85-90.

[6] C. F. Higham, D. J. Higham, *Deep Learning: An Introduction for Applied Mathematicians*, Proc. Indian Acad. Sci. (Math. Sci.), **113**(1), 2018, stranice 9-15.

[7] C. Olah, *Understanding LSTM Networks*, dostupno na: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[8] G. Rajgopal, *Deep Learning using Keras*, dostupno na: https://towardsdatascience.com/deep-learning-using-keras-3c08beda40c1.

[9] A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023, strasnice 389-416.

[10] *Yahoo Finance: Apple Inc. (AAPL)*, dostupno na: https://finance.yahoo.com/quote/AAPL/history?p=AAPL.

Sažetak

Ovaj rad istražuje primjenu Long Short-Term Memory (LSTM) mreža za predikciju cijena dionica. Analizom povijesnih podataka AAPL dionica, model pokazuje značajan potencijal u unapređenju točnosti predikcija i uspješnosti strategije trgovanja. Rad se fokusira na ključne komponente neuronskih mreža, proces treniranja i izazove poput prenaučenosti. Rezultati sugeriraju da LSTM mreže mogu pružiti vrijedne uvide za razvoj investicijskih strategija.

Ključne riječi

neuronske mreže, duboko učenje, long short-term memory (LSTM) mreže, predikcija cijena dionica.

Neural Networks and Stock Price Prediction

Summary

This paper explores the application of Long Short-Term Memory (LSTM) networks for stock price prediction. Through the analysis of historical AAPL stock data, the model demonstrates significant potential in improving prediction accuracy and trading strategy performance. The work focuses on key components of neural networks, the training process, and challenges such as overfitting. The results suggest that LSTM networks can provide valuable insights for the development of investment strategies.

Keywords

neural networks, deep learning, LSTM, stock price prediction.

Životopis

Rođen sam 29. prosinca 1997. godine u Osijeku. Osnovno obrazovanje stekao sam u Osnovnoj školi Ivana Filipovića u Osijeku. Nakon toga, upisao sam Opću gimnaziju, no u drugoj godini odlučio sam se prebaciti u Matematičku gimnaziju u Osijeku. Srednju školu završio sam 2017. godine. Iste godine, započeo sam preddiplomski sveučilišni studij Matematike na Odjelu za matematiku Sveučilišta J. J. Strossmayera u Osijeku. Po završetku studija 2021. godine stekao sam titulu prvostupnika matematike. Iste godine, upisao sam diplomski studij matematike, s usmjerenjem na Financijsku matematiku i statistiku, na Fakultetu primijenjene matematike i informatike, koji je tada bio poznat kao Odjel za matematiku, Sveučilišta J. J. Strossmayera u Osijeku.