Predviđanje kretanja cijena dionica primjenom LSTM rekurentnih neuronskih mreža

Žohar, Hrvoje

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Hrvoje Žohar

Predviđanje kretanja cijena dionica primjenom LSTM rekurentnih neuronskih mreža

Završni rad

Osijek, 2024.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Hrvoje Žohar

Predviđanje kretanja cijena dionica primjenom LSTM rekurentnih neuronskih mreža

Završni rad

Mentor:
izv. prof. dr. sc. Domagoj Matijević

Osijek, 2024.

Sažetak:
U ovom radu istražujemo primjenu long short-term memory rekurentnih neuronskih mreža na predviđanje cijene ZB trend investicijskog fonda Zagrebačke banke koristeći cijene dionica tvrtki sastavnica fonda, tehničku analizu i Fourierove transformacije. Fokusiramo se na sastavne metode tehničke analize te dizajn i treniranje neuronskih mreža. Rezultati pokazuju da je model sposoban pronaći obrasce u povijesnim podacima i dati uvid u buduće kretnje za razvoj investicijskih strategija.

Ključne riječi: RNN, LSTM, dionice, tehnička analiza, Fourierova transformacija, strojno učenje, umjetna inteligencija, Tensorflow

Predicting stock price movement using LSTM recurrent neural networks

Abstract:
In this thesis we explore the applications of long short-term memory recurrent neural networks for predicting price movements Zagrebačka bank's ZB trend mutually traded fund using the prices of the fund's constituent company stock positions, technical analysis and Fourier transforms. We focus on fundamental technical analysis methods and the design and training process of neural networks. Our results show that the model is capable of capturing patterns in the data give insight into future price movements to develop investing strategies.

Keywords: RNN, LSTM, stocks, technical analysis, Fourier transforms, machine learning, AI

Sadržaj

1. Uvod

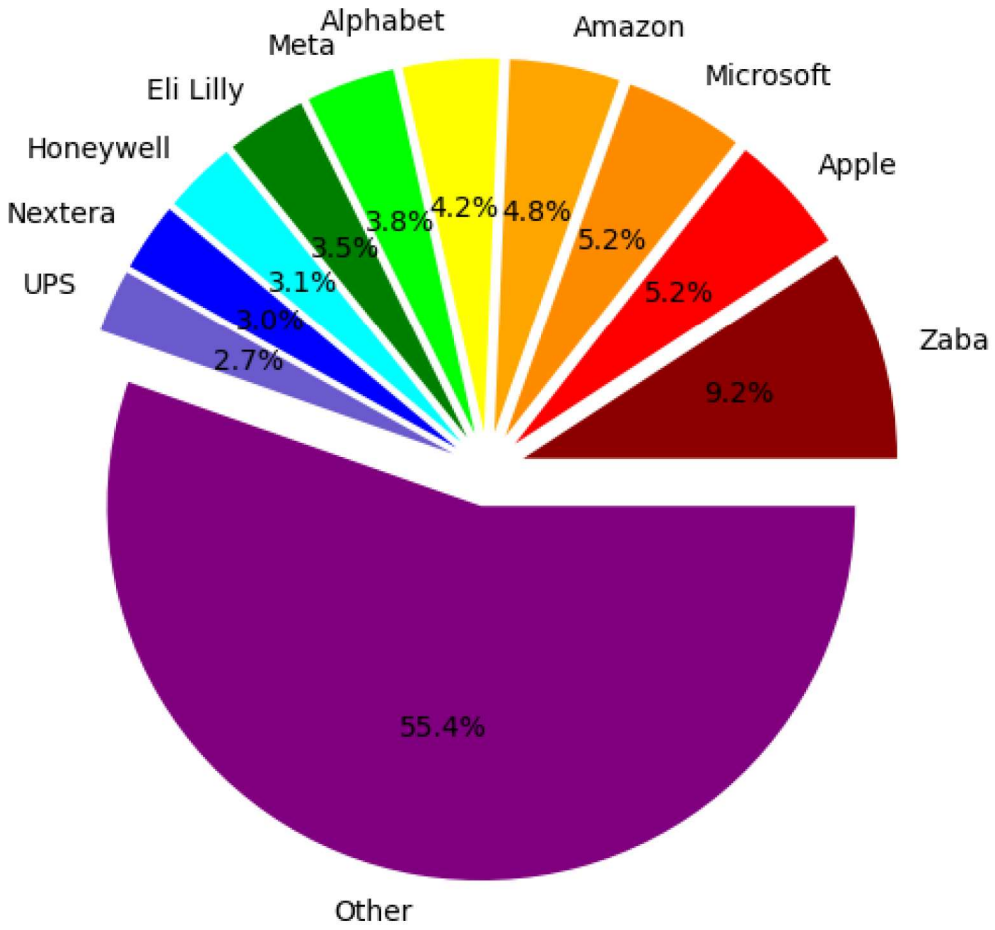
Dionice su vrijednosni papiri koji predstavljaju udio vlasništva u nekom dioničkom društvu ili investicijskom fondu[9]. Investitor dionicom ostvaruje pravo na udio zarade, prinosa od prodaje imovine ili pravo glasanja o poslovnim odlukama tvrtke proporcionalno postotku dionica koje posjeduje[10]. Tvrtke izdaju i prodaju dionice u zamjenu za kapital kojima se trguje privatno ili na burzama vrijednosnih papira (engl. *stock exchange*) kao što su Zagrebačka burza, New York stock exchange, London stock exchange i Frankfurter Wertpapierbörse.

Vrijednost dionice ovisi o više faktora. Fundamentalna analiza je statistička metoda kojom se proučava poslovna usješnost neke tvrtke u svrhu određivanja trenutnih i budućih vrijednosti njenih dionica[11]. Dionice koje su podcijenjene stvaraju potražnju i njihova cijena raste, a dionice koje su precijenjene stvaraju ponudu i njihova cijena pada, dakle vrijednost dionice također ovisi o ponudi i potražnji.[12] Tehnička analiza je statistička metoda koja proučava povijesne podatke kretanja cijena i trgovnih volumena dionica u svrhu određivanja njihovih trenutnih i budućih vrijednosti[13].

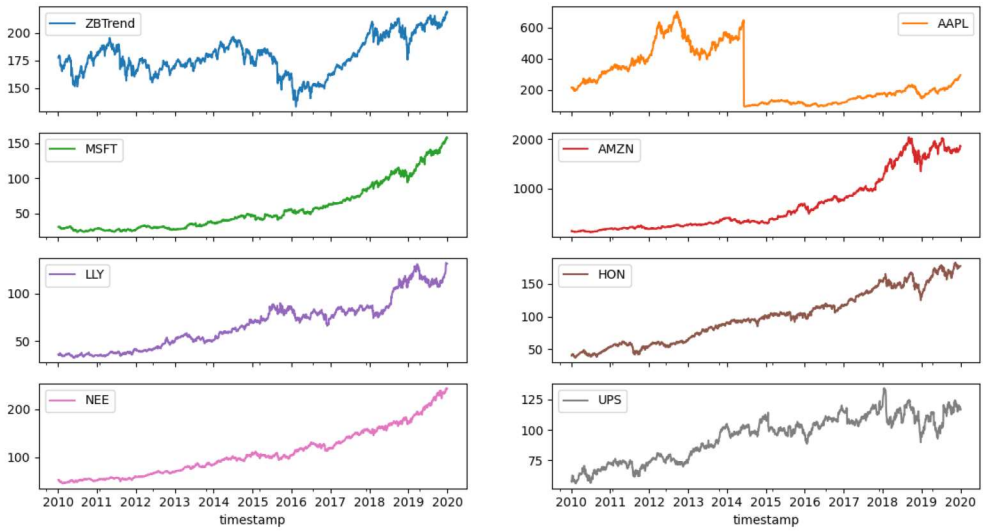
Strojno učenje je disciplina koja proučava statističke modele koji mogu biti trenirani na skupu podataka iz neke domene i generalizirati svoje znanje na neviđene podatke. Ovi su modeli pronašli primjenu u računalnom jezikoslovlju, generiranju medija poput slika, zvuka i videa, računalnom vidu, robotici, samoupravljivim automobilima, video igrama, dizajnu, ekonomiji i medicini[14]. U ovom radu ćemo proučiti korištenje modela rekurentne neuronske mreže u tehničkoj analizi ZB Trend investicijskog fonda Zagrebačke banke.

2. Podaci

Želimo znati hoće li cijena fonda porasti ili pasti. Iz tog razloga važno je iskorisitit što više podataka možemo. Koristit ćemo podatke o cijeni dionice fonda između 2010. i 2020. godine. Međutim, investicijski fondovi ne postoje u vakumu. Mjesečni izvještaj fonda ZB trend za rujan 2024. navodi deset najvećih pozicija fonda kao Zagrebačku banku (9.19%), Apple inc. (AAPL, 5.24%), Microsoft corp. (MSFT, 5.15%), Amazon.com inc. (AMZN, 4.76%), Alphabet inc. (GOOG, 4.16%), META platforms inc. (META, 3.83%), Eli Lilly & Co (LLY, 3.46%), Honeywell international inc. (HON, 3.14%), Nexera energy inc. (NEE, 2.97%) i United parcel service inc. (UPS, 2.72%)[2].



Slika 1: Kružni dijagram pozicija fonda ZB trend u rujnu 2024.

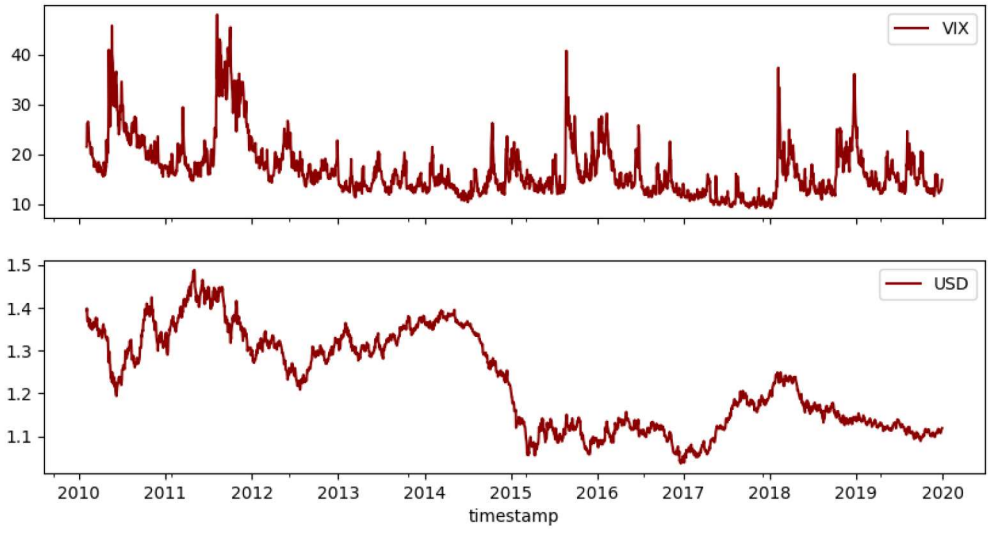


Slika 2: Povijesni podatci cijena dionica

Koristit ćemo povijesne podatke cijena dionica za Apple, Microsoft, Amazon, Eli Lilly, Honeywell, Nexeru i UPS za taj vremenski period.

Također ćemo koristit VIX, koji je mjera volatilnosti tržišta. Izveden je iz brzine promjene cijene udjela u S&P500 fondu i predstavlja razinu nesigurnosti investitora.

Trgovanje financijskim instrumentima, pogotovo na međunarodnim tržištima, ovisi o tečajevima raznih valuta[1]. U ovom skupu podataka koristit ćemo dnevni tečaj američkog dolara u eurima.



Slika 3: Grafovi indeksa volatilnosti i tečaja američkog dolara

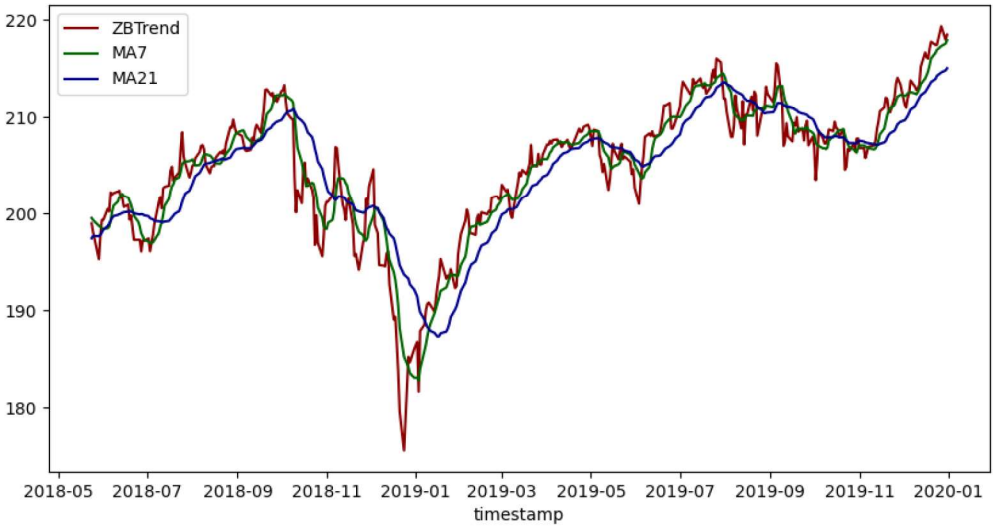
2.1. Tehnički indikatori

2.1.1. Klizni prosjek

Definicija 2.1. Klizni prosjek
Neka je $(a_n)_n$ niz realnih brojeva. Klizni prosjek (engl. *moving average*) širine m niza $(a_n)_n$ je niz $(MA_n)_n$ definiran kao:

$$\begin{aligned} MA_k &= \frac{1}{m}(a_{k-m+1} + a_{k-m+2} + ... + a_{k-1} + a_k,\ k \geq m \\ &= \frac{1}{m} \sum_{i=k-m+1}^{k} a_i,\ k \geq m \end{aligned} \tag{1}$$

To jest, svaki element ovog niza je prosjek zadnjih m elemenata niza $(a_n)_n$. Klizni pomak se ponaša kao nisko propusni filter koji izglađuje kratkoročne oscilacije u podacima kako bi istaknuo skrivene dugoročne trendove[15][5].



Slika 4: Graf cijene zadnjih 400 dana s kliznim prosjekom širine 7 i 21

Napomena 2.1. Kod računanja kliznog prosjeka možemo koristiti činjenicu da

$$\begin{aligned} MA_{k+1} &= \frac{1}{m}\sum_{i=k-m+2}^{k+1} a_i \\ &= \frac{1}{m}(a_{k-m+2} + a_{k-m+3} + ... + a_k + a_{k+1}) \\ &= \frac{1}{m}(a_{k-m+2} + a_{k-m+3} + ... + a_k + a_{k+1} + a_{k-m+1} - a_{k-m+1}) \\ &= \frac{1}{m}(\sum_{i=k-m+1}^{k} a_i + a_{k+1} - a_{k-m+1}) \\ &= MA_k + \frac{a_{k+1} - a_{k-m+1}}{m}. \end{aligned}$$

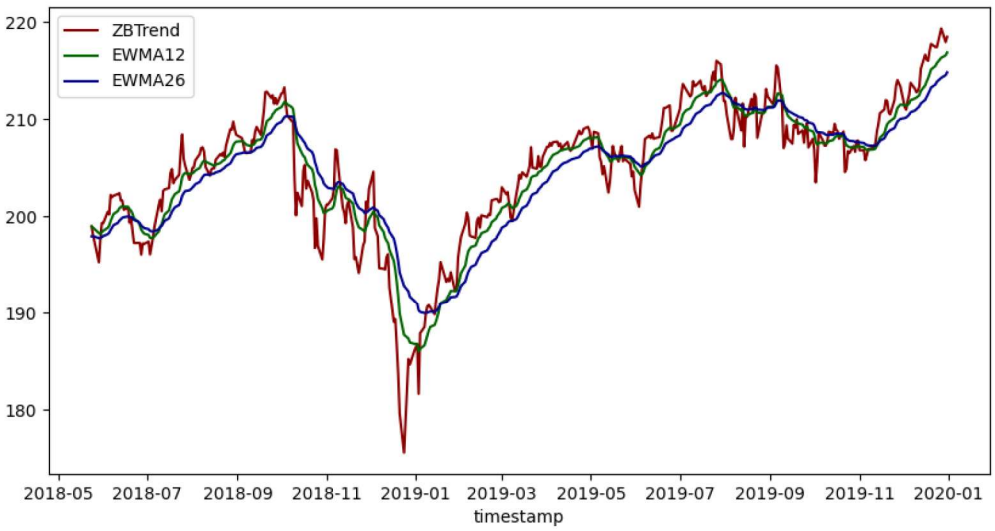
Napomena 2.2. Klizni prosjek nije definiran za prvih m - 1 elemenata niza jer nemamo dovoljno podataka iz prošlosti.

2.1.2. Eksponencijalni klizni prosjek

Definicija 2.2. Eksponencijalni klizni prosjek Neka je $(a_n)_n$ niz realnih brojeva. Eksponencijalni klizni prosjek (engl. *exponential (weighted) moving average*) niza $(a_n)_n$ je niz $(E_n)_n$ definiran kao[16][6]:

$$E_0 = a_0$$
$$E_k = \alpha a_k + (1 - \alpha) E_{k-1}$$

gdje je $\alpha \in (0, 1)$ koeficijent zaglađivanja, odnosno proporcija značajnosti koju dajemo trenutnom podatku a_k. Koeficijent zaglađivanja može biti zadan eksplicitno ili koristeći širinu m tako da $\alpha = \frac{2}{m+1}$. Eksponencijalni klizni prosjek zadan širinom m ekvivalentan je kliznom prosjeku širine m. Pišemo $E^{(m)}$.



Slika 5: Graf cijene zadnjih 400 dana s eksponencijalnim kliznim prosjekom širine 12 i 26

2.1.3. Konvergentnost i divergentnost kliznog prosjeka (MACD)

Definicija 2.3. MACD
Neka je $(a_n)_n$ niz realnih brojeva. MACD(p, q, r), $p, q, r \in \mathbb{N}$ je uređena dvojka (v, s), gdje je v niz razlika eksponencijalnih kliznih prosjeka širine p i q niza a, a s je eksponencijalni klizni prosjek širine r niza v.

$$v = E^{(p)}(a) - E^{(q)}(a)$$
$$s = E^{(r)}(v)$$

MACD je indikator u tehničkoj analizi koji se može interpretirati na više načina. Na primjer, presjecanje nizova v i s se može shvatiti kao najava za akceleraciju trenda u cijeni. Ako niz v presjeca prema dolje, to znači da će cijena početi padati, a ako prema gore, da će početi rasti. Također, ako niz v presječe x-os, to predstavlja mijenjanje smjera trenda[17][7].

Slika 6: Graf cijene zadnjih 400 dana uz MACD(12, 26, 9)

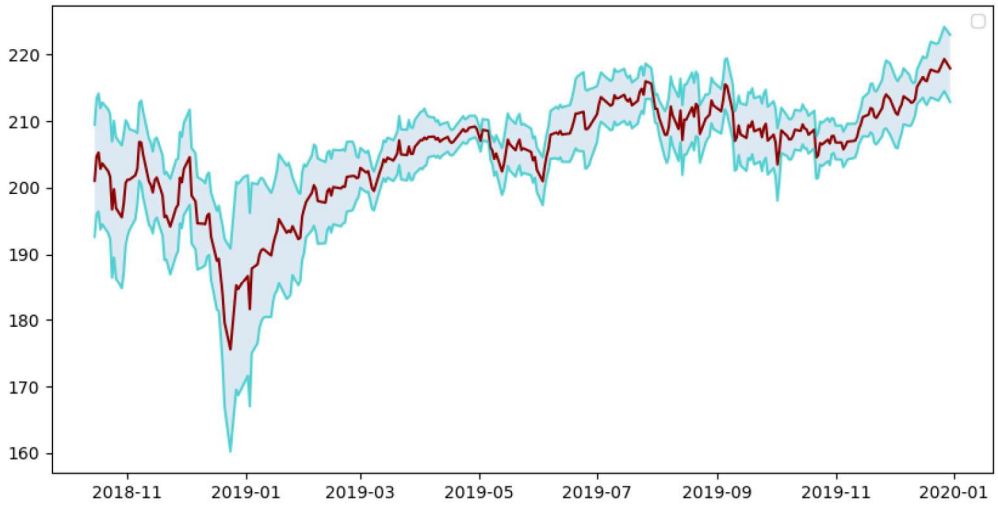
2.1.4. Bollingerov pojas

Analogno kliznom prosjeku, možemo definirati kliznu varijancu i standardnu devijaciju.

Definicija 2.4. Bollingerov pojas Neka je $(a_n)_n$ niz realnih brojeva te $(D_n^{(m)})_n$ klizna standardna devijacija širine m niza $(a_n)_n$). Tada definiramo Bollingerov pojas kao uređeni par (L, U), gdje su:

$$L = a - \lambda D^{(m)}$$
$$U = a + \lambda D^{(m)},$$

gdje je λ neki pozitivan realan broj, najčešće 2[18][8]. Bollingerov pojas definira područje kretanje cijene. Ako je cijena blizu gornje granice, to može biti indikacija da će početi padati i obrnuto.



Slika 7: Graf cijene zadnjih 300 dana uz Bollingerov pojas ($\lambda = 2$)

2.1.5. Diskretna Fourierova transformacija

Cijena dionice fonda je vremenski niz koji možemo shvatiti kao diskretni signal, tj. diskretnu funkciju u vremenskoj domeni[4]. Koristeći diskretnu Fourierovu transofmaciju, možemo ga preslikati u frekvencijsku domenu.

Definicija 2.5. Diskretna Fourierova transformacija
Neka je $x_0, x_1, x_2, ..., x_{N-2}, x_{N-1}$ konačan niz N kompleksnih brojeva. Diskretna Fourierova transformacija (DFT) je preslikavanje ovog niza u drugi niz kompleksnih brojeva $X_0, X_1, X_2, ..., X_{n-2}, X_{n-1}$ zadano formulom:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{k}{N} n} \tag{2}$$

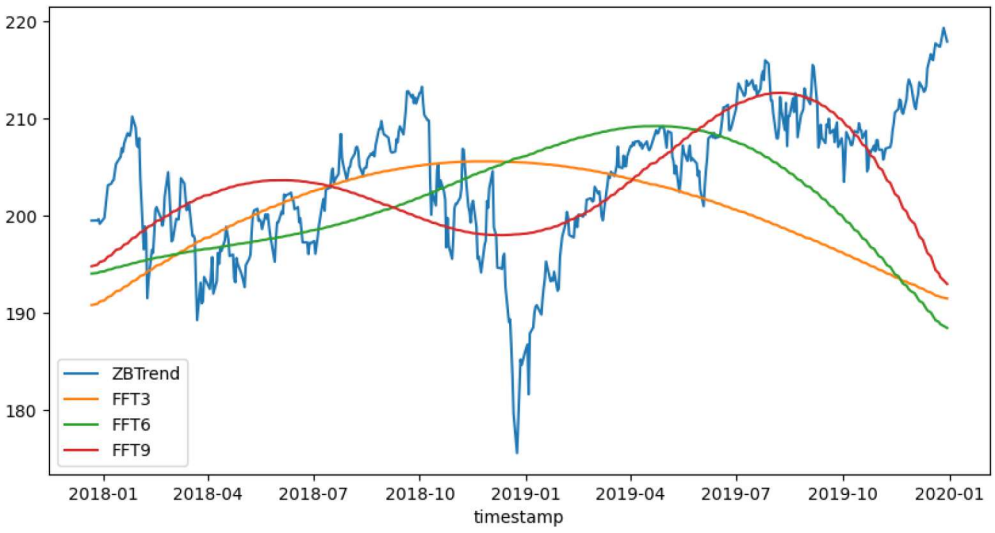
Za svaki X_k jednadžba (2) računa međusobnu korelaciju ulaznog signala s kompleksnom sinusoidom frekvencije $\frac{k}{N}$, tj. doprinos te sinusoide u signalu. Za DFT postoji inverz zadan formulom:

$$x_k = \frac{1}{N} \sum_{n=0}^{N-1} X_n e^{i2\pi \frac{k}{N} n}.$$

Pišemo $X_k = \mathcal{F}(\{x_n\})_k$ i $x_k = \mathcal{F}^{-1}(\{X_k\})_n$.

Ako interpretiramo cijenu kao diskretni signal, tj. niz u vremenskoj domeni, možemo primjeniti diskretnu Fourierovu transformaciju kako bi ga transformirali u niz u frekvencijskoj domeni. Veće frekvencije ovog signala možemo shvatiti kao šum koji prikriva stvarno kretanje cijene, pa odbacivanjem tih komponenti možemo sintetizirati niz koji bolje ukazuje na trendove u tržištu. Ovisno o broju komponenti kojih zadržimo, više ćemo istaknuti kratkoročne ili dugoročne trendove. U ovom radu se koriste nizovi sa 3, 6 i 9 komponenti.

Ako primjenimo DFT na cijenu te u rezultantnom nizu odbacimo sve osim prvih m i zadnjih m elemenata, pa zatim na taj niz primjenimo inverznu DFT, dobit ćemo kretanje cijene s izbačenom velikom količinom šuma, koje bolje ukazuje na stvarno kretanje cijene i ističe trendove u tržištu. U ovom radu se koriste $m = 3, 6, 9$.



Slika 8: Graf cijene zadnjih 500 dana sa sintetiziranim nizovima iz Fourierove transformacije s 3, 6 i 9 komponenti

3. Umjetne neuronske mreže

3.1. Neuron

Umjetna neuronska mreža se sastoji od umreženih jedinica koje zovemo *neuroni*[19] (po uzoru na živčane stanice u živim bićima). Svaki neuron ima n ulaza na kojima prima signale od drugih neurona koji su spojeni s njim i m izlaza na kojima šalje signal drugim neuronima s kojima je spojen. Između dva spojena neurona je usmjereni brid kojemu je pridružena težina koja predstavlja jakost veze između ta dva neurona. Neka su $x = (x_1, x_2, ...x_n)$ signali na ulazima nekog neurona i $w = (w_1, w_2, ..., w_n)$ težine na pripadnim bridovima te w_b pristranost (engl. *bias*) tog neurona. Izlazni signal y ovog neurona je tada:

$$\begin{aligned} y &= h(w_b + w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n) \\ &= h(w_b + w \cdot x), \end{aligned}$$

gdje je $h : \mathbb{R}^n \to \mathbb{R}$ neka, najčešće nelinearna, *aktivacijska* funkcija.

Neuroni su najčešće organizirani u slojeve, svaki od kojih može imati drugu zadaću. Neuronske mreže se najčešće sastoje od jednog *ulaznog* sloja, koji ima onoliko neurona koliko ulazni podatci imaju značajki (različitih skalarnih vrijednosti koje ih opisuju), jednog ili više skrivenih slojeva koji imaju proizvoljan broj neurona i jednog izlaznog sloja, koji može imati jedan ili više neurona, ovisno o primjeni specifične neuronske mreže. Neuronske mreže sa dva ili više skrivenih slojeva se zovu *duboke neuronske mreže*. Neuronske mreže imaju široku primjenu u područjima poput računalnog jezikoslovlja, generiranju medija, računalnom vidu, robotici, samoupravljivim automobilima, video igrama, dizajnu, ekonomiji i medicini.

3.2. Aktivacijske funkcije

Aktivacijske funkcije imaju velik utjecaj na ponašanje i treniranje neuronske mreže. Važno je da aktivacijske funkcije budu nelinearne jer inače bi izlaz neuronske mreže bio kompozicija linearnih funkcija ulaznih značaji, dakle linearna funkcija, što bi znatno ograničilo vrste problema koje neuronske mreže mogu riješiti. Nelinearne aktivacijske funkcije omogućuju neuronskim mrežama da nauče kompleksne nelinearne veze između ulaznih podataka[20].

3.2.1. ReLu aktivacijska funkcija

ReLu (Rectified Linear units) je funkcija zadana formulom

$$f(x) = max\{0, x\}.$$

ReLu je najčešće korištena aktivacijska funkcija u dubokim neuronskim mrežama zbog svoje jednostavnosti i dobrih rezultata koje postiže u praksi.

3.2.2. Leaky ReLu

Leaky ReLu je varijacija na ReLu definirana formulom:

$$f(x) = max\{0.01x, x\},$$

Za razliku od ReLu, gradijent leaky ReLu je uvijek pozitivan, što pomaže kod problema nestajućih gradijenata.

3.2.3. GeLu

GeLU (Gaussian-error Linear unit) je definirana formulom:

$$f(x) = x \cdot \Phi(x),$$

gdje je $\Phi(x)$ funkcija distribucije standardne normalne distribucije.

3.2.4. tanh

Tangens hiperbolni je također jedna od najčešće korištenih aktivacijskih funkcija. Računamo ju formulom:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}.$$

Osim jednostavnosti svoje derivacije, jedna je od glavnih prednosti tanh funkcije to što ima horizontalne asimptote na $y \pm 1$, što pomaže kod problema eksplodirajućih gradijenata.

3.2.5. Sigmoid

Sigmoid (ili logistička) funkcija je funkcija definirana formulom:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

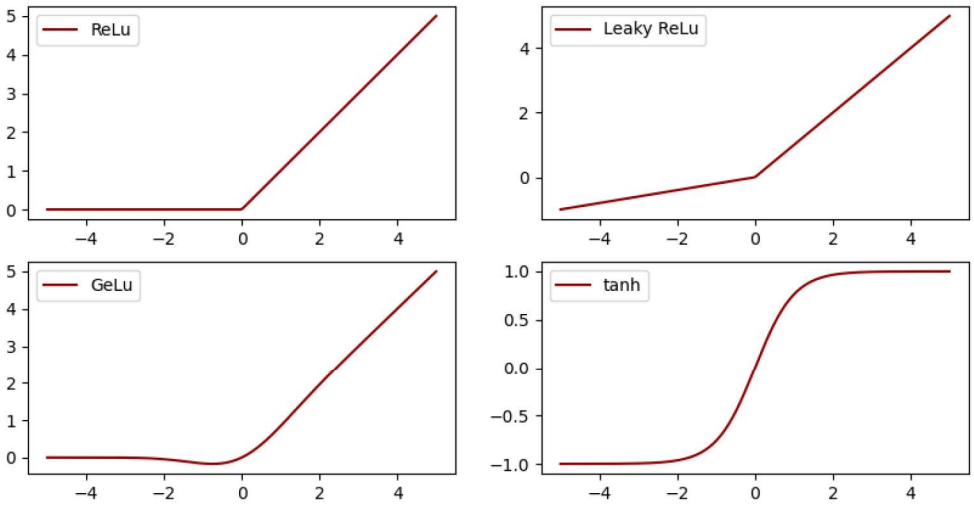
Najčešće se koristi u problemima binarne klasifikacije.

3.2.6. Softmax

Softmax je posebna funkcija koja se najčešće koristi u izlaznom sloju neuronske mreže kod problemima klasifikacije s više klasa. Ako imamo problem klasifikacije s N klasa, zadnji skriveni sloj će imati N neurona, a izlazni sloj će na signale tih neurona djelovati na sljedeći način:

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=0}^{N} e^{x_j}},$$

gdje je x vektor signala. Svaki od $\sigma(x)_i$ je signal x_i normaliziran na interval $(0, 1)$ proporcionalno e^{x_i}, stoga je vektor $\sigma(x)$ vjerojatnosna distribucija za klasu trenutnog ulaza.



Slika 9: Grafovi ReLu, leaky ReLu, GeLu i tanh aktivacijskih funkcija

3.3. Treniranje neuronske mreže

Već smo spomenuli da izlazni signal neurona ovisi o težinama bridova. Ove težine su upravo ono što omogućuje neuronskoj mreži da uči od podataka, na način da mijenja težine bridova tako da minimizira neku funkciju cilja, na primjer prosjek kvadrata razlika (engl. MSE - *mean squared error*) ili prosjek apsolutnih razlika (engl. MAE - *mean absolute error*). Proučimo jednostavan primjer neuronske mreže s ulaznim slojem s N neurona i izlaznim slojem s jednim neuronom i nekom aktivacijskom funkcijom h. Funkcija cilja ove mreže može biti, na primjer:

$$J(x) = \sum_{i=0}^{N} (y_i - h(b + \Theta^T \cdot x_i))^2$$

ili

$$J(x) = \sum_{i=0}^{N} |y_i - h(b + \Theta^T \cdot x_i)|.$$

Tijekom treniranja, cilj nam je pronaći parametre b i Θ koji minimiziraju funkciju cilja:

$$\min_{\Theta, b} J(x)$$

U praksi funkcije cilja često nisu konveksne, pa se traženje minimuma provodi iterativnim metodama poput gradijentnog spusta.

3.3.1. Gradijentni spust

Metoda gradientnog spusta je iterativna metoda minimizacije u kojoj računamo gradijent funkcije cilja i ažuriramo parametre modela u suprotnom smjeru od njega. Naime,

$$\Theta \to \Theta - \alpha \nabla J(x),$$

gdje je α hiperparametar koji zovemo stopom učenja (engl. *learning rate*). Ovaj hiperparametar je vrlo važan jer ima velik utjecaj na vrijeme treniranja i kvalitetu rezultantnog modela, iz kojeg razloga postoje razni algoritmi za računanje dobre stope učenja, pa čak i mijenjanje stope učenja tijekom treniranja.

Međutim, metoda gradijentnog spusta ima jedan veliki nedostatak. Za računanje gradijenta moramo uračunati doprinos svakog podatka iz skupa podataka, što je nepraktično za velike skupove. Umjesto toga, možemo računati gradijent na podskupu podataka.

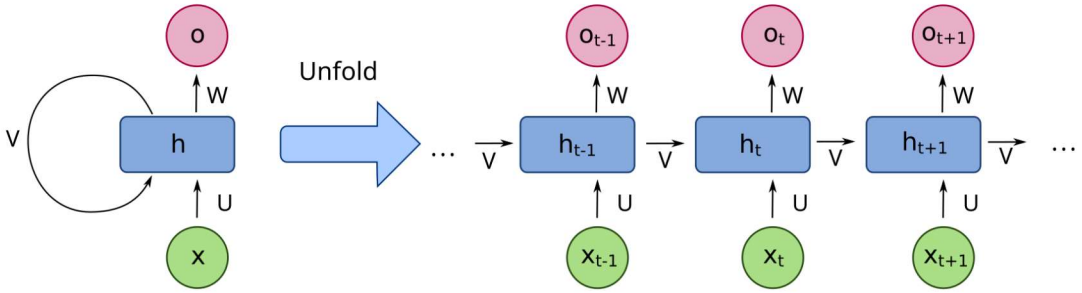
3.3.2. Stohastički gradijentni spust

Metoda stohastičkog gradijentnog spusta je stohastička aproksimacija gradijentnog spusta[21]. Ideja je da skup podataka nasumično podijelimo na podskupove (engl. *batch*) te u svakoj iteraciji gradijentnog spusta računamo gradijent samo na jednom od tih podskupova. Ovime smanjujemo količinu računanja potrebnu za ažuriranje parametara modela, pod cijenu toga da smo potencijalno produljili vrijeme treniranja jer nas gradijent više ne vodi nužno u smjeru najbržeg rasta.

3.4. Rekurentne neuronske mreže

Rekurentne neuronske mreže su klasa umjetnih neuronskih mreža koje uz ulazne signale neurona imaju još i skriveno stanje koje je funkcija prošlog skrivenog stanja i trenutnog

ulaza[22]. Skriveno stanje se ponaša kao oblik pamćenja, što omogućava modelu orijentiranje podataka u vremenu, što je pogodno za rad s vremenskim nizovima, tekstom ili drugim sekvencijalnim podacima. Glavni nedostatak klasičnih RNN-ova je problem nestajućih gra-



Slika 10: Grafički prikaz RNN-a. Izvor: [3]

dijenata. Naime, težine vezane uz neki podatak relativno brzo iščezavaju prema nuli s vremenom, što znači da klasični RNN "nema vrlo dobro pamćenje", pamti samo relativno nedavne podatke. Ovaj je problem riješen uvođenjem LSTM ćelija krajem 90-ih godina.

3.4.1. LSTM ćelije

LSTM (Long short-term memory) ćelije se, uz skriveno stanje, sastoji stanja ćelije c_t i troja vrata: zaboravna vrata f_t, ulazna vrata i_t i izlazna vrata o_t. Ova se vrata ažuriraju po sljedećim jednadžbama:

$$
\begin{aligned}
f_t &= \sigma(\Theta_f x_t + \Psi_f h_{t-1} + b_f) \\
i_t &= \sigma(\Theta_i x_t + \Psi_i h_{t-1} + b_i) \\
o_t &= \sigma(\Theta_o x_t + \Psi_o h_{t-1} + b_o) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tanh(\Theta_c x_t + \Psi_c h_{t-1} + b_c) \\
h_t &= o_t \cdot c_t.
\end{aligned}
$$

Zaboravna vrata odlučuju koje informacije više nisu važne i mogu se zaboraviti pridruživanjem vrijednosti između 0 i 1, gdje je 0 predstavlja zaboravljanje, a 1 pamćenje. Ulazna vrata odlučuju koje ulazne informacije su važne i trebaju biti dodane u stanje ćelije (zapamćene) na isti način kao zaboravna vrata. Izlazna vrata odlučuju koje informacije trebaju biti uključene u trenutni izlaz modela, a koje sačuvane za buduće izlaze[23].

4. Model

Za dizajn i treniranje modela koristio sam Tensorflow, Google-ovu open-source Python biblioteku za strojno učenje. Koristio sam sekvencijalni model, koji omogućuje definiranje modela sloj po sloj.

Počeo sam dizajn modela s pripremon podataka. Podijelio sam podatke na skupove za treniranje i validaciju u omjeru $9:1$. Probao sam i druge omjere, kao na primjer $8:2$, ali ovaj se pokazao najboljim. Zatim sam generirao nizove od m uzastopnih dana koji će biti ulazi neuronskoj mreži, a model treba predvidjeti cijenu $m+1$-vog dana. Eksperimentirao sam s različitim vrijednostima broja m između 3 i 60. Najbolje sam rezultate dobio s relativno malim m-ovima. Odabrao sam $m = 5$. Kod za pripremu podataka:

```
import pandas as pd;
import numpy  as np;
import random;
# m
SEQUENCE_LENGTH = 5;

def PreprocessFeatures(features: pd.DataFrame, shuffle = True):
    sequential_data = [];
    previous_days = deque(maxlen = SEQUENCE_LENGTH);

    for row in features.values:
        # Izostavi zadnji stupac, to je cijena koju
        # model treba predvidjeti.
        previous_days.append(row[ : -1]);
        if len(previous_days) == SEQUENCE_LENGTH:
            sequential_data.append((np.array(previous_days), row[-1]));

    if shuffle:
        random.shuffle(sequential_data);

    X = [];
    y = [];

    for sequence, target in sequential_data:
        X.append(sequence);
        y.append(target);

    return np.array(X), np.array(y);

train_size = int(0.9 * features.shape[0]);
training_features = features.iloc[ : train_size];
testing_features  = features.iloc[train_size : ];

X_train, y_train = PreprocessFeatures(training_features);
X_test , y_test  = PreprocessFeatures(testing_features);
```

Listing 1: Kod sa pripremu podataka

Isprobao sam različite dizajnove modela dok nisam pronašao jedan koji se pokazao obećavajućim. Neki od dizajnova koje sam probao su:

1. Plitka mreža s jednim skrivenim LSTM slojem sa 128, 256 i 512 neurona
2. Plitka mreža s dva skrivena LSTM sloja sa 128 i 256 neurona
3. Plitka mreža s jednim skrivenim LSTM slojem i jednim skrivenim gustim slojem
4. Duboka mreža sa tri skrivena LSTM sloja sa 64 i 128 neurona i jednim skrivenim gustim slojem sa 32 i 64 neurona

Za svaki od ovih primjera sam probao različite kombinacije aktivacijskih funkcija, dropout slojeva, kroz koje mreža zaboravi dio naučenih težina kako bi se spriječilo pretreniranje, uz koeficijente zaboravljanja između 0.05 i 0.5 te slojeve za normalizaciju batch-eva, koji spriječavaju problem eksplodiranja parametara. Na kraju sam završio sa sljedećim modelom:

```
import tensorflow as tf;
from tensorflow.keras.models import Sequential;
from tensorflow.keras.layers import Input, LSTM, Dropout, Dense,
    BatchNormalization;

DROPOUT_RATIO = 0.35;

model = Sequential();
model.add(Input(shape = (X_train.shape[1 :])));

model.add(LSTM(36, activation='relu', return_sequences=True));
model.add(Dropout(DROPOUT_RATIO));
model.add(BatchNormalization());

model.add(LSTM(36, activation='tanh', return_sequences=True));
model.add(Dropout(DROPOUT_RATIO));
model.add(BatchNormalization());

model.add(LSTM(32, activation='tanh', return_sequences=True));
model.add(Dropout(DROPOUT_RATIO));
model.add(BatchNormalization());

model.add(LSTM(32, activation='relu'));
model.add(Dropout(DROPOUT_RATIO));
model.add(BatchNormalization());
# Izlazni sloj.
model.add(Dense(1, activation='relu'));
```

Listing 2: Kod za definiranje modela

Svaki sam dizajn modela isprobao trenirati koristeći *mean squared error* i *mean absolute error* funkcije cilja. Pronašao sam da *mean absolute error* u većini slučajeva postiže bolje rezultate. Za optimizaciju stope učenja koristio sam ADAM algoritam s različitim početnim stopama između 10^{-3} i 10^{-4}. Primjetio sam da su optimalne početne stope učenja vrlo osjetljive na dizajn modela te sam ih morao mijenjati nakon gotovo bilo kakve izmjene, bilo broja m, broja neurona u nekom sloju, aktivacijske funkcije u nekom sloju ili čak funkcije cilja. Također sam pokušao koristiti ExponentialDecay scheduler iz Tensorflow-a, koji eksponencijalno prigušuje stopu učenja s vremenom, ali nisam imao zadovoljavajuće rezultate. Na kraju sam odabrao sljedeću implementaciju:

```
model.compile(loss = 'mae', optimizer = tf.keras.optimizers.Adam(
    learning_rate = 4e-4
));
```

Listing 3: Kod za kompajliranje modela

Tijekom treniranja modela koristio sam stohastički gradijentni spust. Eksperimentirao sam s veličinama batch-eva od 128, 64, 32 i 16. Najbolje sam rezultate postigao s batch-evima veličine 16. Svaki sam model trenirao kroz 512 epoha, u svakoj od kojih je model prošao kroz cijeli skup podataka za treniranje. Pratio sam vrijednost funkcije cilja na skupu za testiranje kroz svaku epohu i spremio parametre koji su do sada postigli minimalnu vrijednost. Ovo osigurava da model ne postane pretreniran.

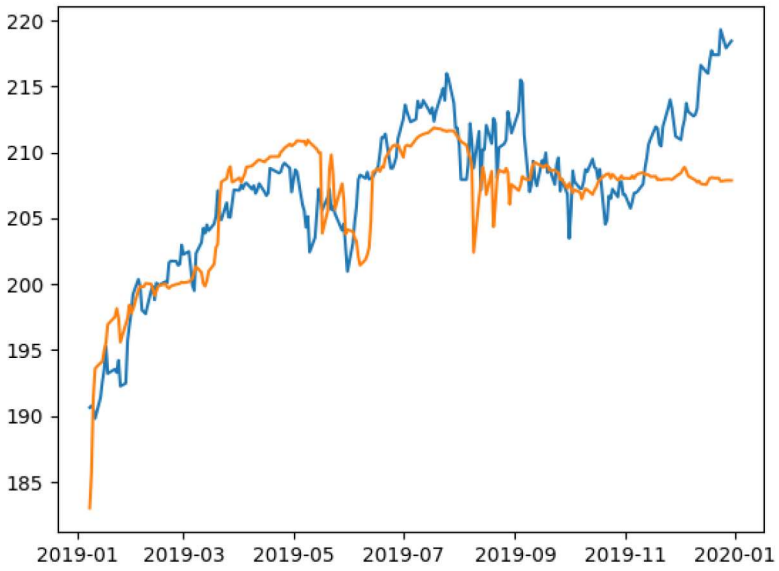
```
from tensorflow.keras.callbacks import ModelCheckpoint;
# Spremi parametre modela koji ima minimalnu funkciju cilja na skupu za
    testiranje.
checkpoint  = ModelCheckpoint("models/checkpoint.model.keras", monitor = '
    val_loss', verbose = 0, save_best_only = True, mode = 'min');

model.fit(
    X_train, y_train,
    batch_size = 16,
    epochs = 512,
    validation_data = (X_test, y_test),
    callbacks = [checkpoint]);
```

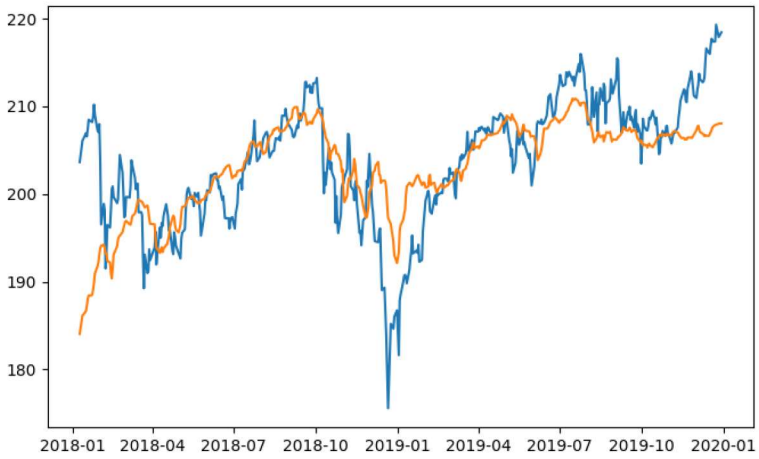
Listing 4: Kod za treniranje modela

5. Zaključak

Treniranje neuronskih mreža je po svojoj prirodi slučajan proces. Svaki sam dizajn modela trenirao više puta kako bih dobio najbolje rezultate. Ovo su neki koje sam izdvojio:

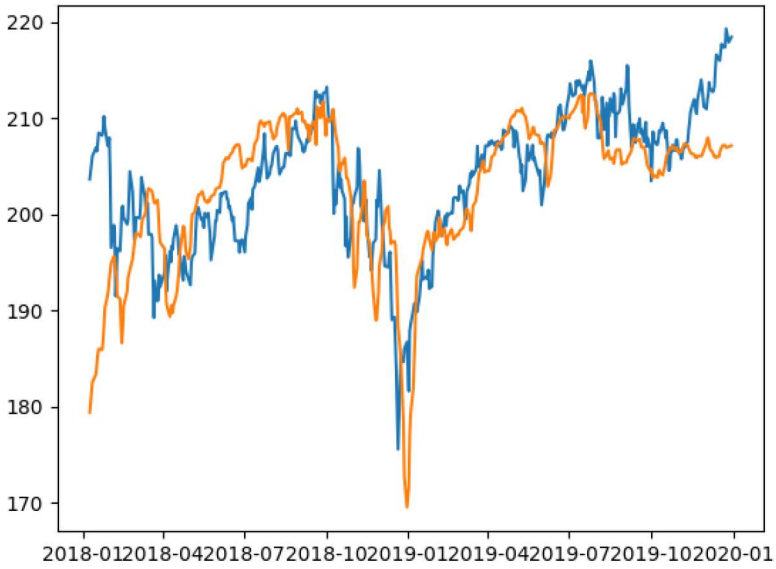


Slika 11: Graf predviđenih i stvarnih podataka u skupu za testiranje. Ovo je najbolji rezultat koji sam uspio postići uz MSE funkciju cilja



Slika 12: Graf predviđenih i stvarnih podataka u skupu za testiranje

Iz grafova vidimo da je model uspio naučiti generalni oblik distribucije podataka i da je sposoban predvidjeti približno kretanje cijene u budućnosti. Model se potencijalno može unaprijediti korištenjem tehnika fundamentalne analize kao analizom financijskih izvještaja i sentimentalnom analizom vijesti iz područja financija.



Slika 13: Graf predviđenih i stvarnih podataka u skupu za testiranje

Zaključno, iz rezultata vidimo da su LSTM rekurentne neuronske mreže moćan alat za predviđanje kretanja cijena dionica. Ovi modeli imaju velik potencijal u razvoju investicijskih strategija u skladu sa pojedinačnim investicijskim ciljevima i sklonosti riziku.

Literatura

[1] Boris Banushev. *Using the latest advancements in deep learning to predict stock price movements*. 2019. URL: https://towardsdatascience.com/aifortrading-2edd6fac689d#f8df.

[2] Zagrebačka banka d.o.o. 2024. URL: https://zbi.hr/home/zbi/download/Fondovi/zb-trend/mjesecni_izvjestaji/2024/ZB_trend_ID_09_2024.pdf.

[3] fdeloche. URL: https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg#/media/File:Recurrent_neural_network_unfold.svg.

[4] Joao P. Hespanha. *Linear systems theory*. 41 William Street, Princeton, New Jersey: Princeton University Press, 2009.

[5] Investopedia. URL: https://www.investopedia.com/terms/m/movingaverage.asp.

[6] Investopedia. URL: https://www.investopedia.com/terms/e/ema.asp.

[7] Investopedia. URL: https://www.investopedia.com/terms/m/macd.asp.

[8] Investopedia. URL: https://www.investopedia.com/terms/b/bollingerbands.asp.

[9] Wikipedia. URL: https://en.wikipedia.org/wiki/Share_(finance).

[10] Wikipedia. URL: https://en.wikipedia.org/wiki/Stock.

[11] Wikipedia. URL: https://en.wikipedia.org/wiki/Fundamental_analysis.

[12] Wikipedia. URL: https://en.wikipedia.org/wiki/Stock#Stock_price_fluctuations.

[13] Wikipedia. URL: https://en.wikipedia.org/wiki/Technical_analysis.

[14] Wikipedia. URL: https://en.wikipedia.org/wiki/Machine_learning.

[15] Wikipedia. URL: https://en.wikipedia.org/wiki/Moving_average.

[16] Wikipedia. URL: https://en.wikipedia.org/wiki/Exponential_smoothing.

[17] Wikipedia. URL: https://en.wikipedia.org/wiki/MACD.

[18] Wikipedia. URL: https://en.wikipedia.org/wiki/Bollinger_Bands.

[19] Wikipedia. URL: https://en.wikipedia.org/wiki/Neural_network_(machine_learning).

[20] Wikipedia. URL: https://en.wikipedia.org/wiki/Activation_function.

[21] Wikipedia. URL: https://en.wikipedia.org/wiki/Stochastic_gradient_descent.

[22] Wikipedia. URL: https://en.wikipedia.org/wiki/Recurrent_neural_network.

[23] Wikipedia. URL: https://en.wikipedia.org/wiki/Long_short-term_memory.