Primjena LTSM modela dubokog učenja za predviđanje kretanja cijena dionica

Rihter, Frano

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Dubrovnik / Sveučilište u Dubrovniku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:155:934002

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-20**

SVEUČILIŠTE U DUBROVNIKU
UNIVERSITY OF DUBROVNIK

Repository / Repozitorij:

Repository of the University of Dubrovnik

SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

Frano Rihter

Primjena LSTM modela dubokog učenja za predviđanje kretanja cijena dionica

ZAVRŠNI RAD

Dubrovnik, rujan 2020

SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

Primjena LSTM modela dubokog učenja za predviđanje kretanja cijena dionica

ZAVRŠNI RAD

Studij: Primijenjeno/poslovno računarstvo

Kolegij: Strojno učenje

Mentor: izv. prof. dr. sc. Mario Miličević

Student: Frano Rihter

Dubrovnik, rujan, 2020

Sadržaj

Uvod 1
Strojno učenje 2
Alati strojnog učenja 3
Cross-validation 3
Linearna regresija 3
Linearna refleksija 3
Hiperbolična trigonometrijska funkcija 3
Klasifikator 3
Clustering 4
Underfitting i Overfitting 4
Gradijentni spust 4
Moment prvog i drugog reda 5
Umjetne neuronske mreže 6
Treniranje neuronskih mreža 7
Aktivacijska funkcija 7
Stopa učenja 8
Duboko učenje 9
Tipovi neuronskih mreža 9
Unsupervised Pretrained Network: 9
Convolutional Neural Network (CNN): 10
Recurrent Neural Network (RNN): 10
Long short-term memory (LSTM) 10
Prolazak kroz LSTM mrežu: 11
Alati za predviđanje 12
Weka 12
Scikit-learn (scikit) 12
Pytorch 12
TensorFlow 12
Define-by-Run 13
Aplikacije 13
Izgradnja LSTM mreže i rezultat 14

Primjena LSTM koristeći Tensorflow (Python) 14
Analiza rezultata 17
Zaključak 22

Sažetak

Rad analizira mogućnosti primjene dubokog učenja na području vremenskih serija. Nakon teoretskog dijela u kojem su objašnjeni koncepti strojnog učenja i neuronskih mreža, prikazan je način rada LSTM algoritma. LSTM (Long short-term memory) algoritam je povratna neuralna mreža (RNN) koja, za razliku od unaprijednih mreža ima povratnu vezu. U praktičnom dijelu rada koriste se podaci cijene dionica Microsoft.corp u različitim vremenskim intervalima. Optimalni oblik modela se pokušava pronaći višestrukim testiranjem i različitim brojem prolaza kroz mrežu. Model je primijenjen koristeći programski jezik Python i TensorFlow biblioteku. LSTM model se pokazao dobar za kratkoročno predviđanje, a za bolje dugoročno predviđanje potrebno je modificirati mrežu.

Ključne riječi: Duboko učenje, strojno učenje, neuronske mreže

Abstract

Paper analyses possible use of deep learning for time series forecasting. Theoretical part explains concepts of machine learning and neural networks. Practical part uses LSTM (Long short-term memory) model, a type of Recurrent neural network algorithm with feedback connection. Model will be used to predict movement od Microsoft.corp stock prices, using Python programing language and TensorFlow library. More ideal state for the model will be found cycling thrughe network different number of times. In this paper LSTM model has shown useful for short-term predictions. For better long-term predictions model needs to be modified.

Key words: Deep learning, machine learning neural networks

Uvod

Duboko učenje je koncept koji se često spominje u medijima zajedno s umjetnom inteligencijom i strojnim učenjem. Umjetna inteligencija je koncept koji obuhvaća veliki broj pojmova iz računarstva ali, generalno riječ je o području koje istražuje mehanizame koji koriste stohastičke metode (metode nasumičnosti). Te metode aktivno mijenjaju okolinu (dosta slično *trigger* metodama) ili okolina aktivno mijenja njih pa ta promjena daje neki rezultat (često vjerojatnost) iz kojih mi donosimo zaključak. Podskup umjetne inteligencije je strojno učenje. Jako popularno područje među znanstvenicima, inženjerima i velikim brojem entuzijastičnih istraživača s pozadinom u matematici ili programiranju. Ta popularnost je stvorila potrebu za velikim brojem raznovrsnih i besplatnih alata. Funkcija strojnog učenja je da omogući računalu da obavi zadatak koji mu nije eksplicitno zadan. Što je naravno neizostavan izvor rješenja koja samo čekaju da budu otkrivena. Strojno učenje uzima jednostavne podatke i obrađuje ih, te daje rezultat u nekom obliku vjerojatnosti. Problemi s kojima se susrećemo u strojnom učenju lako se povezuju sa statističkim konceptima kao što su uvjetna vjerojatnost, algebra događaja, korelacija, kovarijanca, normalna razdioba i slično. Sve od navedenog je koristan alat svakoga tko se planira baviti ovim područjem. Duboko učenje je podskup strojnog učenja. Glavna karakteristika dubokog učenje su višeslojne neuronske mreže koje su ponekad toliko kompleksne da ni inženjeri koji su ih izradili nisu sigurni kako i zašto je dobijen određeni rezultat. Područje uglavnom podrazumijeva izradu i modificiranje višeslojnih mreža.

Duboko učenje do sada nije donijelo neki značajan napredak u ekonomiji. Svi pametni sistemi koje banke i ostale institucije koriste gotovo uvijek imaju kompleksni agent model i tu priča staje, što se strojnog učenja tiče. Ideja je u nastavku rada pokušati otvoriti prostor dubokom učenju u području ekonomije.

Strojno učenje

Iako je tema ovog rada duboko učenje, za pojašnjenje rada potrebno je prvo objasniti Strojno učenje. Strojno učenje je pojam kojim opisujemo algoritme koji nakon unosa podataka, sebe poboljšavaju uz pomoć testiranja s često velikim setom podataka.

Strojno učenje se koristi u znanosti još 1980-ih godina, ali je u široj primijeni postalo popularno tek nedavno (2010-ih). Povećavanjem prometa podataka na internetu stvorila se potreba za njihovom analizom, te povećanjem snage računala omogućilo se korištenje kompleksnijih algoritama.

Strojno učenje se također može opisati vještinama potrebnim za rad u toj struci. Neke od tih vještina su: poznavanje statističkih metoda dokazivanja, linearne algebre, baza podataka i poznavanje osnova programiranja. Tipičan projekt strojnog učenja se odrađuje slijedom:

Skupljaju se podaci na temelju kojih se želi provesti analiza. Ponekad su ti podaci dostupni online, napisani u „uređenom formatu“ a, ponekad je potrebno iz tih podataka filtrirati one dijelove koji su irelevantni i neupotrebljivi. U nekim slučajevima nema podataka pa ih je potrebno prikupiti, često uz pomoću anketa, koristeći data scraping alata za skupljanje podataka na društvenim mrežama ili u zadnje vrijeme sve češće uz pomoć Internet of Things naprava. Uređeni podaci se dijele u skupine za treniranje i testiranje.

Nakon uređivanja podataka postavlja se teorija koja od analitičkih metoda bi proizvela najpovoljniji rezultat. Iako se danas strojno učenje promovira kao nužnost u svakom projektu, ponekad je nepotrebno ili apsolutno krivo. Bolji rezultat bi se postigao algoritmom optimizacije ili evolucijskim (genetskim) algoritam.

Strojno učenje je stohastička metoda koja ne daje „nepogrešivo” rješenje jer „nepogrešivost” nije koncept koji se u stohastičkim metodama koristi, nego ako je dobro izabran ili izrađen algoritam on daje globalni optimum (ovisno o algoritmu, i set lokalnih optimuma) koji predstavlja najbolje rješenje unutar njemu poznatih parametara. Biranje metode zahtjeva dobro poznavanje statistike i linearne algebre.

Kada se dobije rezultat jako je bitno donijeti objektivan i inteligentan zaključak. Moguće da set nije objektivna reprezentacija problema, koji se pokušava promatrati, ili je odabran loš algoritam. U tom slučaju se tvrdnja mora odbaciti i krenuti iz početka. Pri izlaganju rezultata, za koje se vjeruje da su relevantni, nije primjereno upotrijebiti izraze kao „sigurno“ ili „apsolutno“ nego, radije izraze kao „gotovo sigurno“, „vjerojatno“ isl.

Objasniti strojno učenje kroz linearnu algebru, jednostavno rečeno, čita se vrijednosti iz matrice i linearnim operacijama ih pokušavamo približiti stvarnoj vrijednosti. Odnosno rješava se takozvani $Ax = b$ problem gdje mijenjanjem brojeva u vektoru x (vektor

parametara, odnosno nepoznanica na koje model utječe) se pokušava dobiti rješenje s najmanjom greškom. Koristi se funkcija gubitka za prilagodbu matrice težine (*A*) dok vjerojatnost koju daje vektor *b* ne postane zadovoljavajuća.

Alati strojnog učenja

Poglavlje opisuje osnovne pojmove koji se koriste u strojnom učenju.

Cross-validation

Set se dijeli na dvije skupine: testne podatke i podatke za treniranje. Nakon treniranja cilj je testirati podatke koje do sada program nije koristio. Treniranje se koristi za procjenu odabranog modela strojnog učenja. Nakon testiranja model se trenira. Treniranjem se provjerava je li došlo do overfitting ili underfitting problema. Set za treniranje je veći od seta za testiranje. Cross-validation dijeli set na blokove i naizmjenično bira koji će se set koristiti za treniranje, a koji za testiranja (testiranje ima manji broj blokova često oko 25% od ukupno). Proces je gotov kada je svaki blok bio izabran za testiranje.

Linearna regresija

Jednostavna funkcija koja promatra parametre matrica i pokušava od njih stvoriti konkretnu vrijednost je linearna regresija. Linearna regresija crta vezu između dva seta (X,Y) i za znane vrijednosti X pokušava pretpostaviti vrijednost koju poprima Y.

Linearna refleksija

Linearna refleksija (ReLU) se aktivira samo ako je vrijednost prešla zadanu veličinu. Ako je ulaz ispod nula onda je i izlaz nula, u suprotnom ima linearnu ovisnost $f(x)=max(0,x)$. Pomaže u rješavanju problema nestajućeg gradijenta. Koristi se u konvolucijskim mrežama.

Hiperbolična trigonometrijska funkcija

Koristi se kao reprezentacija nekih negativnih brojeva. Funkcija ima domenu [-1,1]. Slično kao i sigmoid koristi se za normalizaciju (svaka vrijednost koja je veća od 1 postaje 1 i svaka koja je manje od -1 postaje -1).

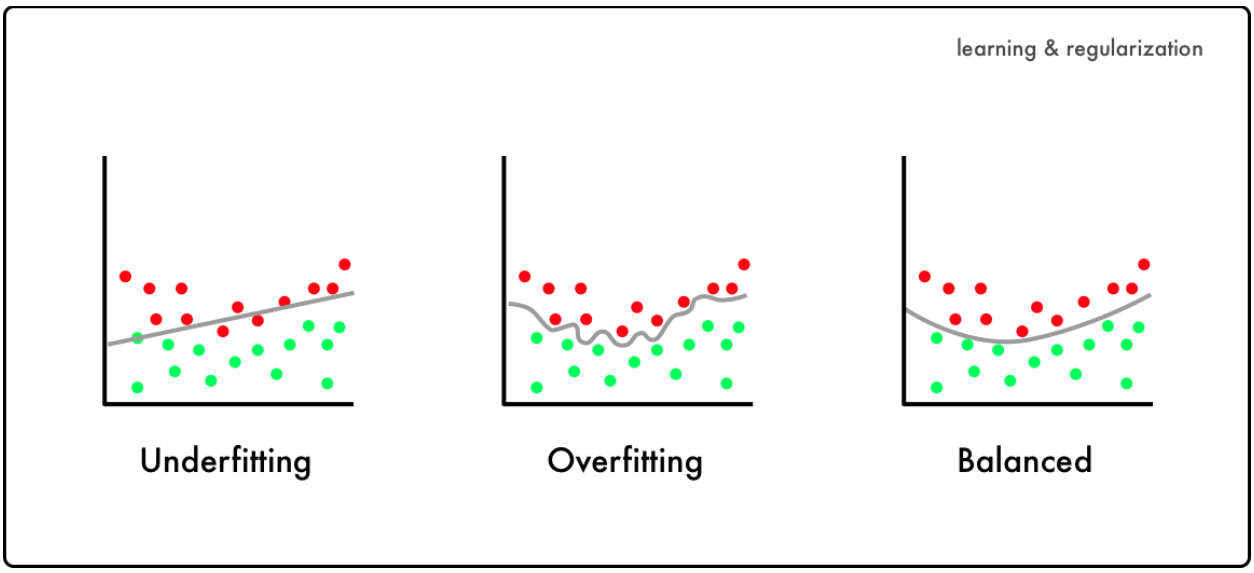
Klasifikator

Klasifikator koristimo za opisivanje vrste vrijednosti koja se dobija. Najjednostavniji oblik je binarni klasifikator s dvije vrijednosti (1,0) kao diskretna slučajna varijabla, ali često se koristi za decimalnu vrijednost, kao neprekinuta slučajna varijabla, koju poslije zaokružujemo na vrijednost 1 ili 0 (naravno može se staviti *n* parametara pa se zaokružuje na najbliži) npr. rast ili pad dionice se označava s 1 (za rast) i 0 (za pad) . Primjenjuje se u neuronskim mrežama gdje decimalni broj prolazom kroz čvor mijenja vrijednost [1].

Clustering
Proces kojim se diskretiziraju vrijednosti ovisno o broju podataka u bliskom okruženju.

Underfitting i Overfitting
Ako je točnost testiranja podataka 1 (odnosno 100%) dogodio se owerfitting. Svrha strojnog učenja je predviđanje, a 100% sigurnost može samo značiti da je model za učenje prekomjerno prilagođen podacima. Neki podaci za testiranje neće biti svrstani u pravu kategoriju radi neke nasumične karakteristike koja je slučajno dobila veću težinsku vrijednost, nego što je trebala. U tom slučaju model nije pouzdana.

Slijedno underfitting se pojavljuje kada je korišteni model prejednostavan. Rezultat toga je preveliki broj podataka koji nije ispravno razvrstan. Različita očekivanja se smatraju prihvatljivima za različite setove podataka [2].



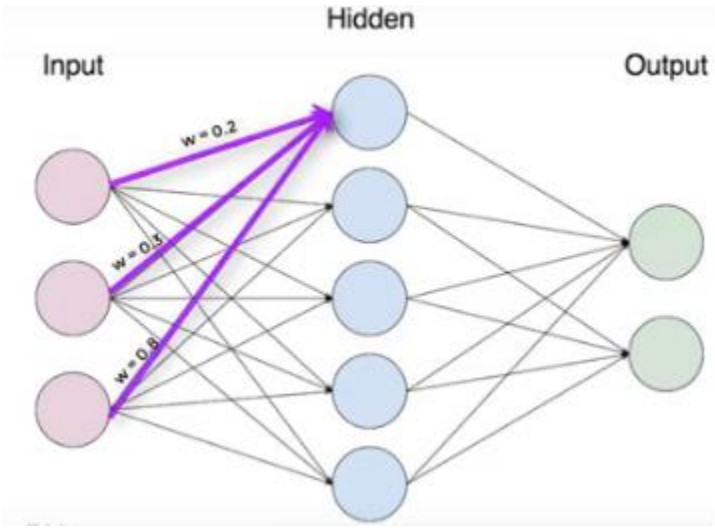
Slika 1 Primjer Overfitting i Underfitting

Gradijentni spust
Gradijentni spust jedna je od najvažnijih metoda koja se koristi u gotovo svakoj neuronskoj mreži. Cilj gradijentnog spusta je promotriti podatak kao matematičku funkciju i pokušati joj pronaći mjesto gdje ima najviše zajedničkih elemenata s prethodnim primjerima. Pronalazak takvog rješenja se postiže kroz globalni minimum. Gradijentno spuštanje deriviranjem pronalazi mjesto gdje se funkcija spušta. Ako se pronađe manji broj od prethodnog, novi broj se koristi za modificiranje mreže, a stari se zaboravlja. Bitno je obratiti pažnju na to, da u nelinearnim modelima nije sigurno je li doista pronađen globalni minimum.

Moment prvog i drugog reda
Metoda u vjerojatnosti koja vrši derivaciju na nasumičnoj vrijednosti x i nasumičnoj vrijednosti y. Računamo kovarijancu te dvije točke i dobijemo neki zajednički element među njima. Metoda nije učinkovita ako se ne ponavlja veliki broj puta.

Umjetne neuronske mreže

Riječ je o modelima koji koriste algoritme za učenje. U ovom slučaju ti algoritmi naizgled dijele neke karakteristike s neuronima u našem tijelu. Mreže se sastoje od čvorova (jezgra stanice) i veza (akson). Čvorovi su podijeljeni u vertikalne slojeve koji imaju različite funkcije. Prvi sloj je ulazni sloj koji ima funkciju prenošenja podataka na sljedeći. Broj čvorova u ulaznom sloju je jednak broju ulaznih karakteristika seta podataka. Izlazni sloj je zadnji sloj kojem se dodjeli podatak. Podatak se zadnjem sloju dodjeljuje ovisno o njegovoj vrijednosti i na temelju te vjerojatnosti korisnik može zaključiti o kakvom je ishodu riječ. Svaki sloj između ulaznog i izlaznog sloja se zove skriveni sloj jer nije direktno vidljiv korisniku. Ako algoritam ima više od jednog skrivenog sloja onda je riječ o algoritmu koji vrši duboku analizu. Funkcija veze između svakog čvora je da pridodaje podatku težinsku vrijednost koja opisuje koliko je vjerojatno da taj podatak pripada tom čvoru u vezi.



Slika 2 Primjer mreže

Na primjer veza *w*=0.2 (uvijek poprima vrijednost između 0 i 1) se pridodaje prvom čvoru u sljedećem sloju jer podatak iz prvog čvora u prvom sloju kaže da ima 20% šanse da taj čvor daje pozitivan rezultat za informaciju koju podatak nosi. Zbroj svih vrijednosti *w* koje prima čvor prolazi kroz aktivacijsku funkciju koja između ostalog određuje koje *w* vrijednosti će čvor prenijeti na sljedeći sloj. Vrijednost koja se dobije nakon aktivacijske funkcije se mijenja treniranjem. Treniranje je događaj u kojem podatak prolazi kroz mrežu. Prolazak seta podataka kroz mrežu zovemo epoch. Aktivacijska funkcija u svakom čvoru jednog sloja je uglavnom ista.

Treniranje neuronskih mreža

Proces učenja u svakom algoritmu koristi koncept težine. Težina je vrijednost koja označava kolika je relevantnost određenog podatka za donošenje izlazne vjerojatnosti. Ispravnim mijenjanjem težinskih vrijednosti dobijemo veću ili bolje reći, čvršću korelaciju. Algoritam stavlja naglasak na dio podatka koji ima čvrstu korelaciju, a smanjuje onaj dio koji ima slabiju. Jedna od funkcija koja mijenja vrijednost težine je funkcija gubitka (Loss function). Koristi jednostavan princip gdje „nagrađuje“ mrežu za dobre pogotke, a „kažnjava“ za loše.

Dobro trenirana neuronska mreža ima povećane vrijednosti koje smatra relevantnima, a smanjuje one koje smatra irelevantnima. U literaturi se proces treniranja često uspoređuje sa signalom, gdje se čvrstoća korelacije naziva snaga signala, a nedostatak korelacije se naziva šum.

Aktivacijska funkcija

Kao što je već spomenuto aktivacijska funkcija se nalazi u svako čvoru mreže. Nije nužno da svaki čvor ima različitu aktivacijsku funkciju. Uglavnom svaki čvor istog sloja ima istu aktivacijsku funkciju. Pojednostavljeno, zadatak funkcije je da za zbroj primljenih vrijednosti odredi koja je vjerojatnost da pripada određenoj klasi (klasa predstavlja određeni čvor u sljedećem sloju ili konačan rezultat ovisi o tipu algoritma).

Linearna Transformacija je aktivacijska funkcija oblika $f(x) = Wx$ gdje ovisna varijabla ima proporcionalnu vrijednost s neovisnom varijablom, odnosno podatak prolazi kroz funkciju nepromijenjen. Koristi se na ulaznom sloju.

Sigmoid je aktivacijska funkcija koja aproksimira neovisne varijable u vjerojatnost između 0 i 1. Koristi se često jer je veliki broj neuronskih mreža napravljenih da rade isključivo s vrijednostima između 0 i 1.

Funkcija gubitka, pojednostavljeno, traži koliko je neuronska mreža blizu svom cilju. Koristi se za prilagođavanje matrice pogreške. Vrši prilagodbu na cijelom setu podataka. Rezultat funkcije je jedan broj. Pri opisu koristimo izraz $L(W, b)$. Promatranjem definicije vidimo da vrijednost težine ovisi o W težini i b (biase) uvjerenju. Mijenjanjem tih vrijednosti stvara se promjena u cijeloj mreži, tako da je potreban veliki oprez, jer mala promjena stvara veliku razliku.

Softmax aktivacijska funkcija je generalizacija logičke regresije, jer se može primijeniti na većem broju klasa (ne samo binarno 0 i 1) i može sadržavati veći broj uvjeta prijeloma. Koristi se često na izlaznom sloju.

Stopa učenja

Stopa učenja je metoda koja množi fiksnu odabranu vrijednost s aktivacijskom funkcijom. U sljedećem prolazu kroz mrežu aktivacijska funkcija se trajno mijenja za vrijednost stope učenja. Koristi se za prilagodbu parametara tijekom optimizacije. Ako je vrijednost stope učenja jako mala, broj promjena koje će mreža postići kroz svaki prolaz će biti jako malene. To je izvrsna metoda za stvaranje optimalnog sustava, ali je jako spora (treniranje velikih setova zna trajati tjednima). Ako je vrijednost stope učenja jako velika, veća se promjena događa u manje vremena, ali takva stopa brzo postaje problem jer prelazi tražene vrijednosti.

Jednostavno objašnjenje stope učenje je gradijentni spust. Naša želja je naravno naći globalni minimum, a mala stopa promjene ponekad rezultira samo pronalaskom lokalnog minimuma. Potrebno je promotriti izlaznu informaciju i objektivno zaključiti da li je stopa učenja ispravno postavljena ili nije.

Duboko učenje

Sva svojstva strojnog učenja izložena do sada vrijede za duboko učenje jer je duboko učenje podskup strojnog. Radi kompleksnosti modernih neuronskih mreža stvara se potreba za „podjelom“ jer iako je u suštini riječ o istom procesu, duboko učenje zahtjeva veći fokus na dijelove projekta koji su se do tada, radi svoje jednostavnosti, gotovo uzimali kao trivijalni, ili su se uvrštavali u matematički dio posla.

Tipovi neuronskih mreža

Ponašanje mreže ovisi o njenoj arhitekturi. Postoji veliki broj mreža, ali za generalno razumijevanje svih, dovoljno je pogledati tri najpopularnije (one imaju svoje podskupove mreža o kojima nećemo puno govoriti).

Unsupervised Pretrained Network:
Koristi se kada je jako teško pronaći minimum funkcije. Umjesto da se mreža trenira s nasumičnim težinama, veza svakog sloja se trenira Auto-Coder algoritmima.

Kodira se samo prvi sloj i s njim se pokušava ponovno stvoriti uneseni podatak. Ispravno odrediti Auto-Coder algoritam zaći izabrati ispravnu aktivacijsku funkciju izlaznog čvora. Ispravna aktivacijska funkcija u ovom slučaju je najčešće ona koja normalizira rang vrijednosti ulaznog podatka. Bitno je naglasiti da određeni tip normalizacije može prouzročiti prevelik gubitak signala. Gubitak signala se može izmjeriti ovisno o obliku podatka koji se dobije na izlaznom sloju. Metoda kojom Auto-Coder trenira mrežu je metodom obnavljanja (Restauration). Metoda obnavljanja mijenja ulazni sloj, najčešće se primjenom filter koji unosi bijeli šum, Gaussov šum ili se dio podatka izbriše. Prolaskom podataka kroz mrežu aktivacijske funkcije se mijenjaju i postaju više otporne na šum.

Kada rezultati prvog sloja budu zadovoljavajući, isto se radi na sljedećem sloju. Sada se mreža sastoji od slojeva koji ne pridodaju nasumično izabranim težinama, u nadi da će se one same modificirati, nego koriste težine treniranog modela. Set prolazi kroz njih i modificira ih. Ovaj tip mreža se također koristi kada je teško obaviti klasifikaciju (preveliki broj parametara kojeg treba rasporediti u veliki broj klasa). Treniranja se uglavnom odvija u tri faze.

- Treniranje bez nadzora: trening s podacima koji nisu podijeljeni u kategorije
- Modifikacija mreže: vrši promjene na čvorovima mreže. Promjene se često vrše na zadnjem sloju, a jako rijetko na unutarnjim slojevima, jer je velika šansa da se mreža toj promjeni neće prilagoditi na očekivani način
- Treniranje pod nadzorom: treniranje s podacima koji imaju oznake kategorija

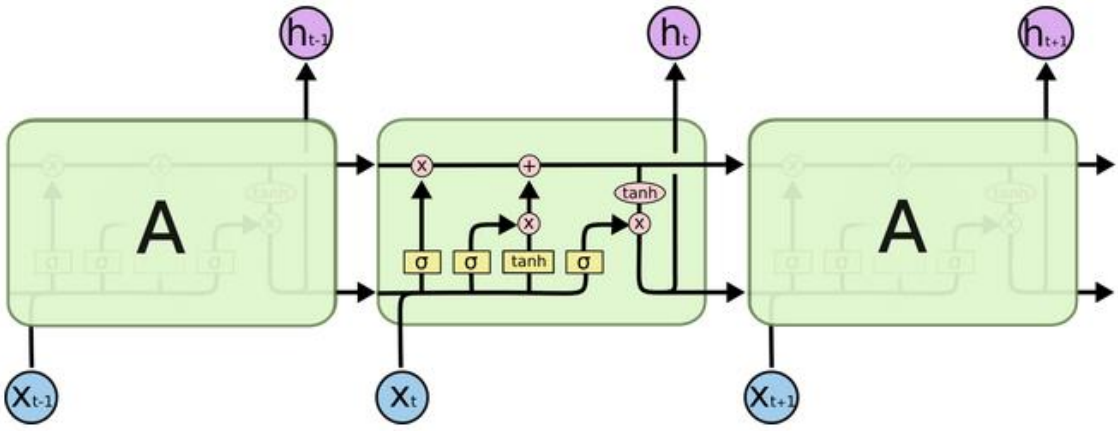
Convolutional Neural Network (CNN):
Često se koristi za analizu slika i videa. Riječ je o potpuno spojenoj mreži (svaki čvor jednog sloja je spojen sa svakim čvorom sljedećeg i svakim čvorom prethodnog sloja) što povremeno može uzrokovati overfitting. Konvolucijske neuronske mreže rješavaju ovaj problem tako da se određeni čvor aktivira samo na neke vrijednosti i time npr. dijeli sliku na više različitih slika od kojih svaka ima svoj set relevantnih podataka. Pojedinačnim analiziranjem tih dijelova slike lakše se prikupljaju relevantni podaci, nego da se slika gleda kao cjelina. Zbroj tih slika daje izvornu sliku (osim ako je neki od elemenata slike irelevantan za analizu, onda se taj element tretira kao šum). Prije pokretanja mreže, postavlja se *bias* funkcija fiksne vrijednosti, čija vrijednost ovisi o tipu podataka koji se analizira. Na taj način mreža prepoznaje o kojem je tipu podataka riječ. Mreža se sastoji od izlaznog i ulaznog sloja te specifičnog tipa skrivenih slojeva. Skriveni slojevi koji se uvijek koriste su slojevi koji vrše podjelu slike na mape. Aktivacijska funkcija konvolucijskog sloja je ReLU. Sljedeći sloj je *pooling* sloj koji detektira određene karakteristike i sprema ih u mapu. Sloj normalizacije iz vrijednosti u mapama izračunava vjerojatnost. Mreža uspješno prepoznaje prostorne ovisnosti [3].

Recurrent Neural Network (RNN):
Čvorovi skrivenih slojeva imaju funkciju *f(x)* koja poziva čvor u kojem se nalazi. Tako u sljedećem pokretanju čvora rezultatu aktivacijske funkcije dodaje *f(x)*. Izlaz je njihov zbroj. Dobra je metoda za prenošenje sekvencijalnih podataka kao prepoznavanje značenja rečenice.

Long short-term memory (LSTM)

Tip Recurrent neural network modela koji se koriste za prepoznavanje teksta. Ovaj tip mreže se koristi kada se zna da će jedan od problema mreže biti nestajući gradijent. Nestajući gradijent je rezultat parcijalne derivacije greške u kojem je vrijednost greške manja od 1. To se samo po sebi ne čini kao veliki problem, ali nakon što ga pomnoži sa stopom učenja koji je jako mali broj, dobije se modificirana vrijednost težine s prevelikim odstupanjem. Tu mrežu postaje gotovo nemoguće trenirati daljnjim prolaskom, nego je potrebno krenuti iz početka. Čvor LSTM mreže se najčešće sastoji od ćelije (prostor rezerviran za pamćenje) i tri regulatora kroz koje putuje informacija. Aktivacijska funkcija koja se koristi u regulatorima je najčešće sigmoid. Aktivacijska funkcija regulatora određuje koliko će se od zadanog podatka prenijeti na sljedeći sloj. Neke od veza s regulatorima su izlazne, a neke pozivaju isti čvor. Rad kojeg regulatori vrše ovisi o vrijednostima koje primaju.

Prikaz mreže:



Slika 3 LSTM mreža

Svaka crta nosi podatak iz ulaza jednog čvora do ulaz drugog. Ružičasti krugovi (unutar) predstavlja matematičku operaciju koja se vrši na podatku (npr. zbrajanje vektora). Žuti kvadrati predstavljaju slojeve mreže. Spajanje strjelica predstavlja konkatenaciju, a razilaženje predstavlja kopiranje podatka, koji se nosi na dvije različite lokacije.

Prolazak kroz LSTM mrežu:

Prvi korak je odlučiti koja informacija se odbacuj iz ćelije. Tu odluku donosi sigma funkcija. Ako je sigma funkcija predala 0 onda se informacija odbacuje, ako je predan 1 zadržava informaciju u ćeliji. Dolaskom sljedeće informacije (u isti sloj) mreža odlučuje koji će podatak zadržati u ćeliji. Zadržana informacija se modificira onom koja će biti odbačena. U sljedećem sloju dodaje novoj informaciji neki unaprijed određeni vektor (koji se nagrađuje ili kažnjava ovisno o rezultatu). Zadnji korak je odlučiti koji dio informacije se prenosi dalje. To se radi pomoću filtera. Nakon filtera informaciju se gura kroz thank algoritam (informaciji se daje vrijednost između -1 i 1) i množi ju sigmoid funkcijom na zadnjem regulatoru. Postoje brojne varijacije na LSTM mrežu.

Alati za predviđanje

Broj alata za predviđanje svaki dan raste. Bitno je izabrati alat koji će uštedjeti što više vremena i dati što precizniji rezultat. Promotrimo trenutno najpopularnija rješenja. Svaki od navedenih okruženja je besplatan i ima kod otvorenog tipa.

Weka

Waikato Environment for Knowledge Analysis ili Weka je alat s grafičkim sučeljem izrađen u Java programskom jeziku. Sadrži set vizualizacijskih alata, biblioteku algoritama za strojno učenje i set prediktivnih modela. Koristi se za propagaciju podataka, regresijsku analizu, klasifikaciju cloustera i rudarenje podataka. Korištenje je jako jednostavno. Odabirom seta, ovisno o broju parametara, Weka dozvoljava ili ne dozvoljava korištenje određenih algoritama te smanjuje šansu da korisnik napravi pogrešku. Nije moguće pisati kod, nego su sve mogućnosti platforme ponuđene u lako razumljivom sučelju. Jedna od glavnih funkcija Weka programa je rudarenje podataka. Algoritam za rudarenje se kreira u obliku dijagrama klasa (ne pisanjem koda). Za rudarenje u ovoj aplikaciji potrebno je biti dobro upoznat s dokumentacijom. Raznolikost izbora i jednostavnost korištenja čini Weku dobrim alatom za stvaranje početnih pretpostavki o setu. Moguće je integrirati Weka program s Python ili R programskim jezikom. Weka ne može koristiti kompleksne neuralne mreže [4].

Scikit-learn (scikit)

Velikim dijelom pisan u Python programskom jeziku. Koristi numpy biblioteku za rad koristeći linearnu algebru. Neki dijelovi programa su pisani u programskim jeziku Cython koji daje izvrsne rezultate u izvođenju koda. Scikit je prilično jednostavan za korištenje.

Pytorch

Python biblioteka nastala kao proizvod firme Facebook. Napravljen je sa svrhom optimizacije izvedbe koda, a da taj kod bude jednostavan za pisanje. Trenutno najpopularniji istraživački alat. Jezik niže razine nego što je scikit pruža neograničenu mogućnost prilagodbe. Ima veliki broj ekstenzija. Pisan je u C.Torch programskom okruženju. C.Toarch je upakiran u Python sučelje. Kod se piše u Python programskom jeziku, ali se izvodi u C++ jeziku koji je optimiziran za Python.

TensorFlow

TensorFlow biblioteka je okruženje, uglavnom namijenjeno strojnom (i dubokom) učenju. Nudi više slojeva apstrakcije tako da je moguće koristiti njihove API za jednostavan rad s postojećim modelima, ili izraditi vlastiti. Proces instalacije TensorFlow je prilično jednostavan. Biblioteka se instalira unosom naredbe u terminal "*pip install tensorflow".* TensorFlow je matematička Python biblioteka simboličkog tipa. Proizvod tvrtke Google, razvijen za njihove potrebe 2011 godine. Koristi se u komercijalne i istraživačke svrhe. Korisnici TensorFlow biblioteke mogu pokretati program u isto

vrijeme na više različitih procesora i grafičkih kartica što omogućuje brzo treniranje seta. TensotFlow sam određuje koje procese odrađuje procesor, a koje grafička kartica. Za pokretanje procesa na grafičkoj kartici koristi se CUDA (Computer Unified Device Architecture) platforma koju je razvila nVidia. TensorFlow ima veliki broj korisnika koji doprinose zajednici objavljujući javno svoj kod koji se poslije može pakirati u biblioteke. Dokumentacija je jako opširna [5].

Define-by-Run
Olakšava izvođenje koda. Omogućuje kreiranje takozvanih „notebook“ datoteka koje omogućuju da se pokrene kod, segment po segment. Olakšava programeru debug proces i čini kod preglednijim. Koncept funkcionira jer Chainer sistem sprema povijest obrada koje su procesor i grafička kartica izvršili, a ne sprema programsku logiku.

Aplikacije
Postoji veliki broj aplikacija za TensorFlow. Neke od njih su:

- CoLab: „notebook“ tip datoteke za lakše i preglednije izvođenje programa
- TensorBoard: vizualizacijsko sredstvo napravljeno sa svrhom lakšeg razumijevanja i optimiziranje programa
- ML Perf: za mjerenje performansi računala u svrhu strojnog učenja te mjerenje performansi programa s kojim se kod izvodi
- XAL (Axcelerated linear algebra): compiler za linearnu algebru koja optimizira TensorFlow izvođenja

Izgradnja LSTM mreže i rezultat

Brojne aplikacije olakšavaju korištenju kompleksnih algoritama predviđanja. U ovom slučaju koristi se TensorFlow.

Primjena LSTM koristeći Tensorflow (Python)

Ideja je napraviti model koji će precizno predviđati cijene rasta i pada dionica. Veliki broj algoritama predviđanja se može primijeniti na ovom primjeru. Radi jednostavnog korištenja izabran je LSTM model. LSTM je također jedan od najčešćih primjera korištenja RN mreža koje su sve popularnije. Kao set je izabrana cijene dionica Microsoft.corp od 13.3.1986. godine do 25.9.2020. godine.

	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062055	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064271	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065379	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.063717	67766400
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.062609	47894400
...	...	...	...	...	...	...	...
8702	2020-09-21	197.190002	202.710007	196.380005	202.539993	202.539993	39839700
8703	2020-09-22	205.059998	208.100006	202.080002	207.419998	207.419998	33517100
8704	2020-09-23	207.899994	208.100006	200.029999	200.589996	200.589996	30803800
8705	2020-09-24	199.850006	205.570007	199.199997	203.190002	203.190002	31202500
8706	2020-09-25	203.550003	209.039993	202.539993	207.820007	207.820007	29416000

Slika 4 Prikaz podataka

Set sa sastoji od: datuma otvaranja trgovanja (Date), cijene pri otvaranja taj dan (Open), najviše cijene taj dan (Heigh), najniže cijene (Low), cijene pri zatvaranju (Close), prilagođenu cijenu ovisno o nekim eventualnim potezima firme (Adj Cost) i količini raspoloživih dionica na burzi (Volume). Iz priloženog se vidi da ima puno parametara koje bi se mogli promatrati. Razlika najniže cijene dana i cijena zatvaranja, broj dionica na tržištu i cijena tih dionica itd. Koristiti će se LSTM da zapamti zadnjih 60 dana trgovanja s očekivanjem da se uspješno predvidi kretanje na 61 dan.

Priprema podataka je prilično jednostavna. Nije potrebno ništa više, nego instalirati python paket panda, biblioteka za upravljanje i uređivanje relacijskim i označenim podacima. Bez sličnih biblioteka, proces čišćenja seta zna biti prilično dug radi nekompatibilnosti programa za obradu podataka s manje zastupljenim tipovima podataka, koji se često ne prenose s aplikacije na aplikaciju bez gubitka.

```
data = df.filter(['Close'])
dataset = data.values
duljinaTrenir = math.ceil( len(dataset) *.8)
#skaliranje podataka između 0 i 1
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
#STVORITI SET ZA TRENIRANJE
train_data = scaled_data[0:duljinaTrenir  , : ]
x_train=[]
y_train = []
for i in range(60,len(train_data)):
    x_train.append(train_data[i-60:i,0])
    y_train.append(train_data[i,0])
```

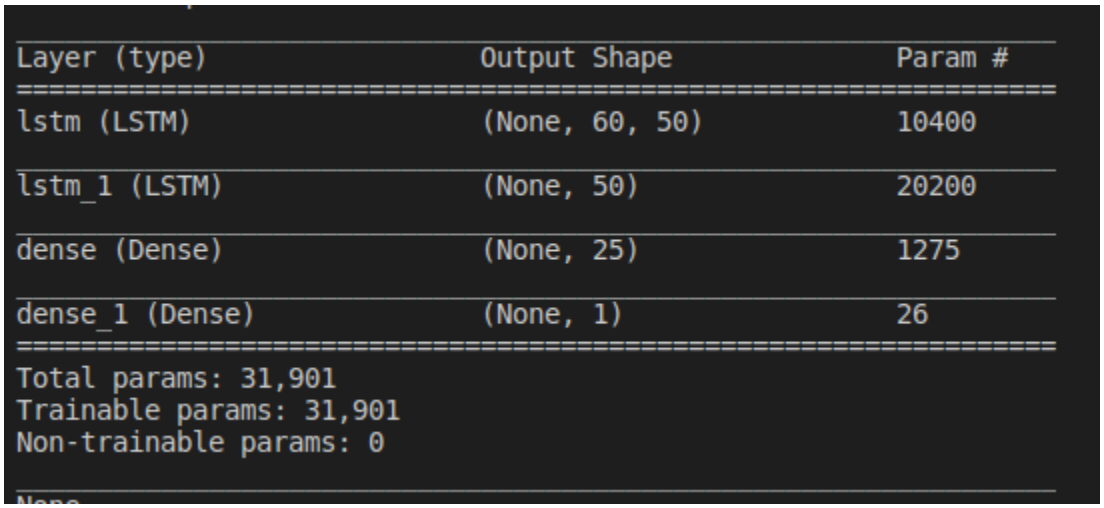
U ovom slučaju potrebni stupci za analizu su zatvaranje i datum. S obzirom na to da su podaci poredani slijedno datum se zanemaruje (datum se teško oblikuje pa je, jako teško raditi s njim) i uzimamo samo vrijednosti zatvaranja. Neke mreže zahtijevaju ulazne parametre u intervalu od 0 do 1 pa tako skaliramo vrijednosti pomoću funkcije *MinMaxScaler().* Koristi se *split()* funkciju da podijelimo set na x_train i y_train.

Korištenje modela:

```
x_train, y_train = np.array(x_train), np.array(y_train)
#MJENJA OBLIK DATOTEKE U OBLIK KOJI ODGOVARA LSTM
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
#IZRADI LSTM MODEL
model = Sequential()
model.add(LSTM(units=50, return_sequences=True,input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dense(units=25))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
#TRENIRA model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

Sekvencijalni model se uglavnom koristi kada ulazni sloj ima samo jedan čvor što je naznačeno u *input_shape* parametru kao 1. Mreža se sastoji od 3 skrivena čvora od kojih su dva tipa LSTM i jedan tipa Dense, izlazni čvor je tipa Dense. Parametar *unite* je broj neurona u sloju. Parametar *returne_sequence* je binarna vrijednost koja odlučuje da li se vraća zadnja izlazna vrijednost operacije ili cijeli podatak. Ne može se provesti compile ako nema parametre optimizator i funkciju gubitka. Adam je stohastička metoda koja primjenjuje gradijentni spust ovisno o adaptivnom predviđanju momenta prvog i drugo reda. Adam zahtjeva malo memorije pa se često koristi u problemima koji imaju veliki broj parametara ili je set jako velik. *loss* parametar je već opisana u prethodnom dijelu teksta. Funkcija *fit()* je funkcija koja provodi treniranje. U ovom slučaju x_train

predstavlja ulazne parametre, y_train predstavlja ciljanu vrijednost podatka (stvarna vrijednost koja se nije koristila u treniranju), batch_size predstavlja koliko podataka ulazi u mrežu, a da se na njima radi odjednom, epoch je broj prolaza kroz set podataka (koristimo velik set pa je izabrani broj prolaza jedan, nije neobično da u nekim slučajevima broj prolaza bude 100, 1000 i više).



Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 50)	10400
lstm_1 (LSTM)	(None, 50)	20200
dense (Dense)	(None, 25)	1275
dense_1 (Dense)	(None, 1)	26

Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0

Tablica sadržaja je jako koristan alat za bolje razumijevanje sadržaja mreže. Ona se sastoji od tipova slojeva koji se koriste. Stupac s oznakom „Parametre“ je zbroj težina i pretpostavki u mreži.

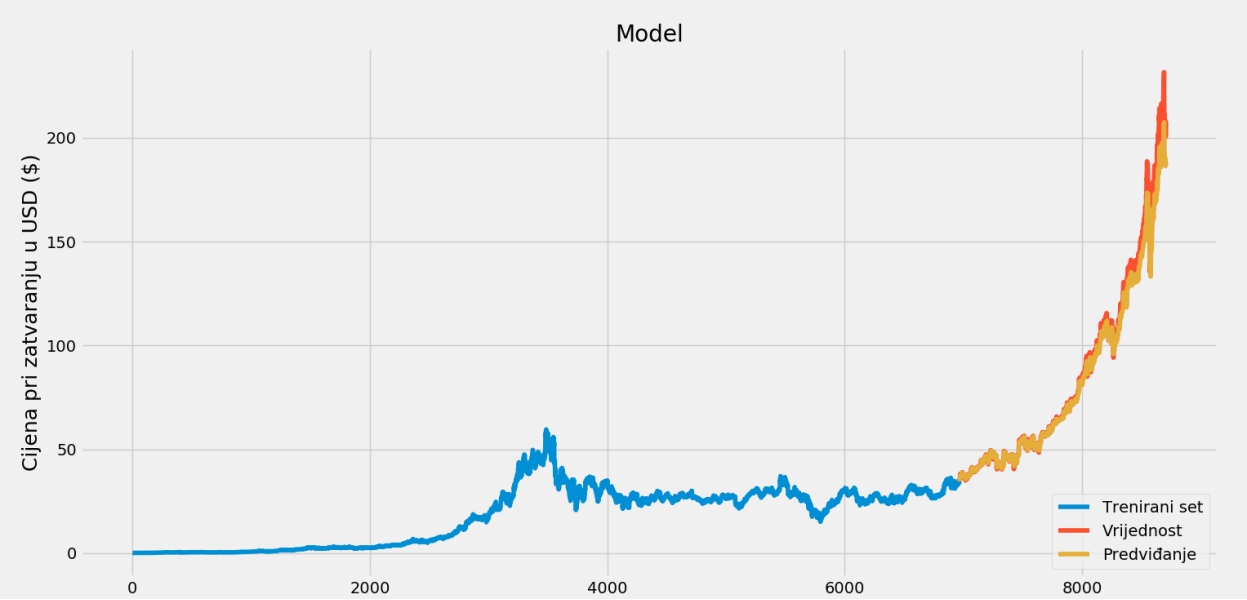
```
x_test = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))
#DOHVAČA CIJENU KOJU JE PREDVIDIO
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
#RAČUNA RMSE VRIEJDNOST
rmse=np.sqrt(np.mean(((predictions- y_test)**2)))
```

Predviđena cijena se sprema u varijablu *predictions*. Primjenom funkcije *invers_transform()* dobija se broj koji više nije u obliku između 1 i 0. RMSD je statistička metoda koja izračunava vrijednost između promatrane i predviđene vrijednosti.

```
#CRTANJE
train = data[:duljinaTrenir]
valid = data[duljinaTrenir:]
valid['Predictions'] = predictions
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Podatak', fontsize=18)
plt.ylabel('Cijena zatvaranja u USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Trenirano', 'Vrijednost', 'Predictions'], loc='lower right')
plt.show()
print(valid)
print(model.summary())
```

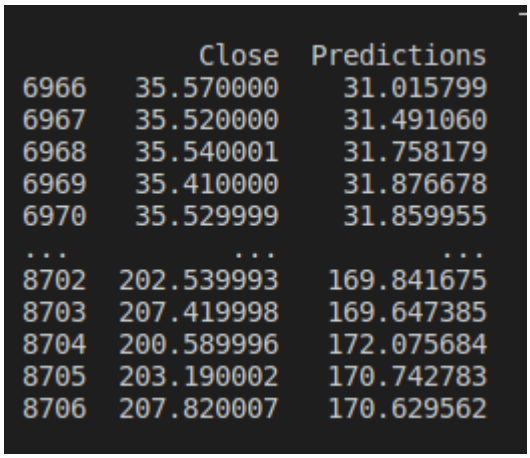
Crtanje grafa krivulje.

Analiza rezultata



Plava krivulja predstavlja podatke koji su se koristili za treniranje i njihova vrijednost je jednaka stvarnoj vrijednosti. Crvena krivulja predstavlja stvarnu vrijednost, ali ti podaci nisu bili poznati LSTM mreži. Narančasta krivulja predstavlja predviđanja. Kao što se vidi iz priloženog, predviđanje se na prvi pogled čini približna stvarnoj cijeni. Graf obuhvaća interval od 34 godine i to znači da iz njega ne možemo dovoljno dobro iščitati predviđanja jer je gustoća prevelika, ali je jasno što je veća vrijednost na x osi to očekivanje više odstupa od stvarnosti. Predviđanje je korisno samo za dugotročnu

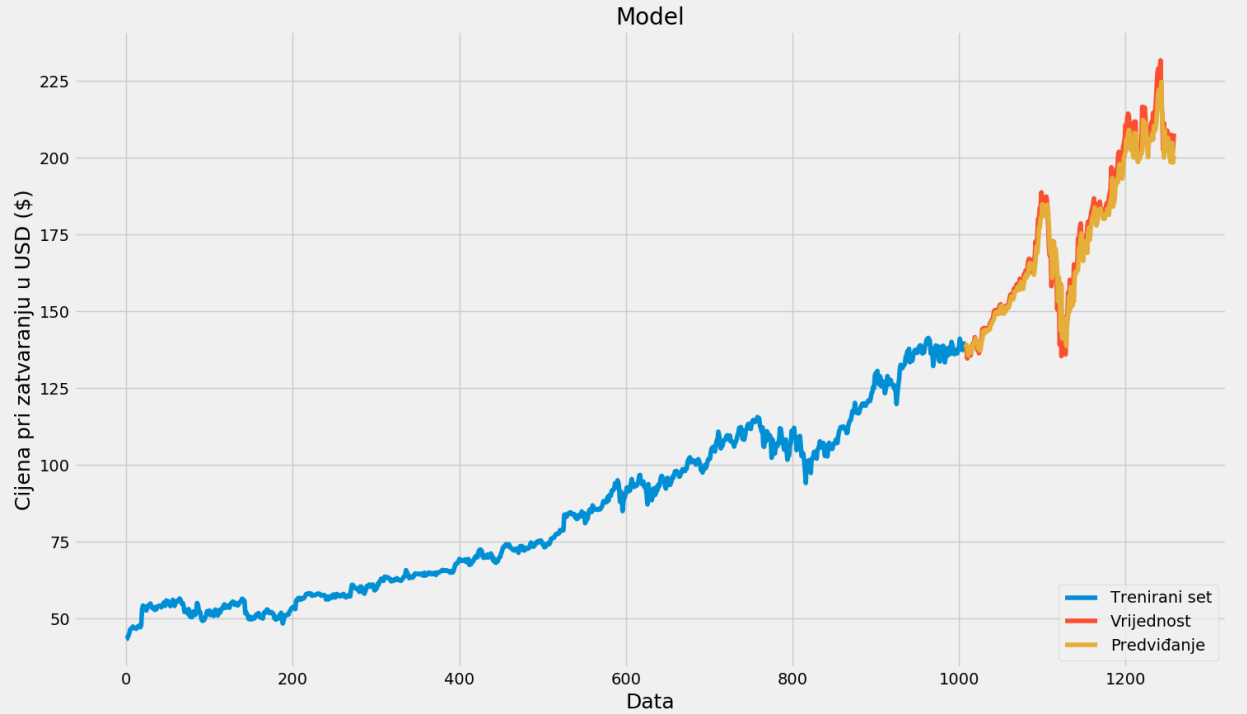
investiciju radi veće „otpornosti“ na manja, nagla kretanja (dio profita se akumulirao s vremenom pa manje oscilacije ne uzrokuju veliku brigu). Vjerojatno je greška u korištenju metode ili je greška napravljena u kodu, jer odstupanja ne bi trebala biti tako velika. Pogreška za sada ostaje nejasna. Razlog zašto ovaj graf nije zanemariv je zato što vrijednost dionica s vremenom, ako se uzme dovoljno dug vremenski period raste. Kad se promatra veliki set u vremenskom intervalu od 34 godine logično je očekivati uzlazno kretanje vrijednosti.



	Close	Predictions
6966	35.570000	31.015799
6967	35.520000	31.491060
6968	35.540001	31.758179
6969	35.410000	31.876678
6970	35.529999	31.859955
...	...	...
8702	202.539993	169.841675
8703	207.419998	169.647385
8704	200.589996	172.075684
8705	203.190002	170.742783
8706	207.820007	170.629562

Tablica sadrži vrijednost zatvaranja i predviđenu vrijednost. Potrebno je promotriti informaciju na što više načina, jer se neki rezultati mogu slučajno krivo protumačiti pa na ovaj način smanjujemo šansu za pogreškom u zaključivanju. Promatranje grafa ponekad nije najbolja metoda promatranja. Vidimo da vrijednost zatvaranja zadnji dan 207.82$, a predviđeno je da će biti 170.63$, to je razlika od 18%. Ako se uzme u obzir da je neto prosječan povrat na investiciju u godinu dana od 5%-11%, može se zaključiti da ovaj model nije najbolji. Mogući razlog za loš rezultat je kaotično stanje burze u zadnjih par mjeseci. Dionice američke burze, zadnjih par mjeseci, imaju jako strm rast i to je možda negativno utjecalo na rezultat, jer svaku neobičnost ovakvi modeli kasno registriraju.

Potrebno je još istraživati promjenom broja podataka i brojem prolaza kroz mrežu za donošenje objektivnijih zaključaka. Ovaj put će set biti proveden kroz mrežu više puta (epoch=10), dijelom zato što jer je broj podataka manji. Period promatranja će biti 5 godina te dalje promatramo vrijednost dionica Microsoft.corp.



Korelacija na grafu ju definitivno veća. Razlog je vjerojatno što dugotrajan spori rast previše utječe na mrežu. Moguće da je to razlog zbog kojeg mreža u prvom primjeru daje slabiji rezultat. Kontinuiran spori rast je istrenirao mrežu za uvjete koji se nisu dogodili i mreža se nije mogla prilagoditi nagloj promjeni, odnosno proces prilagodbe traje duže nego očekivano.

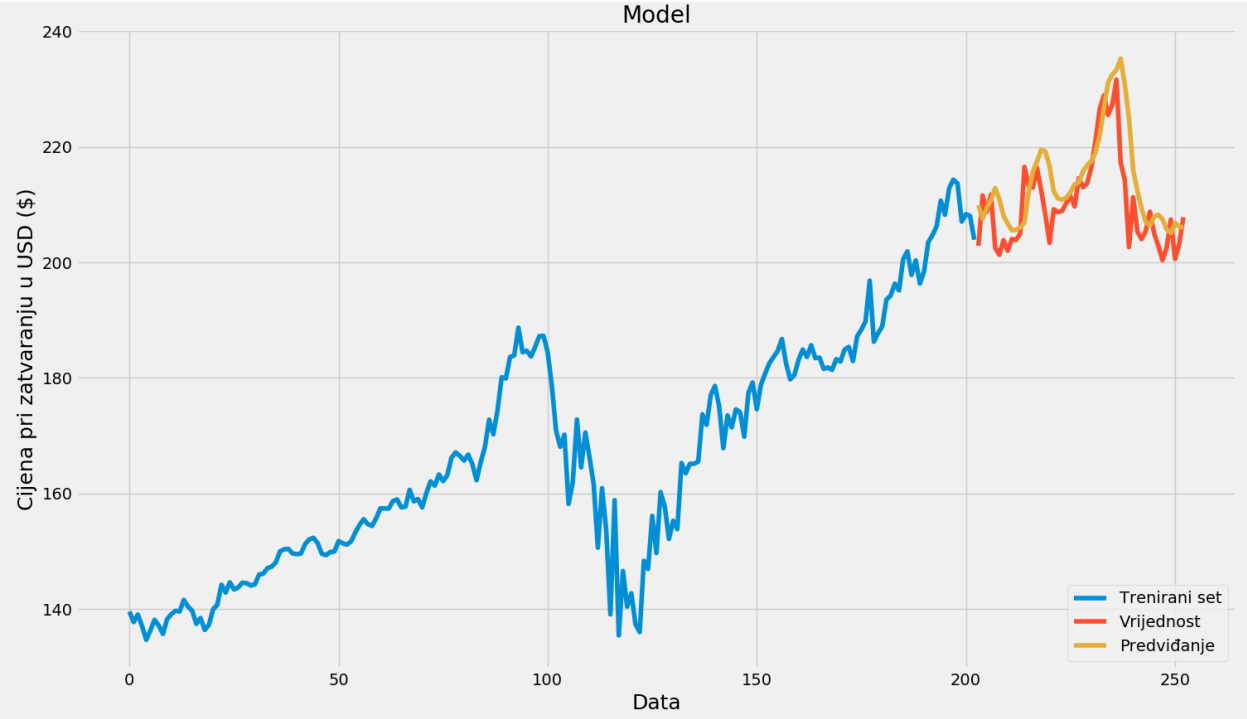
	Close	Predictions
1008	139.029999	138.293747
1009	137.070007	139.111115
1010	134.649994	137.748383
1011	136.279999	135.687073
1012	138.119995	136.694458
...	...	...
1254	202.539993	198.540375
1255	207.419998	200.529175
1256	200.589996	204.807831
1257	203.190002	198.388428
1258	207.820007	200.667282

Je li u drugom testu riječ o povoljnom rezultatu? Razlika između cijene zatvaranja i predviđene cijene je oko 4% (pogreška je puno manja). Je li to indikator da je ovaj algoritam dovoljno dobar za primjenu u praksi? Ako je ideja ulagati na duže vrijeme od npr. dvije godine, mislim da bi se ova informacija mogle uzeti u obzir. Ako je povrat na

investiciju od 5%-11%, a godišnja inflacija je u prosjeku od 2%-5%, pogreška od 4% je prevelika.

Izmjenom epoch varijable na vrijednosti 15 i 5 (s podacima u intervalu od 5 godina), ishod je lošiji nego, za epoch=10. Dakle, dokazano je da je u ovom slučaju broj prolaza jako bitan. Zanimljivo je da je testirana pogreška veća na primjeru koji je imao epoch=15 (greška od 14%) nego, epoch=5 (greška od 7%).

Predviđanje u intervalu od godinu dana vodi do neočekivano povoljnih rezultata.



Greška je u prosjeku oko 2.5%. Što čini grešku dovoljno niskom da se model može koristiti u praksi.

Close	Predictions
202.880005	209.910690
211.600006	207.565948
208.750000	209.394516
211.750000	210.761703
202.539993	212.865067
201.300003	210.927719
203.850006	208.052216
202.020004	206.728043
204.059998	205.553589
203.899994	205.638626
205.009995	206.043945
216.539993	206.902695
213.289993	212.119110
212.940002	215.649063
216.350006	217.478424
212.479996	219.492554
208.250000	219.220596
203.380005	216.662430
209.190002	212.317291
208.699997	211.067642
208.899994	210.846085
210.279999	211.160446

Upitna je korisnost LSTM mreže u ovom slučaju, jer se vidi da predviđanje teško sustiže nagle promjene vrijednosti. Mislim da je algoritam previše osjetljiv s obzirom da koristi zadnjih 60 dana da predvidi sljedeći jedan. Ponavljanjem takvog procesa, pretpostavljam, bi dovelo do prevelike šanse za pogrešku (što rezultati u ovom slučaju ne pokazuju). Možda bi primjena nekog dodatnog mehanizma pomogla da predviđanje ne zaostaje toliko za naglim promjenama u cijeni. Potrebno je još puno testiranja da se može zaključit nešto konkretnije. Jedna stvar se može zaključiti, a to je da pogreška raste proporcionalno s vremenom i da investitor taj faktor mora uzeti u obzir. Investicija se vjerojatno isplati, ako se isplata može čekati dovoljno dugo. Kratkoročna investicija se često isplati, ali profit je manji, a rizik veći. Iznijeti odvažniji zaključak od ovog bi bilo neodgovorno, barem ako ga donosi netko bez iskustva ili dobrog poznavanja tržišta.

Zaključak

Strojno učenje, iako jako zahtjevno ima veliki broj jednostavnih alata za korištenje. Proučavanjem umjetnih mreža vidimo gotovo neizostavne mogućnosti njihove primjene. Za rad u strojnom učenju potrebno je poznavati puno teorije koja je jednako zanimljiva koliko i raznolika. TensorFlow je zabavan i jednostavan alat za korištenje s velikim brojem biblioteka i API programa koji čine rad interaktivnijim. LSTM model se pokazao jako dobar za kratkoročna predviđanja, a pogreška pri dugoročnom promatranju je manja nego očekivano. Pogreška ne znači nužno da nam informacija nije korisna. Nju je moguće gledati kao upozorenje na oprez pri korištenju modela, što zapravo svako od priloženih testiranja čini korisnim. Radom u strojnom učenju stječemo „mudrost“ o polju kojeg istražujemo i vjerojatno bi svim razvojnim inženjerima koristio kao alat za razumijevanje struke za koju izrađuju aplikaciju.

[1] Josh Patterson And, *Deep Learning A Practitioner's Approach*. .
[2] P. Domingos, "A Few Useful Things to Know about Machine Learning."
[3] Xiao Ding, Yue Zhang, Ting Liu, Junwen Duan, "Deep Learning for Event-Driven Stock Prediction."
[4] "Documentation - Weka Wiki." https://waikato.github.io/weka-wiki/documentation/ (accessed Sep. 27, 2020).
[5] "API Documentation." https://www.tensorflow.org/api_docs (accessed Sep. 28, 2020).

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam završni rad izradio samostalno, isključivo znanjem stečenim na studijima Sveučilišta u Dubrovniku, služeći se navedenim izvorima i uz stručno vodstvo mentora izv. prof. dr. sc. Mario Miličević, kojem se još jednom srdačno zahvaljujem.

Frano Rihter