Predviđanje tržišnih vrijednosti strojnim učenjem

Rašetina, Matia

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski studij računarstva

Završni rad

Predviđanje tržišnih vrijednosti strojnim učenjem

Rijeka, rujan 2022.

Matia Rašetina
0069088300

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski studij računarstva

Završni rad

Predviđanje tržišnih vrijednosti strojnim učenjem

Mentor: doc. dr. sc. Goran Mauša

Rijeka, rujan 2022.

Matia Rašetina
0069088300

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2022.

_____

Ime Prezime

Zahvala

Velike zahvale mentoru doc. dr. sc. Goranu Mauši na kontinuiranim savjetima i pomoći tijekom pisanja ovog završnog rada. Također, velike zahvale mojoj obitelji, kolegama i prijateljima na velikoj podršci tijekom studija.

Sadržaj

Poglavlje 1

Uvod

Svaki investitor ili osoba koja se bavi financijama želi predvidjeti tržište u kojem se nalazi - od tržišta kriptovaluta, nekretnina, dionica, sve do valutnih parova. Algoritamsko trgovanje krenulo je tijekom 1970-tih godina i svaka je osoba htjela znati koji je sljedeći korak nepredvidljivog tržišta u kojem se u bilo kojem trenutku može desiti nagli pomak. Što se više približavamo modernijem dobu, kako je sve više i više ljudi na računalima te imaju pristup internetu, počeli su se razvijati servisi poput web-stranice TradingView (slika 1.1) [1] gdje u jednom kliku možete doznati cijenu bilo kojeg valutnog para, dionice ili kriptovalute. Sve više korisnika se okreće algoritamskim rješenjima poput automatskog crtanja i prepoznavanja trendova i tehničkih indikatora i algoritmi za predviđanje buduće cijene na temelju cijena u prošlosti. Svako tržište je u suštini more podataka - kupovna cijena, prodajna cijena, razlika između istim (inače poznato kao *spread*), volumen prodaje, te se svi ovi podaci mogu iskoristiti za pronalazak trenda, što možemo vidjeti na slici 1.1 - X os predstavlja vrijeme, Y os predstavlja cijenu dionice odnosno valutnog para. Svaki od ovih podataka ima potencijal usmjeriti korisnika na pravi put, iako većina korisnika, nažalost, ne završe svoju trgovačku karijeru u profitu, najčešće su razlozi poput neiskustva na tržištu, ulaganje u krivi algoritam ili nespremnost čovjeka i algoritma na događaje koji su nepredvidljivi, imaju ozbiljne posljedice na svjetsku ekonomiju. U ekonomiji, ovakvi događaji spadaju pod teoriju "Crnog labuda".[1]
Iz ovih razloga, proizašla je motivacija za izradom ovog projekta u kojoj se poku-

[1] TradingView - `tradingview.com`

šavaju pronaći modeli umjetne inteligencije koji bi ponudili savjet kako postupiti u bilo kojem trenutku na tržištu, uzeći u obzir sve uspjehe i padove. [1, 2, 3, 4]

Zadatak ovog projekta jest napraviti aplikaciju koja će, korištenjem algoritama Multi-Layer Perceptron i Long Short-Term Memory, raditi predviđanje o cijenama dionica na američkom tržištu obveznica odnosno predviđanje o parovima svjetskih valuta. Multi-Layer Perceptron algoritam će se koristiti za predviđanje cijena na tržištu valuta, dok će se Long Short-Term Memory algoritam koristiti za predviđanje cijena javnih američkih tvrtki na tržištu.

Slika 1.1 Sučelje stranice TradingView koje pokazuje graf cijene kriptovalute Ethereum u američkim dolarima

Poglavlje 2

Korištene tehnologije i metode

2.1 Python

Python je programski jezik koji je stvorio Guido van Rossum 1990. godine. Za razliku od programskih jezika kao što su C, C++ i Java, koji se prevode u binarni kod uz pomoć njihovih prevodioca, Python je jedan od jezika koji se interpretira. Isto tako, Python pripada jezicima koji su objektno orijentirani programski jezici poput C++-a, Jave, Javascripta i C#-a.[5] Danas, Python je jedan od najpopularnijih jezika, a razlozi su sljedeći:

- **Jednostavna sintaksa** - Python je razvijen s naglaskom na jednostavnost sintakse - cilj je bio približiti se engleskom jeziku što je više moguće,
- **Veliki broj knjižnica** - Python podržava veliki broj knjižnica - od knjižnica za crtanje grafova do knjižnica za izračun kompleksnih matematičkih funkcija i jednadžbi,
- **Brzina i pouzdanost** - S obzirom da je Python u svojoj jezgri C, to mu donosi veliku brzinu izvršavanja. Također vrijedi spomenuti i da Python podržava funkcije iz C i C++-a,
- **Umjetna inteligencija i strojno učenje** - Danas, umjetna inteligencija i strojno učenje su sve popularnije teme. Python je jezik u koji se inženjeri mogu pouzdati kako bi riješili zadatak brzo i efikasno.[6]

2.1.1 Korištenje programske knjižnice za ostvarenje projekta

Tkinter

Tkinter je knjižnica koja omogućuje laku izradu korisničkih sučelja koristeći programski jezik Python. U ovom projektu, knjižnica Tkinter će se koristiti za izradu sučelja za korisnika koji želi saznati buduću cijenu neke dionice ili nekog para na tržištu valuta. [1]

Keras

Također jedna od najpoznatijih knjižnica za umjetnu inteligenciju i strojno učenje. Ova knjižnica utemeljena je na drugoj inačici Tensorflowa. Prednost ove knjižnice je što se može izvršiti na više platformi, od operativnih sustava utemeljenih na jezgri UNIX do operativnog sustava Windows te se isto tako može izvršiti unutar preglednika. U ovome projektu, Keras će se koristiti u dijelu za predviđanje cijene dionica na američkom tržištu.[7] [2]

Matplotlib

Matplotlib je sveobuhvatna knjižnica za stvaranje statičnih, animiranih i interaktivnih vizualizacija u Pythonu. U ovom projektu, koristit će se za izradu grafova cijena dionica i parova valuta.[8] [3]

Pandas

Python knjižnica koja se koristi za analizu podataka te sadrži funkcije koje pomažu u manipuliranju istih. Ovdje će se koristiti za obradu Forex podataka. [9] [4]

[1] Tkinter - `docs.python.org/3/library/tkinter.html`
[2] Keras - `keras.io`
[3] MatPlotLib - `matplotlib.org`
[4] Pandas - `pandas.pydata.org`

NumPy

NumPy je temeljni paket za znanstveno računalstvo u Pythonu. To je knjižnica koja pruža višedimenzionalni objekt niza, razne izvedene objekte (kao što su maskirani nizovi i matrice) i asortiman rutina za brze operacije na nizovima, uključujući matematičke, logičke, manipulaciju oblikom, sortiranje, odabir, diskretne Fourierove transformacije, osnovna linearna algebra, osnovne statističke operacije, slučajna simulacija i još mnogo toga. Koristit će se u dijelu za izradu matrica u kojima će biti podaci bitni za *Long Short-Term Memory* algoritam.[10] [5]

Torch

Torch je knjižnica koja sadrži mnogo korisnih funkcija za izradu modela umjetne inteligencije i strojnog učenja. Poanta ove knjižnice je pojednostavljen prijelaz iz prototipa u produkciju. Isto tako, jedna je od rijetkih knjižnica koja se fokusira na jednostavnost izvršavanja "u oblaku" odnosno na servisima poput Amazon Web Services, Microsoft Azure i sl.[11] Knjižnica će se koristiti u Forex dijelu projekta.[6]

[5] NumPy - `numpy.org`
[6] PyTorch - `pytorch.org`

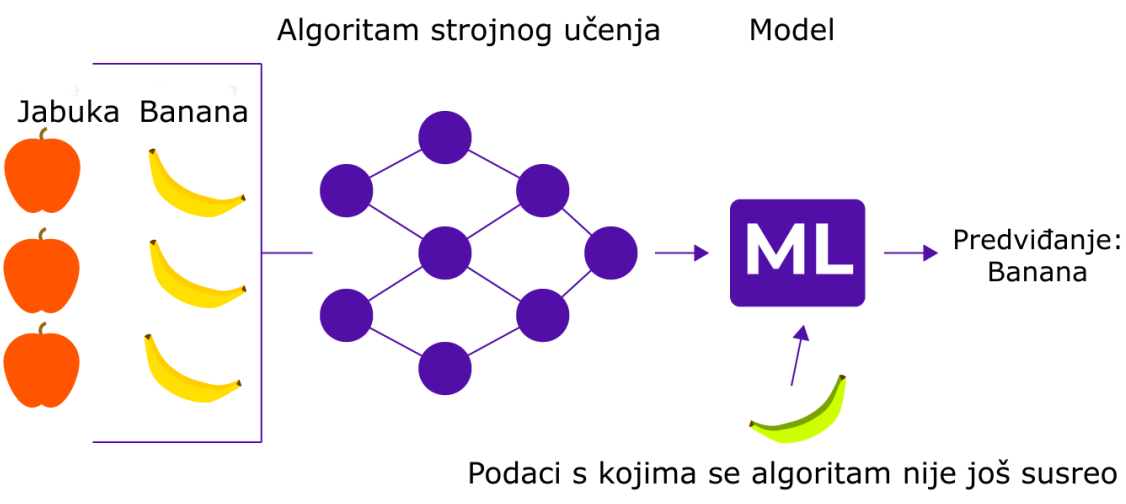
2.2 Strojno učenje

Strojno učenje je polje umjetne inteligencije u kojem se pokušavaju stvoriti i razumjeti metode koje se koriste za poboljšavanje performansi u izvršavanju nekog zadatka. Modeli strojnog učenja zahtijevaju početnu skupinu podataka za treniranje modela te tako model prepoznaje relacije između dostavljenih podataka. Kasnije, na temelju tih korelacija, model je sposoban ispostaviti predviđanje vrijednosti odnosno karakterizaciju novog podatka.

Osnovnu podjelu područja strojnog učenja možemo svesti na:

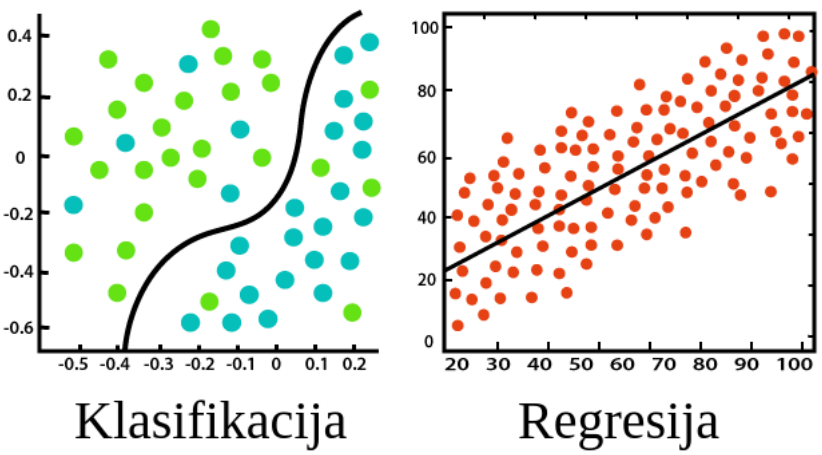
1. Nadzirano učenje,
2. Nenadzirano učenje.

Modeli nadziranog učenja se uvježbavaju odnosno grade primjenom skupa ulaznih i izlaznih podataka prikupljenih eksperimentom kako bi mogli predviđati buduće ishode na temelju dostupnih podataka. Nadzirano učenje možemo podijeliti na klasifikaciju i regresiju.



Slika 2.1 Primjer nadziranog učenja [12]

Klasifikacija koristi predviđanje diskretnih odziva u kojem se podaci razvrstavaju u kategorije, dok se u regresiji predviđaju kontinuirani odzivi - poput promjene temperature, potrošnja električne energije i sl. (slika 2.2) U ovom projektu, s obzirom na to da moramo predvidjeti buduće cijene dionica i Forex parova, korišteni algoritmi pripadaju **regresijskome** dijelu nadziranog učenja.



Slika 2.2 Usporedba klasifikacije i regresije

Neki od modela regresije su sljedeći: linearna regresija i umjetne neuronske mreže. Linearna regresija se koristi kada se želi na temelju jedne ili više ulaznih varijabli predvidjeti vrijednosti budućih varijabli. Glavni cilj modela je pronaći matematičku funkciju koja odgovara svim podacima. Linearna regresija je korisna kada se želi prepoznati smjer gibanja cijene, što možemo vidjeti na slici 2.3. Žuta linija predstavlja vrijednosti koje su definirane s formulom pravca: $y = mx + b$. Kao što se može vidjeti na slici, s obzirom na to da je nagib pravca negativan, to indicira negativan trend, no ipak nije ponuđena opcija previđanja buduće cijene, što je puno vrjednija informacija i zbog tog razloga, umjetne neuronske mreže su puno bolji odabir. [13]

U ovome projektu, koristi se umjetna neuronska mreža kao model strojnog učenja.

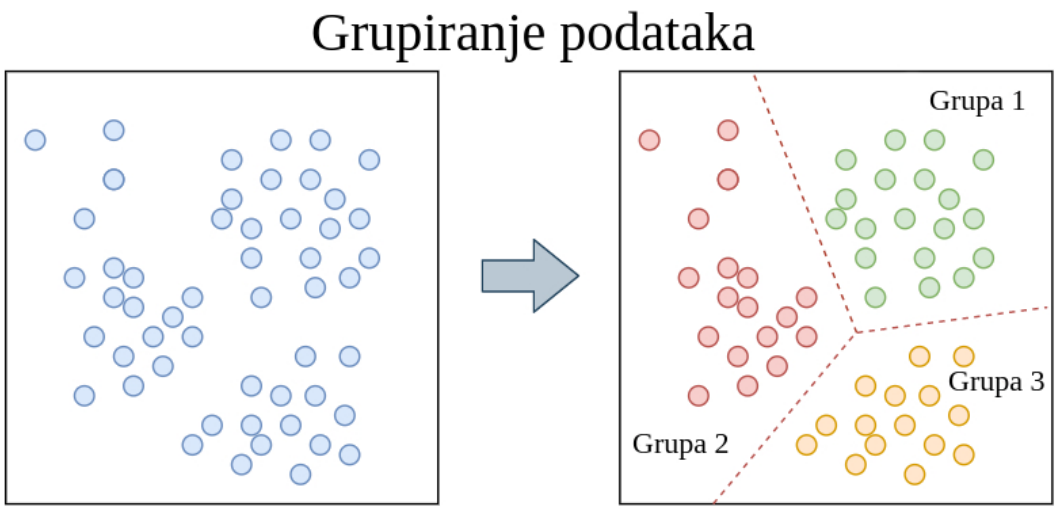
Slika 2.3 Primjer linearne regresije unutar grafa s cijenama u prošlosti

Kao što je spomenuto prije, modeli klasifikacije koriste ulazne podatke kako bi se napravile kategorije. Postoje dvije vrste klasifikacije u strojnome učenju:

1. Binarno klasificiranje - problem koji ima samo dvije moguće klasifikacije npr. je li životinja na nekoj slici pas ili mačka,
2. Višeklasna klasifikacija - problem u kojem je objekt, sa svojim karakteristikama, može biti dio jedne ili više grupa.

Modeli nenadziranog učenja rade na suprotan način od modela nadziranog učenja - omogućuje treniranje modela s podacima koji nisu specificirani za što se koriste. Modeli unutar ovog djela strojnog učenja pokušavaju naći i naučiti skrivene uzorke u podacima koji se nalaze na ulazu algoritma. [14] Neki od modela u nenadziranom učenju su:

1. Grupiranje - tehnika nenadziranog strojnog učenja u kojem se svi dostupni podaci dijele u grupe na temelju njihovih međusobnih sličnosti
2. Učenje asocijativnih pravila - tehnika u kojoj se pokušavaju naći uzorci u velikoj količini podataka. Cilj ove tehnike je uzeti jedan podatak te pokušati naći ovisnost tog podatka o nekom drugom i tako stvoriti odnose između svih dostupnih podataka. [15]



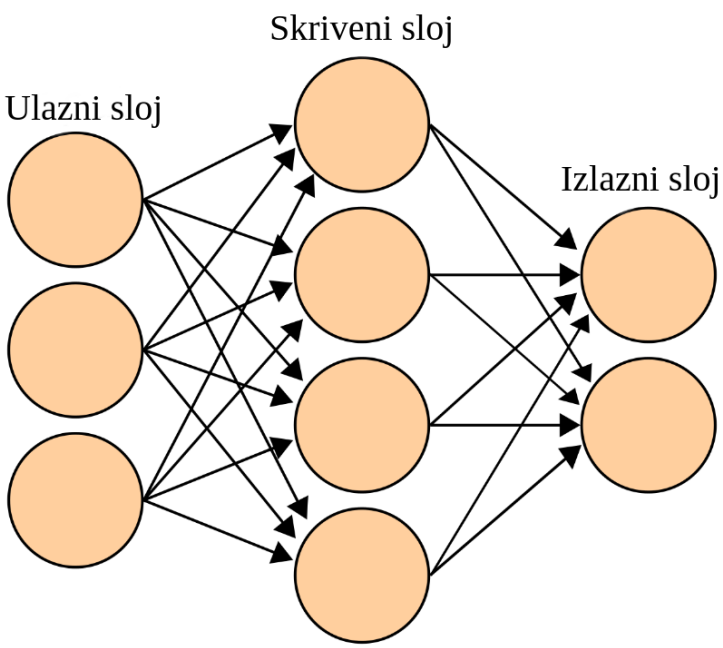
Slika 2.4 Primjer nenadziranog učenja - Grupiranje[16]

2.2.1 Neuronske mreže i korišteni algoritmi

Neuronska mreža je zbir umjetnih neurona koji su međusobno povezani i interaktivni kroz operacije obrade signala. Uređena je po uzoru na ljudski mozak. Bilo koja neuronska mreža može imati jedan ili više ulaza odnosno jedan ili više izlaza, no između ulaza i izlaza nalaze se tzv. skriveni slojevi koji su međusobno povezani i kroz njih idu signali. Za validaciju rezultata dobivenim algoritmima koristit ćemo korijen srednje kvadratne pogreške (eng. *Root-Mean-Squared Error*) čija je matematička formula:

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2} \tag{2.1}$$

gdje n predstavlja veličinu podataka i $d_i - f_i$ predstavlja razliku između predviđenih i pravih podataka te se ta razlika kvadrira kako vrijednost ne bi mogla biti manja od 0. Što je vrijednost veća, to nam govori kako algoritam daje puno grešaka, no što je vrijednost manja i bliža 0, to nam pokazuje da je algoritam precizniji. [17, 18]



Slika 2.5 Pojednostavljeni prikaz umjetne neuronske mreže

Višeslojni Perceptron

Višeslojni perceptron (nadalje *MLP*) pripada algoritmima koji koriste umjetne neuronske mreže kako bi došli do previđanja vrijednosti na temelju podataka u prošlosti. Kao i ostali algoritmi u ovoj grupi, imaju ulazne, skrivene i izlazne čvorove. Svaki čvor, osim ulaznih, koristi nelinearnu aktivacijsku funkciju kako bi se zadaća čvora izvršila. Također, vrijedi spomenuti kako ovaj algoritam pripada polju nadziranog učenja.

U umjetnim neuronskim mrežama, aktivacijska funkcija čvora definira izlaz svakog čvora u neuronskoj mreži s obzirom na skup ulaza. U ovome algoritmu, najčešće se koriste linearne aktivacijske funkcije odnosno funkcija preslikava otežane ulaze u izlaz svakog neurona. Također, najčešće korištene funkcije su sigmoidne funkcije te se definiraju sljedećim formulama:

$$y(v_i) = tanh(v_i) \tag{2.2}$$

$$y(v_i) = (1 + e^{-v_i})^{-1} \tag{2.3}$$

Učenje, odnosno treniranje, se izvršava nakon svake iteracije odnosno nakon procesiranja podatka koji prođe kroz neuronsku mrežu te se težine neurona ažuriraju na temelju grešaka koje se stvore na izlazu u usporedbi s očekivanim brojem grešaka i iz tog razloga MLP algoritam pripada skupini nadziranog učenja.

Možemo izračunati stupanje greške nekog čvora s formulom:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.4}$$

gdje je $\mathbf{d}$ ciljana vrijednost i $\mathbf{y}$ dobivena vrijednost. Težina čvorova se zatim ažurira na temelju korekcija koje smanjuju grešku u dobivenom izlazu:

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{2.5}$$

te zatim dolazi do promjene težine neurona:

$$\Delta w_{ji}(n) = -\eta\frac{\delta\varepsilon(n)}{\delta v_j(n)}y_i(n) \tag{2.6}$$

Treniranje se izvršava algoritma propagiranja pogreške unazad te koraci su sljedeći[19, 20]:

- Inicijaliziranje težine neurona,
- Predstaviti prvi podatak mreži iz podataka za treniranje,
- Primanje izlaza neuronske mreže,
- Izračunavanje signala greške formulama navedenim iznad kako bi se usporedio dobiveni i ciljani signal,
- Vraćanje signala nazad kroz neuronsku mrežu kako bi se ažurirale težine neurona,
- Ponavljanje procesa više puta s novim podatkom kako bi se signal greške što više približio ciljanom signalu.

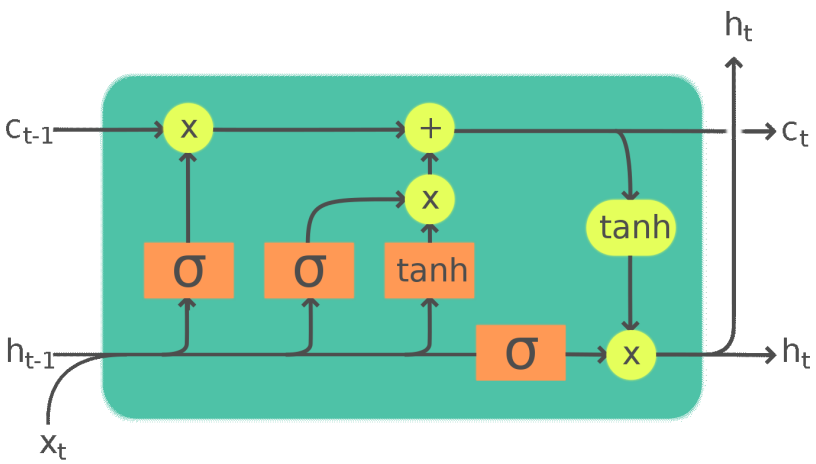
Long Short-Term Memory algoritam

Long Short-Term Memory je vrsta umjetne neuronske mreže koja je specifično dizajnirana kako bi bila u mogućnosti učiti o dugoročnim ovisnostima podataka, što je kod tipičnih neuronskim mreža jedan od glavnih problema. Ovaj algoritam koristi povratne veze, što čini ovaj algoritam drukčiji od drugih - ova vrsta veza omogućava procesiranje sekvenci podataka u cjelini, a ne svaki podatak zasebno.
Kao i Multi-Layer Perceptron, LSTM također sadrži skrivene slojeve, no za razliku od MLP-a, koji u svojem skrivenom sloju ima obične čvorove s matematičkim funkcijama, LSTM sadrži posebne čvorove zvane "memory blocks" koji sadrže veze koje pohranjuju vremensko stanje mreže te unutar svakog čvora se nalaze posebna multiplikativna vrata koja kontroliraju slijed informacija na ulazu u čvor.

Na slici 2.6 možemo vidjeti sljedeće stavke:

- C_{t-1} - stanje čvora koji se nalazi u sloju prije,
- H_{t-1} - prijašnje skriveno stanje,
- X_t- podatak koji se nalazi na ulazu,
- σ - sigmoidna funkcija,

- "tanh" vrata - funkcija hiperboličkog tangensa,
- vrata "X" - množenje funkcija,
- vrata "+" - zbrajanje funkcija,
- C_t - stanje nakon izvršetka čvora,
- H_t - trenutno skriveno stanje.



Slika 2.6 Čvor skrivenog sloja Long Short-Term Memory algoritma (preuzeto iz [21])

Prvi korak procesa su vrata zaborava (eng. *forget gate*). Na početni dio čvora dolaze ulazni podaci te podaci od čvora iz prijašnjeg skrivenog sloja. Na temelju tih podataka, odlučuje koji su dijelovi podataka korisni i koji nisu. Stanje čvora iz prijašnjeg skrivenog sloja prosljeđuje podatke u obliku vektora gdje svaki element ima vrijednost između 0 i 1, što je osigurano aktiviranjem sigmoidne funkcije. Ovaj proces je naučen na sljedeći način - ako algoritam smatra kako podatak nije bitan, njegova vrijednost u vektoru će biti bliža broju 0, dok s druge strane, ako algoritam smatra da je podatak iznimne vrijednosti, vrijednost podatka u vektoru će težiti broju 1.

Drugi korak procesa je stvaranje memorijske mreže i ulaznih vrata (eng. *input gate*). Cilj ovog koraka je odlučiti koji od novih podataka će se dodati u memorijsku mrežu. Vrijedi spomenuti kako vrata zaborava i ulazna vrata imaju iste ulaze. Novi dio memorijske mreže se aktivira unutar funkcije hiperboličnog tangensa koji kombinira prethodno skriveno stanje i novi podatak kako bi stvorio novi memorijski vektor koji definira za koju vrijednost se svaki element memorijske mreže mora ažurirati. S druge strane, ulazna vrata odnosno sigmoidna funkcija je identična vratima zaborava - filtrira podatke te odlučuje koje podatke je vrijedno zapamtiti. Kao i u prvom koraku, svaki podatak dobije vrijednost između brojeva 0 i 1 kako bi se odredila važnost odnosno utjecaj podatka.

Na kraju se nalaze izlazna vrata (eng. *output gate*) koja odlučuju skriveno stanje koje se prosljeđuje u sljedeći čvor. Kako bi se to odlučilo, trebaju se koristiti ažurirano stanje memorijske mreže, prijašnje skriveno stanje te podaci na ulazu čvora. Izlazna vrata su identična zaboravnim i ulaznim vratima, koristi se kao filter podataka te se izlazna vrata koriste za filtriranje podataka koji su se nedavno spremili u memorijsku mrežu čvora kako bi se samo bitne informacije proslijedile u sljedeći čvor. No prije same filtracije podataka, podaci prolaze kroz funkciju hiperboličkog tangensa i podaci dobiju vrijednosti između brojeva -1 i 1. Zatim se podaci iz prijašnjeg skrivenog sloja, zajedno s novim podacima šalju na "X" vrata i tako se stvara novo skriveno stanje.(slika 2.6) [22, 23]

Definiranje broja neurona u skrivenim slojevima

Količina neurona u skrivenim slojevima definirani su na temelju potrage na Internetu gdje se u zajednici najčešće koriste pravila:

1. Broj skrivenih neurona mora biti između veličine ulaznog i izlaznog sloja
2. Broj skirvenih neurona mora biti 2/3 veličine ulaznog sloja te se na taj broj dodaje veličina izlaznog sloja
3. Broj skivenih neurona mora biti manji od dvostruke veličine ulaznoga sloja

[24]

Za oba modela korisit ćemo isti broj ulaznih, skrivenih i izlaznih neurona kako bismo mogli što bolje usporediti rad algoritama.

2.3 Definicija tržišta

Tržište, po najkraćoj definiciji, predstavlja mjesto gdje se susreću ponuda i potražnja. Ovaj projekt sadrži interakciju s dva tržišta koja se nalaze na internetu - Foreign Exchange i New York Stock Exchange.

2.3.1 New York Stock Exchange

New York Stock Exchange, nadalje NYSE, je američka burza čije se središte nalazi u New Yorku. To je najveća svjetska burza čiji prosječni dnevni volumen trgovanja iznosi 169 milijardi američkih dolara. Započela je s radom 1792., kada je došlo do trgovanja prvim obveznicama. Danas, vrijednost svih korporacija na ovoj burzi premašuje 30 bilijuna američkih dolara.
Burza omogućuje elektroničku prodaju obveznica uz pomoć posrednika - pravnog tijela koji spaja kupca i prodavača kako bi se transakcija izvršila što efikasnije. Proces kupnje je identičan kao i u Forex tržištu - najčešće kupac i prodavač dogovore cijenu po zadnjoj prodajnoj cijeni, no prodavač također može ručno namjestiti cijenu po kojoj želi prodati.

2.3.2 Foreign Exchange tržište

Forex, kratica od *"foreign exchange"*, je međubankovno tržište ustanovljeno 1971. godine kada su uvedeni plivajući tečajevi. To je tržište na kojem se valuta jedne zemlje mijenja za valutu druge po cijeni koja se zove tečaj. Forex čini grupa od otprilike 5000 institucija koje se bave trgovanjem valutama, što uključuje međunarodne banke, centralne banke, financijske institucije i pojedince. Postoje dvije vrste pozicija: long i short. Korisnik uzima "long" pozicije ako smatra kako će vrijednost para dobiti na vrijednosti te "short" pozicije ako smatra kako će vrijednost para izgubiti na vrijednosti.

Brojne su prednosti trgovanja unutar ovog tržišta:

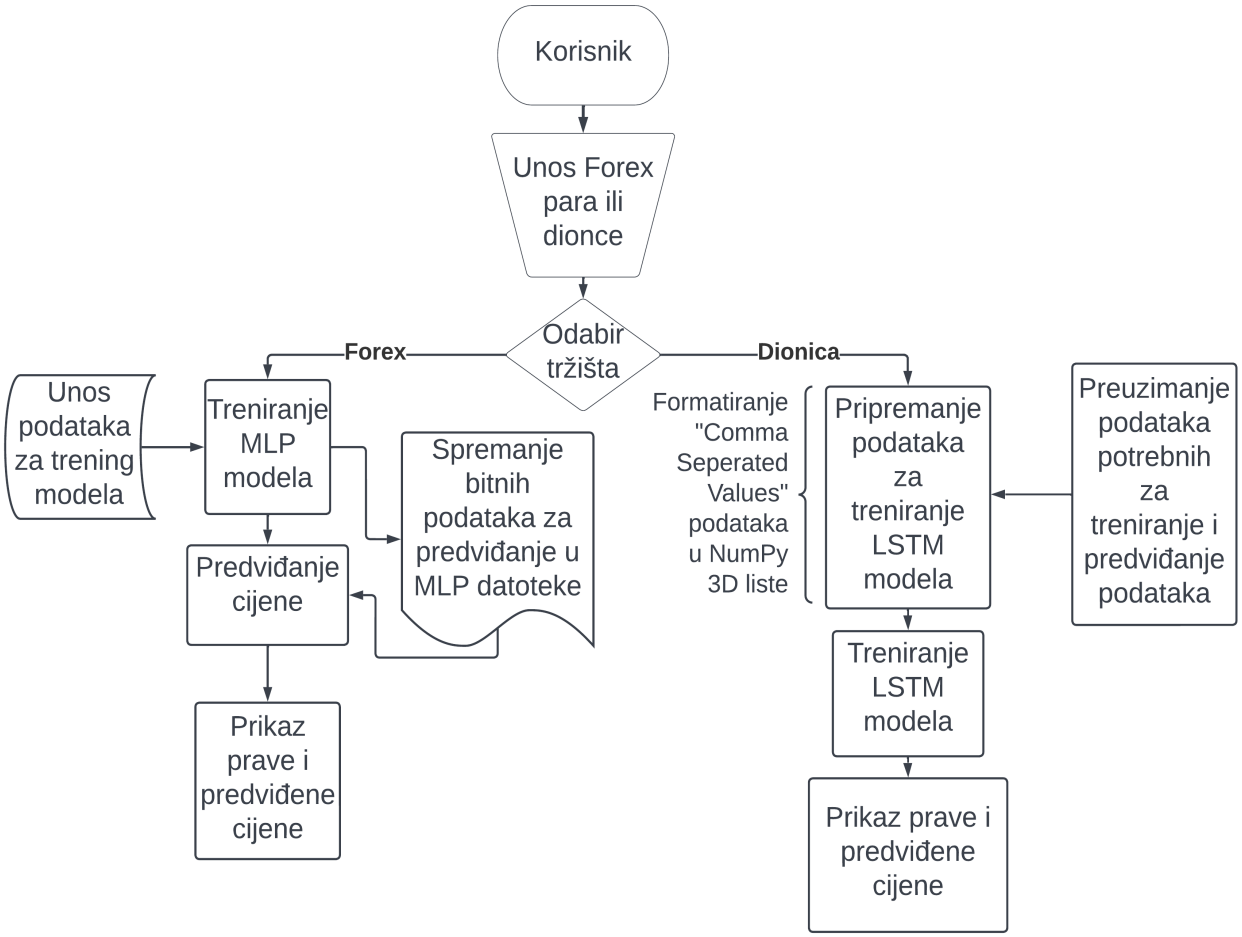
- **Likvidnost** - na tržištu uvijek postoji kupac ili prodavač - tržište je otvoreno tijekom 5 radnih dana bez prestanka, tako da je moguće u bilo kojem trenutku zatražiti neku od pozicija
- **Trgovanje uz polugu** - mnoge firme koje nude uslugu trgovanja raznim valutnim parovima nudi trgovanje na polugu - firma "posuđuje" novac klijentu kako bi korisnik bio u mogućnosti ući u poziciju s većom kupovnom moći. Iako ovo povećava profite, ako se krivo iskoristi, uvelike povećava i rizik za gubitkom. Najčešće "leverage" opcije na tržištu su 100X i 500X što povećava klijentovu kupovnu moć za 100 odnosno 500 puta.
- **Kvaliteta izvršenja** - Forex tržište je jako likvidno te se transakcije izvršavaju u vrlo malom vremenskom periodu. Većina transakcija će se zatražiti i izvršiti po posljednjoj tržišnoj cijeni, no s obzirom da je tržište iznimno brzo te ponekad dođe do naglih promjena u cijeni, korisnik mora biti svjestan rizika za "klizanje" cijene - ulazak u poziciju s cijenom koja je nepovoljnija od početno zamišljene. [25]

Poglavlje 3

Izrada korisničkog sučelja

Kao što je rečeno na početku ovog rada, koristit će se knjižnica Tkinter koja omogućuje da se isto grafičko sučelje prikazuje na različitim operativnim sustavima bez utjecaja na korisničko iskustvo. Nudi razne opcije te neke od opcija ćemo vidjeti u ovom projektu. [26]

Kako bismo opisali rad korisničkog sučelja, koristi će se dijagram tijeka kojim možemo prikazati sve funkcionalnosti aplikacije. Slika 3.1 opisuje rad aplikacije. Korisniku se prikaže korisničko sučelje u kojem može birati želi li dobiti predikciju cijene s Forex ili NYSE tržišta. Isto tako, korisnik ima sposobnost upisati tekstovnu oznaku željene dionice ili valutnog para. Ovisno o odabiru, aplikacija ima dva sljedeća toka rada: predviđanje cijene valutnog para ili predviđanje cijene željene dionice. U slučaju odabira Forex para, prvo se izvršava treniranje modela te nakon toga predviđanje cijene u koje korisnik ima uvid. S druge strane, odabirom dionice prikupljaju se podaci s interneta te isto kao i u Forex djelu projekta, podaci se formatiraju kako bi se LSTM model mogao trenirati i korisnik ima uvid u previđene cijene na grafu. U nastavku rada će svi ovi koraci biti detaljnije opisani.



Slika 3.1 Dijagram tijeka rada

3.1 Implementacija korisničkog sučelja u Python kod

Kako bismo napravili temelj sučelja, moramo pozvati konstruktor knjižnice Tkinter, koju prvo moramo inicijalizirati, te tada možemo stavljati željene elemente unutar prozora. Inicijalizaciju možemo vidjeti na slici 3.2

```
from tkinter import *
from tkinter.ttk import Combobox

window=Tk()
```

Slika 3.2 Korištenje konstruktora za inicijalizaciju prozora

Zatim, potrebno je definirati dimenzije prozora koji će imati interakciju sa korisnikom te pozvati funkciju "mainloop()" koja čeka bilo kakvo korištenje perifernih uređaja računala - bilo to tipkovnica, računalni miš ili nešto drugo[26]. Uz sve to, u ovom projektu se koriste radijski gumbovi, polje za unos teksta te gumba za potvrdu unosa. Svi ovi elementi imaju svoje konstruktore unutar knjižnice Tkinter i svaki element ima svoju skupinu funkcionalnosti koje se nude programeru. Na taj način, elementi postaju zasebni objekti unutar koda koji sadrže razne metode za modificiranje njihove funkcije. U prvom atributu konstruktora definiramo u koji prozor želimo staviti stavku, zatim postoje razne opcije za modifikaciju. Mogu se mijenjati stavke poput: visine, širine, teksta, debljine ruba itd.[26] Nakon što se elementi definiraju, koristi se metoda "place" koja je jedna od metoda za pomicanje elementa korisničkog sučelja na željeno mjesto. Vrijedi spomenuti i objekt IntVar - varijabla koja sadrži broj koja nam definira koji je radio gumb stisnut te nam govori koju ćemo funkciju za predikciju cijene iskoristiti. IntVar varijabla ima tzv. "set" i "get" funkcije s kojima možemo manipulirati s vrijednosti koja je pohranjena, odnosno možemo dobiti vrijednost koju varijabla sadrži i tada izvršiti željeni dio koda. [27] Također se izvršavaju lambda funkcije tijekom pritiska na jednu od radijskih gumbova - to su funkcije koje su jedan od načina pisanja metoda unutar Python programskog jezika. Najčešće sadrži samo jednu liniju koda te se odmah definira funkcija koja se mora pozvati. [28] U ovome projektu, lambda funkcije se koriste za manipuliranje podataka unutar polja za tekst te za poziv funkcije za predviđanje cijene. (slika 3.3)

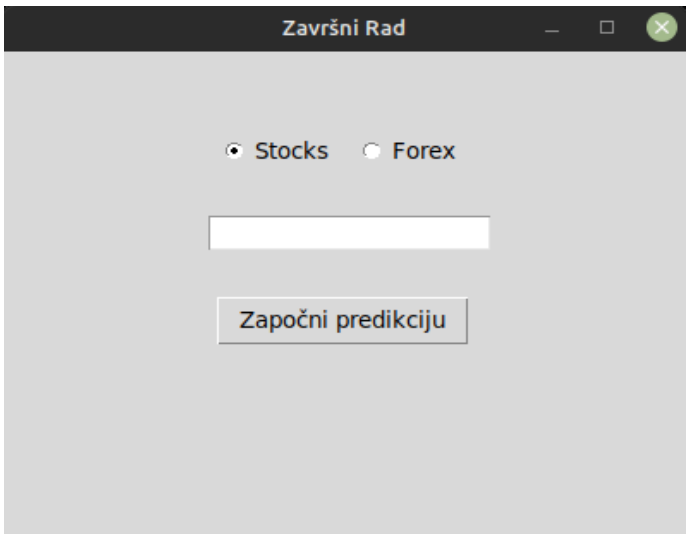
```
v0=IntVar()
v0.set(1)
r1=Radiobutton(window, text="Stocks", variable=v0,value=1, command=lambda : delete())
r2=Radiobutton(window, text="Forex", variable=v0,value=2, command=lambda : e1.insert(0, "EURUSD"))
r1.place(x=120,y=50)
r2.place(x=200, y=50)

e1=Entry(window)
e1.place(x=120, y=100)

button = Button(window, text="Započni predikciju", command=begin())
button.place(x=125, y=150)

window.title('Završni Rad')
window.geometry("400x300+10+10")
window.mainloop()
```

Slika 3.3 Inicijalizacija radio gumbova, polja za tekst i korištenje opcija za uređivanje

Za kraj opisa sučelja, potrebno je definirati što funkcije *delete* i *execute* rade. Funkcija *delete* briše tekst koji se nalazi u polju za unos teksta s obzirom na to da aplikacija može predviđati cijenu na Forex tržištu samo za par EURUSD, dok se za tržište dionica može koristiti bilo koja korporacija, odnosno njezin simbol na tržištu koji je dostupn na "New York Stock Exchange" tržištu. Funkcija *execute* sprema stanje radio gumba u varijablu x i zatim koristimo *if* izraz kako bismo ispitali stanje elemenata te postupili po željama korisnika.

```
from Forex.RunForex import RunForex
from Stocks.RunStocks import RunStocks

def delete():
    e1.delete(0, END)

def begin():
    x = v0.get()
    stock = e1.get()

    if(x == 1):
        RunForex.train()
        RunForex.predict()
    elif(x == 2):
        RunStocks.predict(dionica)
```

Slika 3.4 Provjera stanja elemenata u korisničkom sučelju te inicijalizacija klasa RunForex i RunStocks



Slika 3.5 Korisničko sučelje

Poglavlje 4

Predviđanje cijene na Forex tržištu

4.1 Implementacija MLP algoritma u Python kod

U ovome dijelu projekta koristi ćemo Python knjižnice poput:

- Matplotlib,
- Torch,
- NumPy,
- Pandas.

MLP model definiramo u datoteci "define-network.py" gdje stvaramo klasu "MLP" te ju proširujemo s "nn.Module". "nn.Module" je jedna od vitalnih dijelova Torch knjižnice, taj modul se koristi kao temelj za sve umjetne neuronske mreže koje se koriste. Tako možemo definirati broj ulaza, koliko čvorova čini neki skriveni sloj, koliko slojeva želimo imati te na kraju koliko izlaza algoritam mora imati.
U ovoj aplikaciji, broj ulaza je 10, imamo 2 skrivena sloja - prvi skriveni sloj ima 32 neurona, drugi ima 16 neurona te na kraju, samo 1 čvor predstavlja rezultat.

MLP klasa se koristi u datotekama za treniranje (train-net.py) i predviđanje cijene (test-net.py) te kako bi inicijalizirali umjetnu neuronsku mrežu za oba slučaja, koristimo funkciju "__init__" koja je zapravo konstruktor svih napravljenih klasa u Python kodu. Sa "self" prefiksom definiramo varijable koje su dostupne u svim

funkcijama unutar iste klase. U ovom slučaju, kao što je i prije spomenuto, definiramo broj ulaza, skrivenih slojeva te broj izlaza i uz to koristimo "nn.Linear" funkciju koja je dio Torch knjižnice - na taj način pridodajemo matematičke jednadžbe našim skrivenim slojevima.[29]

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.autograd import Variable

class MLP(nn.Module):
    def __init__(self, input_size=10, layer1_size=32, layer2_size=16, output_size=1):
        super(MLP, self).__init__()

        self.input_size = input_size
        self.layer1_size = layer1_size
        self.layer2_size = layer2_size
        self.output_size = output_size

        self.layer1 = nn.Linear(input_size, layer1_size)
        self.layer2 = nn.Linear(layer1_size, layer2_size)
        self.output = nn.Linear(layer2_size, output_size)
```

Slika 4.1 Inicijaliziranje umjetne neuronske mreže

4.1.1 Treniranje modela

Za početak, MLP model treba istrenirati. Za trening će se koristiti podaci pronađeni na internetu para EURUSD u obliku vrijednosti odvojene zarezom (eng. Comma Seperated Values - nadalje CSV) datoteci koja sadrži sljedeće podatke:

- Datum,
- Cijena tijekom početka dana,
- Cijena tijekom kraja dana,
- Najveća cijena unutar dana,
- Najmanja cijena unutar dana,
- Volumen trgovanja,
- Razlika između volumena i cijene tijekom kraja dana u odnosu na dan prije.

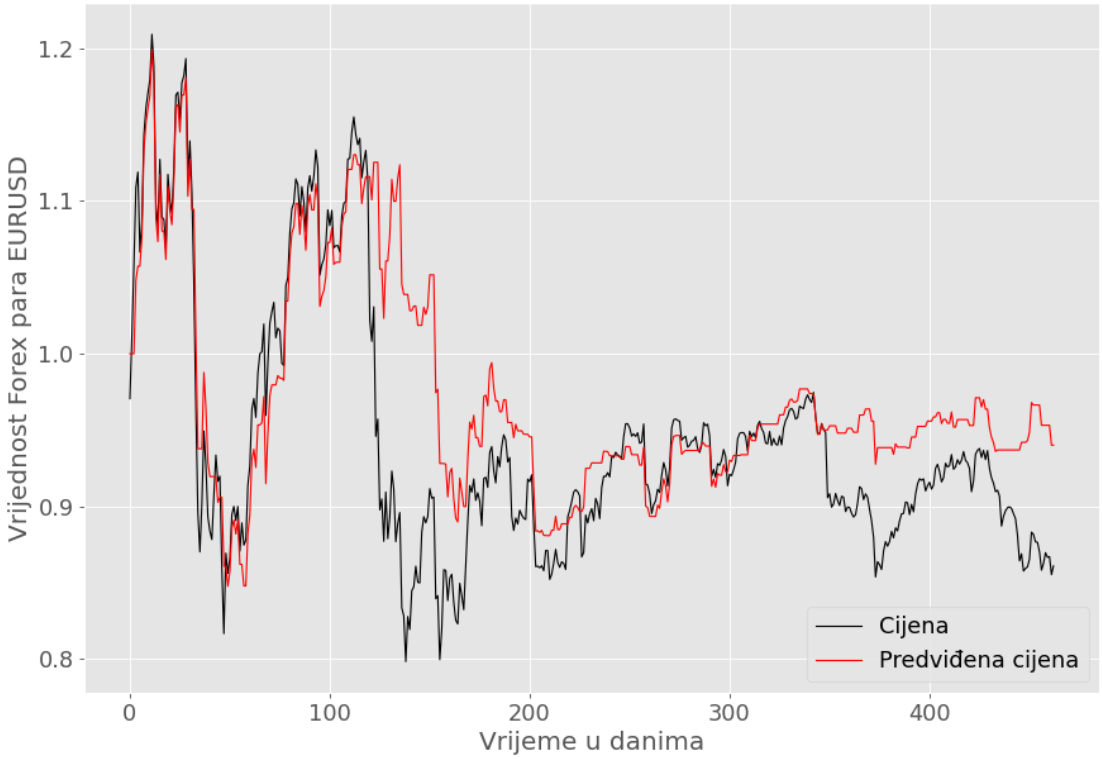
Učitavamo podatke iz CSV datoteke uz pomoć "DataLoader" funkcije koja je dio PyTorch knjižnice. Funkcija kao parametar uzima datoteku u kojoj se nalaze podaci koji će se koristiti za treniranje te kasnije za predikciju cijene. Funkcija "train" prima 2 podatke iz CSV datoteka te na temelju tih informacija šalje podatke kroz skrivene slojeve koji konstantno ažuriraju svoje vrijednosti i prosljeđuju informacije na sljedeće čvorove. Uz to, nakon svake iteracije, u terminalu se ispiše vrijednost dobivene i očekivane greške, što je opisano u poglavlju prije. Rezultati treniranja spremaju se u direktorij "MLPs" te se koriste u svrhu predviđanja cijene.

Vrijedi spomenuti kako je skupina podataka i računalni kod prilagođeni kako tijekom treniranja ne dođe do prekomjernog treniranja (inače poznato kao "overtraining" ili "overfitting"). Prekomjerno treniranje je slučaj kada je algoritam previše naučen na određeni set problema te se ne može prilagoditi ostalim situacijama osim one koju je trenirao. U ovom projektu, korištena je metoda ranog zaustavljanja (eng. early stopping) kako bi spriječili prekomjerno treniranje algoritma.[30]

4.1.2 Predviđanje buduće vrijednosti

Datoteka koja predviđa buduću vrijednost te koja koristi datoteke koje su izrađene tijekom treniranja algoritma je "test-net.py".
Na početku, učitavaju se trenirane vrijednosti iz direktorija "MLPs" te se nakon toga učitavaju vrijednosti cijene para EURUSD i očekuje se predikcija cijene u svakoj točki grafa, što će biti vidljivo na grafu stvarne i predviđene cijene. Kao i tijekom treniranja, podaci koji se trebaju predvidjeti se učitavaju pomoću funkcije "DataLoader". Prava cijena, kao i predviđena cijena, spremaju se u liste koje će se pokazati grafički nakon izvršavanja programa. Crna linija predstavlja pravu cijenu, dok crvena linija predstavlja cijenu koja je predviđena u tom trenutku.

4.1.3 Analiza dobivenoga grafa



Slika 4.2 Graf predviđene i prave cijene

Kao što je i vidljivo na slici, predviđena cijena je na početku približna pravoj cijeni, no kako cijene postaju sve volatilnije, algoritam sve više i više odstupa od prave cijene. Izračunajući korijen srednje kvadratne pogreške dobili smo vrijednost 3.064.

Poglavlje 5

Predviđanje cijene dionica na "New York Stock Exchange" tržištu

5.1 Opis korištenih knjižnica

U ovom djelu projekta, koristit će se knjižnica "yfinance" koja se temelji na stranici "Yahoo Finance" koja je jedna od najboljih stranica za dohvaćanje informacija o cijenama dionica te vijesti vezane za iste. [1]

5.2 Implementacija LSTM algoritma u kod

U ovom djelu projekta, dohvat podataka, pripremanje podataka za treniranje modela, treniranje modela te na kraju predikcija cijene dionica će se izvršavati u jednoj datoteci. Koristimo Python knjižnice numpy, pandas, sklearn, Keras te matplotlib. Prvo se definira dionica čija se cijena želi predvidjeti te se taj podatak sprema u varijablu "ticker". Nakon toga, dohvaćaju se podaci s interneta u obliku CSV datoteke s Yahoo Finance internet portala i zatim se isti pripremaju za treniranje algoritma.

Programi koji služe kao integrirana razvojna okruženja, poput Visual Studio Codea, PyCharma i sl., imaju integrirane čitače vrijednosti odvojene zarezom te auto-

[1] Yahoo Finance web-stranica `finance.yahoo.com`

Date	High	Low	Open	Close	Volume	Adj Close
2020-06-01	80.587502	79.302498	79.437500	80.462502	80791200.0	79.359703
2020-06-02	80.860001	79.732498	80.187500	80.834999	87642800.0	79.727097
2020-06-03	81.550003	80.574997	81.165001	81.279999	104491200.0	80.166000
2020-06-04	81.404999	80.195000	81.097504	80.580002	87560400.0	79.475586
2020-06-05	82.937500	80.807503	80.837502	82.875000	137250400.0	81.739136

Slika 5.1 Primjer dohvaćenih podataka u obliku CSV datoteke

matski prikažu datoteku u obliku tablice (slika 5.1) kako bi osoba koja koristi podatke i manipulira njima, mogla lakše razumjeti kakvi su podaci u pitanju. Na slici možemo redom vidjeti sljedeće podatke: datum, najviša i najniža cijena unutar dana, cijena na otvaranju i zatvaranju tržišta, volumen trgovanja te cijena dionice na kraju dana nakon svih administrativnih poteza koji se mogu desiti tijekom radnog dana.

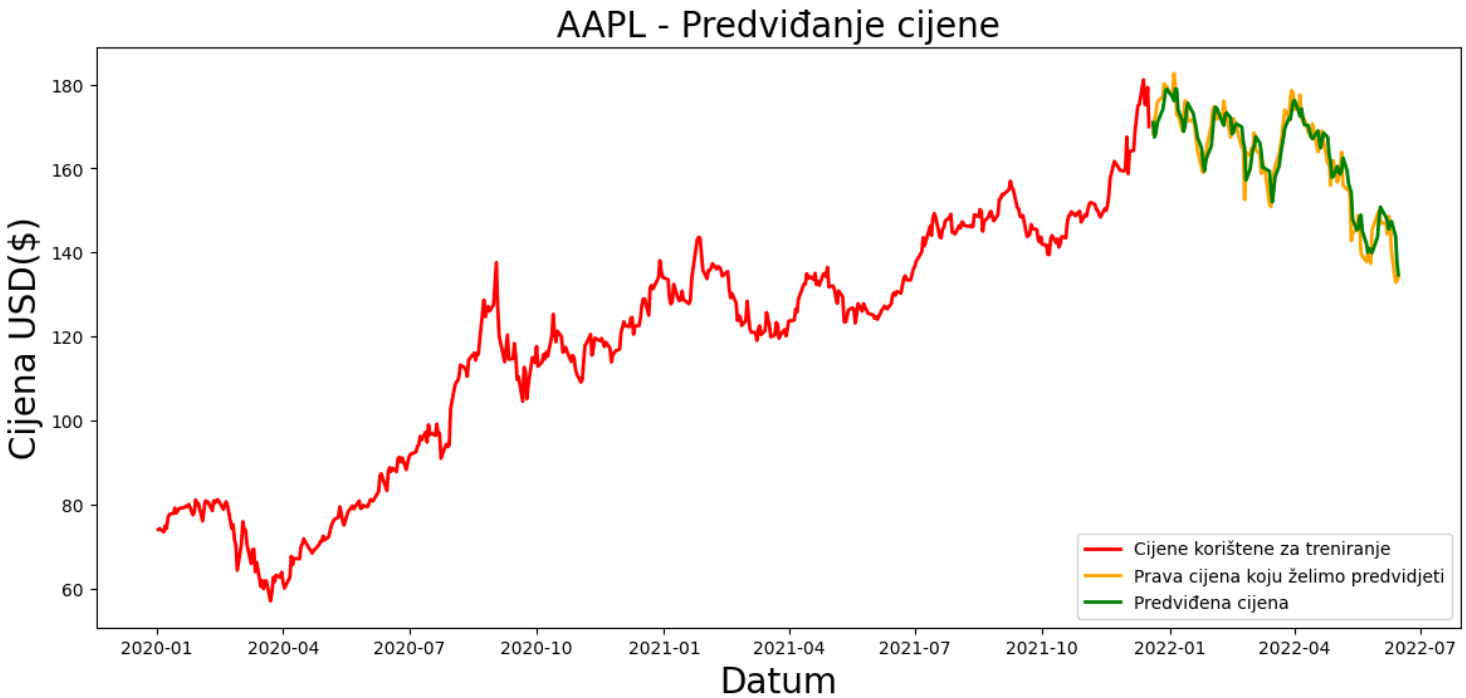
Kako bi LSTM algoritam mogao biti treniran i kasnije mogao raditi predviđanje, potrebno je procesirati podatke na poseban način - s obzirom na to da koristimo LSTM algoritam iz knjižnice za umjetnu inteligenciju Keras, podaci moraju biti u posebnoj listi s 3 dimenzije gdje prva dimenzija opisuje veličinu podataka, druga dimenzija sadrži vrijeme te treća dimenzija sadrži dužinu podatka unutar jedne sekvence ulaza. Nakon što dođe do konverzije u posebne liste, te liste se koriste za treniranje LSTM algoritma. Svaki korak treniranja se opisuje varijablom "epoch". "Epoch" je broj kojim definiramo koliko će puta algoritam iskoristiti postojeće podatke za treniranje. Vrijedi definirati i funkcije gubitka i preciznosti. Funkcija preciznosti nam govori kako naš model napreduje odnosno koliko je predviđena vrijednost blizu vrijednosti koja se koristi tijekom treninga, dok funkcija gubitka uzima u obzir nesigurnost predikcije na temelju razlike između predviđene i prave vrijednosti podatka. [31, 32, 33]

```
#Stvaranje 3D listi za LSTM model
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
x_train.shape

#Izrada LSTM modela
model=Sequential()
#Dodavanje prvog skrivenog sloja
model.add(LSTM(60, return_sequences=True, input_shape=(x_train.shape[1],1)))
#Dodavanje drugog skrivenog sloja
model.add(LSTM(60, return_sequences=False))

#Sastavi model te računaj funkciju gubitka koristeći srednju kvadratnu pogrešku
model.compile(optimizer='adam', loss='mean_squared_error')

#Izvrši treniranje modela
model.fit(x_train, y_train, batch_size=1,epochs=10)
```

Slika 5.2 Stvaranje 3D listi, izrada modela i treniranje u Python kodu

5.3 Dobiveni rezultati

S LSTM algoritmom, pokušali smo predvidjeti cijenu tvrtke Apple, čija je oznaka na dioničkom tržištu AAPL. Na grafu, crvenom bojom su predstavljene cijene s kojima je algoritam treniran, žutom bojom se predstavlja prava cijena dionice koja se pokušava predvidjeti te zelenom bojom se prikazuje predviđena cijena. Isto kao prije, potrebno je izračunati korijen srednje kvadratne pogreške te za ovaj algoritam iznosi 0.200.



Slika 5.3 Cijena dionice tvrke Apple te predikcija njezine cijene

Poglavlje 6

Zaključak

Aplikacije poput ove, iako nisu savršene i ne mogu previdjeti vrijednosti koje će se dogoditi u budućnosti s obzirom na to da je ekonomija jako nepredvidljiva naučna i znanstvena disciplina, mogu dati jako dobar uvid u kretanje određenog tržišta.
Usporedbom korijen srednje kvadratne pogreške dvaju algoritama, *Višeslojni Perceptron* i , *Long Short-Term Memory algoritama*, možemo zaključiti kako je *Long Short-Term Memory* radio manje pogrešaka te je isto tako reagirao bolje na situacije naglog rasta ili pada cijene. U grafu algoritma MLP na slici 4.2 možemo vidjeti kako je predviđanje algoritma relativno blizu (oko 5% odstupanja) , no kako cijena počne naglo oscilirati, algoritam ne nastavlja pratiti pravu cijenu. Postoji par rješenja ovog problema poput povećanja broja neurona u skrivenim slojevima, dodavanje novih skrivenih slojeva, dodavanje novih podataka za treniranje i sl. No iako ovo mogu biti rješenja, ponekad mogu dovesti do suprotnog učinka od očekivanog, a to je pojava pretjeranog treniranja koja je opisana ranije u radu. S druge strane, algoritam LSTM je puno bolje predvidio buduće cijene od algoritma MLP s istim brojem neurona i skrivenih slojeva. Naravno, algoritam nije u potpunosti precizan kao što se može i vidjeti na slici 5.3, ali se može primijetiti kako nagle oscilacije nisu značajno utjecale na predviđenu cijenu u usporedbi s rezultatom MLP modela. Prijedlozi za unapređenje modela su isti kao i za MLP model, no isto tako, treba na umu imati činjenicu kako model može postati pretjerano treniran.

Bibliografija

[1] A. B. i A. Radman Peša, "Bihevioralne financije i teorija „Crnog labuda."

[2] B. Huang, Y. Huan, L. D. Xu, L. Zheng, and Z. Zou, "Automated trading systems statistical and machine learning methods and hardware implementation: a survey," *Enterprise Information Systems*, vol. 13, no. 1, pp. 132–144, 2019. , s Interneta, https://doi.org/10.1080/17517575.2018.1493145

[3] D. J. Jordan and J. D. Diltz, "The profitability of day traders," *Financial Analysts Journal*, vol. 59, no. 6, pp. 85–94, 2003. , s Interneta, https://doi.org/10.2469/faj.v59.n6.2578

[4] N. N. Taleb, "The black swan–the impact of the highly improbable," 2010.

[5] Python (programski jezik). , s Interneta, https://hr.wikipedia.org/wiki/Python_(programski_jezik)

[6] Python (programski jezik). , s Interneta, https://bit.ly/3Dkz0Wi

[7] "Keras," https://keras.io/, 2022.

[8] "MatPlotLib," https://matplotlib.org/, 2022.

[9] "Pandas," https://pandas.pydata.org/, 2022.

[10] "NumPy," https://numpy.org/, 2022.

[11] "PyTorch," https://pytorch.org/, 2022.

[12] What is Supervised Learning? , s Interneta, https://neurospace.io/blog/2020/08/what-is-supervised-learning/

[13] L. Tiong, D. Ngo, and Y. Lee, "Forex trading prediction using linear regression line, artificial neural network and dynamic time warping algorithms," 08 2013.

[14] Osvježimo znanje. , s Interneta, http://silverstripe.fkit.hr/kui/assets/Uploads/Osvjezimo-znanje-KUI-2021-09-10-591-593.pdf

[15] Machine Learning Models. , s Interneta, https://www.javatpoint.com/machine-learning-models

[16] Introduction to Machine Learning: Is AutoML Replacing Data Scientists? , s Interneta, https://neurospace.io/blog/2020/08/what-is-supervised-learning/

[17] Umjetna neuronska mreža. , s Interneta, https://hr.wikipedia.org/wiki/Umjetna_neuronska_mreža

[18] A. M. Lesk, "Extraction of well-fitting substructures: root-mean-square deviation and the difference distance matrix," *Folding and Design*, vol. 2, pp. S12–S14, 1997. , s Interneta, https://www.sciencedirect.com/science/article/pii/S1359027897000576

[19] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627–2636, 1998. , s Interneta, https://www.sciencedirect.com/science/article/pii/S1352231097004470

[20] Multilayer perceptron. , s Interneta, https://en.wikipedia.org/wiki/Multilayer_perceptron

[21] "Long short-term memory." , s Interneta, https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg

[22] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 338–342, 01 2014.

[23] R. Dolphin. LSTM Networks | A Detailed Explanation. , s Interneta, https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

[24] H. Robbani, T. Annisa, and M. Ma'arof, "Variations in the number of layers and the number of neurons in artificial neural networks: Case study of pattern recognition," *Journal of Physics: Conference Series*, vol. 1413, p. 012016, 11 2019.

[25] Z. Maričić. Forex- karakteristike, trgovanje, tehnička i fundamentalna analiza.

[26] J. W. Shipman, "Tkinter 8.5 reference: a gui for python," *New Mexico Tech Computer Center*, vol. 54, 2013.

[27] Tkinter IntVar – Tkinter Tutorial with Examples. , s Interneta, https://www.askpython.com/python-modules/tkinter/tkinter-intvar

[28] Python Lambda. , s Interneta, https://www.w3schools.com/python/python_lambda.asp

[29] PyTorch Documentation Linear. , s Interneta, https://pytorch.org/docs/stable/generated/torch.nn.Linear.html?highlight=nn%20linear#torch.nn.Linear

[30] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, p. 022022, feb 2019. , s Interneta, https://doi.org/10.1088/1742-6596/1168/2/022022

[31] LSTM layer. , s Interneta, https://keras.io/api/layers/recurrent_layers/lstm/

[32] Numpy Reshape. , s Interneta, https://numpy.org/doc/stable/reference/generated/numpy.reshape.html

[33] "Accuracy and Loss," https://machine-learning.paperspace.com/wiki/accuracy-and-loss, 2022.

Sažetak

Motivacija iza ovog projekta jest pitanje može li osobi koja ulazi u vode trgovanja pomoći umjetna inteligencija s modelima koje nudi. Za ovaj projekt, odabran je programski jezik Python kao jedan od najsuperiornijih programskih jezika za umjetnu inteligenciju i strojno učenje. Zadatak rada je usporediti dva algoritma strojnog učenja, *Long Short-Term Memory* i *Višeslojni Perceptron*, i zaključiti koji je bolji za predviđanje buduće cijene valutnog para odnosno dionice na američkom tržištu. Koristeći podatke s Foreign Exchange i New York Stock exchange tržišta, na temelju povijesnih podataka o cijeni dionica, algoritmi imaju zadaću prepoznati trend u kojoj se neki tržišni par nalazi te predvidjeti buduće cijene. Na temelju usporedbe korijena srednje kvadratne pogreške dvaju algoritama, možemo zaključiti kako je *Long Short-Term Memory* algoritam puno bolje predvidio buduće cijene na američkom tržištu.

Ključne riječi **— Python, Umjetna inteligencija, Strojno učenje, Long Short-Term Memory Algorithm, Multi-Layer Perceptron Algorithm, Foreign Exchange i New York Stock Exchange tržišta**

Abstract

Motivation behind this project is the question if a model in the Artificial Intelligence space can help people, who want to start their trading carrers, make better decisions. For this project, programming language Python was used as one of the most superior programming languages for Artificial Intelligence and Machine Learning.

The task of the paper is to compare two machine learning algorithms, *Long Short-Term Memory* and *Multilayer Perceptron*, and to conclude which one is better for predicting future prices of currency pairs or shares on the American market. Using data from the Foreign Exchange and New York Stock Exchange markets, based on historical stock price data, the algorithms have the task of recognizing the trend in which a market pair is located and predicting future prices. Based on the comparison of the root mean square error of the two algorithms, we can conclude that the *Long Short-Term Memory* algorithm predicted future prices of the American financial assets much better.

Keywords — **Python, Artificial Intelligence, Machine Learning, Long Short-Term Memory Algorithm, Multi-Layer Perceptron Algorithm, Foreign Exchange i New York Stock Exchange markets**