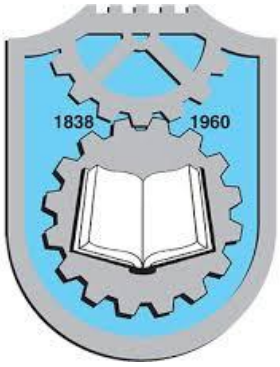
Univerzitet u Kragujevcu

Fakultet inženjerskih nauka



Analiza i projektovanje algoritama

Primena LSTM neuronske mreže na primeru prediktora cena na berzama

Student:

Đorđe Ilić 471/2021

Predmetni nastavnik:

Vladimir Milovanović

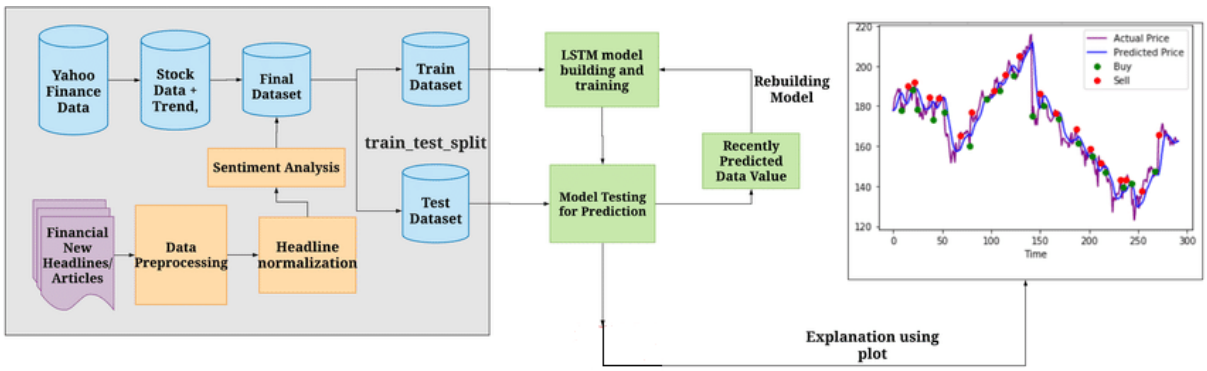
Sadržaj

1.0 Uvod....3
2.0 Skup podataka....4
2.1 Opis podataka....4
2.2 Implementacija i rad sa podacima....5
3.0 LSTM model....8
3.1 Struktura LSTM mreže....8
3.2 Izgradnja modela za predviđanje akcija na tržištu....10
3.3 Kompajliranje modela....10
3.4 Fitovanje modela....11
4.0 Testiranje modela....12
4.1 Testiranje modela nad testnim podacima....12
4.2 Izvršavanje predviđanja nad modelom....12
4.3 Izračunavanje greške....13
4.4 Grafički prikaz....13
4.5 Predviđanje cene na zatvaranju za određeni dan....15
5.0 Zaključak....17
6.0 Literatura....18

1.0 Uvod

Predviđanje akcija na berzi ima za cilj da odredi buduće kretanje vrednosti akcija finansijske berze. Tačno predviđanje kretanja cena akcija će dovesti do većeg profita koji investitori mogu da ostvare. Predviđanje kako će se berza kretati jedno je od najizazovnijih pitanja zbog mnogih faktora koji su uključeni u predviđanje akcija, kao što su kamatne stope, politika i ekonomski rast koji čine berzu nestabilnom i veoma teškom za precizno predviđanje. Predviđanje akcija nudi ogromne šanse za profit i predstavlja glavnu motivaciju za istraživanje u ovoj oblasti; poznavanje kretanja akcija za delić sekunde može dovesti do visokog profita. Pošto je ulaganje u akcije glavna aktivnost na finansijskom tržištu, nedostatak tačnog znanja i detaljnih informacija bi doveo do neizbežnog gubitka ulaganja. Predviđanje tržišta akcija je težak zadatak jer su kretanja na tržištu uvek podložna neizvesnostima. Metode predviđanja tržišta akcija su podeljene u dve glavne kategorije: tehnička i fundamentalna analiza. Tehnička analiza se fokusira na analizu istorijskih cena akcija da bi se predvidele buduće vrednosti akcija (tj. fokusira se na smer cena). S druge strane, fundamentalna analiza se uglavnom oslanja na analizu nestrukturiranih tekstualnih informacija kao što su finansijske vesti i izveštaji o zaradi. Mnogi istraživači veruju da pristupi tehničke analize mogu predvideti kretanje na berzi. Generalno, ova istraživanja nisu dala visoke rezultate predviđanja jer u velikoj meri zavise od strukturiranih podataka zanemarujući važan izvor informacija, a to su onlajn finansijske vesti i osećanja društvenih medija. Ovih dana sve više i više kritičnih informacija o berzi postalo je dostupno na vebu. Primeri uključuju BBC, Blumberg i Yahoo Finance. Teško je ručno izvući korisne informacije iz ovih resursa. Međutim, sa uvođenjem mašinskog učenja i njegovih jakih algoritama, najnovija istraživanja tržišta i napredak u predviđanju tržišta akcija su počeli da uključuju takve pristupe u analizu podataka o berzi. Ukratko, mnoge organizacije naširoko koriste algoritme mašinskog učenja u predviđanju tržišta akcija.

U ovom seminarskom radu će se proći kroz implementaciju analize i predviđanja cena akcija popularne svetske korporacije Apple Inc. u Python-u pomoću različitih algoritama mašinskog učenja. Koristićemo mrežu dugotrajne kratkoročne memorije (LSTM) da kreiramo model mašinskog učenja za predviđanje zaključne cene akcija u poslednjih 60 dana. Dugotrajna memorija (LSTM) je arhitektura veštačke rekurentne neuronske mreže (RNN) dubokog učenja. Za razliku od tradicionalnih neuronskih mreža, LSTM ima povratne veze. Može da obrađuje pojedinačne tačke podataka (kao što su slike) kao i čitav niz podataka (kao što su govor ili video).

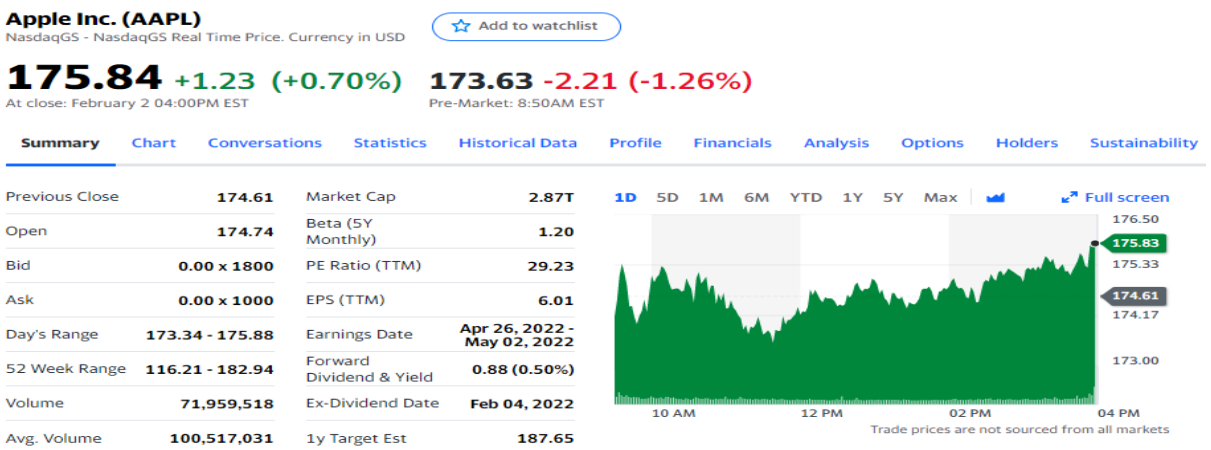


Слика бр. 1 Arhitektura prediktora cene na berzi korišćenjem LSTM neuronske mreže

2.0 Skup podataka

2.1 Opis podataka

U radu će se ispitati cena akcija korporacije Apple Inc. (*APPL*) prema izveštaju sajta Yahoo Finance koji su medijska svojina i predstavljaju deo Yahoo-a [1]. Ovaj sajt pruža finansijske vesti, podatke i komentare uključujući kotacije akcija, saopštenja za javnost, finansijske izveštaje i originalni sadržaj. Podaci o ceni akcija će biti dostavljeni kao frejm podataka (tabela), koja se može otvoriti i analizirati u Excel-u. *APPL*-ove akcije su navedene na Yahoo-u i njihova vrednost se ažurira svakog radnog dana na berzi. Treba napomenuti da tržište ne dozvoljava trgovanje subotom i nedeljom, pa postoji jaz između dva datuma. Početna vrednost akcije („Open“), najviša („High“) i najniža („Low“) vrednost te akcije u istim danima, kao i vrednost na zatvaranju na kraju dana („Close“), sve su naznačene za svaki datum („Date“). Prilagođena vrednost zatvaranja („Adj Close“) odražava vrednost akcija nakon što su dividende proglašene (previše tehnički). Štaviše, obezbeđen je ukupan obim akcija na tržištu. Sa ovim informacijama, posao je stručnjaka za mašinsko učenje/nauka o podacima da pogleda podatke i razvije različite algoritme koji mogu izvući obrasce iz istorijskih podataka Akcije korporacije Apple Inc. U ovom radu će se koristiti *APPL* skup podataka o akcijama iz Yahoo finansija (Finance), podaci koji će se uzeti u obzir o akcijama su u trajanju od 11 godina (početak 2010 i kraj 2021). Informacije u vezi akcija na berzi za ovu kompaniju su predstavljene kao kolone i to su: „High“, „Low“, „High“, „Open“, „Close“, „High“, „Volume“ i „Adj Close“ čije značenje je objasnjeno u gornjem delu teksta. Svaka vrednost ovih kolona je određena po datumu („Date“). Kolone „High“ i „Low“ su korisne, ali definitivno neće moći da pomognu u potpunosti sa predviđanjem cene u narednih nekoliko dana; postoje i drugi faktori koje treba uzeti u obzir. Stvari kao što su datumi („Date“) takođe mogu biti uticajne. Apple često objavljuje proizvode na jesen i ta izdanja proizvoda često snažno utiču na cenu akcija. Kompanije takođe imaju tromesečne izveštaje u kojima se raspravlja o njihovim operacijama koje utiču na cene. Ovi tipovi datumskih faktora mogu biti korisni pri predviđanju cene.



Слика бр. 2 Pregled informacija o akcijama za kompaniju Apple Inc. na sajtu Yahoo Finance

2.2 Implementacija i rad sa podacima

Za izadu prediktora akcija na berzi je korišćeno razvojno okruženje "Python 3.9.0" u kombinaciji sa "Jupyter-om", dok je kod realizovan pomoću *Python* programskog jezika.

Na početku se uvoze potrebne biblioteke koje su potrebne za rad sa podacima, izgradnju modela neuronske mreže i vizuelizaciju podataka. Biblioteke koje su potrebne za rad sa podacima su: *pandas* (služi za učitavanje podataka) i *numpy* (služi za rad sa višedimenzionalnim nizovima)[2].

```
In [137]: #importovanje potrebnih biblioteka
          import math
          import pandas_datareader as web
          import numpy as np
          import pandas as pd
          from sklearn.preprocessing import MinMaxScaler
          from keras.models import Sequential
          from keras.layers import Dense, LSTM , Dropout
          import matplotlib.pyplot as plt
          plt.style.use('fivethirtyeight')
```

Слика бр. 3 Uvoženje potrebnih biblioteka

Potrebni podaci za korporaciju **Apple Inc**. se uzimaju sa sajta *Yahoo Finance* [1] u vremenskom razdoblju od 01.01.2010 do 29.12.2021. Nakon uvoženja podataka se vrši njihovo prikazivanje gde se mogu videti informacije vezane za akcije korporacije (kolone) i vrednosti tih informacija (redovi). Takođe na izlazu se može videti da okvir sa podacima ima ukupno 3019 redova i 6 kolona[2].

```
In [138]: #uzimanje kvota sa berze u određenom vremenskom intervalu
          df= web.DataReader('AAPL',data_source='yahoo',start='2010-01-01',end='2021-12-29')
          #prikaz podataka
          df
```

Out[138]:

Date	High	Low	Open	Close	Volume	Adj Close
2010-01-04	7.660714	7.585000	7.622500	7.643214	493729600.0	6.553026
2010-01-05	7.699643	7.616071	7.664286	7.656429	601904800.0	6.564355
2010-01-06	7.686786	7.526786	7.656429	7.534643	552160000.0	6.459940
2010-01-07	7.571429	7.466071	7.562500	7.520714	477131200.0	6.447998
2010-01-08	7.571429	7.466429	7.510714	7.570714	447610800.0	6.490866
...	...	...	...	...	...	...
2021-12-22	175.860001	172.149994	173.039993	175.639999	92135300.0	175.639999
2021-12-23	176.850006	175.270004	175.850006	176.279999	68356600.0	176.279999
2021-12-27	180.419998	177.070007	177.089996	180.330002	74919600.0	180.330002
2021-12-28	181.330002	178.529999	180.160004	179.289993	79144300.0	179.289993
2021-12-29	180.630005	178.139999	179.330002	179.380005	62348900.0	179.380005

3019 rows × 6 columns

Слика бр. 4 Uzimanje podataka i njihov prikaz

Korišćenjem biblioteke *matplotlib* se mogu vizuelizovati vrednosti cena akcija od početnog do krajnjeg datuma. X osa predstavlja datum po godinama od početne godine (2010) do krajnje godine (2021), dok Y osa predstavlja vrednost cena akcija na zatvaranju u dolarima.

```
In [160]: # prikazivanje cene zatvaranja
          plt.figure(figsize=(16,8))
          plt.title('Istorija cena zatvaranja')
          plt.plot(df['Close'])
          plt.xlabel('Datum',fontsize=18)
          plt.ylabel('Cena zatvaranja USD $',fontsize=18)
          plt.show()
```

Слика бр. 5 Vizuelizacija vrednosti kolone ***Close*** *u vremenskom razdoblju*

Sa slike se može primetiti da cena akcija zatvaranja na berzi raste kako godine odmiču od početnog datuma.

Kolona „Close“ se zatim dodeljuje ciljnoj promenljivoj u sledećem koraku. To je cena akcija Apple Inc. akcija u ovoj situaciji. Dakle, razmotrićemo samo tržišnu cenu na zatvaranju i predvideti cenu na zatvaranju koristeći Python. Nakon zadavanja ciljne promenljive zadaje se kolicina podataka za treniranje, u našem slučaju se uzima 80% od ukupne količine podataka.

```
In [161]: #kreiranje novog frejma podataka sa kolonom 'Close'
          data=df.filter(['Close'])
          #smeštanje vrednosti kolone Close u dataset promenljivu
          dataset=data.values
          #uzimanje 80% podataka za treniranje
          training_data_len=math.ceil(len(dataset)*.8)

          training_data_len

Out[161]: 2416
```

Слика бр. 6 Zadavanje ciljne promenljive i količine podataka za treniranje

Da bismo smanjili troškove izračunavanja podataka u tabeli, skaliraćemo vrednosti podataka na vrednosti između 0 i 1. Jednostavnijim rečima, skaliranje je pretvaranje brojčanih podataka predstavljenih u širokom opsegu u manji. Kao rezultat, svi podaci u velikom broju su smanjeni, a samim tim i potrošnja memorije. Takođe, pošto podaci nisu raspoređeni u ogromnim vrednostima, možemo postići veću preciznost smanjivanjem. Za ovo ćemo koristiti klasu *MinMaxScaler* biblioteke **sci-kit-learn** [2]. Dakle, pristupa se skaliranju svih vrednosti iz kolone „Close“.

```
In [142]: #skaliranje podataka
          scaler= MinMaxScaler(feature_range=(0,1))
          scaled_data= scaler.fit_transform(dataset)

          scaled_data

Out[142]: array([[0.00452113],
                 [0.00459731],
                 [0.00389525],
                 ...,
                 [1.        ],
                 [0.99400471],
                 [0.9945236 ]])
```

Слика бр. 7 Skaliranje svih vrednosti iz kolone „Close“

U nastavku se 80% procenata skaliranih podataka konačno dodaju u promenljivu koja predstavlja podatke za treniranje (ciljna promenljiva) i kreiramo skup podataka za obuku koji sadrže cenu zatvaranja od 60 dana (60 tačaka podataka) kako bismo mogli da uradimo predviđanje za 61. cenu akcija. Sada će skup podataka x_train sadržati ukupno 60 vrednosti, prva kolona će sadržati od indeksa 0 do 59, a druga kolona od indeksa od 1 do 60, itd. Skup podataka y_train će sadržati 61. vrednost u svojoj prvoj koloni koja se nalazi na indeksu 60, a za drugu kolonu će sadržati 62. vrednost koja se nalazi na indeksu 61 i tako dalje.

```
In [145]: #kreiranje seta podataka za treniranje
          train_data=scaled_data[0:training_data_len,:]
          #razdvajanje podataka na x_train i y_train
          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])
              if i<=61:
                  print(x_train)
                  print(y_train)
                  print()
```

Слика бр. 8 Formiranje podataka za obuku

Konvertovanje i nezavisnog i zavisnog skupa podataka za treniranje kao x_train i y_train, respektivno, u NumPy nizove tako da se mogu koristiti za obuku LSTM modela[2]. Takođe, pošto LSTM model očekuje podatke u 3-dimenzionalnom skupu podataka, pomoću funkcije *reshape()* ćemo preoblikovati podatke u 3-dimenzionalni oblik. Kada se radi oblikovanje podataka za treniraje, potrebno je da se navedu sledeći argumenti u oviru f-je *reshape():* broj uzoraka odnosno broj redova koji imamo (2356), broj vremenskih oznaka ili broj kolona (60) i broj karakteristika u našem slučaju samo kolona „Close“ (1).

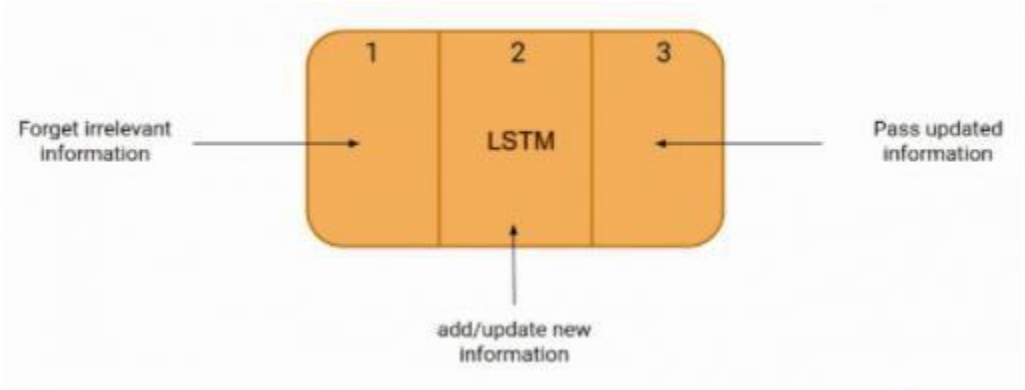
```
In [146]: #konvertovanje x_train i y_train u numpy niz
          x_train, y_train=np.array(x_train),np.array(y_train)

In [147]: #oblikovanje podataka
          x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
          x_train.shape

Out[147]: (2356, 60, 1)
```

Слика бр. 9 Procesiranje podataka

3.0 LSTM model

3.1 Struktura LSTM mreže

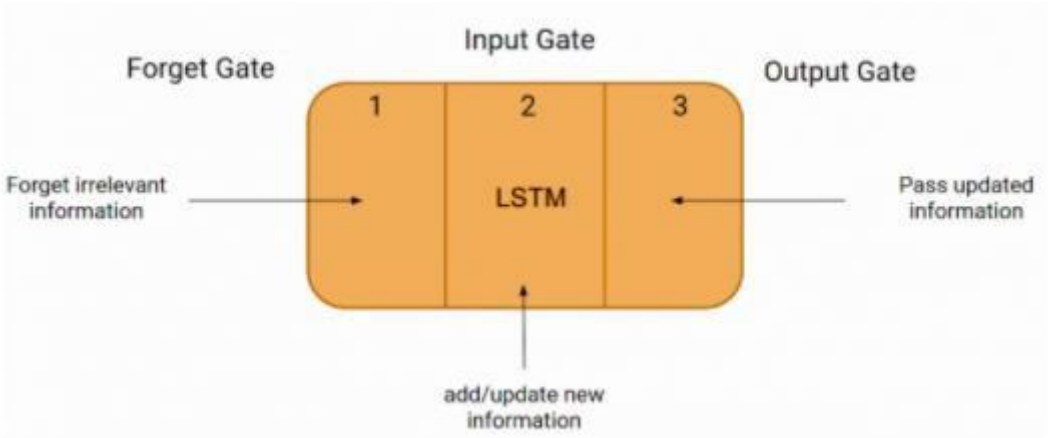
LSTM se sastoji od tri dela kao što je prikazano na donjoj slici i svaki deo obavlja individualnu funkciju[3].



Слика бр. 10 Delovi LSTM

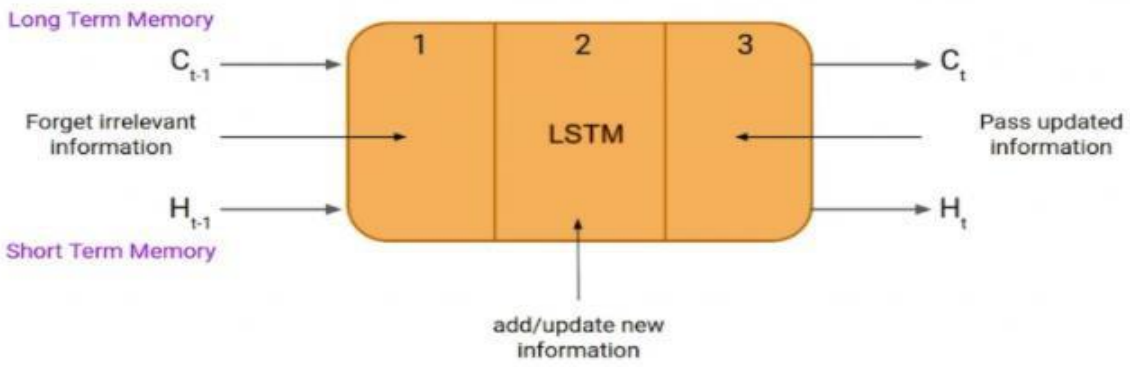
Prvi deo bira da li će informacije koje dolaze iz prethodne vremenske oznake biti zapamćene ili su irelevantne i mogu da se zaborave. U drugom delu ćelija pokušava da nauči nove informacije iz ulaza u ovu ćeliju. Konačno u trećem delu ćelija prenosi ažurirane informacije sa trenutne vremenske oznake na sledeću vremensku oznaku.

Ova tri dela LSTM ćelije su poznata kao kapije. Prvi deo se zove **Forget gate**, drugi deo je poznat kao **ulazna kapija**, a poslednji je **izlazna kapija[3].**



Слика бр. 11 Nazivi ćelija LSTM

Baš kao i jednostavna RNN LSTM takođe ima skriveno stanje gde H(t-1) predstavlja skriveno stanje prethodne vremenske oznake a Ht je skriveno stanje trenutne vremenske oznake. Pored toga LSTM takođe ima stanje ćelije predstavljeno sa (t-1) i C(t) za prethodnu i trenutnu vremensku oznaku[3]. Ovde je skriveno stanje poznato kao kratkoročna memorija a stanje ćelije je poznato kao dugoročna memorija. Zanimljivo je primetiti na sledećoj slici da stanje ćelije nosi informacije zajedno sa svim vremenskim oznakama.



Слика бр. 12 Ćelije sa vremenskim oznakama

3.2 Izgradnja modela za predviđanje akcija na tržištu

Pre nego što budemo mogli da razvijemo LSTM, moramo da napravimo nekoliko uvoza iz **Kerasa**: *Sequential* za inicijalizaciju neuronske mreže, *LSTM* za dodavanje LSTM sloja, *Dropout* za sprečavanje prekomernog prilagođavanja slojevima koji ispadaju, i *Dense* za dodavanje gusto povezanog sloja neuronske mreže.

Neuronske mreže su definisane u **Kerasu** kao sekvenca slojeva[2]. Kontejner za ove slojeve je klasa *Sequential*. Prvi korak je kreiranje instanca klase *Sequential*. Zatim se kreiraju sloejvi i dodaju se po redosledu gde ih treba povezati. LSTM rekurentni sloj koji se sastoji od memorijskih jedinica se zove *LSTM().* Potpuno povezan sloj koji često prati LSTM slojeve i koristi se za izlaz a predviđanje se zove *Dense().* LSTM sloj se dodaje sa sledećim argumentima: 50 units je dimenzionalnost izlaznog prostora, *return_sequences=True* je neophodan za slaganje LSTM slojeva tako da konsekvencionalni LSTM sloj ima ulaz trodimenzionalne sekvence, a input_shape je oblik skupa podataka za treniranje (broj vremenskih koraka i broj karakteristika). Prateći slojeve **LSTM** (ulazni LSTM sloj i drugi LSTM sloj sa 50 memorijskih ćelija) , dodajemo dva **Dense** sloja koji specificira izlaz jedne jedinice. Jedan **Dense** sloj ima 25 neurona, dok drugi ima 1 neuron.

```
In [148]:
#LSTM model
model= Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

Слика бр. 13 LSTM model

3.3 Kompajliranje modela

Kada se mreža definiše potrebna je njena kompajlacija. Kompajlacija predstavlja jedan efikasan korak. Transformiše jednostavan niz slojeva koji je definisan u visokoefikasnu seriju matrice. Transformiše u format namenjen za izvršavanje na GPU-u ili CPU-u, u zavisnosti od toga kako je **Keras** konfigurisan[2]. Zamislite kompilaciju kao korak pre računanja vaše mreže. To je uvek potrebno nakon definisanja modela. Za kompilaciju je potrebno specificirati niz parametara, posebno prilagođenih za obuku mreže. Konkretno, algoritam optimizacije koji se koristi za obuku mreže i gubitak(loss) funkcija koja se koristi za procenu mreže koja je minimizirana algoritmom optimizacije. Za kompajliranje našeg modela koristimo *Adam* optimizator i postavljamo loss kao *mean_skuared_error*.

```
In [149]:  #kompajliranje modela
           model.compile(optimizer='adam', loss='mean_squared_error')
```

Слика бр. 14 Kompajliranje modela

3.4 Fitovanje modela

Kada se mreža sastavi, može da se uklopi, što znači prilagođavanje težinskih koeficijenata na trening skup podacima. Podešavanje mreže zahteva da se specificiraju podaci o obuci, oba matrica ulaznh obrazaca, x, i niz odgovarajućih izlaznih obrazaca, y. Mreža se obučava korišćenjem algoritma propagacije greške unazad kroz vreme i optimizovana je u skladu sa optimizacionim algoritmom i funkcijom gubitka koja je navedena pri kompajliranju modela. Algoritam propagacije unazad zahteva da mreža bude obučena za određeni broj epoha ili izloženosti svim sekvencama u skupu podataka za obuku. Svaka epoha se može podeliti u grupe ulazno-izlaznih parova obrazaca koji se nazivaju serije[2]. Ovo definiše broj šablona kojima je mreža izložena pre nego što se težine ažuriraju u okviru jedne epohe. To je takođe an optimizacija efikasnosti, obezbeđujući da se ne učitava previše ulaznih obrazaca u memoriju na vreme. Nakon toga, prilagođavamo model da radi za 10 epoha (epohe su koliko puta će algoritam učenja raditi kroz ceo skup za obuku) sa veličinom serije od 1 (*batch_size*=1: Težinski koeficijenti se ažuriraju nakon svakog uzorka i postupak se naziva stohastički gradijentni spust).

```
In [12]: #treniranje modela
         model.fit(x_train,y_train,batch_size=1,epochs=10)

         Epoch 1/10
         2356/2356 [==============================] - 84s 26ms/step - loss: 2.4160e-04
         Epoch 2/10
         2356/2356 [==============================] - 55s 24ms/step - loss: 7.4152e-05
         Epoch 3/10
         2356/2356 [==============================] - 56s 24ms/step - loss: 5.6812e-05
         Epoch 4/10
         2356/2356 [==============================] - 58s 25ms/step - loss: 5.2497e-05
         Epoch 5/10
         2356/2356 [==============================] - 67s 29ms/step - loss: 3.8191e-05
         Epoch 6/10
         2356/2356 [==============================] - 59s 25ms/step - loss: 2.8005e-05
         Epoch 7/10
         2356/2356 [==============================] - 57s 24ms/step - loss: 2.9805e-05
         Epoch 8/10
         2356/2356 [==============================] - 55s 23ms/step - loss: 2.5337e-05
         Epoch 9/10
         2356/2356 [==============================] - 55s 23ms/step - loss: 2.2214e-05
         Epoch 10/10
         2356/2356 [==============================] - 55s 23ms/step - loss: 2.4824e-05

Out[12]: <keras.callbacks.History at 0x207b3db7f10>
```

Слика бр. 15 Treniranje modela

4.0 Testiranje modela

4.1 Testiranje modela nad testnim podacima

Kod ispod će dobiti sve redove iz *training_data_len* za kolone cene na zatvaranju („Close“). Zatim konvertujte skup podataka *x_test* u NumPy nizove tako da se mogu koristiti za obuku LSTM modela. Kako LSTM model očekuje podatke u 3-dimenzionalnom skupu podataka, pomoću funkcije *reshape()* ćemo preoblikovati skup podataka u obliku 3-dimenzionalnog[2].

```
In [151]:
#kreiranje seta podataka za testiranje
#kreiranje novog niza koji sadrzi skaliranje vrednosti od indeksa 2356  do  2416
test_data=scaled_data[training_data_len-60: , :]
#kreiranje seta podataka x_test i y_test
x_test=[]
y_test=dataset[training_data_len:, :]

for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])
```

```
In [152]:
#konvertovanje podataka u numpy niz
x_test=np.array(x_test)
```

```
In [153]:
#oblikovanje podataka
x_test=np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

Слика бр. 16 Kreiranje testnih podataka za procenu

4.2 Izvršavanje predviđanja nad modelom

Kada performance modela ispune očekivanje, može se koristiti za predviđanje na novim podacima. Ovo je lako kao pozivanje funkcije *predict()* [2] na modelu sa nizom novih obrazaca unosa. Predviđanja će biti vraćena u formatu koji obezbeđuje izlazni sloj mreže. U slučaju problema regresije što je naš slučaj, ova predviđanja mogu biti u formatu problema direktno, obezbeđeno linearnom aktivacionom funkcijom.

```
In [154]:
#dobijanje predvidjene vrednosti cena modela
predictions=model.predict(x_test)

predictions=scaler.inverse_transform(predictions)
```

Слика бр. 17 Predviđanje modela

Koristeći funkciju *predict()*, dobijaju se predviđene vrednosti iz modela koristeći podatke testa. A funkcija *scaler.inverse_transform()* poništava skaliranje, odnosno pruža nam mogućnost da vidimo stvarne cene koje su predviđenje.

4.3 Izračunavanje greške

Srednja kvadratna greška je jedna metrika greške za procenu tačnosti i stope greške bilo kog algoritma mašinskog učenja za problem regresije. Dakle, **MSE** je funkcija rizika koja nam pomaže da odredimo prosečnu kvadratnu razliku između predviđene i stvarne vrednosti karakteristike ili promenljive[4]. RMSE je akronim za Root Mean Skuare Error, što je kvadratni koren vrednosti dobijen iz funkcije Mean Skuare Error. Koristeći **RMSE**, možemo lako nacrtati razliku između procenjenih i stvarnih vrednosti parametra modela[4]. Po ovome možemo jasno suditi o efikasnosti modela. Obično se **RMSE** rezultat manji od 180 smatra dobrim rezultatom za umereno ili dobro funkcionalan algoritam. U slučaju da vrednost RMSE prelazi 180, potrebno je da izvršimo izbor karakteristika i hiper podešavanje parametara na parametrima modela. Što je niža vrednost, to su bolje performanse modela.

```
In [17]: #RMSE
         rmse=np.sqrt(np.mean(predictions-y_test)**2)
         rmse

Out[17]: 7.192438419778549
```

Слика бр. 18 Izračunavanje RMSE

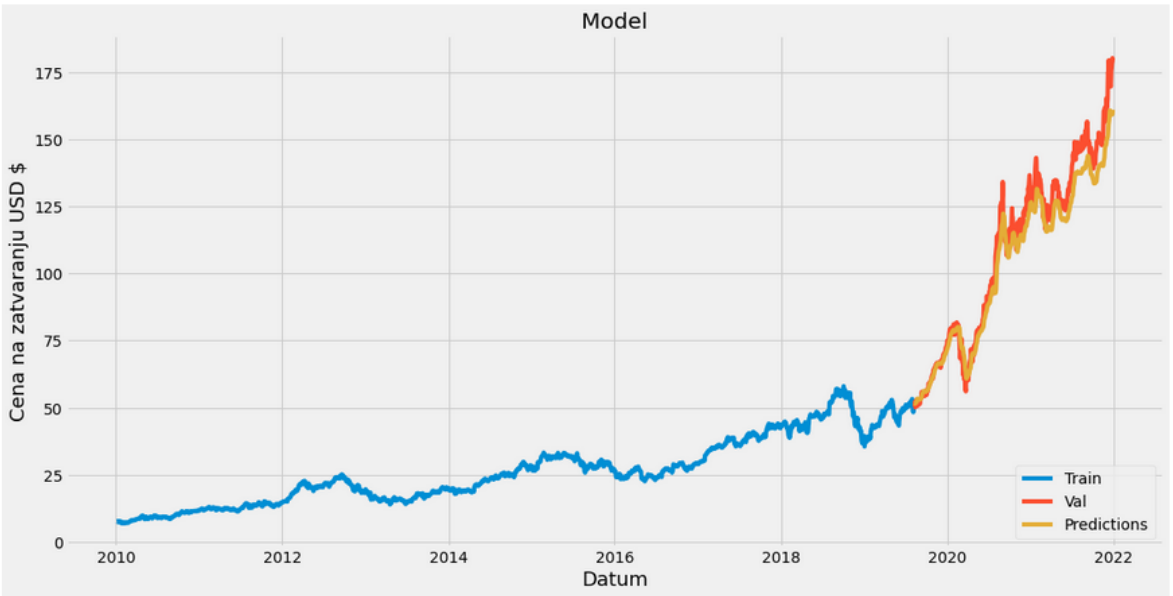
RMSE vrednost koju smo dobili je 7.192438419778549 što je dovoljno pristojno.

4.4 Grafički prikaz

Istorija obuke LSTM modela može se koristiti za dijagnozu ponašanja modela. Mogu da se nacrtaju performanse modela koristeći **Matplotlib** biblioteku. Kreiranje i pregled dijagrama može pomoći kao prelged o mogućim novim konfiguracijama kao pokušaj da se dobiju bolje performanse svog modela. Može se razmatrati veština modela na setu za treniranje i skupove validacije u smislu gubitka koji je minimiziran. Može se koristiti bilo koja metrika koja je značajna za prisutni problem. Nakon svih ovih koraka, možemo koristiti **matplotlib** da vizualizujemo rezultat naše predviđene cene akcija i stvarne cene akcija.

```
In [156]:
#zadavanje vrednosti promenljivama za treniranje i validaciju
train= data[:training_data_len]
valid=data[training_data_len:]
valid['Predictions']=predictions
#vizuelizacija
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Datum',fontsize=18)
plt.ylabel('Cena na zatvaranju USD $', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','Val','Predictions'],loc='lower right')
plt.show()
```

Слика бр. 19 Postavljanje grafika



Слика бр. 20 Prikaz grafika

Za prikaz validacionog skupa je uzeto poslednjih 60 vrednosti koje se porede sa predviđenim vrednostima. Iako tačne cene iz naše predviđene cene nisu uvek bile bliske stvarnoj ceni, naš model je i dalje ukazivao na opšte trendove kao što su porast ili pad.

```
In [19]: #prikaz predvidjene i prave cene
         valid
```

Out[19]:

Date	Close	Predictions
2019-08-09	50.247501	50.053787
2019-08-12	50.119999	49.943600
2019-08-13	52.242500	49.872730
2019-08-14	50.687500	50.076374
2019-08-15	50.435001	50.214577
...	...	...
2021-12-22	175.639999	156.511459
2021-12-23	176.279999	156.492126
2021-12-27	180.330002	156.816574
2021-12-28	179.289993	157.752441
2021-12-29	179.380005	158.708786

603 rows × 2 columns

Слика бр. 21 Prikaz poređenja prave i predviđene vrednosti

4.5 Predviđanje cene na zatvaranju za određeni dan

U nastavku ćemo se baviti previđanjem cene zatvaranja za određeni datum, u našem slučaju: 30.12.2021. Prvo se uzimaju vrednosti kvota iz Apple Inc. tabele podataka akcija za berzu u razdoblju od 01.10.2010 do 29.12.2021. Iz te tabele se uzima samo kolona „Close“ jer želimo da predvidimo cenu na zatvarnju. Nakon uzimanja pomenute kolone selektuju se poslednjih 60 vrednosti i te vrednosti se skaliraju u domenu od 0 do 1. Kada su podaci spremni za korišćenje koji su pre toga prebačeni u 3D oblik jer LSTM model zahteva takav unos podataka, može se predvideti cena za datum 30.12.2021. Kod za pomenuti deo rada je prikazan u nastavku:

```
In [20]: #vrednost
apple_quoute=web.DataReader('AAPl',data_source='yahoo',start='2010-01-01',end='2021-12-29')
#novi okvir podataka
new_df=apple_quoute.filter(['Close'])
#uzimanje cene na zatvaranju prethodnih 60 dana i konvertovanje u niz
last_60_days=new_df[-60:].values
#skaliranje podataka izmedju 0 i 1
last_60_days_scaled=scaler.transform(last_60_days)
#kreiranje prazne liste
X_test=[]
#ubacivanje dana u listu
X_test.append(last_60_days_scaled)
#konvertovanje X_test seta podataka u numpy niz
X_test=np.array(X_test)
#oblikovanje u 3d
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
#skalirana cena koja je predvidjena
pred_price=model.predict(X_test)
#reiskaliranje
pred_price=scaler.inverse_transform(pred_price)
print(pred_price)

[[159.55528]]
```

Слика бр. 22 Предвиђање цене акције за одређени датум

Nakon dobijanja predviđene vrednosti cene akcije, možemo prikazati stvarnu cenu za datum 30.12.2021 prikazivanjem iz tabele podataka Apple Inc. akcije na berzi.

```
In [159]: apple_quoute2=web.DataReader('AAPL',data_source='yahoo',start='2021-12-30',end='2021-12-30')
print(apple_quoute2['Close'])

Date
2021-12-30    178.199997
Name: Close, dtype: float64
```

Слика бр. 21 Приказивање стварне цене акције за одређени датум

Previđena vrednost cene akcije je 159.55528 dok je stvarna cena akcije 178.199997. Razlika između ove dve vrednosti je oko 18 dolara što je prilično dobar rezultat.

5.0 Zaključak

Sa uvođenjem mašinskog učenja i njegovih jakih algoritama, najnovija istraživanja tržišta i napredak u predviđanju tržišta akcija su počeli da uključuju takve pristupe u analizu podataka o berzi. Početna vrednost akcije, najviša i najniža vrednost te akcije u istim danima, kao i vrednost na zatvaranju na kraju dana, sve su naznačene za svaki datum. Štaviše, dat je ukupan obim akcija na tržištu. Uz ove informacije, posao je naučnika za mašinsko učenje podataka da pogleda podatke i razvije različite algoritme koji mogu pomoći u pronalaženju odgovarajućih vrednosti akcija.

Predviđanje tržišta akcija je bila dugotrajna i naporna procedura pre nekoliko godina ili čak deceniju. Međutim, primenom mašinskog učenja za prognoze berze, procedura je postala mnogo jednostavnija. Mašinsko učenje ne samo da štedi vreme i resurse, već i nadmašuje ljude u pogledu performansi. Uvek će radije koristiti obučeni kompjuterski algoritam jer će savetovati samo na osnovu činjenica, brojeva i podataka i neće uzimati u obzir emocije ili predrasude.

6.0 Literatura

[1] NasdaqGS - NasdaqGS Real Time Price. Currency in USD, Apple Inc. (AAPL)

https://finance.yahoo.com/quote/AAPL/

(01.02.2022.)

[2] Your First Deep Learning Project in Python with Keras Step-By-Step, Jason Brownlee

https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/ (03.02.2022.)

[3] An High Level Introduction to Long Short Term Memory Neural Networks, Aleia Knight

https://towardsdatascience.com/lstm-neural-network-the-basic-concept-a9ba225616f7 (03.02.2021.)

[4] What does RMSE really mean? , James Moody

https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e (05.02.2022.)

[5] Stock Market Predictions with LSTM in Python

https://www.datacamp.com/community/tutorials/lstm-python-stock-market (06.02.2022.)