

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I  
INFORMACIJSKIH TEHNOLOGIJA**

**Raspoznavanje uzoraka i strojno učenje**

**Predviđanje dionica kriptovaluta uz LSTM neuronsku mrežu**

**Ivan Gudelj**

**Diplomski studij računarstva, DRB**

**Osijek, 2022.**

<b>UVOD</b>	<b>3</b>
<b>PREGLED PROBLEMATIKE</b>	<b>4</b>
<b>OPIS ZADATKA</b>	<b>9</b>
<b>PRIKAZ REZULTATA I OPTIMIZACIJA MODELA</b>	<b>13</b>
Utjecaj promjene batch_size-a na točnost modela	13
Utjecaj promjene broja epoha na točnost modela	16
Utjecaj promjene broja prethodnih dana na točnost modela	18
<b>ZAKLJUČAK</b>	<b>20</b>
<b>LITERATURA</b>	<b>21</b>

# 1. UVOD

Projektni zadatak pod temom “Predviđanje dionica kriptovaluta uz LSTM neuronsku mrežu” obuhvaća predobradu podataka o dionicama koje su javno dostupne [1], te “predviđanje” budućeg stanja dionica koristeći model treniran na određenom skupu podataka koji korisnik unese, što ujedno znači da optimalnost predikcije ovisi također o unešenim parametrima od strane korisnika.

Za izradu programskog rješenja korišten je programski jezik *Python* u razvojnom okruženju *Anacoda*. Sam cilj projektnog zadatka je istrenirati model nad dostupnim podacima te izvršiti njegovo testiranje sa preostalim podacima do trenutnog trenutka.

Napisana programska podrška se može primjeniti za sve dionice, čije se ‘*Closing*’ vrijednosti (vrijednost dionice pri kojoj je izvršena zadnja sigurnosna transakcija tijekom dana) zabilježavaju [1]. Iz [1] je moguće dobiti informacije vezane za dionice kao što su:

*Prethodno zatvaranje (Closing price)*

*Dnevni raspon*

*Početni datum izdavanja dionice*

*Tržišna kapitalizacija*

Za izradu modela korištena je *Keras* biblioteka neuronskih mreža otvorenog koda napisana u Python-u. *Keras* biblioteka je izvođena nad *TensorFlow* platformi za strojno učenje. *Keras* je dizajniran kako bi omogućio jednostavno eksperimentiranje s dubokim neuronskim mrežama; jednostavan je za korištenje, modularan i proširiv. Glavne strukture podataka u *Keras* biblioteci su *Slojevi* i *Modeli*.

U jednom od poglavlja je pokriveno testiranje odnosno provjera uspješnosti ovog projektnog rada unutar lošijih uvjeta. Kvaliteta i količina podataka za testiranje ovisi o tome koliki vremenski raspon podataka za učenje korisnik odabere.

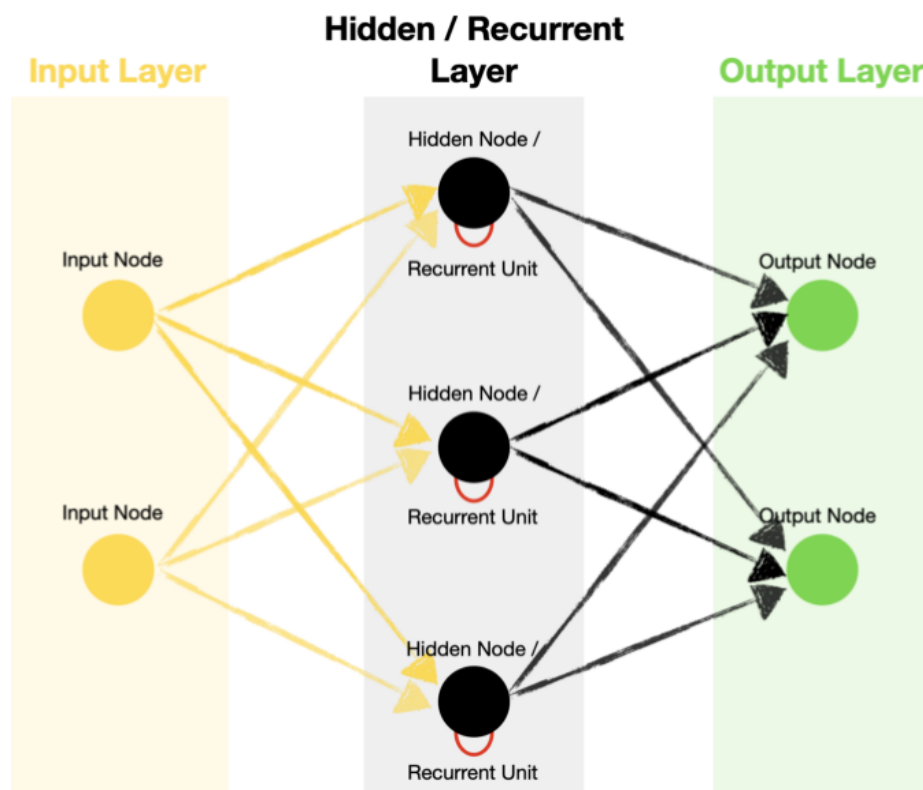
## 2. PREGLED PROBLEMATIKE

Ukoliko ste kupac dionica potrebno je modelirati stanje dionica kako bi odlučili kada kupiti dionice a kada ih prodati kako bi ostvarili profit. Potrebno je poznavanje modela strojnog učenja kako bismo što bolje obradili povijesne podatke (podatke za treniranje) te kako bismo što točnije predvidjeli moguće nadolazeće elemente odnosno nadolazeću vrijednost dionice.

Bitno je naglasiti da su tržišne cijene dionica veoma nestabilne. To znači da ne postoje predefrirani uzorci u podacima koji bi dozvolili modeliranje sustava za nadzor stanja dionica gotovo pa savršeno. Burton [2] spominje kako bi, u slučaju da je tržište zaista efikasno i da se svi faktori, čim postanu javni, odražavaju na stanje dionice, no to nije današnji slučaj zbog velikog utjecaja vanjskih faktora kao što su utjecaji javnosti i poznatih osoba, rat i sl.

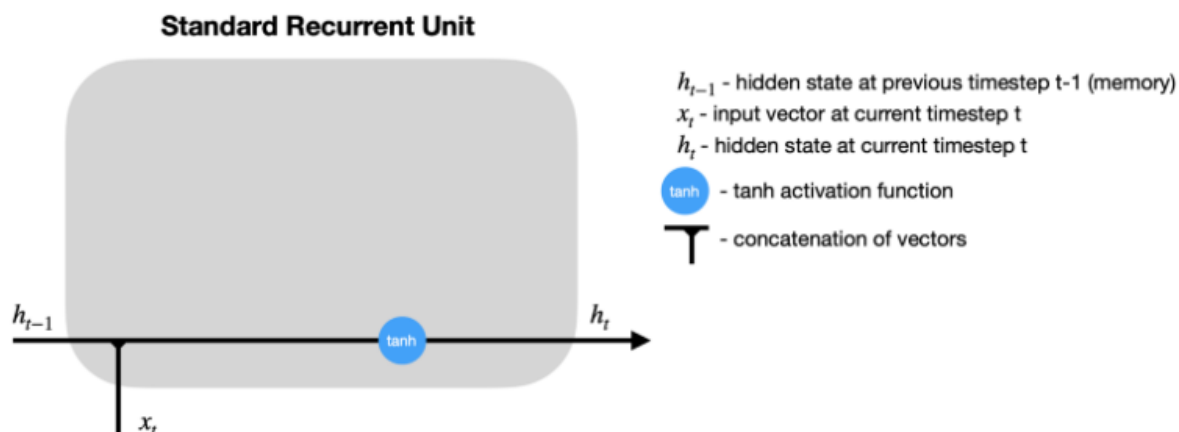
Standardne Cikličke neuronske mreže (*Recurrent Neural Network (RNN)*) pate od manjka kratkotrajne memorije zbog nestajućeg gradijenta koji proizilazi pri radu s više podataka. Zbog toga su nastale mreže ***Long Short-Term Memory (LSTM)*** [3] i ***Gated Recurrent Units (GRU)***, koje su napredne verzije RNN-a koje imaju mogućnost očuvanja bitnih informacija iz prijašnjih obrađenih nizova podataka te mogu te informacije proslijediti.

Kako bismo pobliže objasnili što to razlikuje LSTM od standardne RNN potrebno je poznavati osnovnu RNN strukturu. RNN se sastoji od nekoliko slojeva slično unaprijednoj neuronskoj mreži; Ulazni sloj, Skriveni sloj i Izlazni sloj (Sl.2.1). Potrebno je reći da RNN sadrži rekurzivne jedinice u svom skrivenom sloju što omogućava algoritmu obrađivanje sekvencijalnih podataka (npr. *Time-Series Data*). To se vrši rekurzivnim prosljeđivanjem skrivenog stanja iz prethodnog koraka i kombiniranjem ga s ulazom trenutnog stanja



Slika 2.1. Uobičajena struktura unaprijedne neuronske mreže

Poznato je da RNN koristi rekurzivne jedinice kako bi učila iz nizova podataka. To isto radi i LSTM, no iz dijagrama standardnog RNN (bez težina), moguće je primjetiti da postoje samo dvije glavne operacije a to su; spajanje vrijednosti prethodnog skrivenog stanja s novim ulazom i predavanje te vrijednosti aktivacijskoj funkcij



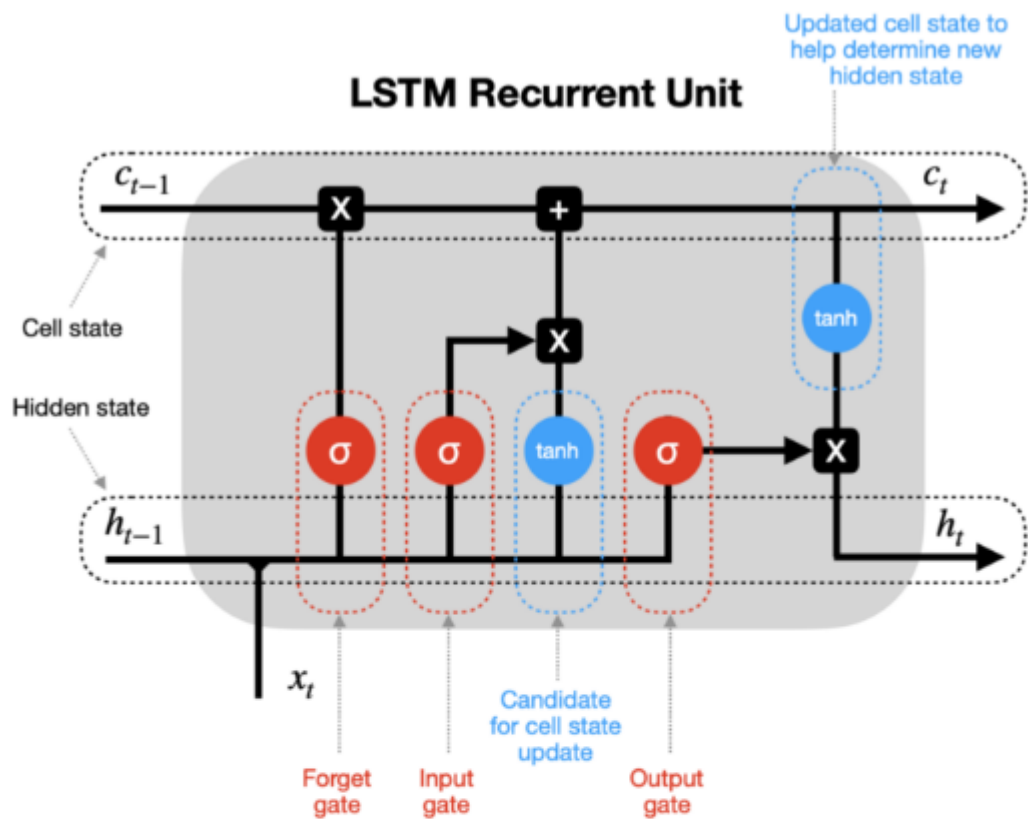
Nakon što izračuna skrivenog stanja u određenom koraku  $t$ , ta se vrijednost proslijeđuje rekurzivnoj jedinici te se kombinira s ulazom koraka  $t+1$  kako bi se

izračunala nova vrijednost skrivenog sloja u koraku  $t+1$ . Taj se proces ponavlja za korake  $t+2, t+3, \dots, t+n$  sve dok se ne dostigne predefiniiran broj koraka.

U međuvremenu LSTM koristi različita “vrata” kako bi odredio koje informacije zadržati, a koje odbaciti. Također dodaje i **stanje ćelije** kao vrijednost *long-term* memorije. Dakle rekurzivna jedinica LSTM-a je prilično kompliciranija za razliku od RNN-ove, što ima dobar utjecaj na ičenje neuronske mreže ali zahtjeva više računalnih resursa.

Na Slici 2.2 se nalazi pojednostavljeni dijagram (bez težina) kako bi shvatili kako rekurzivna jedinica LSTM-a obrađuje informacije.

1. Skriveno stanje i novi ulazi - skriveno stanje iz prethodnog koraka ( $h(t-1)$ ) i ulaz u trenutnom koraku ( $x(t)$ ) se kombiniraju prije nego se vrijednosti šalju prema raznim *vratima*
2. *Vrata zaborava* - ova vrata kontroliraju koje bi informacije treble biti zaboravljene. Budući da sigmoid funkcija ima raspon između 0 i 1, određuje koje vrijednosti u stanju ćelije će biti odbačene (pomnožene s 0), upamćene (pomnožene s 1) ili djelomično upamćene (pomnožene s vrijednošću iz  $[0,1]$ ).
3. Ulazna vrata - pomažu pri identificiranju važnih elemenata koji trebaju biti dodani trenutnom stanju ćelije. Treba naglasiti da se rezultati ulaznih vrata množe s kandidatom za stanje ćelije, pri čemu se u stanje ćelije dodaju samo informacije koje ulazna vrata smatraju važnima.
4. Ažuriranje stanja ćelije - prvo, prethodno stanje ćelije ( $c(t-1)$ ) se množi s rezultatom *vrata zaborava* te se potom dodaju nove informacije [ulazna vrata  $\times$  kandidat za stanje ćelije] kako bismo dobili najnovije stanje ćelije ( $c(t)$ ).
5. Ažuriranje skrivenog stanja - Posljednji dio ažuriranja je zapravo ažuriranje skrivenog stanja. Najnovija vrijednost stanja ćelije ( $c(t)$ ) se prosljeđuje kroz tanh aktivacijsku funkciju te množi s rezultatima izlaznih vrata.



$h_{t-1}$  - hidden state at previous timestep t-1 (short-term memory)

$c_{t-1}$  - cell state at previous timestep t-1 (long-term memory)

$x_t$  - input vector at current timestep t

$h_t$  - hidden state at current timestep t

$c_t$  - cell state at current timestep t

$\times$  - vector pointwise multiplication     $+$  - vector pointwise addition

$\tanh$  - tanh activation function

$\sigma$  - sigmoid activation function

$\top$  - concatenation of vectors

  - states

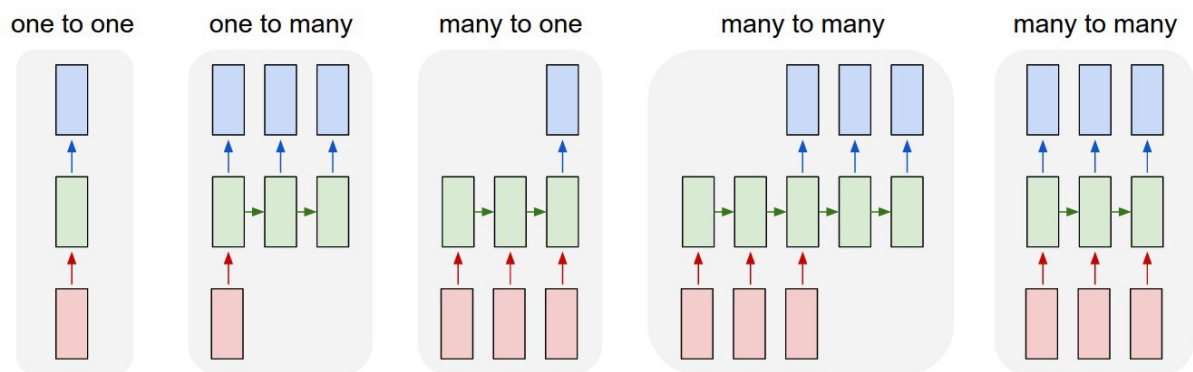
  - gates

  - updates

Slika 2.2 Dijagram rekurzivne jedinice LSTM-a

LSTM je moguće koristiti na 4 načina:

- One-to-one – Teoretski moguće, no s obzirom da jedna stavka ne predstavlja sekvencijalni niz podataka, ne dobijamo nikakve pogodnosti koje LSTM nudi. Stoga je u takvom scenariju bolje koristiti unaprijednu neuronsku mrežu
- Many-to-one – Koristi nizove podataka kako bi predvidjeli buduću vrijednost.
- One-to-many – Koristi jednu vrijednost kako bi predvidjeli niz vrijednosti
- Many-to-many – Koristeći nizove podataka kako bi predvidjeli sljedeći niz vrijednosti.





### 3. OPIS ZADATKA

Za realizaciju projekta, tj. za treniranje, validaciju i testiranje modela, koristi se podatkovni skup preuzet koristeći se `DataReader()` metodom iz biblioteke *pandas-datareader* koja omogućava preuzimanje vrijednosti u stvarnom vremenu s [1].

Kako bismo koristili neuronske mreže i ostale alate za obradu podataka, koristimo se bibliotekama koje je potrebno uvesti u naš projekt. Biblioteke korištene u ovom projektu su: *NumPy*, *Pandas*, *Matplotlib*, *Pandas-datareader*, *datetime*, *Scikit-learn* te *Keras*.

*NumPy* predstavlja jedan od najvažnijih modula za Python, a namijenjen je prije svega za rad s nizovima, vektorima i matricama što je poprilično važno pri radu s podacima o stanju s dionicama. *Pandas* je open-source biblioteka koja omogućuje rad s strukturiranim podacima te pruža alate za analizu podataka. *Matplotlib* je jedna od glavnih Python-ovih biblioteka za ispis grafova. *Tensorflow* je također open-source biblioteka koju su razvili Google-ovi stručnjaci za potrebe suočavanja s izazovima na području strojnog učenja i umjetne inteligencije, koju je također moguće koristiti i u C-u, Go-u i Javi. *Scikit-learn* je najkorisniji modul u Pythonu za područje strojnog učenja a sadrži razne algoritme za klasifikaciju, regresiju i klasteriranje. *Keras* je biblioteka visoke razine namijenjena dizajniranju i treniranju dubokih neuronskih mreža, a može koristiti *Tensorflow* kao pozadinski kod (Sl.3.1)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pandas_datareader as web
import datetime as dt

from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
```

Slika 3.1 Učitavanje biblioteka u skriptu

Nakon učitavanja biblioteka u skriptu, na samom početku programa, korisniku se nudi da izabere za koju kriptovalutu bi htio predvidjeti stanje dionice. Ponuđene opcije su **Bitcoin**, **Etherum** te **Doge Coin** (kako bi preuzeli vrijednosti kroz Yahoo Finance potrebno je poznavati *tickers* odnosno skraćenicu pod kojom je spremljena dionica (BTC-USD, ETH-USD, DOGE-USD)). Nakon što korisnik izabere koju kriptovalutu želi predvidjeti, postavlja mu se pitanje nad kojim podacima učiti neuronsku mrežu, stoga korisnik mora unijeti period vremena za koji će biti preuzeti podaci za treniranje neuronske mreže i to 'Closed' vrijednosti za cijeli odabrani period (SI 3.2).

```
#Selecting time range for training data of neural network
startDate = input("Please enter wanted start Date in form (YYYY-MM-DD)\n")
yearStart, monthStart, dayStart = map(int, startDate.split('-'))
startDate = dt.date(yearStart,monthStart,dayStart)
endDate = input("Please enter wanted end Date in form (YYYY-MM-DD)\n")
yearEnd, monthEnd, dayEnd = map(int, endDate.split('-'))
endDate = dt.date(yearEnd,monthEnd,dayEnd)
#Loading the data from Yahoo Finance
dataToRead = web.DataReader(selectedCrypto,'yahoo',startDate,endDate)

#Preparing data for network
scaler = MinMaxScaler(feature_range=(0,1))
scaledData = scaler.fit_transform(dataToRead['Close'].values.reshape(-1,1))
```

Slika 3.2 Omogućavanje unosa perioda za podatke treniranja

Preuzimanjem podataka, moguće je početi s izgradnjom modela neuronske mreže no prije toga potrebno je dodatno obraditi preuzete podatke kako bi mogli koristiti ih za sam trening i testiranje mreže. Nakon obrade podataka od korisnika se zahtjeva unos broja koji označava koliko dana unazad će model razmatrati kako bi izračunao vašu vrijednost.

Odlučeno je da model neuronske mreže za predviđanje dionica izgleda kao na slici ispod. Unaprijed definiranom modelu, potrebno je podesiti/odrediti broj epoha (broj prolazaka kroz čitav dataset za treniranje) i *batch size* (parametar koji nam definira broj uzoraka koji će propagirati kroz mrežu). Prednosti manje vrijednosti *batch\_size*-a su manji zahtjevi za memorijom i veća brzina učenja. Međutim ukoliko je *batch\_size* malen, veća je pogreška. U projektu su korištene sljedeće vrijednosti: *batch\_size* = 32, *epochs* = 40. Te vrijednosti nije moguće jednostavno unijeti nego ih je potrebno testirati dok nismo zadovoljni s rezultatom gubitka (SI 3.3)

```

model = Sequential()

model.add(LSTM(units=50,return_sequences=True,input_shape = (x_train.shape[1],1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True,))
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True,))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1)) #Actual prediction of next price

model.compile(optimizer='adam',loss='mean_squared_error')
history = model.fit(x_train, y_train, epochs = 40, batch_size= 32)

```

Slika 3.3 Model neuronske mreže za predviđanje dionica

Kako bismo testirali izrađeni model potrebno je prvo obraditi i pripremiti podatke za testiranje. Tako, za testne podatke koristimo vremenski period od završnog trenutka za učenje neuronske mreže do današnjeg dana. Podatke je potrebno preoblikovati (Sl 3.4)

```

#-----TESTING MODEL ACCURACY on Existing data-----
#Test data from last day of training data until now
testStartDate = endDate
testEndDate = dt.datetime.now()
testData = web.DataReader(selectedCrypto, 'yahoo', testStartDate, testEndDate)

actual_prices = testData['Close'].values
totalDataset = pd.concat((dataToRead['Close'],testData['Close']),axis=0)
model_inputs = totalDataset[len(totalDataset)-len(testData)-predictionDays:].values
model_inputs = model_inputs.reshape(-1,1)
model_inputs = scaler.transform(model_inputs)

#Prepare test data
x_test = []
for x in range (predictionDays,len(model_inputs)):
    x_test.append(model_inputs[x-predictionDays:x, 0])
x_test = np.array(x_test)
x_test = np.reshape(x_test,(x_test.shape[0],x_test.shape [1],1))

```

Slika 3.4 Dio programske podrške za obradu testnih podataka

Vrijednosti koje se nalaze u varijabli `x_test` su vrijednosti koje ćemo koristiti za testiranje neuronske mreže. Nakon predviđanja u koji interval buduća vrijednost dolazi, potrebno je inverzno transformirati ju (`MinMaxScaler()`, Sl 3.5).

```
#Make prediction on Test Data
predictedPrices = model.predict(x_test)
predictedPrices = scaler.inverse_transform(predictedPrices)
```

Slika 3.5, Predviđanje nad testnim podacima i post-modalna obrada

Također kao što je navedeno u zadatku potrebno je omogućiti predviđanje dionica za vrijednosti koje nemamo te je stoga omogućeno predviđanje dionica za sutrašnji dan (Sl 3.6).

```
#Predict for Tomorrow!!!
real_data = [model_inputs[len(model_inputs)+1-predictionDays:len(model_inputs+1),0]]
real_data = np.array(real_data)
real_data = np.reshape(real_data,(real_data.shape[0],real_data.shape[1],1))

predictionForTomorrow = model.predict(real_data)
predictionForTomorrow = scaler.inverse_transform(predictionForTomorrow)
print(f"Prediction: {predictionForTomorrow}")
```

Slika 3.6 Predviđanje stanja dionica za sutrašnji dan

Nakon predviđanja potrebno je uraditi evaluaciju modela. Evaluaciju modela moguće je dobiti uz pomoć metode `model.summary()` koja prikazuje vrijednosti kao što su ime i tip svakog sloja modela, izlazni oblik podataka svakog sloja, broj težina parametara svakog sloja kao i ukupan broj parametara koje je moguće trenirati (Sl 3.7).

```
print("")
print('----- Model Summary -----')
model.summary() # print model summary
print("")
#print('----- Weights and Biases -----')
#print("")
#for layer in model.layers:
#    print(layer.name)
#    for item in layer.get_weights():
#        print(" ", item)
#print("")

# Print the last value in the evaluation metrics contained within history file
print('----- Evaluation on Training Data -----')
for item in hisotry.history:
    print("Final", item, ":", hisotry.history[item][-1])
print("")

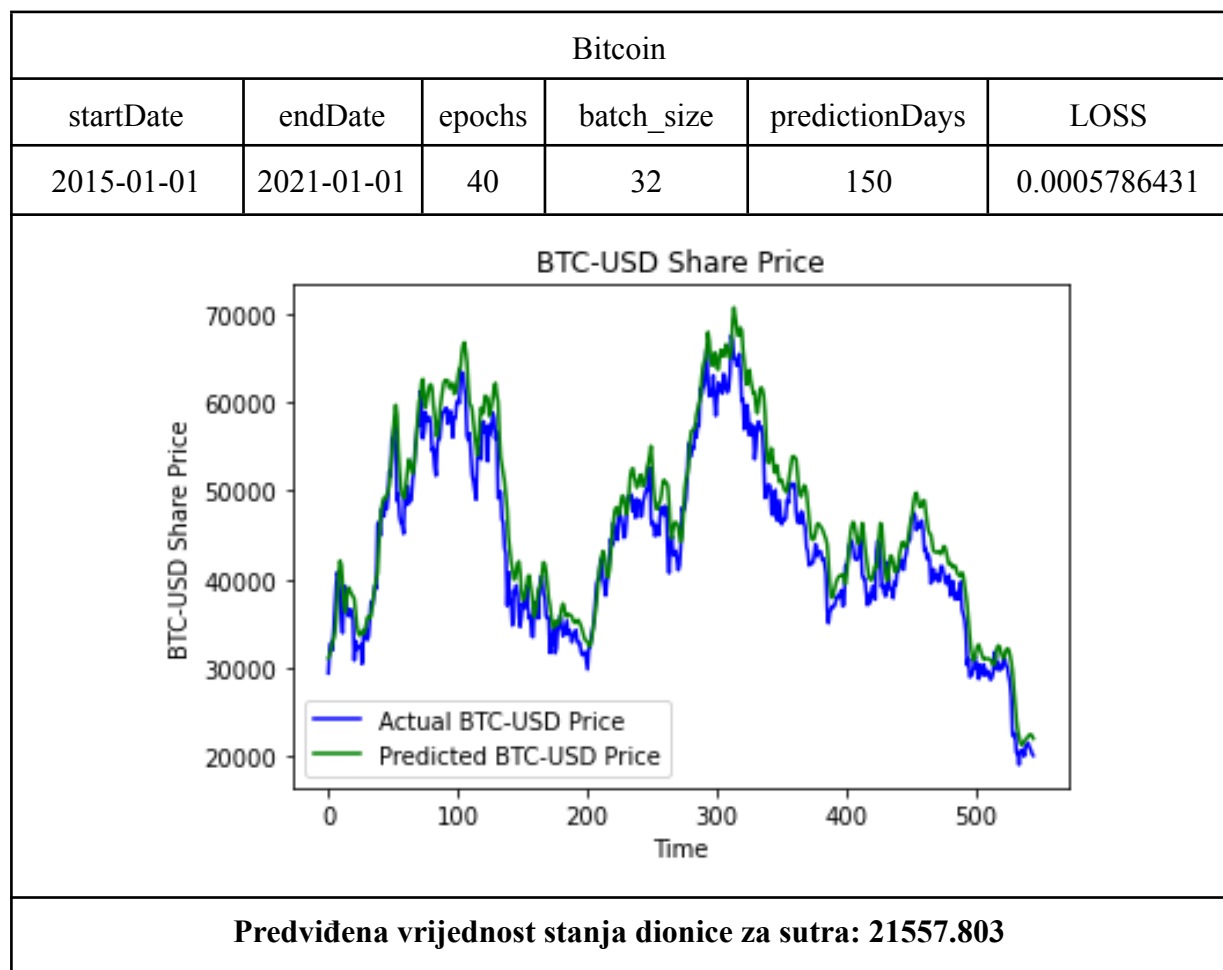
# Evaluate the model on the test data using "evaluate"
print('----- Evaluation on Test Data -----')
results = model.evaluate(x_test)
print("")
```

Slika 3.7 Programska podrška za analizu stanja modela

## 4. PRIKAZ REZULTATA I OPTIMIZACIJA MODELA

U sljedećem poglavlju ćemo mijenjati parametre modela kako bismo vidjeli utjecaj istih na točnost i brzinu izvođenja. Vrijednosti prikazane u tablicama odgovaraju vrijednostima i varijablama iz koda radi lakšeg pisanja tablica (*startDate* - datum početka za treniranje mreže, *predictionDays* - koliko dana unazad želimo razmatrati za izračun buduće vrijednosti)

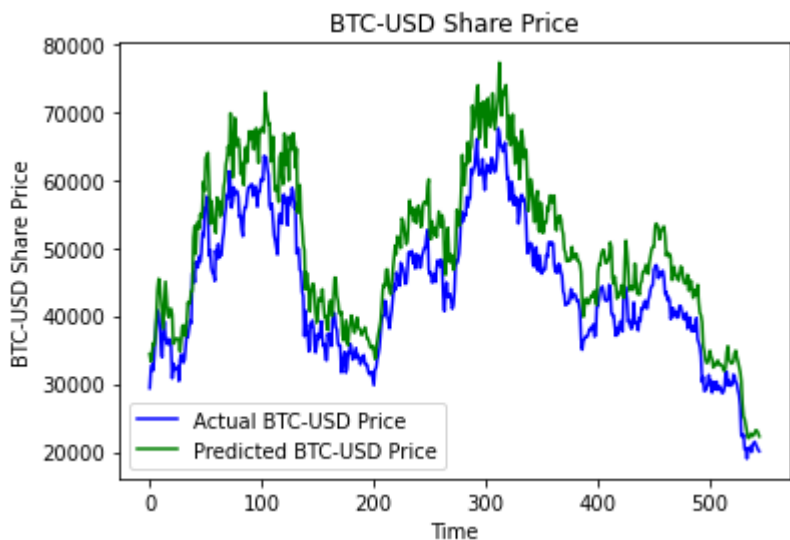
### 4.1. Utjecaj promjene batch\_size-a na točnost modela

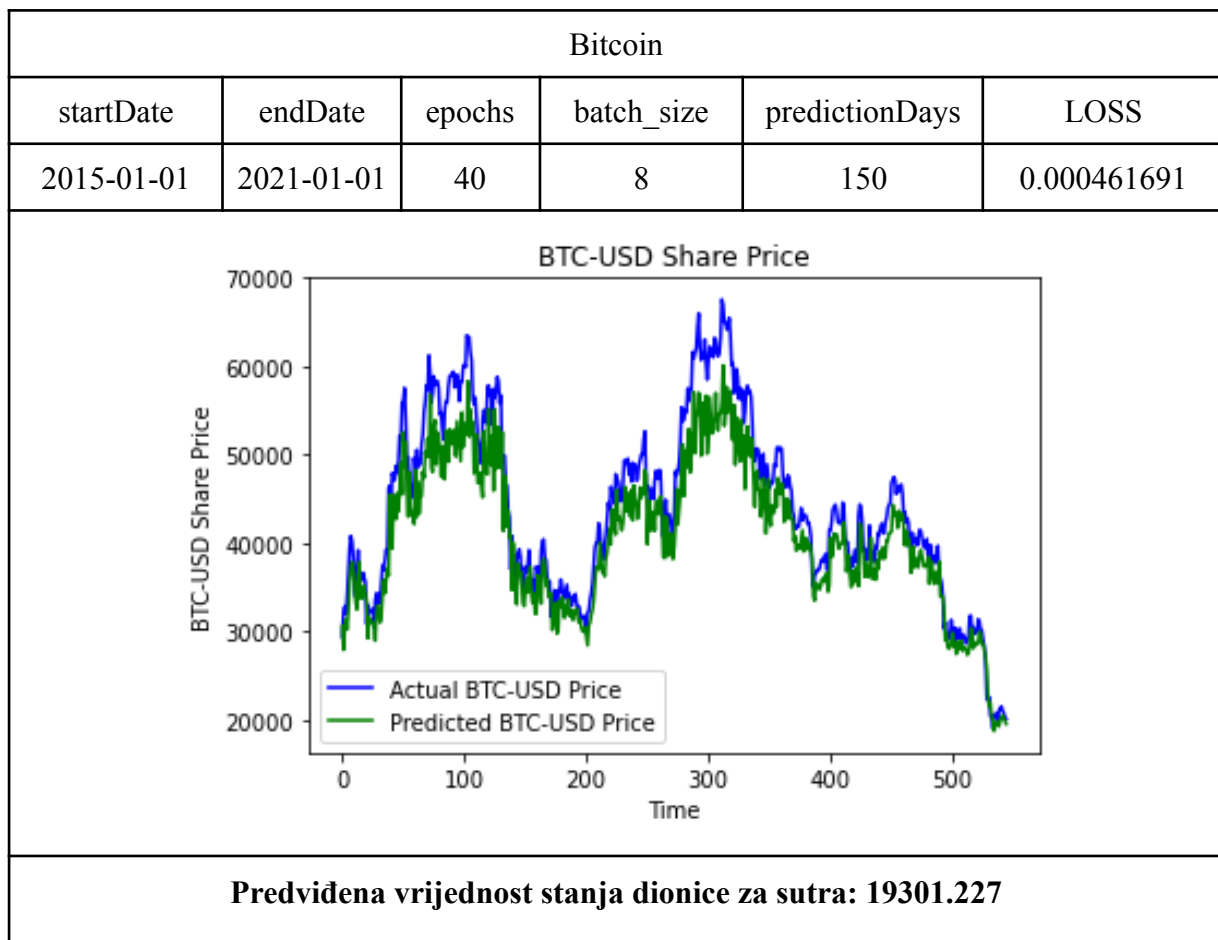


```

----- Model Summary -----
Model: "sequential"
Layer (type)                Output Shape                Param #
-----
lstm (LSTM)                  (None, 150, 50)            10400
dropout (Dropout)            (None, 150, 50)            0
lstm_1 (LSTM)                (None, 150, 50)            20200
dropout_1 (Dropout)          (None, 150, 50)            0
lstm_2 (LSTM)                (None, 150, 50)            20200
dropout_2 (Dropout)          (None, 150, 50)            0
lstm_3 (LSTM)                (None, 50)                 20200
dropout_3 (Dropout)          (None, 50)                 0
dense (Dense)                (None, 1)                  51
-----
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0

```

Bitcoin					
startDate	endDate	epochs	batch_size	predictionDays	LOSS
2015-01-01	2021-01-01	40	16	150	0.0005919876
					
Predviđena vrijednost stanja dionice za sutra: 21968.424					

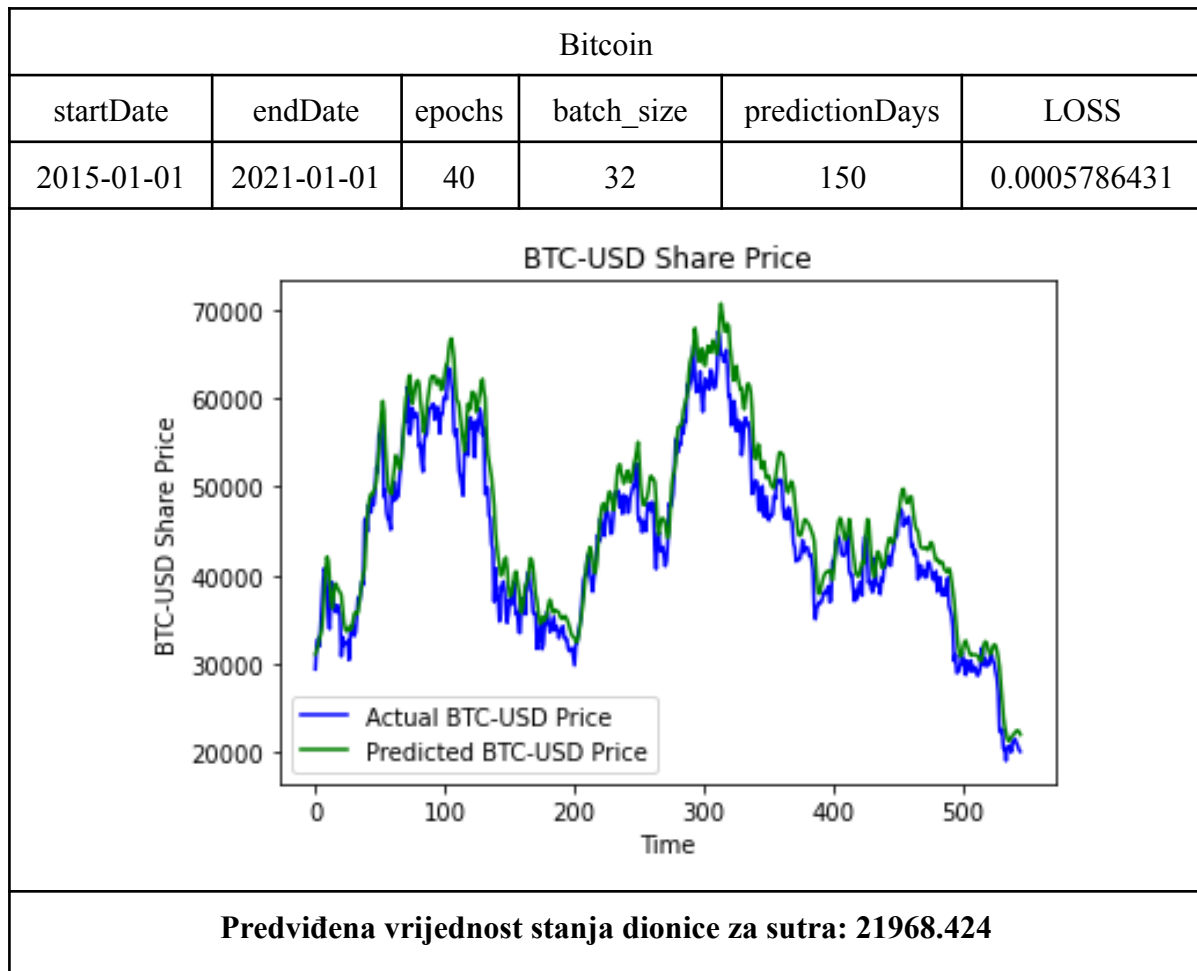


Vidimo da promjenom batch\_size-a utječemo na točnost predviđanja cijene dionice što je i navedeno u prethodnom poglavlju. Smanjenjem batch\_size-a dobijamo nešto bolji model iako po mom mišljenju kombinacija parametara iz prvog slučaja (epohe=40, batch\_size=32 i predictionDays = 150) daje najbolje rješenje.

Također iz sljedećih poglavlja vidimo da smanjenjem broja epoha odnosno smanjenjem broja prolazaka kroz cijeli dataset smanjujemo i točnost predviđenih podataka.

Promjenom broja dana koji će biti uzeti u obzir kako bi izračunali buduću vrijednost isto direktno utječemo na točnost modela na način da; smanjenjem broja dana za predviđanje dobijamo nešto bolju vrijednost. Ukoliko je dionica bila više promjenjiva u prošlosti može doći do loše predikcije i samim time lošeg rezultata.

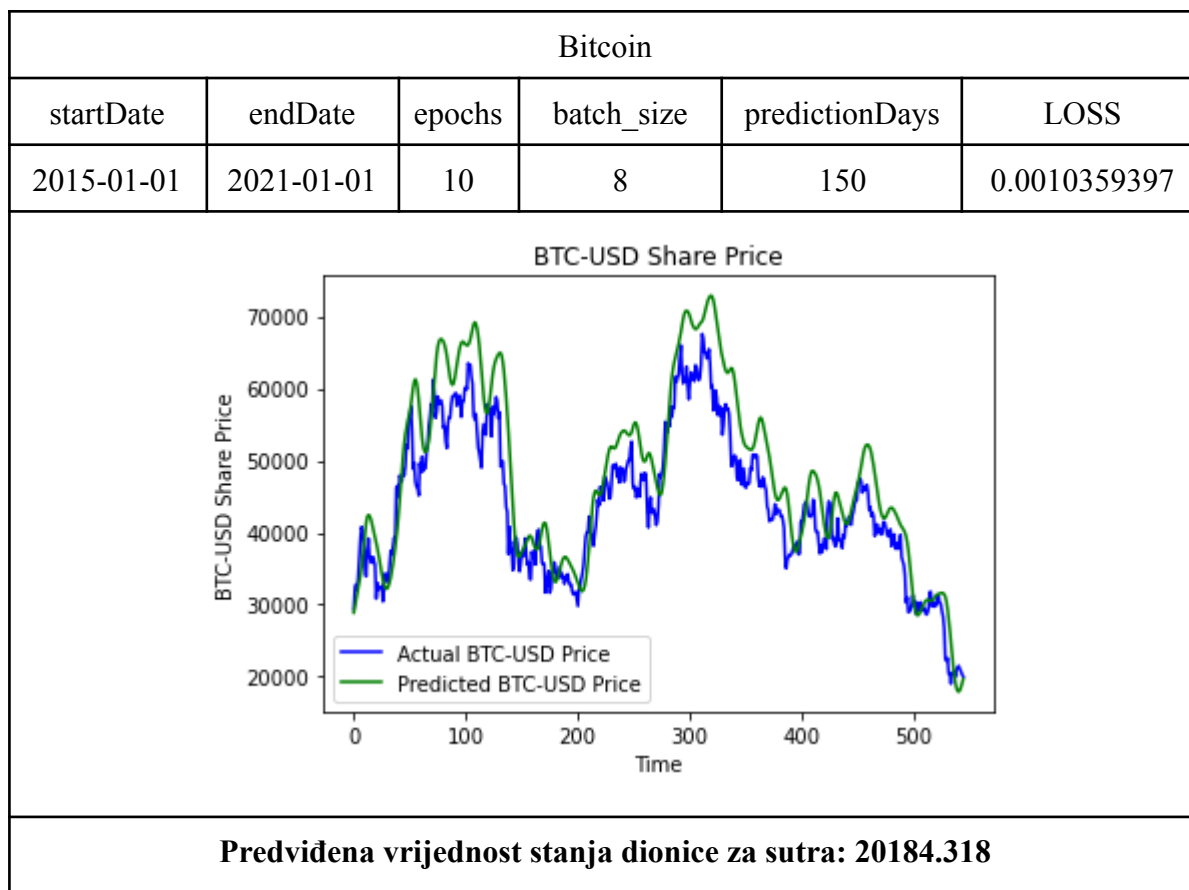
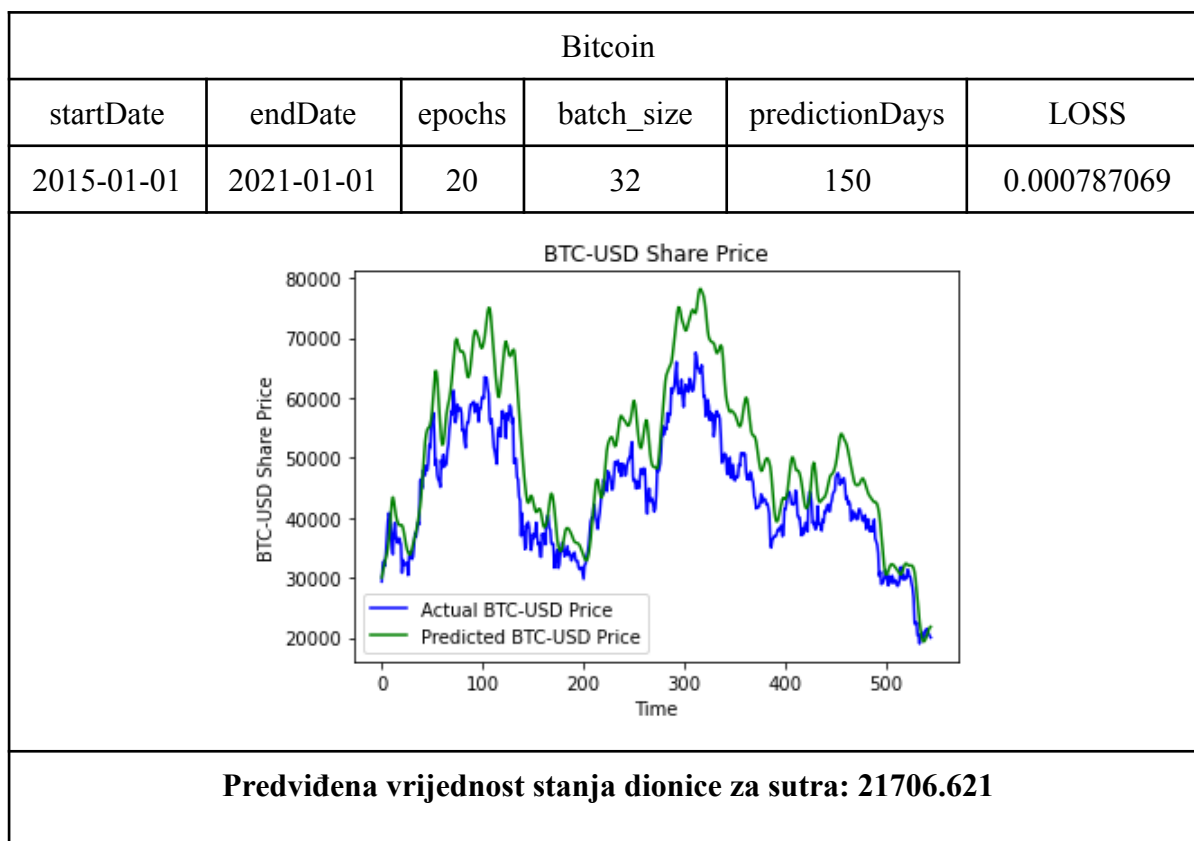
## 4.2. Utjecaj promjene broja epoha na točnost modela



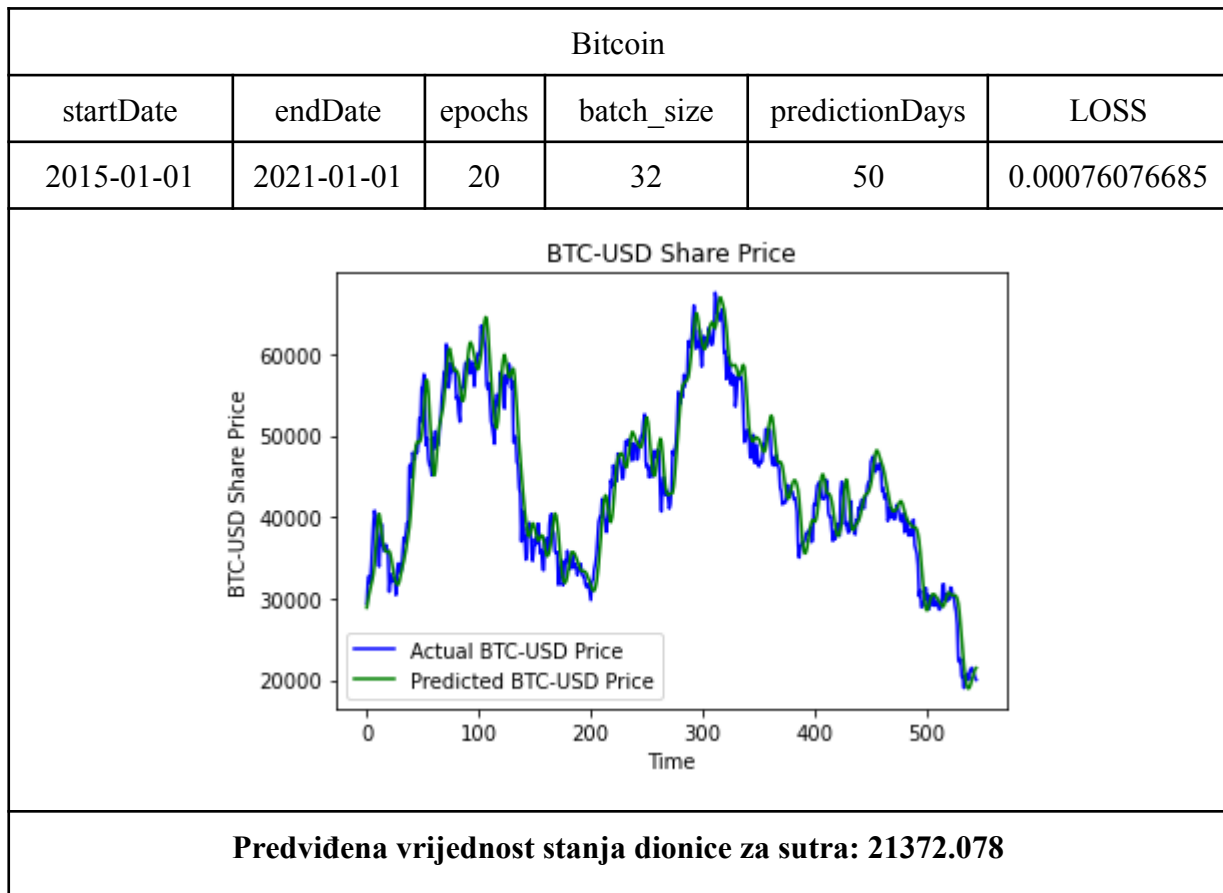
```

----- Model Summary -----
Model: "sequential_3"
Layer (type)                Output Shape                Param #
-----
lstm_12 (LSTM)               (None, 150, 50)            10400
dropout_12 (Dropout)         (None, 150, 50)            0
lstm_13 (LSTM)               (None, 150, 50)            20200
dropout_13 (Dropout)         (None, 150, 50)            0
lstm_14 (LSTM)               (None, 150, 50)            20200
dropout_14 (Dropout)         (None, 150, 50)            0
lstm_15 (LSTM)               (None, 50)                  20200
dropout_15 (Dropout)         (None, 50)                  0
dense_3 (Dense)              (None, 1)                   51
-----
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
  
```



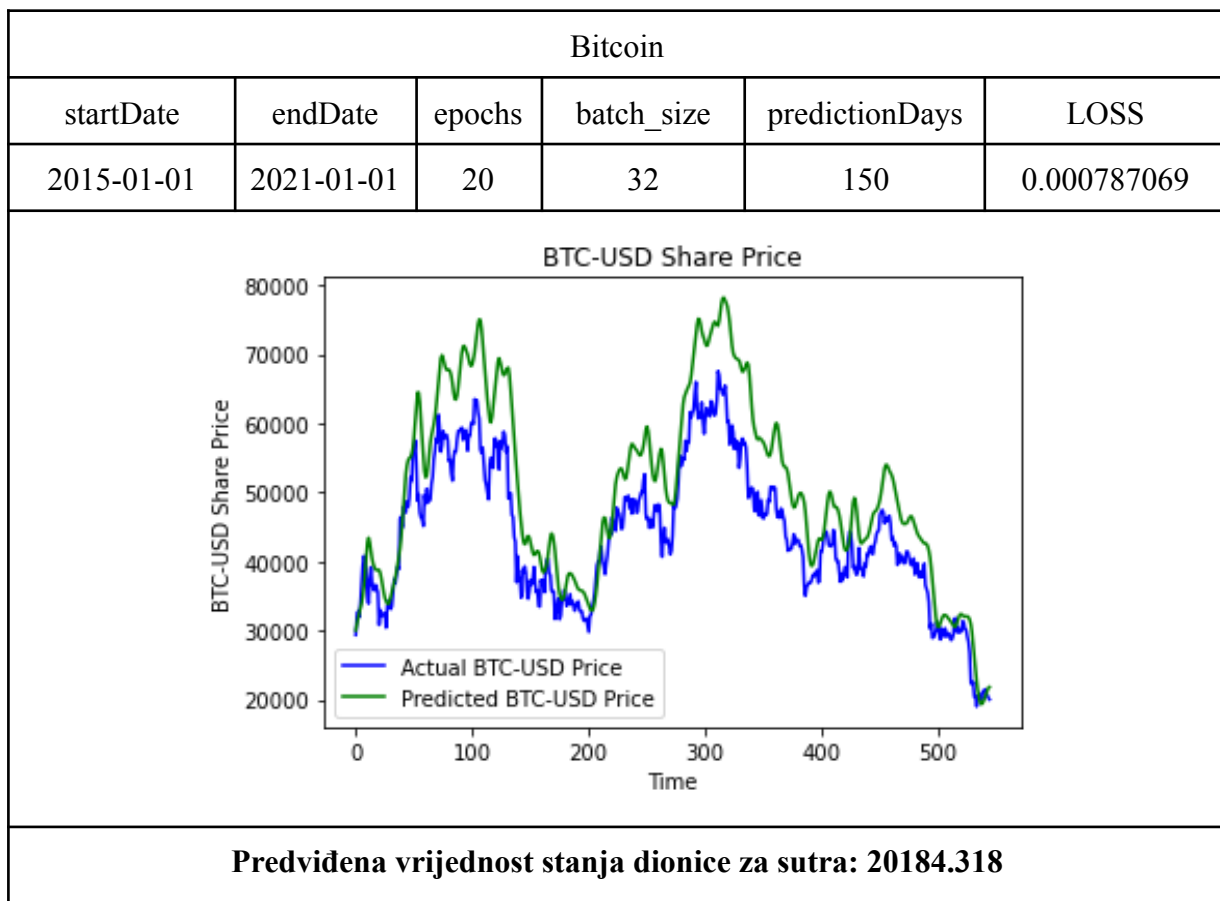
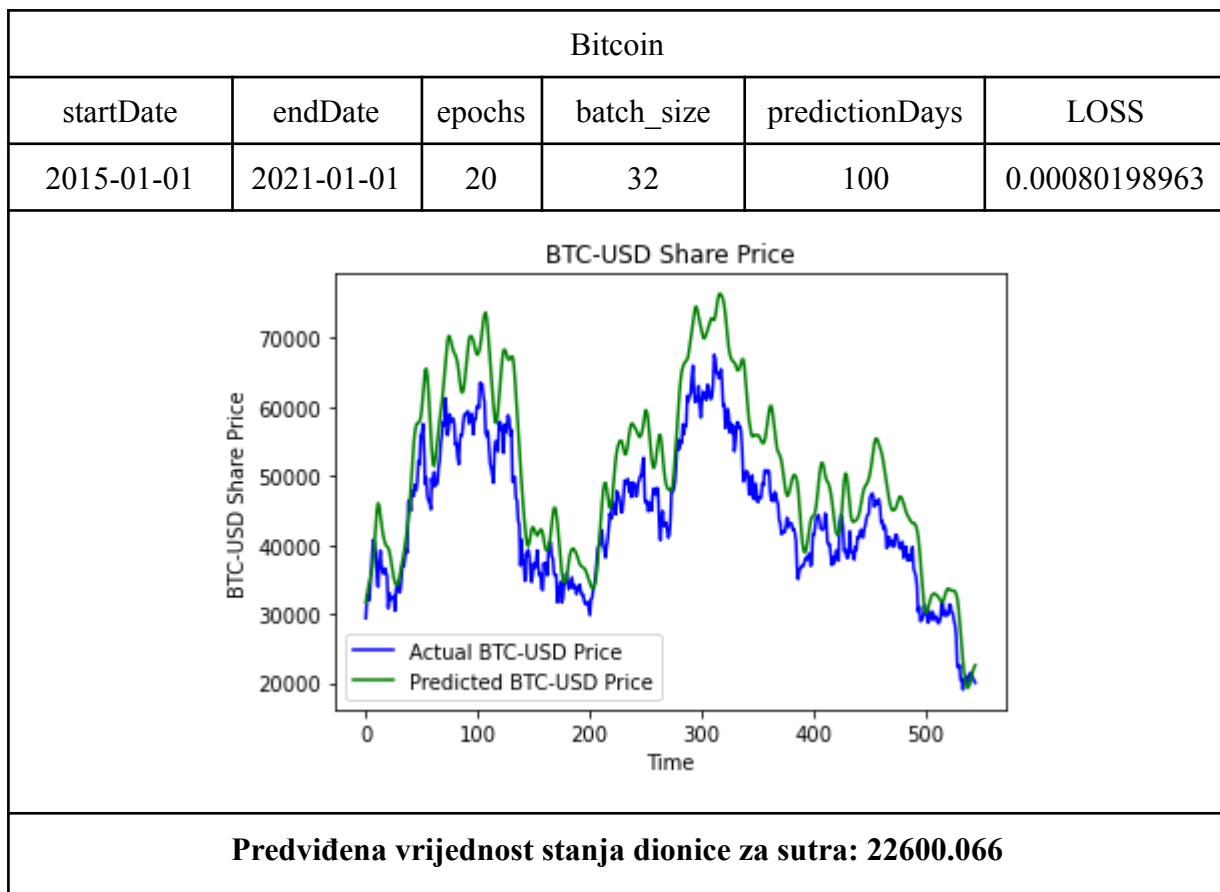


### 4.3. Utjecaj promjene broja prethodnih dana na točnost modela



```

----- Model Summary -----
Model: "sequential_6"
Layer (type)                Output Shape                Param #
-----
lstm_24 (LSTM)               (None, 50, 50)             10400
dropout_24 (Dropout)         (None, 50, 50)             0
lstm_25 (LSTM)               (None, 50, 50)             20200
dropout_25 (Dropout)         (None, 50, 50)             0
lstm_26 (LSTM)               (None, 50, 50)             20200
dropout_26 (Dropout)         (None, 50, 50)             0
lstm_27 (LSTM)               (None, 50)                 20200
dropout_27 (Dropout)         (None, 50)                 0
dense_6 (Dense)              (None, 1)                  51
-----
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
  
```

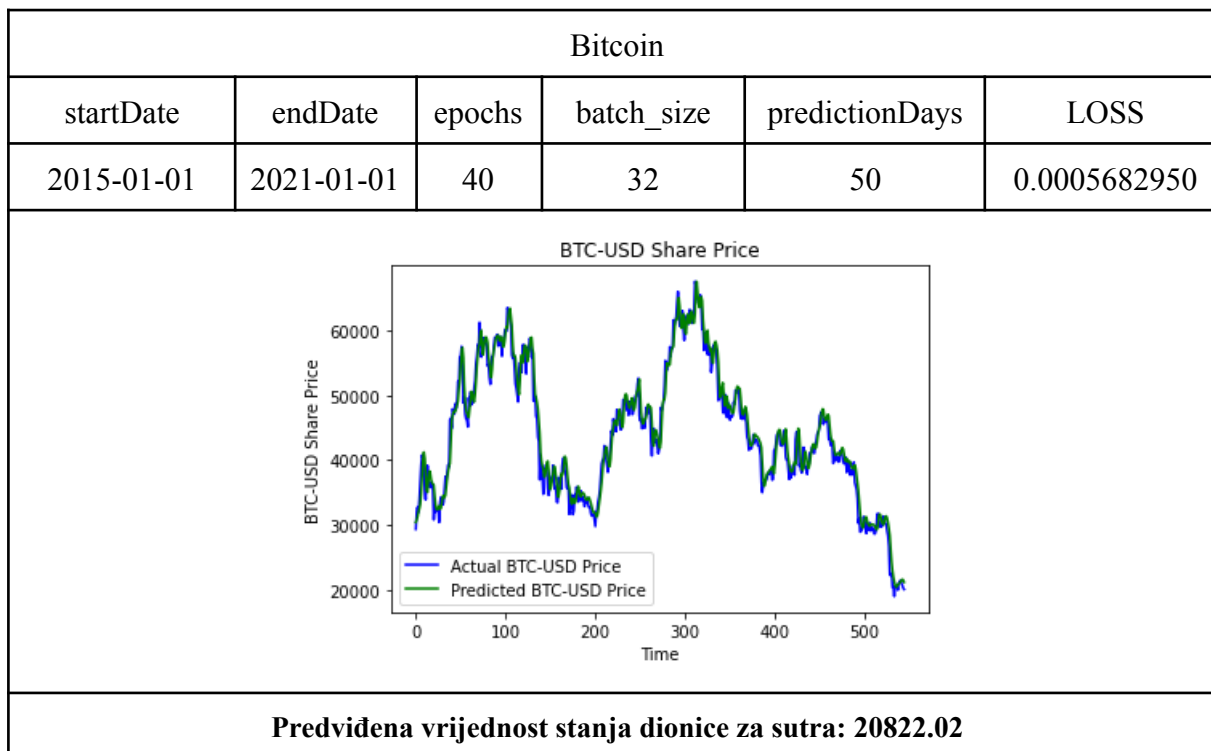


## 5. ZAKLJUČAK

Koristeći programski jezik Python i biblioteku Keras u ovom projektnom zadatku je napravljen model za predviđanje stanja dionica neke od kriptovaluta. Vidimo da Bitno je naglasiti kako se ne preporučuje korištenje ovog projekta kako biste vršili kupovinu odnosno prodaju dionica budući je sam model prilično jednostavan. Iz prethodnih poglavlja vidimo kako predviđanje cijene dionica nije jednostavan proces te je gotovo nemoguće predvidjeti buduće stanje dionica zbog previše vanjskih utjecaja.

Hiperparametri ovog modela su ekstremno osjetljivi na rezultate koje dobijemo tako da bi bila odlična stvar odraditi optimizaciju hiperparametara kao što su *nivo učenja optimizer*, *broj slojeva* i *broj skrivenih jedinica u svakom sloju*, *optimizer (adam se pokazao najbolji)*, itd. Zanimljiv podatak je taj da je strojno učenje postiglo veliki napredak tijekom prethodnih nekoliko godina. Naime 2018. Je OpenAI stvorio botove koji su pobijedili profesionalne igrače u igrici Dota 2. Ta mreža se sastojala od 5 neovisnih no povezanih neuronskih mreža te svaka mreža sadrži jedan sloj od 1024 LSTM jedinice koje prate trenutno stanje igre i vrše akcije kroz nekoliko *akcijskih glava*.

Kroz testiranje je zaključeno da kombinacija sljedećih parametara daje najbolje rezultate za predviđanje dionice Bitcoin-a;



## 6. LITERATURA

[1] <https://finance.yahoo.com/>

[2] Malkiel, Burton G. 2019. A Random Walk down Wall Street.

[3] Hochreiter and Schmidhuber, 1997. “Long Short Term Memory”, Neural Computation