УНИВЕРЗИТЕТ „Св. КИРИЛ И МЕТОДИЈ“ - СКОПЈЕ

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

-ПРОЕКТ-
по предметот
ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА

Тема

Прогнозирање на цена на биткоин со користење на модел трениран со рекурентни невронски мрежи

Ментор:

Проф. Д-р Андреа Кулаков

Изработиле:

Тамара Костова, Евгенија Попчановска

Скопје, септември 2023

Содржина

Апстракт

Биткоин е една од најпопуларните и највредните криптовалути на моменталниот пазар и таа привлекува голем број на инвеститори. Токму затоа претставува примамлива тема за голем број на истражувачи во однос на предвидување на нејзината цена. Постојат бројни истражувања кои вклучуваат разни типови на алгоритми кои се служат со машинско учење за изведување на предвидувањето.

Вовед

Нашиот проект обработува метод за тренирање на едноставен модел со Long Short Term Memory (LSTM) и Gated Recurrent Units (GRU) за предвидување на цената на биткоин и споредба меѓу двата алгоритми и добиените резултати. При тренирање на ваквите модели со значително множество на податоци, се добиваат задоволителни резултати.

Рекурентни невронски мрежи

Рекурента невронска мрежа (РНМ) е тип на невронска мрежа која содржи меморија и најдобро служи за обработка на секвенцијални податоци. Ваквиот тип на невронски мрежи главно има примена во обработка на природните јазици и препознавање на говор, а исто така е имплементиран и кај дигиталниот асистент Siri на компанијата Apple и гласовното пребарување на Google. Заради овие подобности на рекурентните мрежи, во нашиот проект за предвидување на цена на криптовалута имплементираме рекурентни невронски мрежи од тип Long Short-Term Memory и Gated Recurrent Unit.

Рекурентната невронска мрежа има повторлива природа бидејќи ја пресметува истата функција за секој влез на податоци додека излезот на тековниот влез зависи од минатите пресметки. По генерирање на еден излез, тој се копира и се испраќа назад во рекурентната мрежа. За носење на одлука, се зема предвид и моменталниот влез и излезот што моделот го научил од претходниот влез. На пример, при генерирање на реченица, за да се предвиди следниот збор потребно е да се паметат сите минати зборови. Така, РНМ во скриениот слој ги меморира податоците од низата. Ова е главната карактеристика на рекурентните невронски мрежи и се нарекува уште скриена или мемориска состојба.

Друга карактеристика на рекурентните мрежи е тоа што параметрите во секој слој на мрежата се идентични. „Feedforward“ мрежите имаат различни тежини за еден ист јазол во различните слоеви, додека рекурентните невронски мрежи го делат истиот тежински параметар во секој слој на мрежата.

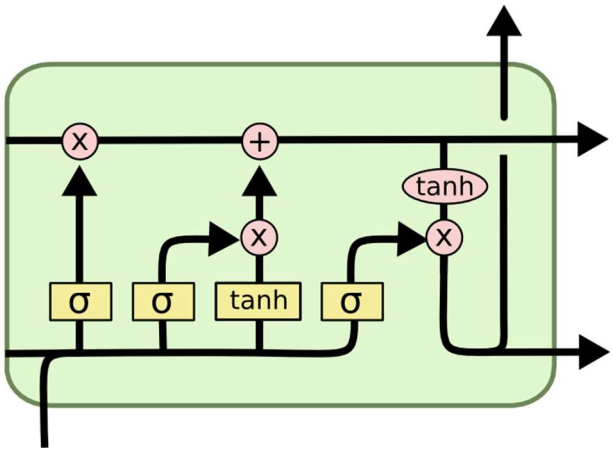
Важен аспект кој треба да се спомене кај рекурентните невронски мрежи е градиентот. Градиентот претставува парцијален извод во однос на неговите влезови и тој всушност мери колку се менува излезот на функцијата при мала промена на нејзините влезови. Колку е повисок градиентот, кривата на учење е пострмна, па моделот учи побрзо. Заради проблемот со градиентот и ратата на учење направени се модификации во РНМи, односно LSTM (Long Short Term Memory) и GRU (Gated Recurrent Unit) тип на мрежа.

Long Short-Term Memory

Рекурентните мрежи не може долго време да ги меморираат податоците и почнуваат да ги забораваат претходните влезови. За да се надмине овој проблем, кој е поврзан со изчезнување на градиентот, се користи Long Short-Term Memory, кој е посебен тип на

рекурентна невронска мрежа. Кај РНМ, кога се додава нова информација, мрежата целосно ги менува постоечките информации. Таа не е во состојба да прави разлика помеѓу важни или не толку важни информации. За разлика од тоа, кај LSTM има мала промена на постојните информации кога се додава нова информација на влез бидејќи LSTM содржи порти кои го одредуваат протокот на информации.

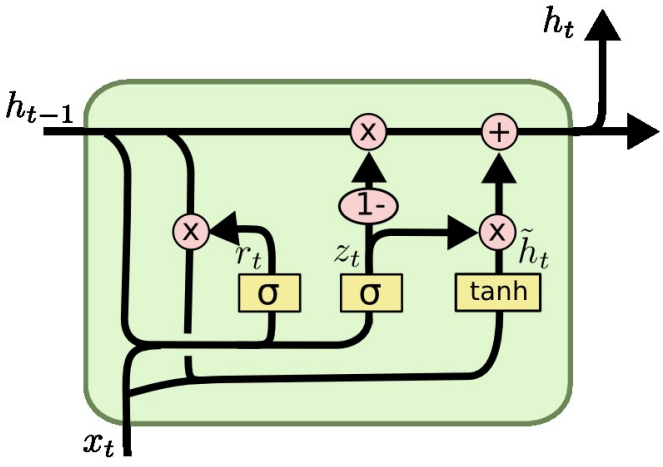
Портите одлучуваат кои податоци се важни и можат да бидат корисни во иднина и кои податоци треба да се избришат. Трите порти се влезна порта, излезна порта и „порта на заборав“ (forget gate). Портата на заборав одлучува кои информации се важни и треба да се складираат, а кои информации да се заборават. Влезната порта се користи за додавање информации во јазолот. Излезна порта е одговорна за избирање важни информации од тековниот јазол и нивно прикажување како излез.



Сл.1 Графички приказ на LSTM.

Gated Recurrent Unit

Покрај LSTM постои и друг вид на рекурентни невронски мрежи таканаречен Gated Recurrent Unit (GRU). GRU има помал број на порти и помал број на параметри во споредба со LSTM, што го прави поедноставен и побрз, но исто така и помалку моќен и адаптибилен. Кај GRU мрежите присутни се само 2 порти: ажурирачка порта (update gate) и ресетирачка порта (reset gate). Ажурирачката порта одлучува дали состојбата на ќелијата треба да се ажурира со моменталната активациска вредност или не. Ресетирачката порта служи за одлучување дали претходната состојба на ќелијата е важна или не.



Сл.2 Графички приказ на GRU.

Анализа на проблемот

Дефиниција на проблемот

Да се развие и истренира модел кој со користење на рекурентни невронски мрежи (LSTM и GRU) врз база на одредени карактеристики ќе ги предвидува цените на криптовалутите со мала рата на грешка.

Опсег на проблемот

При користење на овој модел, прво треба да се земе множеството на податоци. Множеството податоци содржи карактеристики врз база на кои моделот понатаму ќе го врши предвидувањето. Карактеристиките кои се искористени кај овој модел се: цена при отворање на временскиот интервал, највисока цена од сите тргувања за време на интервалот, најниска цена од сите тргувања за време на интервалот, цена при затворање на временскиот интервал, волумен за BTC (сума од сите трансфери на биткоин во интервалот), волумен (сума од сите трансфери на долари во интервалот) и тежинска цена (волумен на долари/ волумен на биткоин). Врз база на наученото од тренирачкото множество, после одреден број на епохи моделот е спремен да врши предвидувања на нови множества од податоци.

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Volume (BTC)	Volume (Currency)	Weighted Price
2	2014-01-07	874.6704	892.06753	810	810	15.62237812	13151.47284	841.8355223
3	2014-01-08	810	899.84281	788	824.98287	19.18275555	16097.32958	839.1562694
4	2014-01-09	825.56345	870	807.42084	841.86934	8.1583345	6784.249982	831.5729126
5	2014-01-10	839.99	857.34056	817	857.33056	8.02451046	6780.220188	844.9387936
6	2014-01-11	858.2	918.05471	857.16554	899.84105	18.74828487	16698.56693	890.6717092
7	2014-01-12	899.96114	900.93989	833.00001	860	25.42943314	21880.87898	860.454846
8	2014-01-13	847.32152	859.99999	815	835	25.86912661	21529.8395	832.2600071
9	2014-01-14	835	877.293	805	831	31.66288131	26756.27845	845.0361224
10	2014-01-15	831	864	828	850.00364	6.70756459	5698.139372	849.5094301
11	2014-01-16	853	865	824	826.97077	28.6020137	24229.47835	847.1249124
12	2014-01-17	824.1264	835.06427	788	804	26.35803928	21249.58041	806.1897237
13	2014-01-18	812.90538	844.14159	807.74074	819.51	18.76832727	15488.33894	825.2381109
14	2014-01-19	819.51	848.9	819.51	835	11.93959203	9816.159199	822.1519775
15	2014-01-20	843.94804	855	830.1	843.76589	4.44675857	3747.005869	842.6375775
16	2014-01-21	840	845.98999	820.1	829.01	6.89537424	5771.281599	836.9787335
17	2014-01-22	829	834.53144	821	830.88999	1.71261648	1421.643261	830.1001872
18	2014-01-23	822.4	830.88899	821	830.88897	11.58125085	9553.06187	824.8730637
19	2014-01-24	830.88897	835.48427	783	803.01935	25.00906729	20467.14709	818.3890608
20	2014-01-25	803.01936	830	795.07478	812.02	9.92963832	8121.589126	817.9138921
21	2014-01-26	818	840	813	826.72139	6.56973636	5460.465083	831.1543697
22	2014-01-27	826.70102	840	780	780	35.69920751	28647.24852	802.4617496
23	2014-01-28	780	830	775	815.99	15.75364425	12683.62099	805.1229788

Сл.3 Табеларен приказ на множеството податоци со карактеристиките.

Имплементација на решение во Пајтон

Прво, се импортираат сите потребни библиотеки за обработка на податоците, исцртување на графиците и тренирање на моделот.

```
from math import sqrt
import numpy as np
from numpy import concatenate
from matplotlib import pyplot
import pandas as pd
```

```
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dence, GRU, Dropout
import plotly.offline as py
import plotly.graph_objs as graph
from sklearn.preprocessing import MinMaxScaler
```

Се читаат податоците од документ bitcoininputdata сместен по директориум со име input. Потоа податоците со вредност 0 се заменуваат со последно прочитаната ненулта вредност со цел да се добијат поточни предвидувања.

```
dataset=pd.read_csv(filepath_or_buffer="input/bitcoininputdata", index_col="Date")
dataset['Weighted Price'].replace(0, np.nan, inplace=True)
dataset['Weighted Price'].fillna(method='ffill', inplace=True)
```

Со користење на скалер (MinMaxScaler) податоците се скалираат во вредности помеѓу 0 и 1.

```
values = dataset['Weighted Price'].values.reshape(-1,1)
values = values.astype('float32')
scaler = MinMaxScaler(feature_range=(0, 1))
values = scaler.fit_transform(values)
```

Првите 70% од добиените скалирани податоци од множеството ги користиме за тренирање, а останатите 30% за тестирање на моделот.

```
train_size = int(len(values) * 0.7)
test_size = len(values) - train_size
train, test = values [0:train_size, :], values [train_size:len(values), :]
```

Функција за припрема на множеството податоци која користи целобројен параметар („look back“) кој специфицира колку временски чекори наназад треба да се земат предвид при креирање на влезно-излезните парови за моделот.

```
def prepare_data(dataset, look_back=1):
        data_x, data_y = [], []
        for i in range(len(dataset) - look_back):
            element = dataset[i:(i + look_back), 0]
            data_x.append(element)
            data_y.append(dataset[i + look_back, 0])
        return np.array(data_x), np.array(data_y)
```

Се генерираат податоците за тренирачкото и тестирачкото множество со повик до наведената функција. „Look back“ параметарот го поставуваме на 1, што значи дека секој влезен примерок ќе се состои само од податоци од моменталниот временски чекор и соодветниот излез ќе биде вредноста во следниот чекор.

```
look_back = 1
train_x, train_y = prepare_data(train, look_back)
test_x, test_y = prepare_data(test, look_back)
train_x = np.reshape(train_x, (train_x.shape[0], 1, train_x.shape[1]))
test_x = np.reshape(test_x, (test_x.shape[0], 1, test_x.shape[1]))
```

Се креира секвенцијален модел со LSTM со 100 неврони, на кој се додава Dense слој со 1 излезен неврон. Моделот се компајлира со Mean Absolute Error (MAE) како функција на загуба (ја мери разликата помеѓу предвидената и вистинска вредност) и Adam како оптимајзер. Се поминуваат 300 епохи при што на секои 100 изминати податоци се ажурираат тежините.

```
model_lstm = Sequential()
model_lstm.add(LSTM(100, input_shape=(train_x.shape[1], train_x.shape[2])))
model_lstm.add(Dense(1))
model_lstm.compile(loss='mae', optimizer='adam')
metadata = model_lstm.fit(train_x, train_y, epochs=300, batch_size=100, validation_data =
(test_x, test_y), verbose=0, shuffle=False)
```

Во објектот metadata се сместени податоците околу загубата за време на тренирањето при секоја епоха. Тие може да се прикажат на график.

```
pyplot.plot(metadata.history['loss'], label='train')
pyplot.plot(metadata.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

Се прави предвидувањето на моделот на тест множеството и се прикажува на график за да се направи споредба со вистинските вредности.

```
yhat = model_lstm.predict(test_x)
pyplot.plot(yhat, label='predicted')
pyplot.plot(test_y, label='true')
pyplot.legend()
pyplot.show()
```

На нов график се прикажуваат добиените предвидувања така што на x оска ги имаме датумите, а на y оска предвидените и вистинските цени.

```
dates = dataset.tail(len(test_x)).index
test_y_reshape = test_y_inverse.reshape(len(test_y_inverse))
yhat_reshape = yhat_inverse.reshape(len(yhat_inverse))
actual_prices_chart = graph.Scatter(x=dates, y=test_y_reshape, name= 'Actual Price')
predict_pices_chart = graph.Scatter(x= dates, y=yhat_reshape, name= 'Predict Price')
py.plot([predict_prices_chart, actual_prices_chart])
```

Се креира модел на рекурентна невронска мрежа GRU (Gated Recurrent Unit) со 100 неврони. Се додава „dropout“ слој кој служи за спречување на т.н. overfitting, а доделената вредност 0.2 значи дека 20% од податоците ќе бидат изедначени на 0. На крај

се додава Dense слојот со еден неврон. Моделот се компајлира со Mean Absolute Error како функција на загуба и Adam како оптимајзер и поминува низ учење од 300 епохи.

```
model_gru = Sequential()
model_gru.add(GRU(100, input_shape=(train_x.shape[1], train_x.shape[2])))
model_gru.add(Dropout(0.2))
model_gru.add(Dense(units=1))
model_gru.compile(loss='mae', optimizer='adam')
metadata = model_gru.fit(train_x, train_y, epochs=300, batch_size=100,
validation_data=(test_x, test_y), verbose=0, shuffle=False)

pyplot.plot(metadata.history['loss'], label='train')
pyplot.plot(metadata.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

За крај ја имплементираме функцијата series_to_supervised која го изменува податочното множество за да биде адаптабилно во процесот на надгледувано учење. Тоа се прави со реорганизација на податоците во структура слична на табела каде што секој ред претставува одреден временски чекор. Надгледуваното учење го спроведуваме преку LSTM рекурента мрежа, идентична на првиот модел.

```
def convert_to_supervised(data, n_in=1, n_out=1, dropnan=True):
	n_vars = 1 if type(data) is list else data.shape[1]
	df = pd.DataFrame(data)
	cols, names = list(), list()
	for i in range(n_in, 0, -1):
		cols.append(df.shift(i))
		names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
	for i in range(0, n_out):
		cols.append(df.shift(-i))
		if i == 0:
			names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
		else:
			names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
	ready = pd.concat(cols, axis=1)
	ready.columns = names
	if dropnan:
		ready.dropna(inplace=True)
	return ready
```

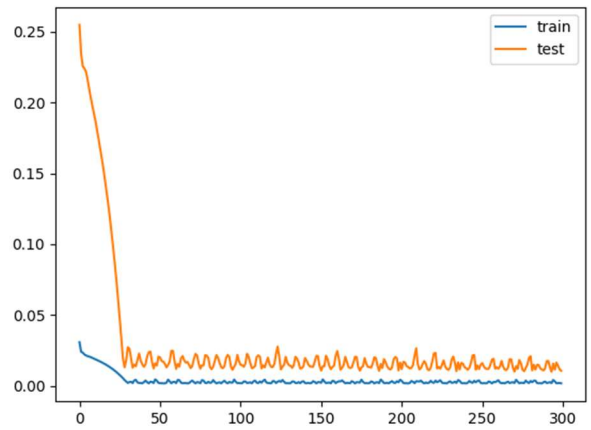
Резултати

Среднoквдратно отстапување (RMSE) е начин да се измери колку добро функционира моделот за машинско учење. Всушност, тоа покажува колку се блиски предвидувањата направени од моделот до вистинските вредности. Колку е помала вредноста на отстапувањето, толку моделот е подобар и предвидените вредности се поблиски до вистинските. Среднoквдратно отстапување се пресметува со следнава формула:

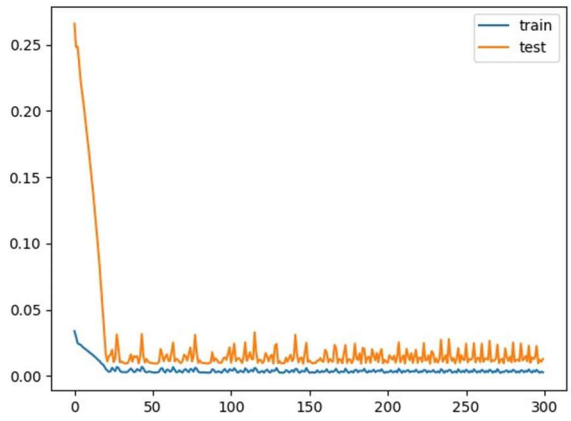
$\sqrt{\frac{\sum(xi-xi*)^2}{n}}$, каде xi е вистинската вредност на *i*-тиот податок, xi* е предвидената вредност на *i*-тиот податок, а n е вкупниот број на податоци.
После тренирање на моделите го пресметуваме средноквадратното отстапување за секој од нив со дадениот код:

```
rmse = sqrt(mean_squared_error(test_y_inverse, yhat_inverse))
```

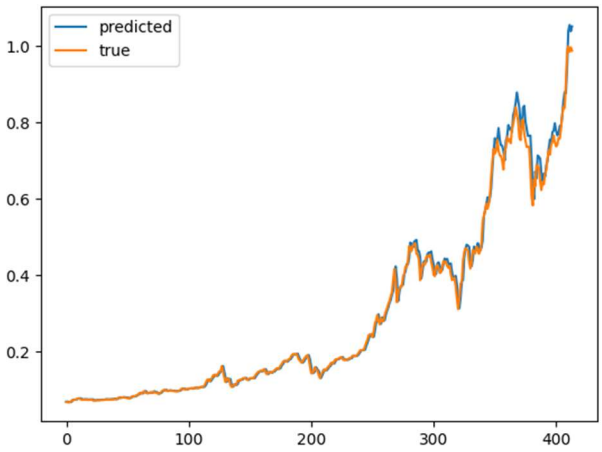
За првиот модел кој користи LSTM тип на рекурентна мрежа, добиваме на екран излез „ RMSE: 128.649“, за вториот GRU модел добиваме излез „ RMSE: 86.152“, и за моделот кој применува надгледувано учење добиваме излез „RMSE: 127.049“. Ова значи дека најдобро предвидување добиваме користејќи го моделот на GRU рекурентна невронска мрежа.
Во прилог дадени се графици за приказ на резултатите добиени со истренираните модели.



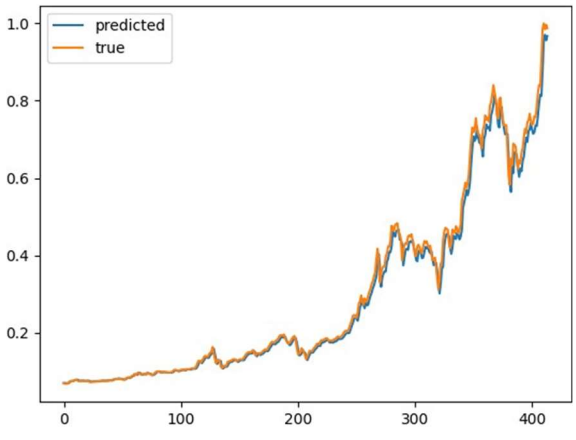
Сл.4а График за функција на загуба од тренирачко и тестирачко множество со LSTM.



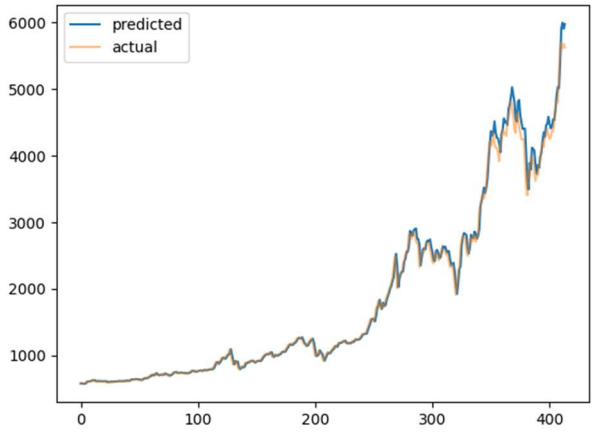
Сл.4б График за функција на загуба од тренирачко и тестирачко множество со GRU.



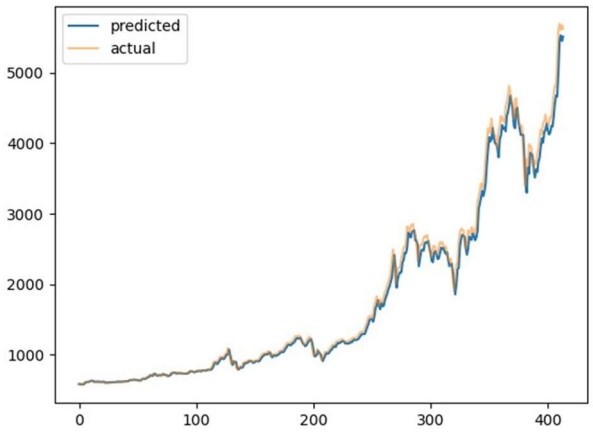
Сл.2а График на предвидени вредности скалирани помеѓу 0 и 1 во однос на епхохите со LSTM.



Сл.2б График на предвидени вредности скалирани помеѓу 0 и 1 во однос на епхохите со GRU.

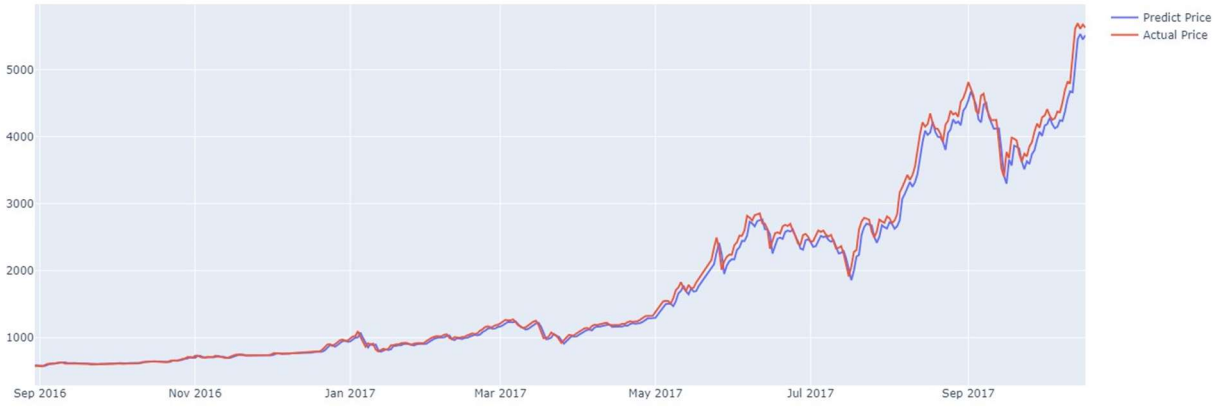


Сл.3а График на нескалирани предвидени и вистински вредности во однос на епхохите со LSTM.

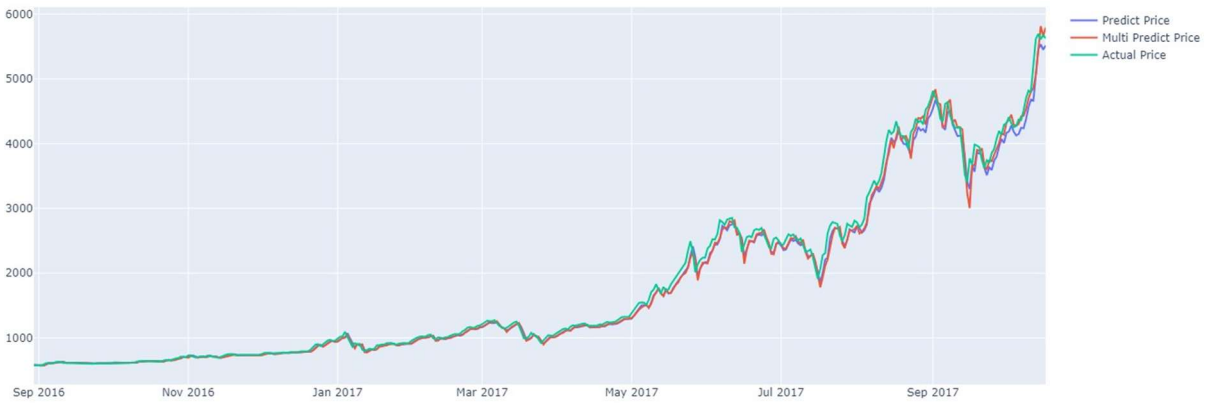


Сл.3б График на нескалирани предвидени и вистински вредности во однос на епхохите со GRU.

Сл.4а График на нескалирани предвидени и вистински вредности во однос на датумите со LSTM.



Сл.4б График на нескалирани предвидени и вистински вредности во однос на датумите со GRU.



Сл.4б Споредба на предвидени и вистински вредности во однос на датумите со надгледувано учење.

Заклучок

Овој проект вклучува 2 модели на рекурентни невронски мрежи - Long Short Term Memory и Gated Recurrent Unit, кои се тренираат за предвидување на цената на биткоин криптовалутата. Моделите кои ние ги имплементиравме се едноставни и учат од мал број на карактеристики кои влијаат врз цената на биткоин, но се значително точни во предвидувањата кои ги вршат. За да се зголеми нивната ефикасност, треба да се земат предвид повеќе карактеристики кои влијаат на цената.

Референци

- Податоци:
  https://coinmarketcap.com/
- Intro to Recurrent Neural Networks LSTM | GRU,
  https://www.kaggle.com/code/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru
- Hemanth Pedamallu (2020, 14 ноември), RNN vs GRU vs LSTM,
  https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573
- Vijaysinh Lendave (2021, 28 август), LSTM Vs GRU in Recurrent Neural Network: A Comparative Study,
  https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/
- Xinchen Zhang, Linghao Zhang, Qincheng Zhou, Xu Jin (2022, 5 мај), A Novel Bitcoin and Gold Prices Prediction Method Using an LSTM-P Neural Network Model,
  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9098287/