Value of cryptocurrencies - What the value of cryptocurrencies depend on? What causes the differences among various currencies?

Pavel Kostarev

Mendelova Univerzita v Brně

June 1, 2020

1 Introduction

Cryptocurrencies have experienced significant increases in value in the last year. But things have looked less clear in the past few months. As there has been talk of the bubble bursting in the stock market, so too has there been talk of a cryptocurrency bubble pop. Is now a good time to assess the value of cryptocurrencies from a more reasoned perspective?

2 Factors that influence the cryptocurrency's value

A cryptocurrency's value is generally derived from its **utility**, **use case** and, ultimately, security, **demand**, among other factors.

2.1 Utility

A coin must have a strong function (or use-case) to incentivize people to hold the coins; a coin's utility is strongly correlated with its value. Let's look at the Ether (ETH) as a case study. In order anyone to execute commands and develop applications in the Ethereum Blockchain one needs to have ETH, as it will be converted into gas and represents the 'fuel' for the Ethereum ecosystem. Therefore, ETH is used as a currency within its system to fuel transactions and development. The more people that execute transactions and develop apps, the greater the demand for ETH and will, therefore, push prices up.

Some utility includes voting rights, dividend payments or most commonly a medium of exchange within their respective ecosystem. A coin without a use is simply speculative in nature that's substantiated without any fundamental value.

2.2 Scarcity

Scarcity refers to the finite nature of the coins. In economics, a fixed supply of a certain item would increase its value in the long term, assuming its demand increases. This creates scarcity, as there is only a limited supply of coins in circulation. For instance, Bitcoin's maximum supply is fixed at 21 million coins. Especially for coins that have a great utility, demand will undeniably push its value up.

Some coins even employ a “burning” mechanism, which refers to the act of destroying a portion of the coin supply. This would increase the value of the coin as there is now a much lesser supply of coins.

2.3 Perceived value

A coin is only as valuable as what the markets deem it to be, and how a project is valued depends on factors that are core to the development of the project. Therefore, projects that persistently achieve their milestones that were set out in their white paper. Additionally, collaborations and partnerships with credible companies or other projects is a good sign of expansion. Other news that can enhance the perceived value in the eyes of the market includes a successful launch of their Minimum Viable Product (MVP) or the beta version of their protocol/software. These are positive indicators of good progress by the project and will enhance the value of their coin. Here we have to answer the question - does it pay off to invest in the currency?

2.4 Liquidity

If a token has a high trading volume and tight spread, this makes it easier to buy and sell at a fair price.

If a token is easier to buy, more people can buy it at any given time. This can be a factor in how accessible a token is.

2.5 Market cap

Market cap = Total Circulating Supply * Price of each coin.

For example, if Ethereum has 200,000 coins circulating on the market with each one worth 3$, the market cap of the crypto would be 200, 000*3=$600,000.
In the same way, if Bitcoin has 100,000 in circulation with each worth $4, the market cap would be 100,000*4= $400,000.
Even though the price of Bitcoin is individually higher, the total value of Ethereum appears much more than Bitcoin. Thus, the index of the coin market cap is a better way to indicate the true price of a cryptocurrency.

3 Differences between cryptocurrencies

Wokring on the project I found a very interesting python library - Cryptory. It is a great tool for wokring with cryptocurrencies dependencies analysis. It makes easy for example to access Stock Prices and Bitcoin prices, plot them with matplotlib and find the correlations.

3.1 Ethereum example

The popularity of Ethereum cryptocurrency is caused by several factors:

- → High security.
- → The speed of transactions that take place in a few seconds.
- → The presence of a minimum commission.
- → Confidentiality (anonymity) for users.
- → Ease of exchange (Ether is included in the listing of the largest exchange platforms)

Market speculations

Dumping - a sharp drop in the value of a virtual coin caused by the sale of a large amount of an asset by one or more market participants. The decrease in the exchange rate during this period can reach 5–10%, which is what traders use.

Pumping is the opposite process when market participants create a stir around cryptocurrency and actively buy Ethereum. In such a situation, the value of a virtual coin grows.

Technology Development and Implementation

Vitalik Buterin and his team improved the system in three years and eliminated many errors. For example, today work is underway to create a second blockchain protocol, which has better characteristics than its predecessor.

In addition, NASA was interested in blockchain technology. The organization plans to use smart contracts in the future to increase the maneuverability of spacecraft in space, as well as improve the connection of astronauts with a base on Earth.

Intrinsic Economic Factors

→ The activity of using virtual coins
→ The difficulty of mining
→ Average number of transactions per day
→ Transactions fee

Network Security Level

For example, in June 2016, when The DAO fund was attacked by attackers. As a result of the attack, it was possible to steal cryptocurrency for $ 50 million. At the same time, the cost of ETH decreased by almost 40% - to $ 13.76. This factor was one of the reasons for the appearance of the Ethereum fork, Ethereum Classic.

News

Different Countries Policies

Countries that support the circulation of cryptocurrencies. This category includes Japan, Sweden, the USA, Estonia, the Netherlands and a number of other states. India is showing increased interest in Ethereum. And if earlier the investors of this country preferred Bitcoin, today the vector of attention is shifting specifically to ETH. In 2018, cryptocurrency interested many investors in India. According to statistics, almost 35% of market participants are interested in this particular virtual coin. For comparison, only 30% of investors show interest in Bitcoin. In third place is Dash, who is interested in just over 5% of respondents.

States that have not decided or banned cryptocurrency. In another "camp" are countries that do not accept the use of virtual currency or are at the stage of reflection. So, virtual money cannot be used in Ecuador, Bolivia, Kyrgyzstan, Bangladesh and other countries. A number of restrictions exist in Vietnam, Thailand, and Iceland. As for the largest “player” (China), the government of the country cannot decide on the fate of the virtual coin. At this stage, the work of exchanges is prohibited, but lately there have been rumors about the future legalization of cryptocurrency.

USD - ETH Price pair since April 2017 - June 1st of 2020 [color lines in the bottom are stocks] candle histogram

Middle of March

Up to the 1st of June period

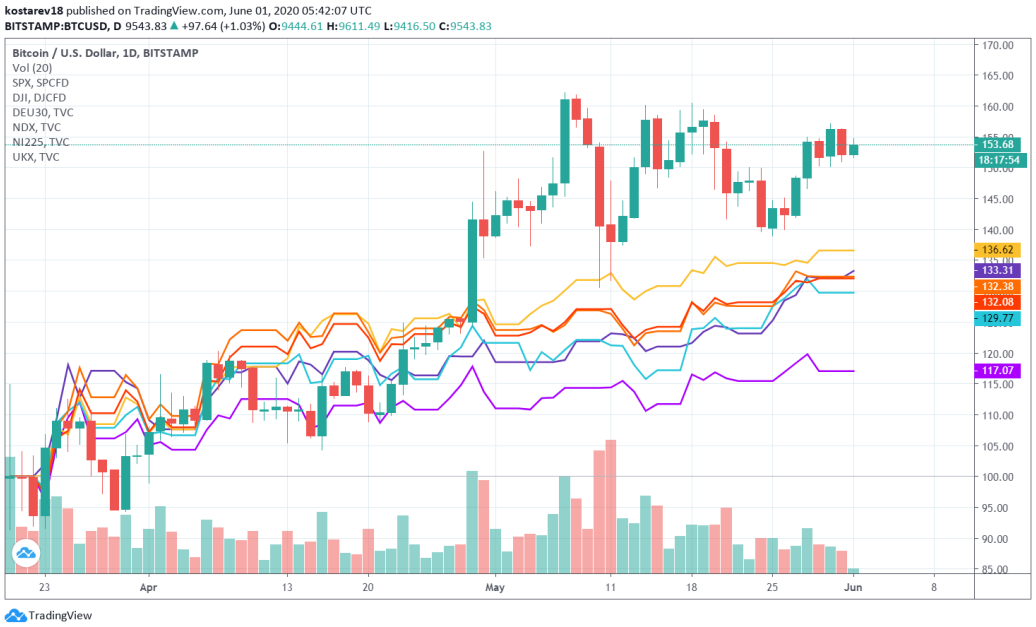
Trading Volume	$12,227,550,214
Volume / Market Cap	0.4615
24h Low / 24h High	$230.90 / $241.75
7d Low / 7d High	$200.96 / $243.04
Market Cap Rank	#2
All-Time High	$1,448.18 -83.5% Jan 13, 2018 (over 2 years)
All-Time Low	$0.432979 55188.6% Oct 20, 2015 (over 4 years)
Ethereum/Bitcoin Ratio	1 BTC = 39.98 ETH

3.2 Bitcoin example

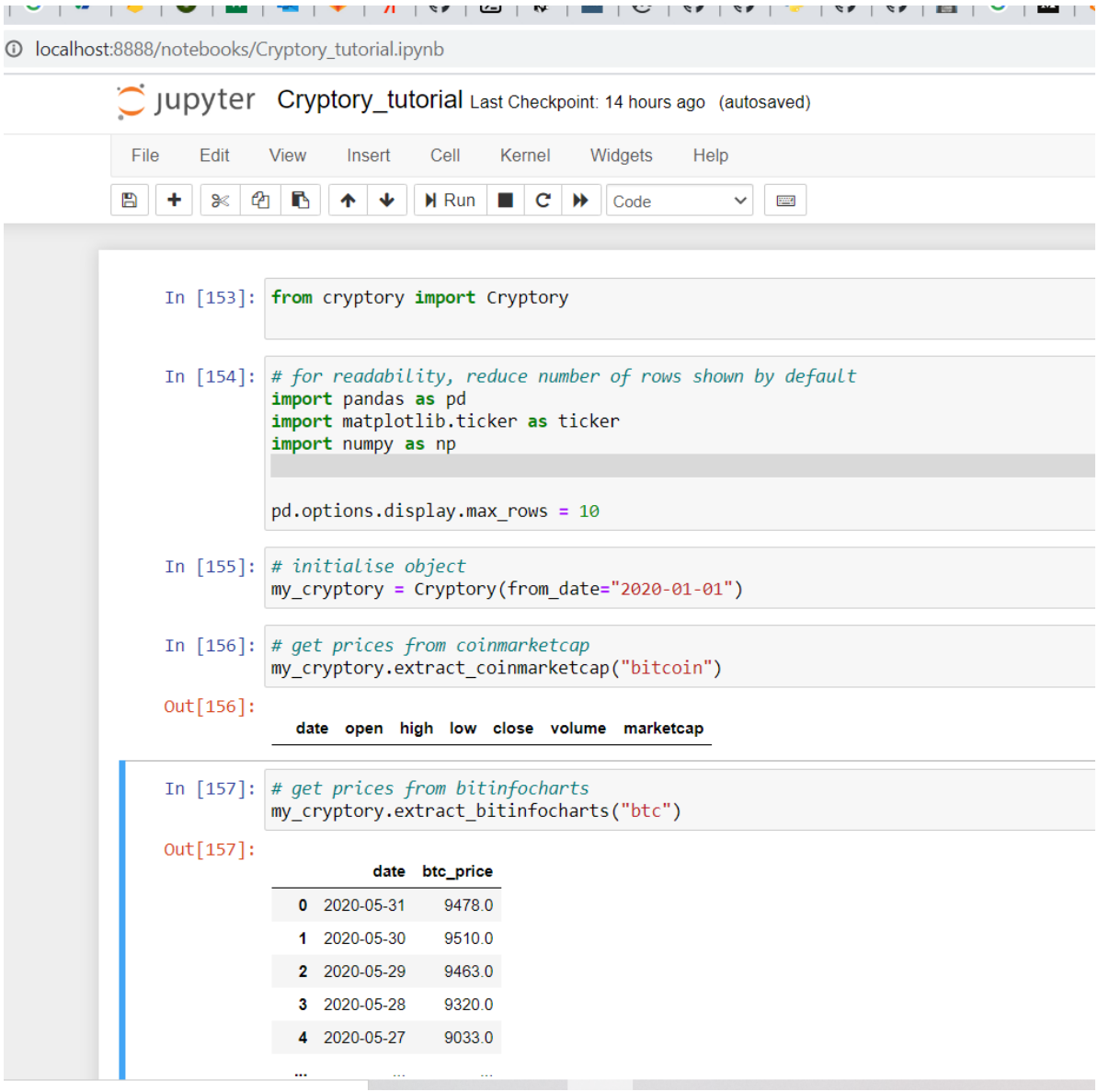
USD - BTC Price pair since April 2017 - June 1st of 2020 [color lines in the bottom are stocks] candle histogram

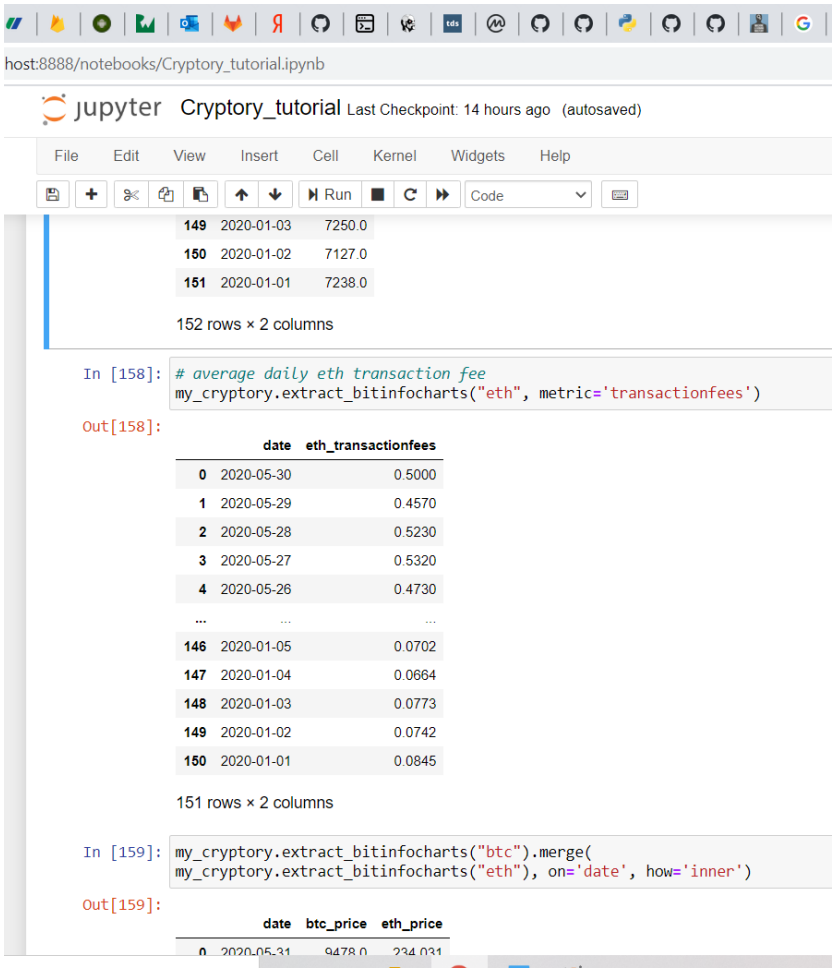
Recession of the economy in the middle of March. Bitcoit showed a strong correlation here with the stock prices, although the prices did not fall so fast as stocks' ones.

Up to the 1st of June



Trading Volume	$27,490,195,912
Volume / Market Cap	0.1577
24h Low / 24h High	$9,416.26 / $9,609.38
7d Low / 7d High	$8,839.13 / $9,678.04
Market Cap Rank	#1
All-Time High	$19,665.39 -51.5% Dec 16, 2017 (over 2 years)
All-Time Low	$67.81 13967.6% Jul 06, 2013 (almost 7 years)
Bitcoin/Bitcoin Ratio	1 BTC = 1.0 BTC


localhost:8888/notebooks/Cryptory_tutorial.ipynb
jupyter Cryptory_tutorial Last Checkpoint: 14 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [153]:
from cryptory import Cryptory
In [154]:
# for readability, reduce number of rows shown by default
import pandas as pd
import matplotlib.ticker as ticker
import numpy as np
pd.options.display.max_rows = 10
In [155]:
# initialise object
my_cryptory = Cryptory(from_date="2020-01-01")
In [156]:
# get prices from coinmarketcap
my_cryptory.extract_coinmarketcap("bitcoin")
Out[156]:
date open high low close volume marketcap
In [157]:
# get prices from bitinfocharts
my_cryptory.extract_bitinfocharts("btc")
Out[157]:
date btc_price
0 2020-05-31 9478.0
1 2020-05-30 9510.0
2 2020-05-29 9463.0
3 2020-05-28 9320.0
4 2020-05-27 9033.0



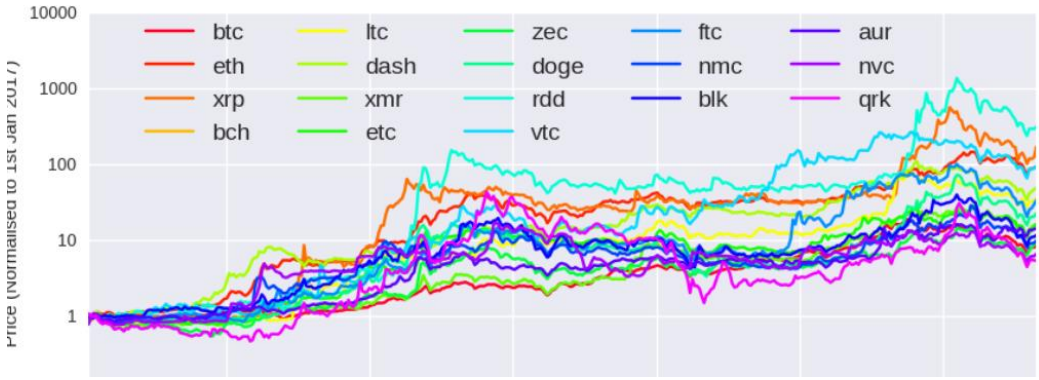
152 rows × 2 columns

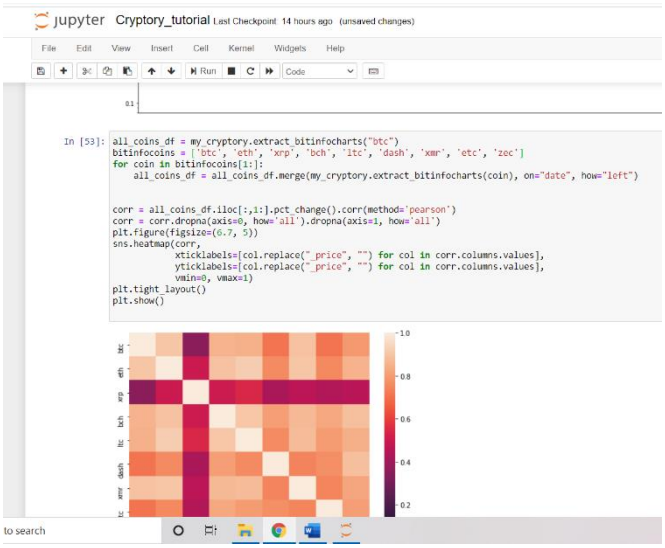
In [7]:
```
my_cryptory.extract_bitinfocharts("btc").merge(
my_cryptory.extract_bitinfocharts("eth"), on='date', how='inner')
```

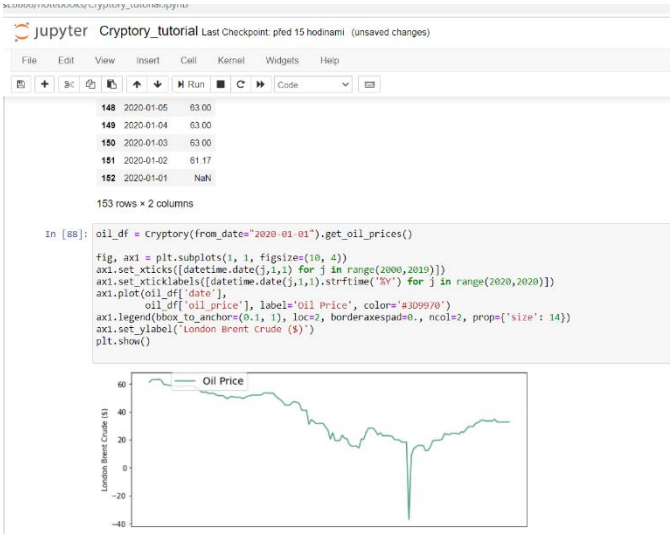
Out[7]:

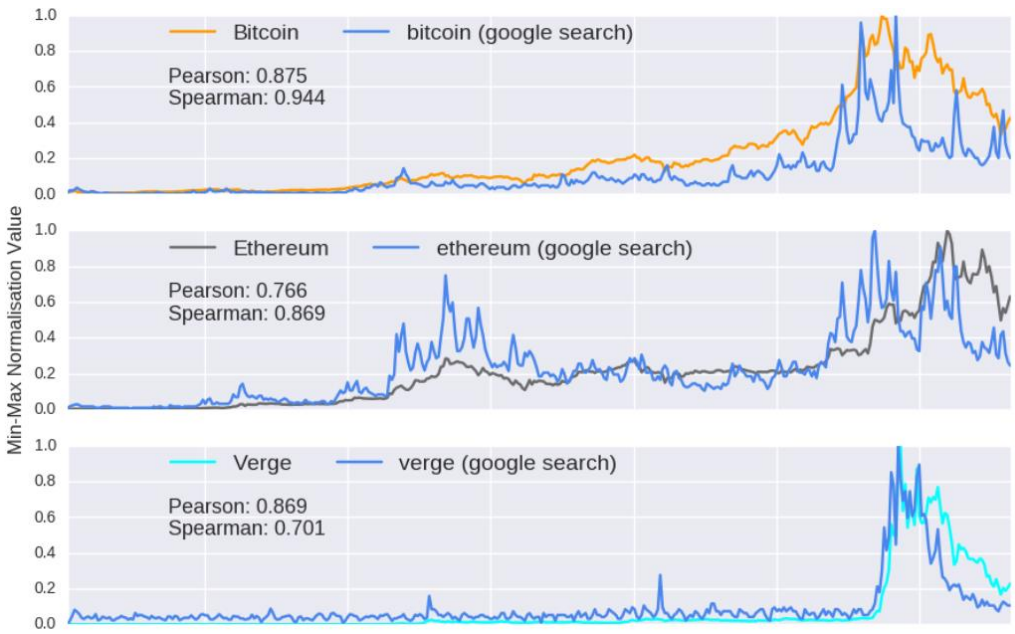
	date	btc_price	eth_price
0	2020-06-01	9555.0	239.104
1	2020-05-31	9558.0	237.985
2	2020-05-30	9510.0	232.733
3	2020-05-29	9463.0	220.948
4	2020-05-28	9320.0	210.438
...	...	...	...
148	2020-01-05	7503.0	137.648
149	2020-01-04	7390.0	134.662
150	2020-01-03	7250.0	131.522
151	2020-01-02	7127.0	129.500
152	2020-01-01	7238.0	131.352

153 rows × 3 columns


Price (Normalised to 1st Jan 2017)
10000
1000
100
10
1
btc
eth
xrp
bch
ltc
dash
xmr
etc
zec
doge
rdd
vtc
ftc
nmc
blk
aur
nvc
qrk


jupyter Cryptory_tutorial Last Checkpoint 14 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
In [53]:
all_coins_df = my_cryptory.extract_bitinfocharts("btc")
bitinfocoins = ['btc', 'eth', 'xrp', 'bch', 'ltc', 'dash', 'xmr', 'etc', 'zec']
for coin in bitinfocoins[1:]:
all_coins_df = all_coins_df.merge(my_cryptory.extract_bitinfocharts(coin), on="date", how="left")
corr = all_coins_df.iloc[:,1:].pct_change().corr(method='pearson')
corr = corr.dropna(axis=0, how='all').dropna(axis=1, how='all')
plt.figure(figsize=(6.7, 5))
sns.heatmap(corr,
xticklabels=[col.replace("_price", "") for col in corr.columns.values],
yticklabels=[col.replace("_price", "") for col in corr.columns.values],
vmin=0, vmax=1)
plt.tight_layout()
plt.show()
btc
eth
xrp
bch
ltc
dash
xmr
1.0
0.8
0.6
0.4
0.2
to search


jupyter Cryptory_tutorial Last Checkpoint: pfed 15 hodinami (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
148 2020-01-05 63.00
149 2020-01-04 63.00
150 2020-01-03 63.00
151 2020-01-02 61.17
152 2020-01-01 NaN
153 rows × 2 columns
In [88]:
oil_df = Cryptory(from_date="2020-01-01").get_oil_prices()
fig, ax1 = plt.subplots(1, 1, figsize=(10, 4))
ax1.set_xticks([datetime.date(j,1,1) for j in range(2000,2019)])
ax1.set_xticklabels([datetime.date(j,1,1).strftime('%Y') for j in range(2020,2020)])
ax1.plot(oil_df['date'],
oil_df['oil_price'], label='Oil Price', color='#3D9970')
ax1.legend(bbox_to_anchor=(0.1, 1), loc=2, borderaxespad=0., ncol=2, prop={'size': 14})
ax1.set_ylabel('London Brent Crude ($)')
plt.show()
Oil Price
London Brent Crude ($)
60
40
20
0
−20
−40


Min-Max Normalisation Value
Bitcoin
bitcoin (google search)
Pearson: 0.875
Spearman: 0.944
Ethereum
ethereum (google search)
Pearson: 0.766
Spearman: 0.869
Verge
verge (google search)
Pearson: 0.869
Spearman: 0.701
1.0
0.8
0.6
0.4
0.2
0.0

4 Methodology

4.1 Social perspective

Bitcoin example

The structure of the Bitcoin user community (in percent) is as follows

- financial services, investment services - 8.14;
- software / design - 3.72;
- financial, banking services - 3.30;
- employment center - 2.76;
- consumer electronics / mobile phones - 2.70;
- computers and peripherals - 2.65;
- dating service - 2.14;
- travel / hotels - 2.12;
- real estate / residential property - 1.72;
- software / business - 1.71.

In terms of gender, among bitcoin users the proportion of men prevails and amounts to 96.57%, while the proportion of women is only 3.43%

Age	% (percentage)
18-24	8.36
24-34	45.71
35-44	30.62
45-54	12.30
55-64	1.83
65+	1.19
Total	100.0

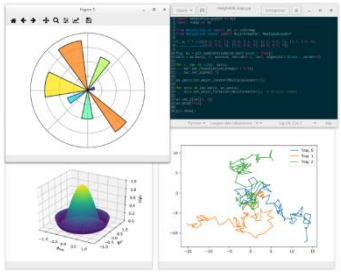
Obviously, the most active are two population groups at the age from 24 to 34 years old and from 35 to 44 years old, both groups come to 76.33%.

4.2 SWOT analysis

Internal factors	
Strengths → high growth potential → considered as an investment instrument with a high profitability → pseudo-anonymity of transactions; → deflationary model (Bitcoin), i.e. limited emissions (21 mil BTC)	**Weaknesses** → high volatility → it is impossible to cancel a transaction (comparing with simple banks) → if a "Bitcoin wallet" was hacked, then they cannot be returned can; → not everywhere can pay with cryptocurrency; → complex scalable system (Bitcoin - 7 operations per second, payment systems - up to 500 operations per second)
Opportunities → Payments will drop sharply. Distribution of cryptocurrencies make people to reject from having to pay commissions to payment systems, card issuing banks, acquiring banks and other participants of financial transactions. → Micropayments will be possible (e.g. amounts less than 5-10 cents), and all this without the use of cash; → No need for tax services as fees will be already calculated accurately and automaticaly	**Threats** → Threat to the "reputation" of central banks; → The possibility of using money laundering, criminalism and financial terrorism; → Threat to classic commercial organizations (bank institutions), necessity for them can disappear → There is no regulatory body for controlling the movements of the cryptocurrency capital
External factors	

5 Predicting the value

5.1 Getting started - importing the libraries and getting the dataset (STEP 1)



What will we need for the project?

1. **Python 3.6**

I learned how to use the next libraries which are essential for the Data Science, Data Analysis or using the Python for the Machine Learning

- KERAS - it is a very powerful library that helps us to work with the neural networks as part of the machine learning and deep learning. In this example we will use Recurrent Neural Networks (RNN), it allows us to write a neural network just in few lines of code.
- TENSORFLOW AND TENSORBOARD AS PART OF KERAS LIBRARY - it contains the most useful instruments for the deeplearning, for instance, Sequence model, architectures like LSTM or CuDNNLSTM (for tensorflow-gpu)
- NUMPY - for working with arrays, matrices ets.

- PANDAS - (basically data-preprocessing), for example, time series, reshaping the datasets, manipulations with dataframes, data merging and joining, spliting
- SKLEARN - linear algebra/array operations etc.
- MATPLOTLIB and SEABORN for the plots and graphics (provides matlab-alike interface for working with plots)

2. Anaconda environment - Jupyter : I used the Anaconda python interpreter and Jupyter notebook, it is convenient for making the plots and graphics
3. PyCharm - as Anaconda alternative.
4. Rstudio+Excel: I made the data visualisation in the Rstudio(tidyverse, ggplot2 packages), Excel was needed for working with .csv files and converting the timestamps into the data format

```
import pandas as pd
import os
from sklearn import preprocessing
from collections import deque
import random
import numpy as np
import time
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Dropout, LSTM, CuD-
NNLSTM, BatchNormalization
from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' #This is for avoiding the er-
ror message I got while running the main.py
```

This is data preprocessing part. Here we have to reshape our data, normalize it, get rid of NA values, splitting the data into the sets (training and validation in our case), form the arrays for training and validation data as well as shuffle our data, better multiple times.

And here we always have to define the constants we are going to use later, such as. We will analyse last 60 minutes and will try to predict the value of cryptocurrency for the next 3 ones:

```
#CONSTANTS

SEQ_LEN = 60 #USING LAST 60 MIN
```

```
FUTURE_PERIOD_PREDICT = 3 #PREDICTING NEXT 3 MIN
RATIO_TO_PREDICT = "BTC-USD"
EPOCHS = 10 #10
BATCH_SIZE = 30 #64
NAME = f"{RATIO_TO_PREDICT}-{SEQ_LEN}-SEQ-{FUTURE_PERIOD_PREDICT}-
PRED-{int(time.time())}"
```

What is Epoch = number of iterations for training the data
What is Batch Size = number of training examples utilized in one iteration
We use f strings for python to have unique names while running the file to remember which model and ratio we used

→**Close Price** — It is the market close price for currency for that particular day.

→**High Price** — It is highest price of currency for the day.

→**Low Price** — It is the lowest price for currency for that day.

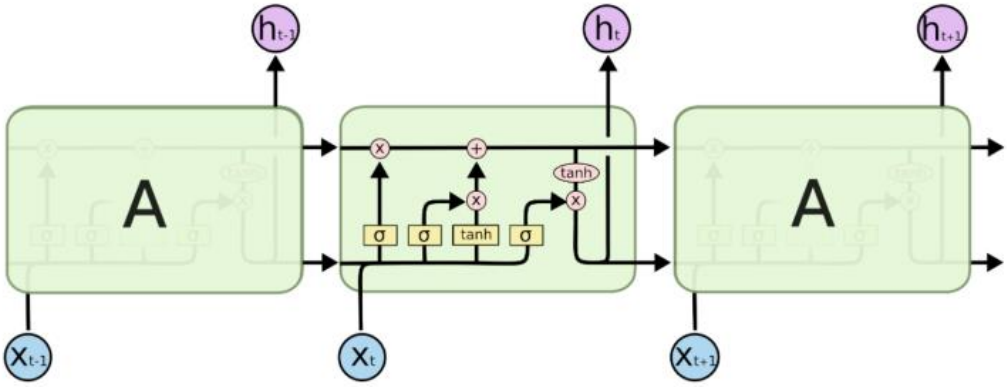
→**Open Price** — It is market open price for currency for that day.

→**Volume** — The volume of currency that is being in trade for that day.

5.2 What is LSTM and why do we use RNN?

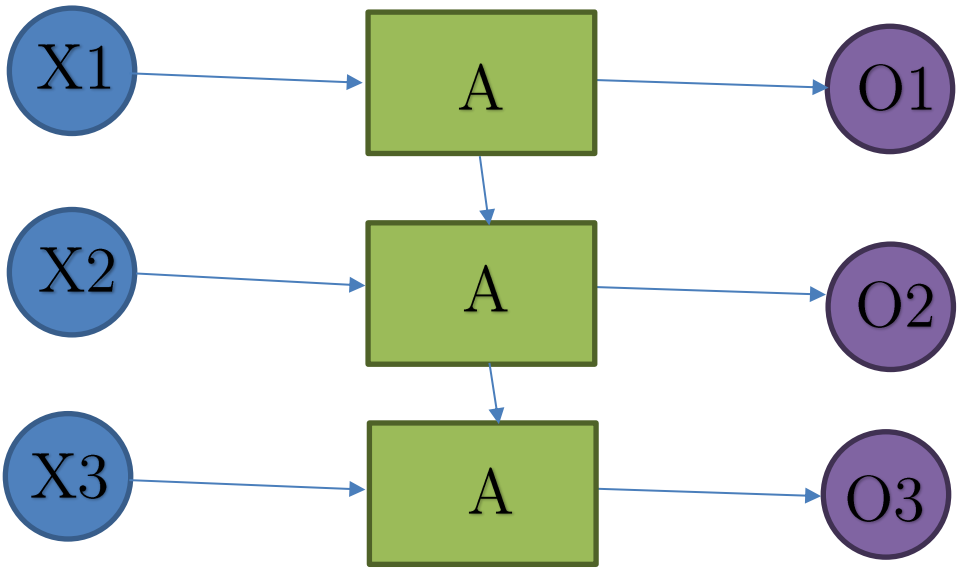
Why not colvolutional or forward neural networks??

As we are working with time and will have time series, sequences of data we have to remember, it is beneficial for us to use RNN, because it has exactly these feautures

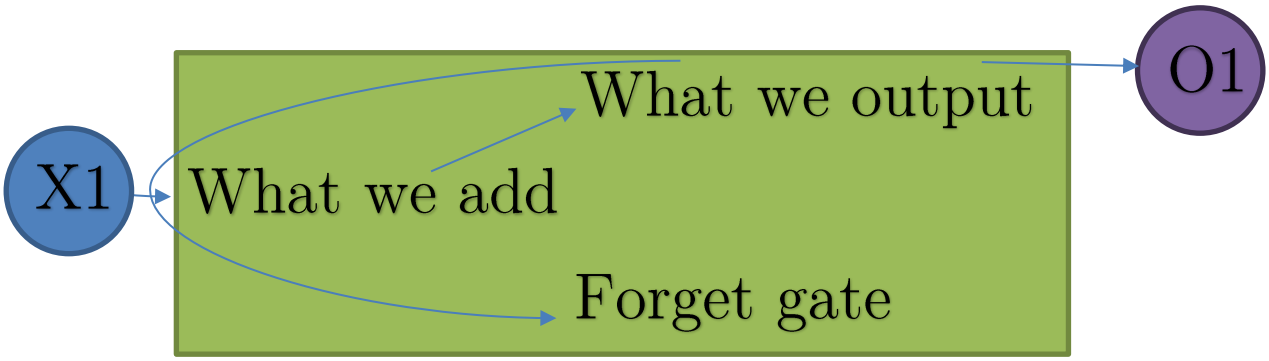


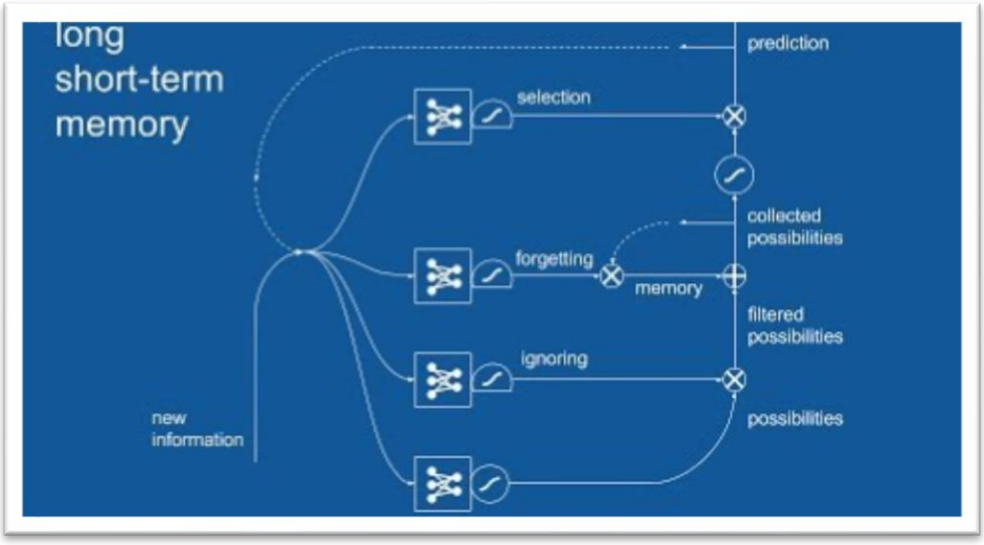
The repeating module in an LSTM contains four interacting layers.

Recurrent Network Idea

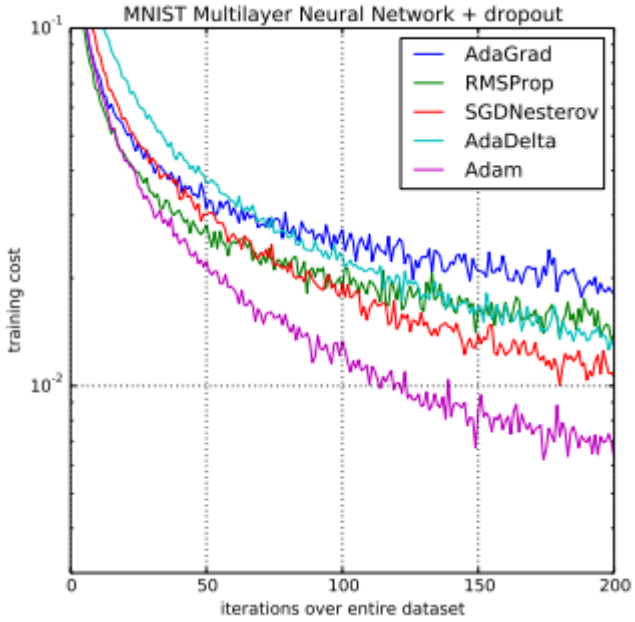

X1
A
O1
X2
A
O2
X3
A
O3

LSTM Idea


X1
What we output
What we add
Forget gate
O1


long
short-term
memory
prediction
selection
collected
possibilities
forgetting
memory
filtered
possibilities
ignoring
new
information
possibilities

We use Adam function for the loss. It has the best performance (Adaptive Moment Estimation)



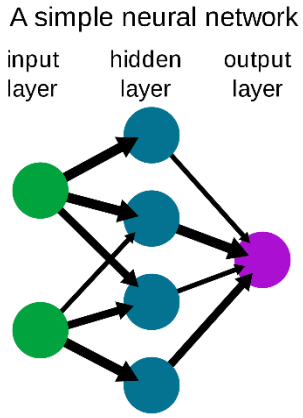

```
def classify(current, future):
    if float(future) > float(current):
        return 1 #if price is higher -> we should buy it
    else:
        return 0

def preprocess_df(df):
    df = df.drop('future',1)

    for col in df.columns: # go through all of the columns

        if col != "target": # normalize all ... except for the target
itself!

            #Currency fluctations
            df[col] = df[col].pct_change() # pct change "normalizes"
the different currencies (each crypto coin has vastly diff values,
we're really more interested in the other coin's movements)

            #getting rid of NA values
            df.dropna(inplace=True)
            #scaling the value by keras
            df[col] = preprocessing.scale(df[col].values) # scale be-
tween 0 and 1.


    df.dropna(inplace=True)
```

```
    sequential_data = [] # this is a list that will CONTAIN the se-
quences

    prev_days = deque(maxlen=SEQ_LEN) # These will be our actual se-
quences. They are made with deque, which keeps the maximum length by
popping out older values as new ones come in

    for i in df.values: # iterate over the values

        prev_days.append([n for n in i[:-1]]) # store all but the tar-
get

        if len(prev_days) == SEQ_LEN: # make sure we have 60 se-
quences!

            sequential_data.append([np.array(prev_days), i[-1]])
    random.shuffle(sequential_data)

    #balancing the data
    buys = [] # list that will store our buy sequences and targets

    sells = [] # list that will store our sell sequences and targets


    for seq, target in sequential_data: # iterate over the sequential
data

        if target == 0:
            sells.append([seq,target]) # if it's a "not buy"  append
to sells list

        elif target == 1: # otherwise if the target is a 1..(buy case)
            buys.append([seq,target])

    random.shuffle(buys) # shuffle for the good measure

    random.shuffle(sells)

    lower = min(len(buys),len(sells))
    #what is the shorter length?
    #30K
    buys = buys[:lower] #up tp lower
    sells = sells[:lower] # make sure both lists are only up to the
shortest length.

    sequential_data = buys + sells # add them together
    random.shuffle(sequential_data) # another shuffle, so the model
doesn't get confused with all 1 class then the other.

    X = []
    y = []

    for seq, target in sequential_data: # going over our new sequen-
tial data
```

```
        X.append(seq) # X is the sequences

        y.append(target) # y is the targets/labels (buys vs sell/not-
buy)


    return np.array(X), y # return X and y...and make X a numpy array!


main_df = pd.DataFrame() # begin with an empty df


ratios = ["BTC-USD","LTC-USD","ETH-USD","BCH-USD"]

# the 4 ratios we want to consider

for ratio in ratios:
     dataset = f"D:\crypto_data\crypto_data\{ratio}.csv"

     df = pd.read_csv(da-
taset,names=["time","low","high","open","close","volume"])

# split away the ticker from the file-name
    # rename volume and close to include the ticker so we can still
which close/volume is which

     df.rename(columns={"close": f"{ratio}_close", "volume": f"{ra-
tio}_volume"}, inplace=True)
#WE PUT ONLY THE CLOSE VALUE AND VOLUME INTO THE DATA FRAME, REDUCING
THE OTHER COLUMNS
     df.set_index("time",inplace=True)

# set time as index so we can join them on this shared time

     df = df[[f"{ratio}_close",f"{ratio}_volume"]]

# ignore the other columns besides price and volume

     if len(main_df) == 0:
         main_df = df
     else:
         main_df = main_df.join(df) # otherwise, join this data to the
main one

main_df.fillna(method="ffill", inplace=True)  # if there are gaps in
data, use previously known values (current ones)
main_df.dropna(inplace=True) #NEW

main_df['future'] = main_df[f"{RATIO_TO_PREDICT}_close"].shift(-FU-
TURE_PERIOD_PREDICT)
main_df['target'] = list(map(classify,main_df[f"{RATIO_TO_PRE-
DICT}_close"],main_df["future"]))
print(main_df[[f"{RATIO_TO_PREDICT}_close", "future",
```

```
"target"]].head())
main_df.dropna(inplace=True) #NEW

#we have to separate out of sample data - part of normalizing
times = sorted(main_df.index.values)
last_5pct = times[-int(0.05*len(times))]
print(last_5pct)

validation_main_df = main_df[(main_df.index >= last_5pct)]
main_df = main_df[(main_df.index < last_5pct)]

#lets make sequences
#preprocess_df(main_df)
train_x, train_y = preprocess_df(main_df)
validation_x, validation_y = preprocess_df(validation_main_df)

print(f"train data: {len(train_x)} validation: {len(validation_x)}")
print(f"Dont buys: {train_y.count(0)}, buys: {train_y.count(1)}")
print(f"VALIDATION Dont buys: {validation_y.count(0)}, buys: {valida-
tion_y.count(1)}")
```

5.3 Building the model



- → What is Sequential?
- → What does return sequences mean?
- → What does dropout mean?
- → What is Batch Normalization?

This is the first layer → input

```
model = Sequential()
model.add(LSTM(128, input_shape=(train_x.shape[1:]), return_se-
quences=True)) #128 train shape 1:a
model.add(Dropout(0.2)) #0.2
model.add(BatchNormalization())
```

This is the second layer → hidden layer

```
model.add(LSTM(128, return_sequences=True)) #128
model.add(Dropout(0.1)) #0.1
model.add(BatchNormalization())
```

This is the third layer → hidden layer

```
model.add(LSTM(128)) #128
model.add(Dropout(0.2)) #0.2
model.add(BatchNormalization())
```

```
model.add(Dense(32, activation='relu')) #32
model.add(Dropout(0.2))
```

Here we use rectified linear activation function

```
model.add(Dense(2, activation='softmax')) #2 softmax

opt = tf.keras.optimizers.Adam(lr=0.001,decay=1e-6)
```

lr is a learning rate and decay can be explained as a time of step (the smaller steps we make, the better is the performance)

```
model.compile(loss='sparse_categorical_crossentropy', #sparse_categor-
ical_crossentropy
              optimizer=opt, #opt
              metrics=['acc'])

tensorboard = TensorBoard(log_dir="D:\crypto_data\logs\{}".for-
mat(NAME))

filepath = "RNN_Final-{epoch:02d}-{val_acc:.3f}"  # unique file name
that will include the epoch and the validation acc for that epoch
checkpoint = ModelCheckpoint("D:\crypto_data\models\{}.model".for-
mat(filepath, monitor='val_acc', verbose=1, save_best_only=True,
mode='max')) # saves only the best ones

train_x = np.asarray(train_x)
train_y = np.asarray(train_y)
validation_x = np.asarray(validation_x)
validation_y = np.asarray(validation_y)
```

5.4 Training and fitting the model

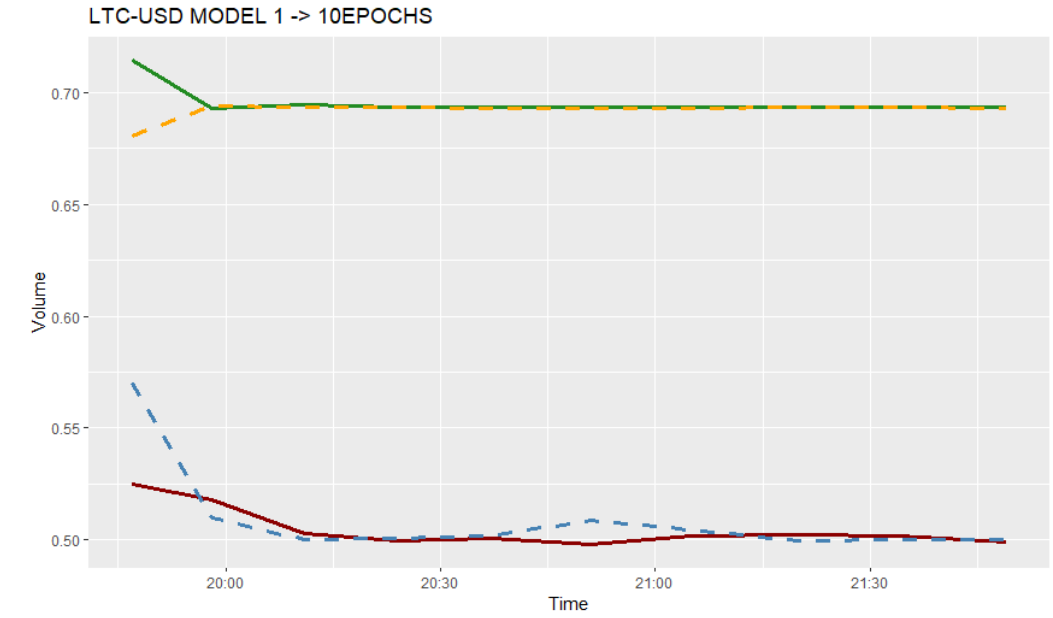
→ Activation, loss and optimizer functions

Which optimizer to pick?

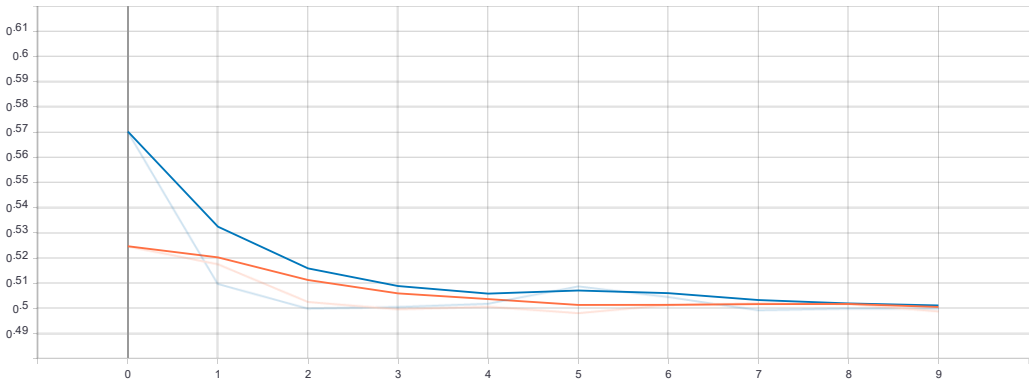
```
history = model.fit(
    train_x, train_y,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=(validation_x,validation_y),
    callbacks=[tensorboard,checkpoint])

# Score model
score = model.evaluate(validation_x, validation_y, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

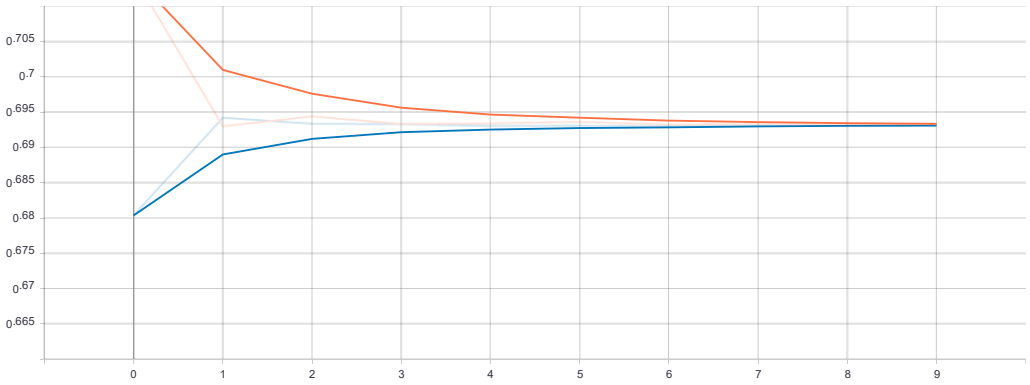
```
# Save model
model.save("D:\crypto_data\models\{}".format(NAME))
```

5.5 Conclusion

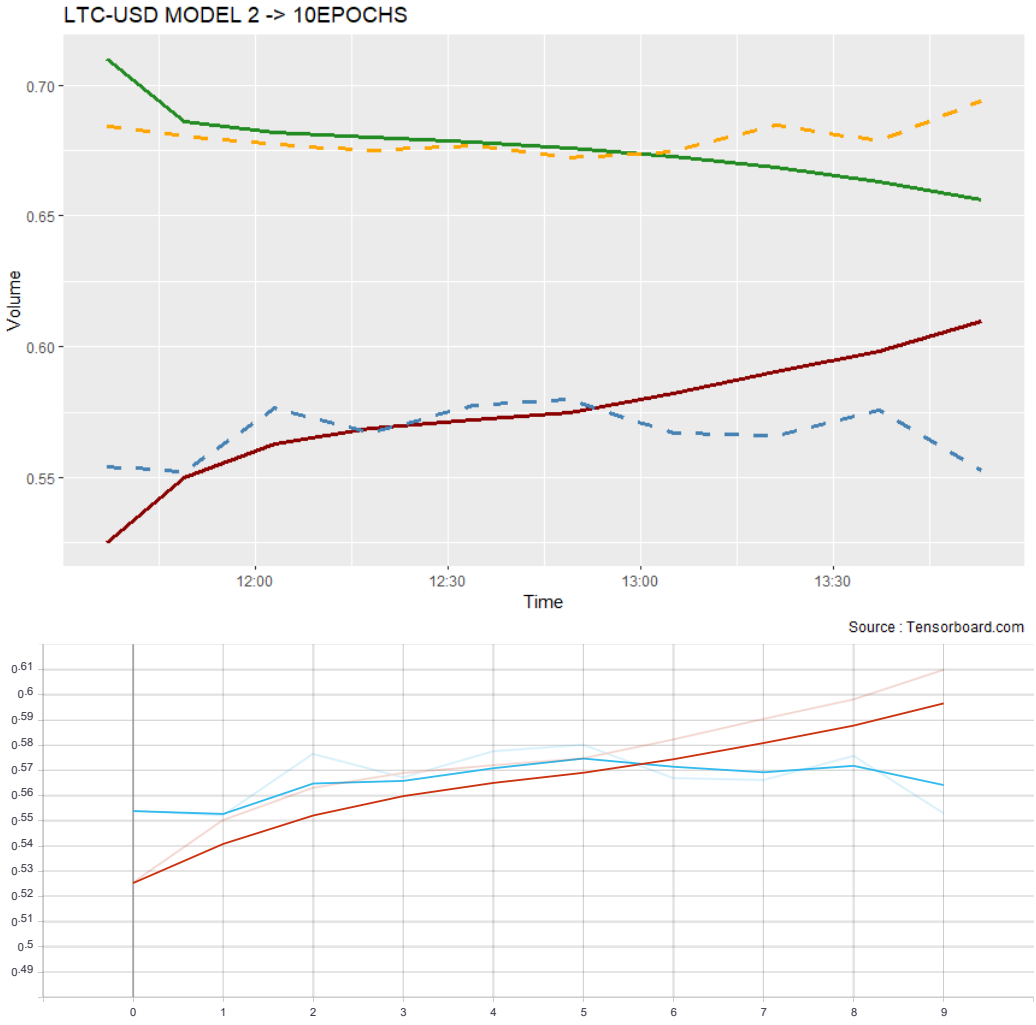


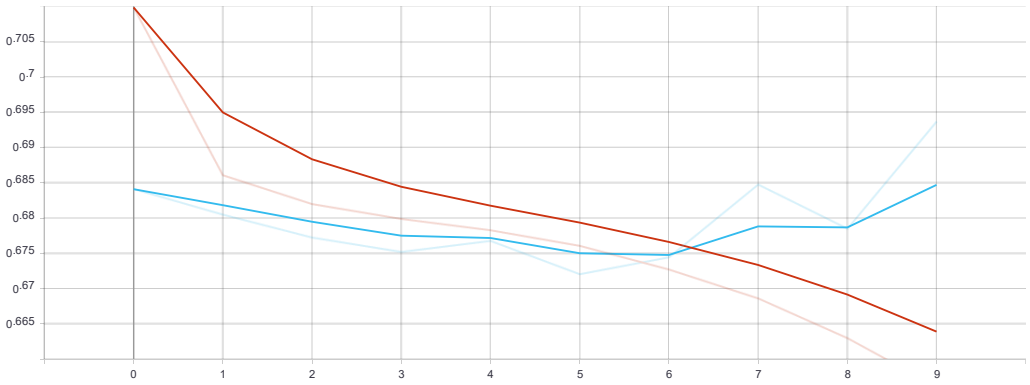
Source : Tensorboard.com



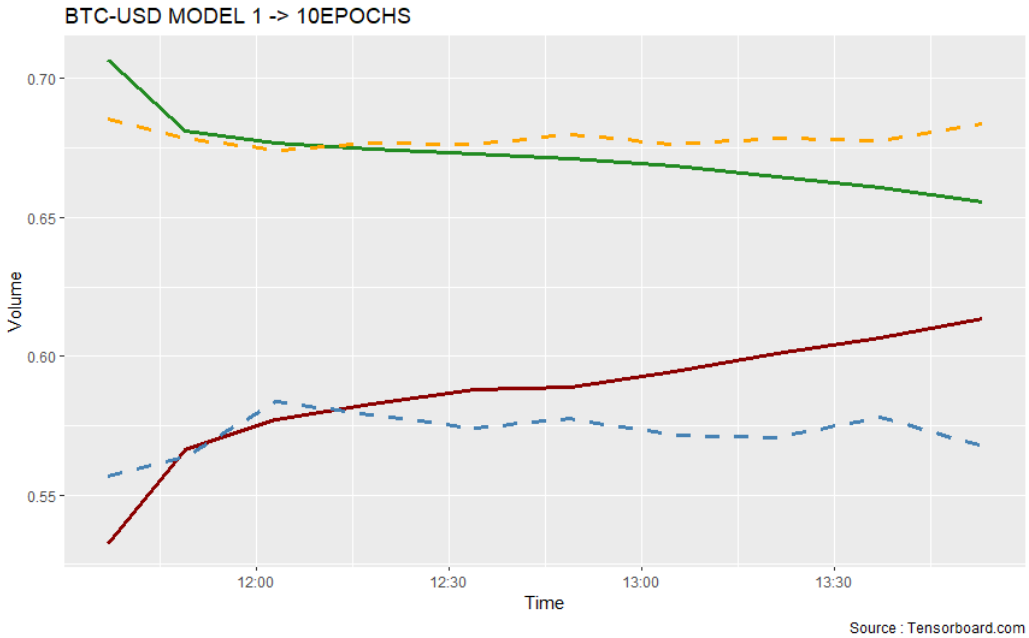


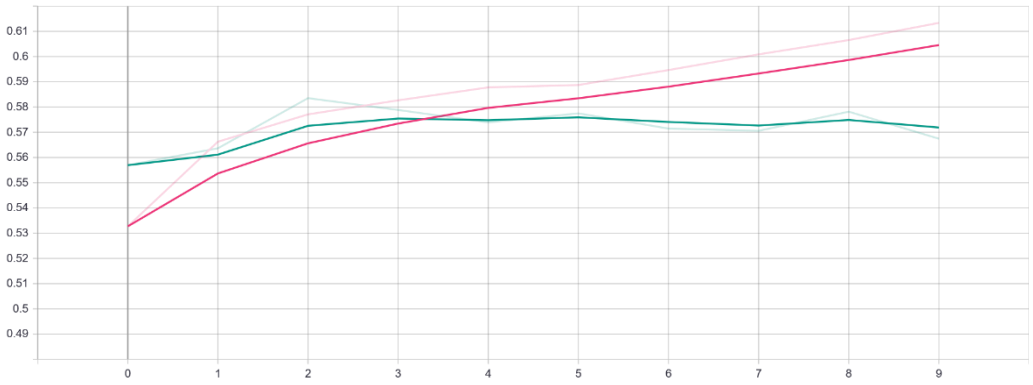
Time	Steps	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
19:47	0	0.524686	0.570216	0.714342	0.680371
19:58	1	0.517619	0.509798	0.69299	0.694194
20:11	2	0.502659	0.5	0.69438	0.693341
20:24	3	0.499725	0.500653	0.693313	0.693275
20:37	4	0.500766	0.50196	0.693392	0.693028
20:51	5	0.498179	0.508818	0.693608	0.693051
21:05	6	0.50146	0.504572	0.693216	0.69299
21:20	7	0.502154	0.499347	0.693286	0.693192
21:34	8	0.501836	0.5	0.693227	0.693186
21:49	9	0.498844	0.5	0.693232	0.693154

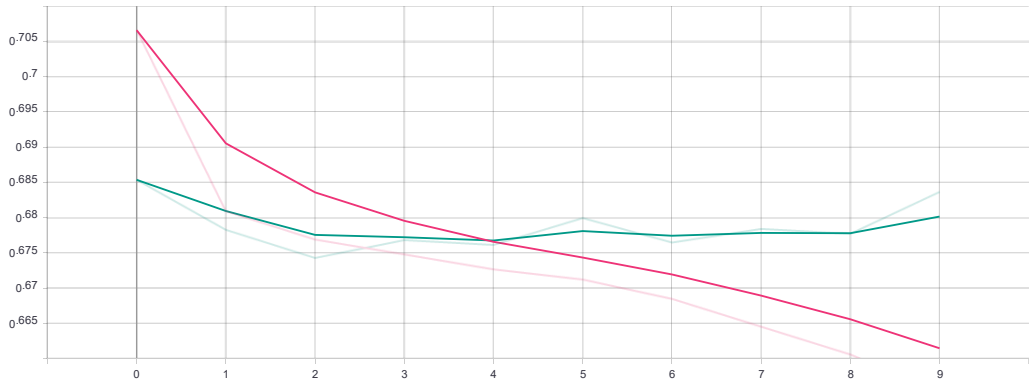




Time	Steps	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
11:37	0	0.52532	0.553886	0.709859	0.684099
11:49	1	0.550166	0.552073	0.686055	0.680473
12:03	2	0.562986	0.576425	0.681964	0.677227
12:18	3	0.568877	0.567098	0.679862	0.675188
12:34	4	0.571905	0.577461	0.678259	0.676747
12:49	5	0.57469	0.580052	0.676048	0.672031
13:05	6	0.582133	0.566839	0.672692	0.674423
13:21	7	0.590295	0.566062	0.668602	0.684728
13:37	8	0.598021	0.575648	0.662974	0.678476
13:53	9	0.609751	0.55285	0.656101	0.693667

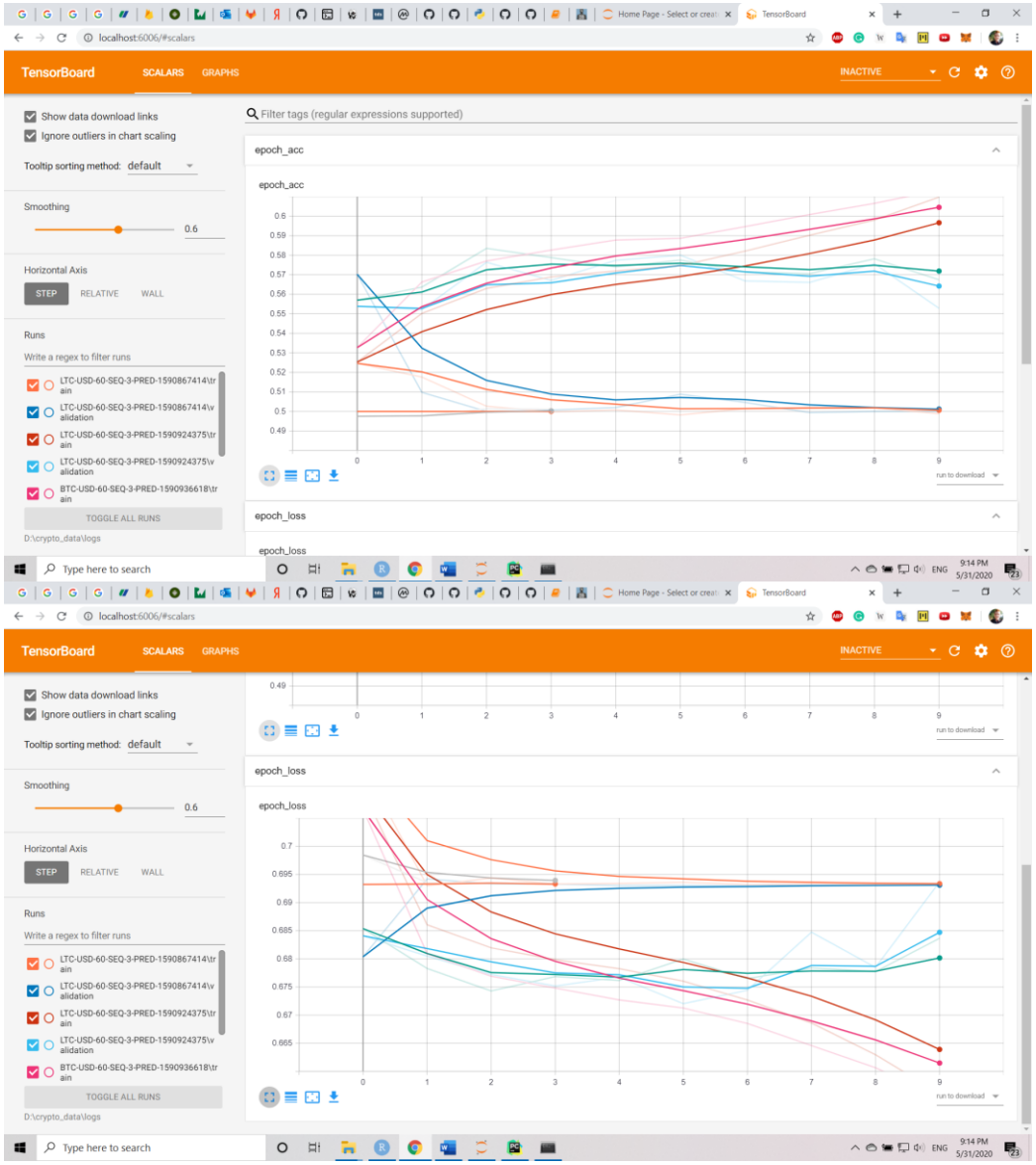






Time	Steps	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
14:58	0	0.53271	0.556945	0.706582	0.68537
15:07	1	0.566249	0.563644	0.680919	0.678297
15:17	2	0.577084	0.583519	0.676921	0.674312
15:27	3	0.582652	0.57883	0.674815	0.676839
15:38	4	0.587775	0.573917	0.672702	0.676159
15:49	5	0.588725	0.57749	0.671262	0.679958
16:01	6	0.594653	0.57146	0.668525	0.676498
16:12	7	0.600871	0.570567	0.664583	0.678427
16:24	8	0.606535	0.57816	0.660655	0.677743
16:36	9	0.613341	0.567441	0.655382	0.683665

All together models presented -> BTC USD BY FAR WAS THE MOST EFFICIENT ONE



Which problems did I face? How to improve the model?

The very first model I have created works very wrong, then I found out that I forgot to remove the NA values, that is why the data was not normalized properly.

In the second model I changed the batch size to 30, reduced the number of neurons, because my computer did not have enough CPU resources to evaluate the good results.

In the third model I decided to change the ratio and the new dataset was bigger than the previous one that affected on the train efficiency relatively much

In the fourth model I have increased the learning rate trying to fit the model and reduced the number of EPOCHS to 4, I also changed the IDE to Jupyter notebook, it affected to the speed of training the network.

What are the disadvantages?

I am using simple LTSM, that uses tensorflow-cpu, that is much slower than the tensorflow-gpu package, I have installed NVIDIA Dev-Kit, it helped to solve the problems with LTSM, as I did not have required drivers, but I still did not find the solution for running the CuDNNLSTM on my laptop.

Therefore if we take under consideration the time of training, it becomes very consuming part and is even delayed, so the future price is not actual.

Algorithm still has pretty high loss

What did I discover
The bigger number of neurons gave me the less loss

6 Conclusion

//
In the process of researching the experience of cryptocurrency regulation in foreign countries, we found
some similarities. First, each country seeks to create a favorable climate for development.
latest technologies (blockchain) and sees the high potential for using technology not only in
private but also public sector.
Secondly, the cryptocurrency market is growing rapidly.
and the state, without adapting the tax code to the modern challenges of the digital economy,

Gains revenue to the budget, since cryptocurrencies are outside legal jurisdiction. Thirdly, on

today there are more than 1 thousand different cryptocurrencies, therefore, it is necessary

develop common standards for regulating cryptocurrencies.

//Using Machine Learning

Using LTSM algorithm we can better estimate should we invest into the cryptocurrency today or not

//Using cryptory package for python

For having the full report about all the factors we should consider

//Checking the stocks and the FOREX markets

Candle charts are very convenient for analysing the market situation, if it is white(green) inside it means that the close price was higher than open one and we should invest into the currency

7 Literature and websites

- → https://masterthecrypto.com/guide-how-to-value-a-cryptocurrency/
- → https://blog.liquid.com/what-gives-cryptocurrencies-value
- → https://www.investopedia.com/news/does-crypto-have-intrinsic-value-bitcoin-ethereum/
- → Babkin A. V., Burkaltseva D. D., Guk O. A., Tyulin A. S. Analysis of the Development and Regulation of Crypto-currency: Foreign and Russian Experience. MIR (Modernizatsiia. Innovatsii. Razvitie) = MIR (Modernization. Innovation. Research). 2017; 8(4):554–565. DOI: 10.18184/2079–4665.2017.8.4.554–565
- → Anohin N.V., Shmyreva A.I. Criptocurrency as a tool of fi nancial market. Idei i idealy –Ideas and Ideals, 2018, no. 3, vol. 2, pp. 39–49. doi: 10.17212/2075-0862-2018-3.2-39-49.

→ https://github.com/dashee87/blogScripts/blob/master/Jupyter/2017-11-20-predicting-cryptocurrency-prices-with-deep-learning.ipynb
→ https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a
→ https://coinmarketcap.com/currencies/bitcoin/historical-data/
→ https://dashee87.github.io/deep%20learning/python/predicting-cryptocurrency-prices-with-deep-learning/
→ https://www.wikiwand.com/en/Pandas_(software)
→ https://www.wikiwand.com/en/Scikit-learn
→ https://www.wikiwand.com/en/NumPy
→ https://machinelearningmastery.com/improve-deep-learning-performance/
→ https://pythonprogramming.net/crypto-rnn-model-deep-learning-python-tensorflow-keras/
→ https://towardsdatascience.com/cryptocurrency-price-prediction-using-deep-learning-70cfca50dd3a
→ https://keras.io/api/metrics/accuracy_metrics/#binaryaccuracy-class
→ https://pandas.pydata.org/pandas-docs/version/0.22.0/generated/pandas.to_datetime.html
→ https://community.rstudio.com/t/plotting-times-with-hours-and-minutes/37788/2
→ https://www.unixtimestamp.com/index.php
→ https://www.statmethods.net/advgraphs/axes.html
→ https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781849513067/4/ch04lvl1sec12/plotting-date-and-time-on-the-x-axis [R GRAPHS COOKBOOK]
→ https://github.com/abhinavsagar/cryptocurrency-price-prediction
→ https://www.wikiwand.com/en/Training,_validation,_and_test_sets

→ https://github.com/llSourcell/ethereum_future/blob/master/A%20Deep%20Learning%20Approach%20to%20Predicting%20Cryptocurrency%20Prices.ipynb
→ https://developer.nvidia.com/
→ https://realpython.com/
→ https://github.com/abhinavsagar/cryptocurrency-price-prediction/blob/master/price_prediction.ipynb
→ https://www.wikiwand.com/en/Matplotlib
→ https://www.wikiwand.com/en/RStudio
→ http://colah.github.io/posts/2015-08-Understanding-LSTMs/
→ https://medium.com/@sdoshi579/optimizers-for-training-neural-network-59450d71caf6
→ https://www.nature.com/articles/s41598-018-37999-1
→ https://www.tradingview.com/
→ https://naga.com/blog/where-do-cryptocurrencies-get-their-value-10996744
→ https://github.com/dashee87/blogScripts/blob/master/Jupyter/2018-02-11-analysing-the-factors-that-influence-cryptocurrency-prices-with-cryptory.ipynb
→ https://tehnoobzor.com/cryptolife/ethereum/2601-ot-chego-zavisit-kurs-efiriuma-chto-na-nego-vliyaet.html
→ https://www.investopedia.com/ask/answers/100314/why-do-bitcoins-have-value.asp
→ https://www.analyticsvidhya.com/blog/2015/12/improve-machine-learning-results/