Cryptocurrency Prediction using Machine learning

A Report for the Review

S.No.	Enrollment Number	Admission Number	Student Name	Degree/Branch	Sem
1.	201010988	20SCSE1010035	Sajid Azam	B.Tech.(CSE)	5
2.	201010969	20SCSE1010133	Rahul kumar	B.Tech.(CSE)	5

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

SCHOOL OF COMPUTER SCIENCE &ENGINEERING

Galgotias University, Greater Noida

Fall 2020-2021

Declaration

I hereby declare that this Project Report titled “Cryptocurrency price prediction using machine learning” submitted to the “Department of Bachelor of Engineering in Computer Science & Engineering”.

It is a record of original work done by my team members under the guidance of **Mr. Mukesh Kumar Jha**

The information and data given in the reports is authentic to the best of my knowledge.

This project Report is not submitted to any other university or institute for the award of any degree, diploma, or published any time before.

Sajid Azam 20SCSE1010035

Rahul Kumar 20SCSE1010133

ABSTRACT

Advancements of Machine Learning (ML) in the field of computer vision have paved the way for its potential application in many other fields. Researchers and hardware domain specialist are exploring possible applications of Machine Learning in optimizing many aspects of hardware development process. In this paper we are using forcast algorithum to predict the future price of cryptocurrency.

Solana, as the most popular cryptocurrency, has received increasing attention from both investors and researchers over recent years. One emerging branch of the research on Solana focuses on empirical Solana pricing. Machine learning methods are well suited for predictive problems, and researchers frequently apply these methods to predict Solana prices and returns. In this study, we analyze the existing body of literature on empirical Solana pricing via machine learning and structure it according to four different concepts. We show that research on this topic is highly diverse and that the results of several studies can only be compared to a limited extent. We further derive guidelines for future publications in the field to ensure a sufficient level of transparency and reproducibility representation from a smaller dataset.

Forecast model is a very popular model, and it works on anything with a numerical value based on learning from historical data. For example, in answering how much lettuce a restaurant should ordernext week or how many calls a customer support agent should be able to handle per day or week, the system looks back to historicaldata.

S.No	Particulars	Page No
1	Abstract	3-4
2	Introduction	5-6
3	Literature Survey	7
4	Code implementation	7-9
5	Conclusion	10
6	References	11-12

Introduction

This paper explains the working of the linear regression and Long Short-Term Memory model in predicting the value of a Bitcoin. Due to its raising popularity, Bitcoin has become like an investment and works on the Block chain technology which also gave raise to other cryptocurrency.. This makes it very difficult to predict its value and hence with the help of Machine Learning Algorithm and Artificial Neural Network Model this predictor is tested. Methodology: In this study, we have used data sets for Bitcoin for testing and training the ML and AI model. With the help of python libraries, the data filtration process was done. Python has provided with a best feature for data analysis and visualization. After the understanding of the data, we trim the data and use the features or attributes best suited for the model. Implementation of the model is done and the result is recorded. Finding: It was discovered that the linear regression model's accuracy rate is very high when compared to other Machine Learning models from related works; it was found to be 99.87 percent accurate

ANALYSIS

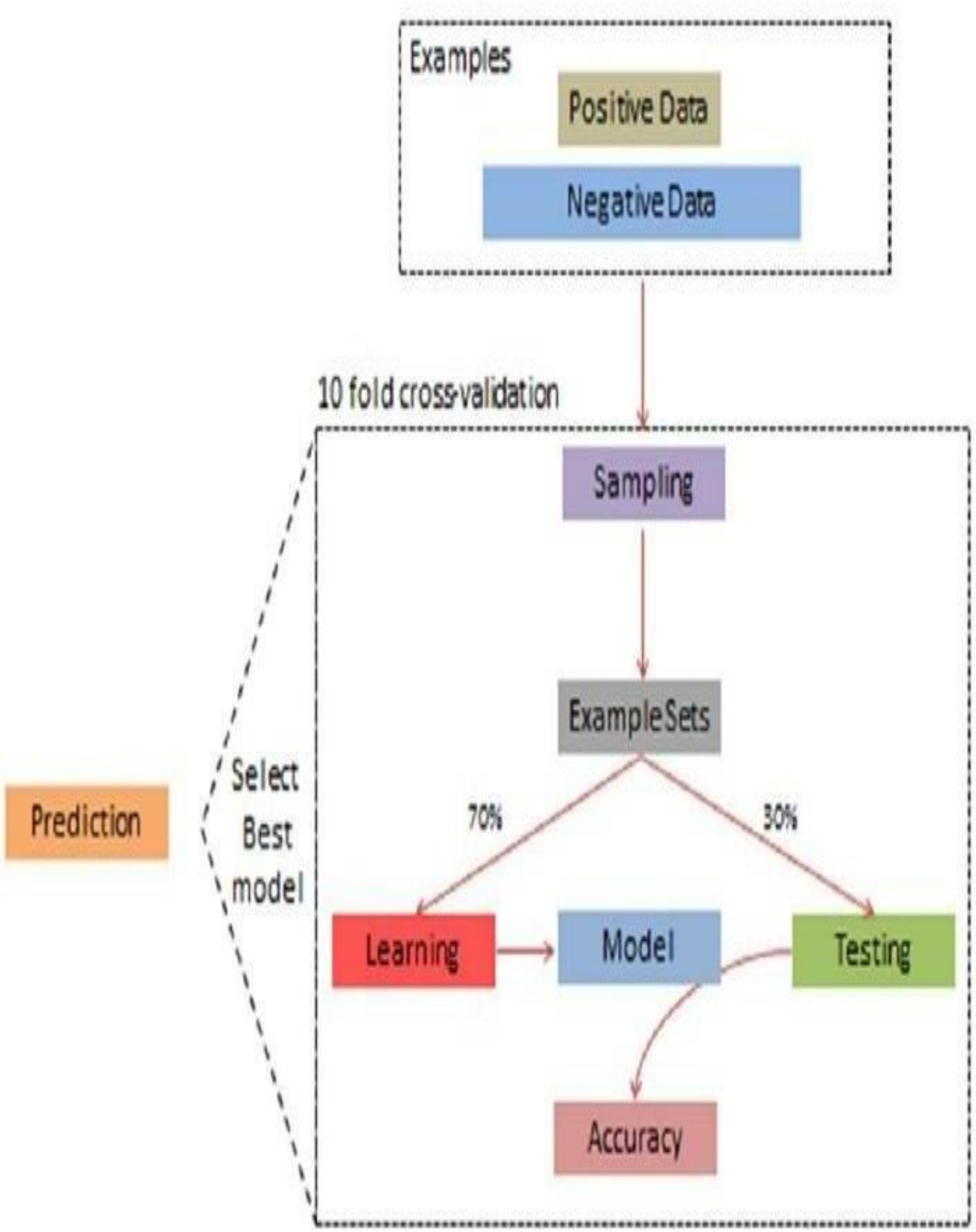
How Algorithms Work In Forecasting

In demand planning, where the cake we are baking is a forecast, our recipe generally entails different prediction methods and approaches, along with layers built from inputs from various sources. The steps and sequence of the inputs, the configuration of the methods, the repeating of steps, and the outputs all come together to form an algorithm. And this can easily consist of multiple methods and inputs reduced to three logical operations: AND, OR, and NOT. While these operations can chain together in extraordinarily complex ways, at their core, algorithms are built out of simple rational associations and a limited series of steps.

What this means is that an algorithm can be anything you like, for example, an exponential smoothing model that takes an input, uses a set of rules, parameters and steps to deliver an output to your forecasting process.

After you have properly defined the need and have the right data in the right format, you get to the predictive modeling stage which analyses different algorithms that to identify the one that will best future demand for that particular dataset.

Flowchart



Examples
Positive Data
Negative Data
10 fold cross-validation
Sampling
Example Sets
70%
30%
Learning
Model
Testing
Accuracy
Select
Best
model
Prediction

Introduction About Python Programming Language

#Python is a popular programming language. It was created byGuido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting. #

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read andmodify files.
- Python can be used to handle big data and perform complexmathematics.
- Python can be used for rapid prototyping, or forproduction- ready software development.

- Python works on different platforms (Windows, Mac, Linux,Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programswith fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code canbe executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientedway or a functional way

Literaturereview

Inthelastthreeyears,therehasbeenanincreasinginterestonforecastingandprofitingfromcryptocu rrencieswithML techniques.Table1summarizes several of those papers, presented in chronologicalorder since the work of Madan etal. (2015), which, to the best ofourknowledge, is one of the firstworks to address this issue. We do notintend to provide a complete list of papers for this strand of literature;instead, our aim is to contextualize our research and to highlightits maincontributions. For a comprehensive survey on cryptocurrencytrading andmanymorereferencesonMLtrading,see,forexample,Fangetal.(2020).

Advantages of forcast model

Embedding exterior variables into the model

Increase prediction accuracy by reducing overfitting and addressing the curse of dimensionality

Adapt to exceptional circumstance and random events

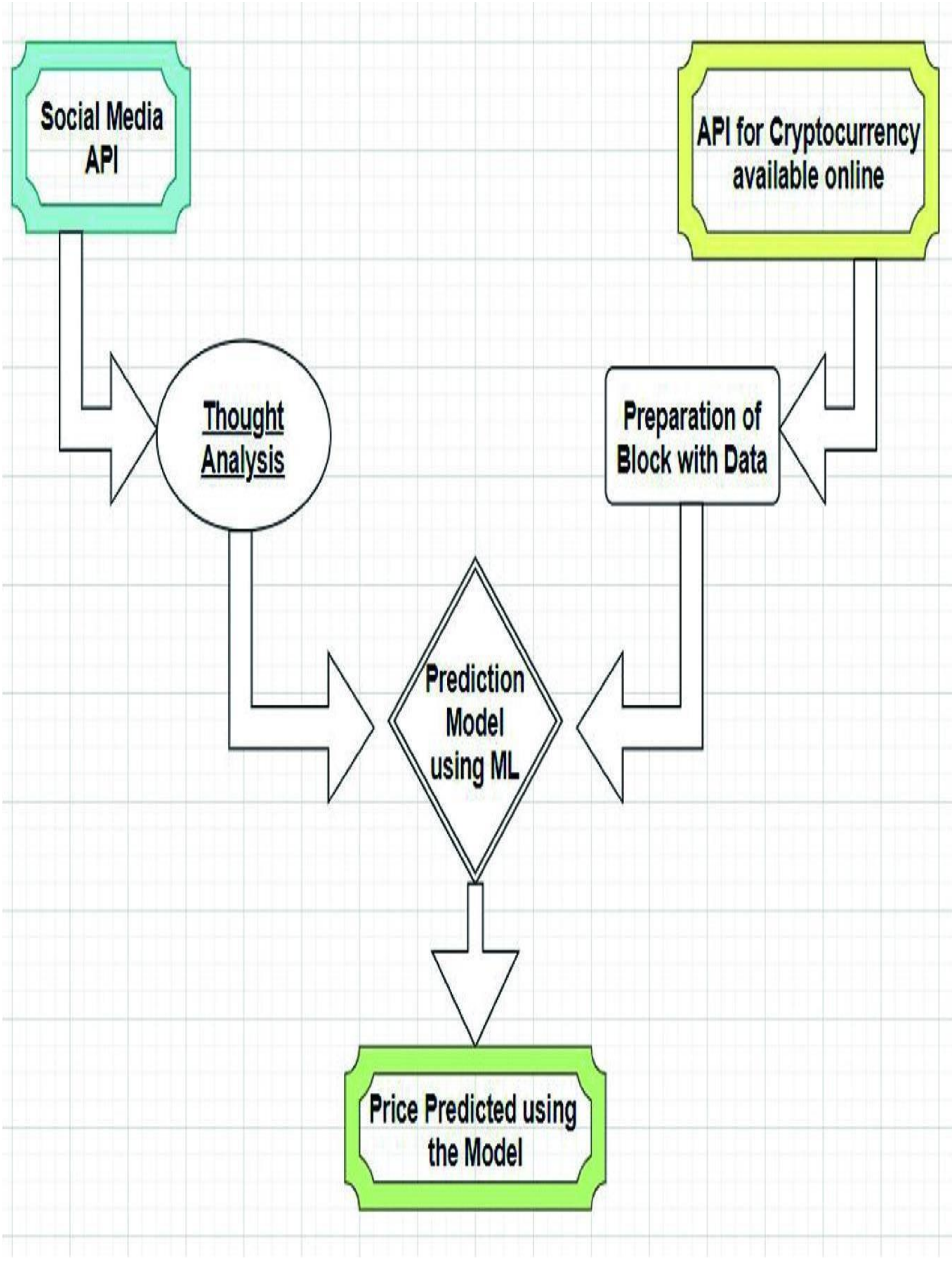
No previous knowledge about the system is needed

Disavantages of Forcast model

No comprehensive correlation between variables Difficulty in identifying

exogenous variables

The horizon of forcast is limmitted up to acouple of days ahead
Limited search explanatory



Block Diagram

Implementation

```
[1] import gc
    from tqdm import tqdm
    import numpy as np
    import pandas as pd
    from pylab import rcParams
    import matplotlib.pyplot as plt
    from sklearn.metrics import mean_squared_error
    from scipy.stats import probplot

    import datetime as dt
    from datetime import date
    from datetime import timedelta
    from fbprophet import Prophet

    %matplotlib inline
    import os
    import warnings
    warnings.filterwarnings("ignore")
```

```
[2] from google.colab import files
    files.upload()
```

```
Choose Files SOL1-USD.csv
• SOL1-USD.csv(text/csv) - 40284 bytes, last modified: 11/27/2021 - 100% done
Saving SOL1-USD.csv to SOL1-USD.csv
{'SOL1-USD.csv': b'Date,Open,High,Low,Close,Adj Close,Volume\n2020-04-10,0.832005,1.313487,0.694187,0.951054,0.951054,87364276\n2020-04-11,0.951054,1.049073,0.765020,0.776819,0.776819,
```

```
[3] data0 = pd.read_csv('SOL1-USD.csv')
    data0[-5:]
```

	Date	Open	High	Low	Close	Adj Close	Volume
592	2021-11-23	215.842957	226.061737	211.312225	221.836899	221.836899	2.328764e+09
593	2021-11-24	221.860611	222.459274	200.502609	214.921539	214.921539	2.293934e+09
594	2021-11-25	205.886169	216.400513	202.323410	206.252991	206.252991	2.586719e+09
595	2021-11-26	210.048279	210.504578	184.328278	191.511749	191.511749	3.380123e+09
596	2021-11-27	192.004044	198.408478	192.004044	194.469986	194.469986	3.353766e+09

```
[4] data1=data0
    data1['date']=pd.to_datetime(data1['Date'])
    data1=data1.drop(['Date'],axis=1).reset_index(drop=True)
    data1
```


```
[4] data1=data0
    data1['date']=pd.to_datetime(data1['Date'])
    data1=data1.drop(['Date'],axis=1).reset_index(drop=True)
    data1
```


	Open	High	Low	Close	Adj Close	Volume	date
0	0.832005	1.313487	0.694187	0.951054	0.951054	8.736428e+07	2020-04-10
1	0.951054	1.049073	0.765020	0.776819	0.776819	4.386244e+07	2020-04-11
2	0.785448	0.956670	0.762426	0.882507	0.882507	3.873690e+07	2020-04-12
3	0.890760	0.891603	0.773976	0.777832	0.777832	1.821128e+07	2020-04-13
4	0.777832	0.796472	0.628169	0.661925	0.661925	1.674761e+07	2020-04-14
...	...	...	...	...	...	...	...
592	215.842957	226.061737	211.312225	221.836899	221.836899	2.328764e+09	2021-11-23
593	221.860611	222.459274	200.502609	214.921539	214.921539	2.293934e+09	2021-11-24
594	205.886169	216.400513	202.323410	206.252991	206.252991	2.586719e+09	2021-11-25
595	210.048279	210.504578	184.328278	191.511749	191.511749	3.380123e+09	2021-11-26
596	192.004044	198.408478	192.004044	194.469986	194.469986	3.353766e+09	2021-11-27

597 rows × 7 columns

```
[10] data1=data0
     data1['date']=pd.to_datetime(data1['Date'])
     data1=data1.drop(['Date'],axis=1).reset_index(drop=True)
     data1
```

	Open	High	Low	Close	Adj Close	Volume	date
0	0.832005	1.313487	0.694187	0.951054	0.951054	8.736428e+07	2020-04-10
1	0.951054	1.049073	0.765020	0.776819	0.776819	4.386244e+07	2020-04-11
2	0.785448	0.956670	0.762426	0.882507	0.882507	3.873690e+07	2020-04-12
3	0.890760	0.891603	0.773976	0.777832	0.777832	1.821128e+07	2020-04-13
4	0.777832	0.796472	0.628169	0.661925	0.661925	1.674761e+07	2020-04-14
...	...	...	...	...	...	...	...
592	215.842957	226.061737	211.312225	221.836899	221.836899	2.328764e+09	2021-11-23
593	221.860611	222.459274	200.502609	214.921539	214.921539	2.293934e+09	2021-11-24
594	205.886169	216.400513	202.323410	206.252991	206.252991	2.586719e+09	2021-11-25
595	210.048279	210.504578	184.328278	191.511749	191.511749	3.380123e+09	2021-11-26
596	192.004044	198.408478	192.004044	194.469986	194.469986	3.353766e+09	2021-11-27

597 rows × 7 columns

```
[11] !pip install --upgrade mplfinance
```

```
Collecting mplfinance
  Downloading mplfinance-0.12.7a17-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB 728 kB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from mplfinance) (1.1.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from mplfinance) (3.2.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mplfinance) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mplfinance) (1.3.2)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mplfinance) (1.19.5)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mplfinance) (3.0.6)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mplfinance) (0.11.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->mplfinance) (1.15.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->mplfinance) (2018.9)
Installing collected packages: mplfinance
Successfully installed mplfinance-0.12.7a17
```

```
data1a=data1.copy()
data1a.index=data1a['date']
data1a=data1a.drop('date',axis=1)
data1a
```

date	Open	High	Low	Close	Adj Close	Volume
2020-04-10	0.832005	1.313487	0.694187	0.951054	0.951054	8.736428e+07
2020-04-11	0.951054	1.049073	0.765020	0.776819	0.776819	4.386244e+07
2020-04-12	0.785448	0.956670	0.762426	0.882507	0.882507	3.873690e+07
2020-04-13	0.890760	0.891603	0.773976	0.777832	0.777832	1.821128e+07
2020-04-14	0.777832	0.796472	0.628169	0.661925	0.661925	1.674761e+07
...	...	...	...	...	...	...
2021-11-23	215.842957	226.061737	211.312225	221.836899	221.836899	2.328764e+09
2021-11-24	221.860611	222.459274	200.502609	214.921539	214.921539	2.293934e+09
2021-11-25	205.886169	216.400513	202.323410	206.252991	206.252991	2.586719e+09
2021-11-26	210.048279	210.504578	184.328278	191.511749	191.511749	3.380123e+09
2021-11-27	192.004044	198.408478	192.004044	194.469986	194.469986	3.353766e+09

597 rows × 6 columns

```
[20] import mplfinance as mpf
     mpf.plot(data1a[-60:], type='candlestick',mav=(4,6))
```


```
data2=data1[['date','Close']]
data2
```

	date	Close
0	2020-04-10	0.951054
1	2020-04-11	0.776819
2	2020-04-12	0.882507
3	2020-04-13	0.777832
4	2020-04-14	0.661925
...	...	...
592	2021-11-23	221.836899
593	2021-11-24	214.921539
594	2021-11-25	206.252991
595	2021-11-26	191.511749
596	2021-11-27	194.469986

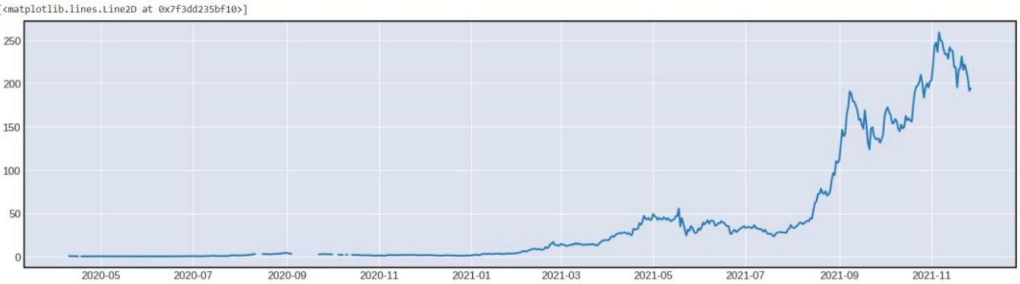
597 rows × 2 columns

```
[22] item1=data2

     item1.columns = ['ds','y']
     item1.y = item1.y.astype('float')
     item1.ds = item1.ds.astype('datetime64')

     rcParams['figure.figsize'] = 20,5
     plt.plot(item1.ds, item1.y)
```

[<matplotlib.lines.Line2D at 0x7f3dd235bf10>]



[23]
```
from datetime import timedelta
start0 = dt.datetime.strptime('2021-01-01','%Y-%m-%d').date()
end0   = dt.datetime.strptime('2022-12-31','%Y-%m-%d').date()
print((end0-start0).days)
```

```
729
```

[24]
```
def daterange(start,end):
    for i in range((end-start).days):
        return start+timedelta(i)
```

[25]
```
dates0=[]
for i in range((end0-start0).days):
    dates0+=[(start0+timedelta(i)).strftime('%Y-%m-%d') ]
print(dates0[0:10])
```

```
['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04', '2021-01-05', '2021-01-06', '2021-01-07', '2021-01-08', '2021-01-09', '2021-01-10']
```

[26]
```
dates0_df=pd.DataFrame(dates0)
dates0_df.columns=['ds']
dates0_df
```

[26]

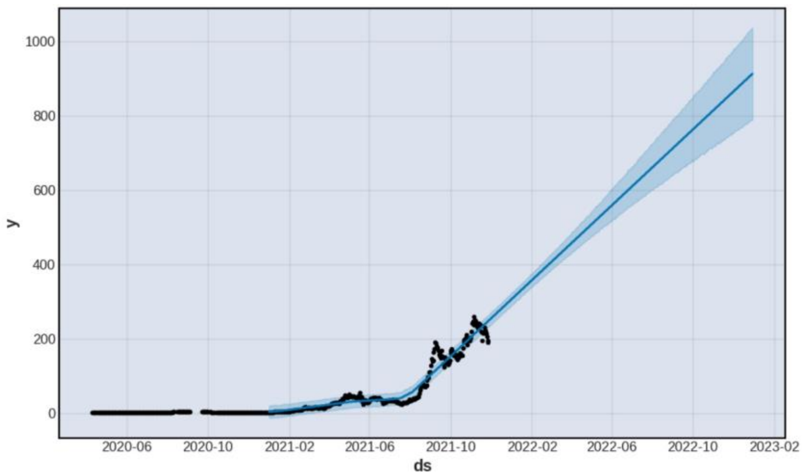
0	2021-01-01
1	2021-01-02
2	2021-01-03
3	2021-01-04
4	2021-01-05
...	...
724	2022-12-26
725	2022-12-27
726	2022-12-28
727	2022-12-29
728	2022-12-30

729 rows × 1 columns

[27]
```
ph = Prophet()
ph.fit(item1)
forecast3=ph.predict(dates0_df)
figure = ph.plot(forecast3)
figure.show()
```

<u>Predicted model:-</u>

[27]



```
[28] forecast3[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
724	2022-12-26	907.199740	784.944939	1029.450578
725	2022-12-27	908.219736	789.639084	1032.282521
726	2022-12-28	910.238118	789.478568	1033.498276
727	2022-12-29	911.831482	788.604290	1037.508002
728	2022-12-30	913.516718	788.665268	1036.930339

References

(1) Lahmiri S, Bekiros S (2019) Cryptocurrency forecasting with deep learning chaotic neural networks. Elsevier, 35–40

(2) Saxena A, Sukumar TR (2018) Predicting bitcoin Price using lstm and compare its predictability with Arima model. Int J Pure Appl Math 2591–2600

(3) Paresh Kumar N, Narayan S, Rahman RE, Setiawan I (2019) Bitcoin price growth and Indonesia's monetary system. Emerg Mark Rev 38:364–376

(4) Pant DR, Neupane P, Poudel A, Pokhrel AK, Lama BK (2018) Recurrent neural network based bitcoin price prediction by twitter sentiment analysis. IEEE, pp 128–132

(5) Dr. Jain ,The Institute of Business Forecasting & Planning (IBF)-est. 1982

Steps to implement in prediction

Import the Libraries.

Load the Training

Dataset.

Use the Open Stock Price Column to Train Your

Model. Normalizing the Dataset.

Creating X_train and y_train Data

Structures. Reshape the Data.

Building the Model by Importing the Crucial Libraries

Fitting the Model.

Building LSTM model

Predicting Bitcoin price

CONCLUSION

All in all, predicting a price-related variable is difficult given the multitude of forces impacting the market. Add to that, the fact that prices are by a large extent dependent on future prospects rather than historic data. However, using deep neural networks has provided us with a better understanding of Bitcoin, and LSTMarchitecture. The work in progress, includes implementing hyperparameter tuning, in order to get a more accurate network architecture. Also, other features can be considered (although from our experiments with Bitcoin, more features have not always led to better results). Microeconomic factors might be included in the model for a better predictive result. Anyway, maybe 6 Conclusions All in all, predicting a price-related variable is difficult given the multitude of forces impacting the market. Add to that, the fact that prices are to a large extent depended on future prospects rather than historic data. However, using deep neural networks has provided us with a better understanding of Bitcoin, and LSTM architecture. The work in progress, includes implementing hyperparameter tuning, in order to get a more accurate network architecture. Also, other features can be considered (although from our experiments with Bitcoin, more features have not always led to better results). Microeconomic factors might be included in the model for a better predictive result. Anyway, maybe the data we gathered for Bitcoin, even though it has been collected through the years, might have become interesting, producing historic interpretations only in the last couple of years.

Furthermore, a breakthrough evolution in peer-to-peer transactions ongoing and

transforming the landscape of payment services. While it seems all doubts have not been settled, time might be perfect to act. We think it's difficult to give a mature thought on Bitcoin for the future.