A NOVEL CHAOS THEORY BASED HYBRID LSTM MODEL FOR BITCOIN PRICE PREDICTION

A DISSERTATION

submitted in partial fulfillment of the requirements

for the award of the degree of

Master of Technology

in

COMPUTER SCIENCE & ENGINEERING

by

Gopal Khatri

ROLL NO. 15MI520

Under the supervision of

Dr. Jatoth Chandrashekhar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY HAMIRPUR

HAMIRPUR – 177005, HIMACHAL PRADESH (INDIA)

June, 2020

CERTIFICATE

I hereby certify that the work which is being presented in the M.Tech. Dissertation entitled **“”**,Thesis Title Here! in partial fulfillment of the requirements for the award of the **Master of Technology in Computer Science & Engineering** is an authentic record of my own work carried out during a period from July 2019 to May 2020 under the supervision of **Dr. Jatoth Chandrashekhar, Assistant Professor**, Computer Science & Engineering Department, National Institute of Technology Hamirpur.

The matter presented in this thesis has not been submitted for the award of any other degree elsewhere.

Signature of Candidate

GOPAL KHATRI

Roll No. 15MI520

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Supervisor

Date: **DR. JATOTH CHANDRASHEKHAR,**

ASSISTANT PROFESSOR

The M.Tech Viva-Voce Examination of Gopal Khatri, M.Tech student has been successfully held on.........

Signature of Supervisor **Signature of External Examiner**

ACKNOWLEDGEMENT

First of all, I express my gratitude to the Supreme Intelligence who blessed me with the zeal, enthusiasm and intelligence, and guided me with his power to complete this research work successfully. I am extremely grateful to my supervisor **Dr. Jatoth Chandrashekhar**, Assistant Professor, Department of Computer Science & Engineering, National Institute of Technology Hamirpur, Himachal Pradesh for the his motivation and tireless efforts to inspire and help me to get deep knowledge of the research area and supporting me throughout the life cycle of my M. Tech. dissertation work. Especially, the extensive comments, fruitful guidance through the times of chaos and confusions, and his good nature made it very comfortable to work with him and produce a good quality M. Tech. dissertation work.

I would also like to extend my heartfelt thanks to all the staff members of the **CSE Department** for their constant support and helping with the topics that were difficult to understand,and **Computer Center** for providing all the resources without the virtue of which this research would not have come the the end stage.

This thesis would not have been possible without the hearty support of my friends during my low times. My deepest regards to my Parents for their blessings, affection and continuous support. Also, Last but not the least, I thank to the GOD, the almighty for giving me the inner willingness, strength and wisdom to carry out this research work successfully.

Gopal Khatri

ABSTRACT

Bitcoin is a digital cryptocurrency that is used as a medium of exchange. After its inception in 2008, it has captured the attention of many investors and big companies because it shows huge returns on investments. However, due to its volatile nature, and high uncertainty, it is a very risky asset to invest in. Investors lose a huge amount of money due to the lack of highly precise prediction models. The high fluctuations in bitcoin prices may be due to the influence of complex nonlinear dynamics that the current bitcoin studies are unable to understand. The unstable nature of bitcoin may be due to the presence of chaos in the market. In this study, we rigorously investigated the presence of chaos in bitcoin prices and found that they exhibit chaotic dynamics. There are many deep learning-based state-of-the-art prediction models that are used to predict bitcoin prices. However, they don't provide predictions precise enough to make significant financial gains. In traditional financial markets literature, the hybrid models are best known to yield much better predictions as compared to their stand-alone models. However, in bitcoin literature, there are not many hybrid models. Moreover, the stand-alone models show modest predictions due to their inability to learn subtle and hidden non-linear patterns behind bitcoin price fluctuations. This motivated us to introduce a new hybrid prediction model based on chaos theory and phase space reconstruction. We detected and modeled the chaos present in bitcoin prices and exploited the application of phase space reconstruction to build a more efficient model that can learn hidden and complex nonlinear patterns. The performance of the proposed model was compared with other state-of-the-art hybrid and stand-alone bitcoin price prediction models, and we found that our model outperformed all those models.

Contents

List of Figures

Chapter 1

INTRODUCTION

Bitcoin is a digital currency, launched in 2008, which provides a medium for transactions between participants to make exchange quickly. It is a currency with no government to issue it and no banks are needed to manage accounts and verify transactions. After its inception, bitcoin was not popular for the first few years. Its price index was very low due to its unstable nature and lack of understanding of the technology behind it. However, after 2017, bitcoin has been gaining immense popularity among traders and investors due to its large market capitalization. But there will never be more than 21 million bitcoin circulated. The model of the bitcoin operation is similar to the model of any precious metal i.e. the amount generation of bitcoin decreases with time. It is a technology that is fully decentralized, open-source with a transparent operation, and its underlying infrastructure guarantees full security. The backbone underlying system is not dependent on any of the government-regulated or private banks because it is based on a clever system of decentralized trustless verification based on cryptography. It consists of a publicly accessible ledger that contains verified transactions. All the transactions are verified and approved through digital signatures that are encrypted through a private key and a public key. A digital signature provides optimal security because it is unique for different messages or transactions. A digital signature is a string of bits 0's and 1's, commonly 256 bits, and altering messages even slightly completely changes the signature on that message.

There is a diverse market for cryptocurrencies, with a market capitalization of more than $860 billion in 2019, and it provides a variety of options for investors to invest in. The growth in popularity of bitcoin has also fueled immense interest among the research and computing community. Historic cryptocurrencies market data shows that bitcoin has faced periodic unexpected rises and sudden dips during specific periods of time. Therefore, bitcoin traders have a need for efficient methods to identify trends and patterns in market behavior. The uncertain and drastic change in the bitcoin market trends increases the need to develop accurate prediction models to predict fluctuating prices. Bitcoin prices are relatively very difficult to predict as compared to other traditional financial instruments. Due to its very volatile nature and relatively young age, there are a lot of opportunities for research and the need for novel methods for Bitcoin market forecasting. Indeed, a lot of new research on this market is required because its dynamics are unique in relation to the traditional markets, and its properties as a currency are also very unique compared to traditional fiat currencies.

Bitcoin price prediction is a time series prediction problem. Time series data is a sequence of data points measured at consistent time intervals. It is a series of numeric values, each with a timestamp defined by a name and a set of labeled dimensions. Time series has various applications in the fields of data and time-driven decisions. In the past few years, time-series databases have emerged as the fastest-growing category of databases. Autonomous trading algorithms continuously collect data on market fluctuations over a period of time. Time series are used to monitor a company's assets that are moving across the world with a fine-grained level of precision and efficiency. Time series prediction involves the analysis of changes that occurred in the past values, monitoring of changes occurring in the present values, to predict future values. It has diverse applications in the fields of stock market analysis, gold price prediction, cryptocurrencies price prediction, sales forecasting, utility studies, inventory studies, quality control, census analysis, etc.

The randomness alone is not responsible for the presence of uncertainty and the nonlinear behavior of a particular time series. Sometimes, it can also be due to a complex deterministic phenomenon, classified as chaos, which makes a time series highly sensitive to initial conditions. A bitcoin time series can be influenced by many factors such as economical, social, public sentiments on social media, investors' perception towards risk, political factors, prices of other cryptocurrencies, etc. All these factors, that are difficult to determine, introduce uncertainty, noise, volatility, and randomness in the bitcoin time series. Despite the uncertainty and volatility, the highly accurate price predictions provide investors the tremendous potential and opportunities to make huge financial gains. Therefore, predictions of bitcoin prices, being complex and chaotic, if one wants to make huge gains, call for the development of more sophisticated and powerful hybrid price prediction models that can provide both high levels of accuracy and performance.

1.1 Chaos Theory

Chaos theory provides a way to detect and model the chaos present in nonlinear and deterministic complex dynamical systems. The chaos in terms of scientific context refers to disorder and confusion in a system that nevertheless complies with certain rules and deterministic laws. The chaos theory is based on the idea which is identified as *sensitive dependence on initial conditions*, meaning that any small change in the system propels the system to exhibit the completely random states of disorder, confusion, and irregularities. The chaos theory states that a chaotic complex system, despite the fact that it is under apparent randomness and confusion, has interconnectedness and underlying patterns among its components.

Mathematically, according to [1], a complex dynamical system is said to be chaotic if:

1. It contains a collection of points that form periodic orbits when projected to higher dimensions.

2. There is a sensitive dependence on the initial conditions of the system.

3. It is topologically transitive.

The *butterfly effect* is the underlying principle of chaos theory, which relies on the metaphor that a butterfly flapping its wing in India can cause a hurricane in Texas. The main idea that butterfly effect tries to convey is that even small changes in previous or current states of a nonlinear dynamical system can produce a significantly larger difference in the next or later states of the system.

When the data of a stochastic or chaotic system is plotted in a graph, the chaotic systems are almost similar to the stochastic system. So we cannot distinguish a chaotic system from a stochastic system just by looking at the time series. The best-known method to detect the presence of chaos in a time series is the Lyapunov exponent test. The main application of chaos theory for a chaotic time series is that the underlying nonlinear dynamics of a complex dynamical system (or time series) or the chaos present in the system can be modeled by reconstructing its phase space using two parameters; time delay (lag) and embedding dimension, which is the number of components that are required to represent the whole dynamics of the nonlinear chaotic system.

The phase space reconstruction (PSR) allows the reconstruction of the complete dynamics of a nonlinear system using a single chaotic time series [2]. Thus phase space reconstruction lays the foundation for the analysis of a complex system like the Bitcoin market. Taken's delay embedding theorem [3] is the most commonly used approach for reconstructing phase space of a time series. Using this theorem, an univariate time series can be reconstructed into a multidimensional time series by regenerating multidimensional vectors of observations form a single vector of observations. If the regenerated multidimensional time series has enough number of embedding dimensions (large enough to represent every data point vector in the reconstructed phase space), then it can display numerous essential properties of its system.

The Bitcoin time series as concluded by [4] is chaotic. A time series of a dynamical system is considered to be chaotic if and only if the system is nonlinear, deterministic, and sensitively dependent on initial conditions. The high sensitive dependence on the initial conditions is the prime reason for the presence of chaos in the time series (or the system). A chaotic time series can be generated by various real-life complex systems; for example, the beating of the heart, if measured numerically, produces a chaotic time series. Other vivid examples of chaotic systems are: financial stock markets, international currency exchange rates, the change in the weather, Lorenz system, Rossler's system, the production and consumption of electricity, etc.

The application of chaos theory can be found in the international exchange rates [1,5,6], crude oil market [7], stock market exchange [8,9], and many commodities and currencies. The aforementioned literature studied the nonlinear statistical dynamics and complex behavior of many markets using chaos theory, but the Bitcoin market is missing such meticulous investigations and lacks the advantage of the application of chaos theory. Only a limited number of works tried to detect chaos in the Bitcoin market. For instance, [4] conducted a study on the Bitcoin market to investigate, assess and detect chaos by analyzing the market throughout two distinct time periods; one period during which prices increased slowly (low-level regime period), and the other where the prices exploded exponentially and showed unexpected returns (high-regime period). Although [4] detected the presence of chaos during both time periods, they found that the chaos and uncertainty were very high during the high-regime period as the Bitcoin market showed highly nonlinear dynamical patterns during this period.

1.2 Fluctuations in bitcoin's price & bitcoin price prediction problem

The past studies in the literature [10–14] mostly used traditional approaches that were used for analyzing financial markets and their predictions, to analyze and forecast price fluctuations of the cryptocurrencies market. The prediction of short-term and long-term prices of cryptocurrencies is highly dependent on many significant factors and predictors such as trading volume, volatility, market beta, public perception, etc. [15]. Since the dynamics of the cryptocurrencies market are very different from that of traditional financial markets, the need for accurately forecasting price fluctuations and the market direction in the future gives rise to an important business challenge for potential investors. [16] showed that although cryptocurrencies trading is a time series prediction problem, the methods and predictive models that are used for traditional financial market forecasting cannot be directly applied to solve cryptocurrency price forecasting problem due to the unstable and highly volatile nature of this market. [17] revealed that Bitcoin, due to its characterization with high volatility, shows a low correlation to other financial assets, but promises extremely high returns at very high risk. [18, 19] studied and documented the high volatility of Bitcoin very well. [20] applied some econometric methods to develop theories to model and predict the volatility of Bitcoin.

[21, 22] researched the time series forecasting of traditionally mature financial markets such as the stock market at length. The bitcoin market prediction is an interesting parallel time series prediction problem (in a transient stage) to the stock market prediction problem. The traditional time series prediction methods are effective to make predictions for traditional financial markets because data from such markets can be broken down into various trends, patterns, seasonal, and noise. For instance, Holt-Winters exponential smoothing models [23] rely on such data. This type of approach is suitable for the markets or tasks where the dependency on seasonal effects is present e.g. for forecasting sales, as the consumers buy a particular product only on a particular season. However, these methods are not very effective when applied in the context of Bitcoin (or any other cryptocurrency) market due to its high volatile nature and the lack of seasonality.

1.2.1 Available solutions for bitcoin price prediction problem

The past studies have focused on the analysis and prediction of fluctuating prices using traditional approaches that were used for financial and stock market predictions. Considering Bitcoin as a financial asset, many researchers used various statistical, analytical, and experimental methods to investigate various factors that influence the Bitcoin and to identify the patterns behind its fluctuations [24]. Univariate autoregressive (AR), simple moving averages (MA), simple exponential smoothing (SES), and Autoregressive Integrated Moving Average (ARIMA) are the traditional time series prediction approaches that were used to forecast Bitcoin prices. ARIMA was used by [25]. [26] investigated seasonality patterns in Bitcoin price fluctuations by the previously described traditional time series models. The seasonality patterns can be best investigated by the seasonal ARIMA (SARIMA) model. The traditional time series models also fail to capture insights on long term dependencies due to the high volatility of the Bitcoin market.

On the contrary, machine learning models and deep learning models like neural networks can be more effective than traditional models because they employ many advanced techniques like iterative optimization algorithms such as gradient descent. They are also able to determine the best-fitted optima by tuning proper hyperparameters. The artificial neural networks (ANN) can also capture the complex nonlinear dynamics of a dynamical system. In recent years [27, 28] incorporated nonlinearity to machine learning models to predict price/returns for a financial asset with higher prediction accuracy than that of the traditional prediction models. However, in the case of Bitcoin price prediction, there is a dearth of application of the machine learning models and deep neural networks in the literature. The prediction models using neural networks can capture the nonlinear dynamics of the Bitcoin market even in the presence of high volatility. Due to the recent advancements in machine learning algorithms, many studies [11, 29] proposed deep learning-based prediction models for predicting Bitcoin's future prices.

In the literature, various machine learning regression models like random forests [30], Bayesian neural networks [31], and deep neural networks [29] have been used to predict Bitcoin prices. The artificial neural networks have been applied to financial time series problems in numerous previous works. However, [32] concluded that even though neural networks can successfully approximate Bitcoin log return distribution, the highly accurate predictions can be yielded by the application of more complex neural networks such recurrent neural networks (RNN) and Long Short-term Memory (LSTM) networks. [11, 29] compared the Bitcoin price predictions of RNN and LSTM with the predictions of the traditional ARIMA model. The RNN and LSTM model, due to the utilization of more advanced algorithms, yielded better predictions than the traditional multilayer perceptron (MLP) for Bitcoin prices [29]. The deep learning LSTM model estimated future prices of Bitcoin with a significantly mean absolute error (MAPE) than the ARIMA time series model [26]. Thus given the complexity of the task of predicting the prices of highly volatile cryptocurrency, Bitcoin, the RNN, and LSTM are far better than multilayer perceptron (MLP) and any other traditional time series due to the temporal nature of Bitcoin price data.

Although there are several studies in the literature that study and compare various deep learning methods for making Bitcoin price predictions, most previous work [33] suggests that the most accurate predictions could be yielded only by more advanced and complex neural networks, mostly based on RNN. For example, prediction models based on convolutional neural networks [34, 35] and its variants, such as a residual network (ResNet) [36] did not gain much attention for Bitcoin price prediction problem, even though they had applications in effectively analyzing long data sequences. Furthermore, most previous studies address only a regression problem, where the future prices of Bitcoin are predicted based on past prices, but not a classification problem, where the upward or downward movement of future prices with respect to the past prices is predicted by the prediction model. Although the root mean squared error (RMSE) or the mean absolute percentage error (MAPE) is commonly used to measure the performance of the prediction model for the regression problem, a low RMSE or MAPE is the most accurate measure of the effectiveness of the prediction model; for instance, the same prediction model with a low RMSE or MAPE values, that performs very well for a regression problem might not have the same high performance for a classification problem.

1.2.2 Characteristics of bitcoin time series and hybrid solutions to price prediction

The Bitcoin time series prediction, due to its high uncertainty, randomness, and high volatility is a highly complex task as a Bitcoin time series exhibits the following characteristics:

1. Bitcoin prices often fluctuate following a random-walk, which makes it unpredictable from a theoretical perspective.

2. Bitcoin time series is very noisy and chaotic, any initial market conditions can cause a high degree of fluctuations in price.

3. The change in Bitcoin price is a time-varying process, and its statistical properties change at different points in time.

Considering these characteristics, the most challenging problem in its price prediction is to build an appropriate prediction model that detects and captures the subtle and hidden patterns in data. Although the literature abounds with various deep neural network models that are statistically more powerful than traditional models both in terms of performance and accuracy and yielding better results, these models have many demerits. The investors cannot always rely on the predictions of those models because of the level of accuracy with which they predict Bitcoin prices are not up to the level of expected accuracy required to make significant profits. For larger investments, an investor can lose a large amount of money (in millions), if a single prediction turns wrong.

Several studies have demonstrated that we can yield superior predictions by hybrid prediction models, which can be formed by combining different prediction models and methods in different permutations and combinations. Several pieces of research in the field of typical financial time series prediction have demonstrated the superiority of hybrid or ensemble models over their stand-alone models. [37] laid the foundation on how to propose and develop hybrid time series models. [37] showed that the hybrids built by the befitting combination of the different methods can yield much accurate predictions compared to their stand-alone methods. Makridakis et al. [38] hybrid or ensemble models improve prediction accuracy. Pelikan et al. [39] combined the various artificial neural networks (ANN) and improved time series prediction accuracy to high levels compared to their stand-alone counterparts. Usually, a good hybrid prediction model should be able to:

1. Improve prediction performance.
2. Overcome the demerits of stand-alone models and provide better results.
3. Reduce uncertainty and provide reliable predictions.
4. Ensure the correctness of predictions.

We can exploit the advantage of the superiority of hybrid models over their stand-alone models, to utilize them for Bitcoin price predictions. There are few traces of the hybrid Bitcoin price prediction models in the literature. [40] combines the Hidden Markov Model, which describes the historical movements of bitcoin prices, and the LSTM model that is optimized using genetic algorithms to build a hybrid for predicting bitcoin's future prices. [32] uses a hybrid of convolution neural networks (CNN) and the recurrent neural networks (RNN) to predict bitcoin prices.

1.3 Working of bitcoin as a digital currency

If the reader wants to understand the working of bitcoin, then he/she must first understand the key concepts of bitcoin, which are discussed as follows:

1. **Bitcoin blockchain**

   It is a live running record of all Bitcoin transactions. It is a list of transactions where payments are sent to or from one person to another are indexed one after another. Once a certain amount of transactions in the list is reached, a block is formed. Each block has a limit of the maximum amount of transaction data it can store. Once the maximum transactions are reached, the block is added behind the previous blocks. The blocks of transactions are linked together through the chains (hash values). Thus a bitcoin blockchain is a group of bitcoin transaction data linked together through chains. The structure of the bitcoin blockchain consists of a network of computers distributed around different geographical regions in the world. Bitcoin software is installed on all of those computers. When bitcoin transactions occur, the data is communicated to this network of computers that validate those transactions, add those transactions in their copy of the Bitcoin ledger, and broadcast changes that are entered in the ledger to the other computer on the network. Since there is a limit on the maximum amount of data that can be saved per block, a new block of transactions is created approximately every 10 minutes, verified, and published to the bitcoin blockchain.

2. **Bitcoin Mining- Verification of transactions on the bitcoin network**

   The verification and posting of transactions on the blockchain are completed by bitcoin miners by a process called mining. Miners are people or pools of people that utilize computer processing power to maintain the bitcoin blockchain and keep the ledger consistent. Miners group new transactions into blocks and broadcast them to the rest of the bitcoin network for verification. Each new block of transactions contains the cryptographic hash of the block that was published before it. The hash links all of the blocks of transactions together to form the blockchain. For a new block to be accepted by the network, miners are required to follow a proof-of-work system which involves creating a new cryptographic hash of the newly completed block. Each new block of transactions has a unique hash from the previous block. The miners compete with each other using computational power to create a new unique hash for a new block of transactions that is less than or equal to the current difficulty target of the network. The miners are rewarded with bitcoin each time they verify a new block of transactions. Mining rewards are a combination of newly minted bitcoin units that were not previously circulating and transaction fees of bitcoin that were already circulating. These rewards are provided to incentivize miners to participate in the mining process to ensure that the bitcoin network continues to be audited and essentially maintained.

3. **Bitcoin supply**

   There are similarities between bitcoin and gold because bitcoin intentionally harbors the characteristics of gold. Prospecting for new gold deposits, building a mine, and operating in mine, to extract gold is laborious similar to the immense computing power required by bitcoin miners to perform guesswork to create the new unique hashes to validate new blocks of transactions. Another gold-like characteristic programmed into bitcoin is the maximum supply. The maximum number of bitcoin that can ever exist is 21 million units. This 21 million unit maximum was established to mirror gold's stable inflation rate. Currently, 18 million of 21 million bitcoins are in circulation. According to its current track, the last bitcoin will be mined in the year 2140.

4. **Bitcoin Halving**

   It refers to the reduction in the bitcoin block rewards issued to miners by half. The reward per block before 11th May 2020 was 12.5 units of newly minted bitcoin that were not previously in circulation. After halving the reward per block is 6.25 units. Halving was programmed to occur every 210,000 blocks. The halving occurs every four years because every new block of transactions is completed every 10 minutes. In 2009, the block reward was 50 bitcoins. In 2012, the first halving reduced block reward to 25 units, and in 2016 the second halving event reduced it to 12.5 units. The halving of newly minted bitcoin every 210,000 units was programmed to prevent inflation from decreasing the purchasing power of bitcoin.

1.3.1 Implications of halving for bitcoin

The fixed supply of gold makes it one of the best stores of value. It makes gold more scarce. Similarly, the algorithm which imposes bitcoin's fixed supply makes bitcoin even scarcer than gold. So if demand remains steady or increases for the fixed supply of bitcoin, the price of bitcoin will experience positive long-term effects. The supply of bitcoin becomes more scare after each halving, which means it's price will increase in the future after every halving. For easier understanding, bitcoin was around \$11 when the first halving occurred in November 2012. Then in 2013, bitcoin spiked to \$1100, the highest bitcoin had ever been before dropping back down to around \$220 and remaining \$1000 for the next few years. In July of 2016 during the second halving bitcoin was around \$600 and then spiked to \$20000 near the end of 2017. The 2017 bitcoin was more due to market manipulation than the halving, but halving had been playing a role in maintaining its price over \$3000 since then.

1.4 Problem statement

Bitcoin, due to its highly volatile nature and high uncertainty, is a very risky asset to invest in. The general public and even the experienced investors often lose a huge amount of money due to the lack of highly reliable price prediction models. The current state-of-the-art bitcoin price prediction models mostly use complex deep learning techniques based on recurrent neural networks(RNN) such as LSTM and GRU. However, even these models are inefficient in predicting future prices with the level of expected accuracy that is required to make significant financial gains. A potential investor can lose a tremendous amount of money if the predictions of price contain even a small amount of error. Thus, there is a dire need for highly accurate prediction models to minimize investment losses as much as possible. As discussed earlier, ensemble or hybrid prediction models can yield greater predictions than their stand-alone models. But there are very few hybrid models in the literature, so more hybrid models need to be introduced. Additionally, the bitcoin market is novel and hasn't been around for long enough to have matured, so limited research is done in this domain. Due to the limited research on the chaotic behavior of bitcoin, there is very little evidence of whether the chaos is present in the bitcoin market or not. If we can detect and effectively model chaos present in this market, then we will be able to unravel many mysterious factors that are accountable for the highly fluctuating prices. If we have enough knowledge to understand all the hidden variables behind bitcoin's volatility and uncertainty, then either we will be able to stabilize the fluctuating prices of this asset, or we will be able to develop sophisticated, powerful, and highly accurate prediction models having great performance.

1.5 Objectives of research

In this study, we propose a novel hybrid prediction model based on the chaos theory and phase space reconstruction (PSR) that accounts for the subtle and hidden patterns and information at multiple time scales to predict bitcoin prices, and compare its performance with various traditional models like ARIMA and ARMA, and with state-of-the-art deep learning-based models like RNN, GRU, and LSTM. In particular, the objective of this study are:

- To detect the presence of chaos in the bitcoin prices.
- To develop a chaos and PSR-based hybrid LSTM bitcoin price prediction model by exploiting the application of chaos theory and phase space reconstruction to model and unfold the nonlinear dynamics of the complex bitcoin market.

The main reason behind modeling the chaos present in our bitcoin time series is that the deep LSTM model can be trained more effectively to learn more complex and hidden patterns to yield highly accurate predictions than if it was trained without modeling the chaos.

1.6 Contributions to the literature

Our study makes two contributions to the Bitcoin literature:

- The first contribution is a methodological approach to examine and detect chaos in the prices of Bitcoin. The study attempts to detect the presence of chaos in the Bitcoin market by using the largest Lyapunov exponent (LLE).

- The second contribution relates to introducing a new hybrid model to effectively predict future prices with higher accuracy than other traditional and state-of-the-art models.

1.7 Dissertation structure

The rest of this thesis is structured as follows:

Chapter 2 documents all the previous work that has been done in the domain of bitcoin price prediction and existing state-of-the-art models that are being currently used by the investors to predict prices before buying or selling this digital asset.

Chapter 3 explains the proposed methodology and all the techniques used by the proposed algorithm in detail and describes the dataset used for predicting prices.

Chapter 4 discusses the simulations that were performed while conducting this research and explain outputs. Simulations for detecting and modeling chaos are done using *MATLAB*, and the LSTM model is trained and simulated using *Python* in *Jupyter Notebook*.

Chapter 5 concludes the outcomes of our research and offers room for future improvements.

Chapter 2

LITERATURE REVIEW

The hybrid models that are formed by combining various time series forecasting models can yield better predictions than their single stand-alone forecasting models. Unlike for traditional financial markets prediction problems, there are very few hybrid bitcoin price prediction models in the Bitcoin literature at the current time. Although there are few deep neural networks based hybrid and ensemble models for predicting bitcoin prices, we didn't find any evidence of the presence of chaos-based hybrid models in the bitcoin literature. In this chapter, we reviewed a few relevant hybrid bitcoin price prediction time series models. We also reviewed the traditional and state-of-the-art models that are used for making price predictions at the present time. First, we present various hybrid models that are currently being used for solving bitcoin price prediction problems, second, we present traditional, state-of-the-art machine learning, and deep learning based prediction models. Finally, we present the studies based on chaos theory that were conducted to detect chaos in the financial markets.

2.1 Hybrid models for predicting bitcoin prices

Iman Abu Hashish et al. 2019 [40] proposed a hybrid model based on the Hidden Markov Model and LSTM which is optimized by the Genetic Algorithm to predict Bitcoin prices. The Hidden Markov Model captures the hidden information and patterns present in Bitcoin prices, and this information is integrated with the dataset while training the LSTM model. They optimized hyperparameters of LSTM using genetic algorithms which improved the model's performance greatly. They compared the performance of the HMM-based hybrid LSTM model with the performance of stand-alone LSTM and a traditional ARIMA model and found that hybrid outperformed the other models and had the lowest MSE, RMSE, and MAE among all the tested models. The limitation of this model is that it doesn't consider the internal details of transactions present in the blockchain. If it were to consider all other features of blockchain while training the model, it would have yielded highly accurate results.

Edwin Sin and Lipo Wang (2017) [41] proposed a hybrid model based on an ensemble of artificial neural network (ANN)s called Genetic Algorithm based Selective Neural Network (GASEN), with multilayer perceptron (MLP) as the base model, to explore the correlation between Bitcoin features (such as the volume of transactions, cost per transaction, etc.) and the next day change in Bitcoin prices. The proposed model solves the binary classification problem of how the next day prices of Bitcoin will change when influenced by features of Bitcoin. The study concluded that the model had a good performance and solved the classification task with an accuracy of 58%-68%. The hybrid was applied with a simple trading strategy to invest and yielded appealing results as it helped in making a profit of 85%.

Chih-Hung Yu-Feng et al. 2018 [42] proposed a new foresting framework for predicting daily prices of bitcoin LSTM model. The performance of the proposed LSTM with AR(2) model is compared with that of the conventional LSTM model using the evaluation metrics Mean squared error (MSE), root mean squared error (RMSE) and mean absolute error (MAE). The study concluded that the forecasting accuracy of the proposed model is excellent and the model outperforms the conventional LSTM model. The study also improves the problem of selecting appropriate input variables that are required for efficiently training the LSTM.

Indera Yassin et al. 2017 [43] proposed a neural network-based Nonlinear Autoregressive hybrid model with exogenous input (NARX) to predict bitcoin prices using Moving Average (MA) technical indicators over different intervals of days and past bitcoin prices. The number of hidden units, input delay (lag), and the output delay (lag) of the NARX model were optimized using a Particle Swarm Optimization (PSO) algorithm. The hybrid predicted the next day Bitcoin prices with high accuracy.

Ravi Dadabada Pradeepkumar et al. 2018) [44] proposed two chaos-based hybrid models that are optimized using multiobjective particle swarm optimization algorithms (MOPSO) and Non-dominated Sorting Genetic Algorithm-II (NSGA-II) respectively, for predicting the future prices of 3 international exchange rates (JPY/USD, GBP/USD, EUR/USD) and gold prices. Both hybrid models ($Chaos + MLP + MOPSO$) and (Chaos+MLP+NSGA-II) detect and model the presence of chaos in financial time series before training the MLP neural network, which is the base prediction model used in this study. They concluded that Chaos+MLP+NSGA-II outperformed $Chaos + MLP + MOPSO$, their stand-alone models (Chaos+MLP and MLP), which are not optimized, in terms of both performance and accuracy. The modeling of chaos by reconstructing phase space for the specified time series before training MLP improved the performance of both hybrids.

2.2 Machine learning and deep based bitcoin price prediction models

Aniruddha Dutta et al. 2019 [45] proposed a gated recurrent unit (GRU) model to predict daily Bitcoin prices. This study investigated the framework of the Bitcoin market using simple trading strategies and found that the GRU model can be more efficient than other advanced machine learning prediction methods if trained properly using crucial and important information. They trained a simple neural network multilayer perceptron (MLP), an LSTM network, and a GRU model using Bitcoin closing price data, and compared the performance of all three models in terms of RMSE value. The prediction results of all three models showed that the GRU outperformed MLP with RMSE of GRU being 0.010 and RMSE of MLP being 0.020. Both GRU and LSTM were equal in terms of performance with the same RMSE value of 0.010. They concluded that complex recurrent neural networks such as LSTM and GRU can effectively learn nonlinear patterns, even with the limited data, than other traditional machine learning and deep learning models, if they are trained properly by tuning accurate hyperparameters.

Madan et al. 2015 [30] conducted investigations to investigate daily Bitcoin prices by applying random forest and generalized linear machine learning models. The investigations were done in two different phases. In the first phase, the study gained insights into the features that surrounded the bitcoin price and identified daily trends in the market. This valuable information was fed to the random forest and generalized linear models to predict upward or downward movements in the daily Bitcoin prices. The model predicted the sign of daily price movements with an accuracy of 98.7%. The second phase evaluated predictions of the first phase at different levels of granularity and noise. In the second phase, they leveraged the results of both random forest and generalized linear models to model the Bitcoin price prediction problem as a binomial classification task. The research was successful at predicting the sign of future price movements with an accuracy of 50-55%.

Devavrat Shah et al. 2014 [46] proposed a prediction model based on Bayesian regression for predicting variations in Bitcoin price. This study utilized Bayesian regression for "latent source model" to devise a Bitcoin trading strategy. This prediction model, combined with the trading strategy was able to make predictions that doubled the investments in 60 days.

Huisu Jang and Jaewook Lee. 2017 [31] proposed a model based on Bayesian neural networks (BNN) to analyze bitcoin price time series and the relevant features from blockchain information that influence bitcoin's demand and supply. The authors found that the BNN model performed better than other models based on benchmark traditional methods such as log price and log volatility. Unlike benchmark models, the BNN model was successful at predicting the direction of the movement of prices with high accuracy. The limitation of this study is that it didn't model the variability of bitcoin prices and the nonlinear dynamics that were present in the bitcoin market. Although this study contributed to the analysis of the Bitcoin time series at that and in predicting the sign of bitcoin price movements, it failed to predict the future prices of bitcoin.

McNally et al. 2018 [29] proposed two models; Bayesian optimized recurrent neural network (RNN) and Bayesian optimized LSTM (Long short-term memory) model to predict the direction of Bitcoin price movements. They compared the performance of both models with the traditional ARIMA time series forecasting model. In terms of accuracy, the LSTM (52.78%) model outperformed both RNN (50.25%) and ARIMA(50.05). The performance was evaluated in terms of RMSE value. The RNN, in terms of RMSE, outperformed both models with the lowest RMSE of 5.45%. For the second position, LSTM followed RNN with an RMSE of 6.87% whereas ARIMA as at last position with an RMSE of 53.74%. The study concluded that while both Bayesian optimized LSTM and RNN are more effective than ARIMA at forecasting bitcoin prices, LSTM can effectively recognize long-term dependencies in bitcoin prices than the RNN.

However, if the models are overfitted and the dropout is not optimized to guarantee good validation results, and if the variance of noise present in the time series is very high, then the models would become ineffective at making good predictions. Despite the performance of LSTM being better than RNN marginally, RNN takes less time than LSTM to train.

Pichl and Kaizoji (2017) [47] conducted an empirical invention into the volatility of the Bitcoin price time series. They used a feed-forward neural network to predict the daily log-returns of bitcoin prices in USD. Also, the logarithmic return distribution and translation volume distribution of bitcoin prices were used to analyze volatility. They also compared the volatility of bitcoin time series with EUR/USD exchange rates and found that bitcoin prices are more volatile than EUR/USD exchange rates as there are abundant bubbles in the market. The proposed model was capable of learning the statistical distribution, but its prediction ability was limited while predicting the bitcoin market trends.

Thearasak Phaladisailoed and Thanisa Numnonda (2018) [48] compared the accuracy and efficiency of various machine learning models at predicting fluctuating bitcoin prices. The study compared two best machine learning-based regression models; Theil-Sen Regression and Huber Regression models, and two deep recurrent neural networks based models; LSTM and GRU, in terms of mean squared error (MSE). The comparison concluded that deep learning-based models; GRU and LSTM outperformed and predicted better results than both Theil-Sen regression and Hubber regression model. Out of all the models, the MSE of GRU was the lowest. From the obtained results, the study infers that the GRU is the most accurate model for predicting bitcoin prices among all the compared models but the Huber regression model takes the least time to train and computes results faster than the other three models.

Anshul Saxena and T.R. Sukumar. 2018 [49] proposed deep learning based LSTM predictive model for predicting bitcoin prices and compares its performance with the traditional ARIMA model. The model is trained using only the closing price of bitcoin. The study revealed that the proposed LSTM model captures longer-term dependencies and learned faster than the ARIMA model. However, LSTM took more time than ARIMA to train. The loss was minimal for LSTM. The LSTM model had a relatively low RMSE value (458.78) than the ARIMA model (700.69), which proves that the LSTM model is better than the ARIMA model and should be preferred over it to predict bitcoin prices. The model proposed by this study, however, has very low accuracy and high RMSE, as compared to the same LSTM models proposed by other studies. This may be due to the fact that the author trained the LSTM model only with the closing price of bitcoin. If other factors such as public perception, market sentiments, government policies, etc. or other advanced techniques were combined while training the model, its performance, and predictability would have increased.

Alex Greaves and Benjamin Au. 2015 [50] used different regression models; Logistic Regression, Support Vector Regression(SVR), and a simple feed-forward neural network to predict one-hour future prices of bitcoin by using blockchain-network features of bitcoin transactions. The study predicted a one-hour future by utilizing prices by adding the average percent increase in price per hour to the current price. A baseline model was used for the comparison. The 2-hidden-layer neural network had the highest accuracy (55.1%) compared to all the models, followed by the Logistic regression model (54.3%), followed by SVM (53.7%) and the baseline model with least accuracy (53.4%). Even though the neural network classifier had the highest accuracy at predicting the increase or decrease in the bitcoin prices, it is only slightly higher than that of the baseline model. The study found that the two most useful features of blockchain that influenced bitcoin's price the most were the 'net flow through an account per hour' and the total number of transactions made by new addresses per hour', both these features were highly correlated.

Peter T Yamak et al. 2019 [51] proposed and compared ARIMA, LSTM, and GRU models for predicting bitcoin price time series. The study found that the ARIMA model outperformed both LSTM and GRU, and gave the best predictions with the lowest RMSE and MAPE values. However, GRU predicted prices better than LSTM. As shown in previous studies, the RNN models should have performed better than the ARIMA model. But in this study, the case is the opposite. This study used a very limited amount of data while training the models. Since LSTM and GRU can learn hidden patterns when they are fitted with sufficient information, but due to the insufficiency of data while training these models, these models failed to provide accurate results. Considering other factors such as blockchain information, social and political factors, etc would have increased the performance of both LSTM and GRU.

Suhwan Hyeonseung et al. 2019 [32] developed and compared various ensemble models and hybrid models of convolution recurrent networks (CNN) and recurrent neural networks (RNN), and various state-of-the-art artificial intelligent models such as LSTM, deep neural networks (DNN) and the combination of CNNs and RNNs (CRNN), for solving bitcoin price prediction problem. They trained and test previously described regression models (LSTM and DNN) and classification models (CNN, deep residual network; ResNet, and ensemble model) by using the bitcoin blockchain data and compared predictions of different models with each other to evaluate their performance with respect to each other. For the regression problem i.e. to predict the future price, the LSTM model performed better than other models, but deep neural network-based models outperformed models for the classification problem i.e. to predict the direction of price movements (upward or downward). Moreover, they applied the regression models to make profits by developing a simple trading strategy i.e if the price predicted by regression models is greater than or near the current price, they would buy the bitcoin. Otherwise, they sold the bitcoin. They doubled the investment in less than 30 days by using this strategy. Although CNN and ResNet were good for predicting the direction of bitcoin price movements, they were not efficient at predicting the future prices of bitcoin.

Roy Chakrabarty et al. 2018 [52] predicted bitcoin prices 10 days into the future by using different traditional time series models that use statistical analysis methods. These models are based on autoregressive moving averages methods such as Autoregressive Integrated Moving Averages (ARIMA), Autoregressive Moving Average(ARMA), and Moving Averages (MA). The study predicted prices using all of these models and compared their results. After analyzing the results, they found that ARIMA was the best model among all these three models as it had the highest performance with an accuracy of 90.31% while the Moving Averages model had the least performance with an accuracy of 87.58%.

Monteiro Zaparoli et al. 2019 [53] provided a general vision for predicting cryptocurrencies by a system mapping approach. This study identifies all the existing gaps in the current research of the cryptocurrencies market and provides a clear revision of cryptocurrencies price prediction techniques for an interdisciplinary perspective, by carrying technical and fundamental analysis. It mapped all the previous researches related to cryptocurrencies using a systematic mapping process and excluded all the technological aspects of blockchain that don't play important roles in influencing highly volatile price fluctuations. This study contributed a state-of-the-art perspective for cryptocurrencies prediction methods and also listed future research opportunities in this domain.

Tian Guo and Nino Antulov-Fantulin (2018) [54] proposed a temporal mixture model which uses buy and sell orders data of bitcoin of different time scales to predict the short-term volatility in bitcoin price. Before training the model, the order book data was reformatted into a feature series and a volatility series. The volatility series was obtained by modeling the volatility present in the bitcoin orders data series. The temporal mixture model was efficient at capturing the dynamic effects of the order book features on the change of volatility in prices with time. The study also compares the prediction performance of the temporal mixture with the conventional time series models and ensemble models. They found that the conventional time series model predicts the volatility in prices with degraded performance, but the temporal mixture model performs and predicts better results than both ensemble and time series models. Moreover, they studied the robustness of the proposed model by utilizing rolling and incremental learning and evaluation techniques with respect to the historical data and found that even with the redundant historical time-varying data, the temporal model shows great performance in predicting the volatility of the bitcoin market. But regression and ensemble models showed degraded performance while predicting the volatility in the redundant historical data.

2.3 Chaos detection methods

Salim Lahmiri and Stelios Bekiros (2018) [4] provide the first attempt to investigate and detect the presence of chaos in the bitcoin prices time series. This study investigated the state of the bitcoin market during two time periods; low-regime, where the change in the prices of bitcoin was very low, and a high-regime period where prices increased exponentially. From the application of the largest Lyapunov exponent (LLE) test, they found that the chaos was present during both time periods. Furthermore, they also found that the price fluctuations were very high during the low price regime period and prices showed a high correlation. However, the chaos was highest during the high price regime period as compared to the low price regime period, and the bitcoin market exhibited very complex nonlinear dynamical patterns during this period.

Ahmed BenSaïda Litimi (2013) [1] rigorously investigated the presence of chaos in highly noisy time series financial data of six stock indexes, and six international exchange rates by exploiting the application of the largest Lyapunov exponent (LLE) test. To provide strong evidence about the consistency of the test they performed simulations on a know chaotic system; a logistic map, a highly noisy times series (random numbers generated by random number generator), and the six stock indexes (S&P, 500, NASDAQ, Nikkei 225, CAC 40, FTSE, DAX) and six international exchange rates (USD/EUR, USD/JPY, USD/JBP, USD/CAD, USD/SEK, and USD/CHF). According to the results of the test, it was concluded that the chaos was absent in all the six stock indexes and international exchange as the value of Lyapunov exponent was negative for all of them. By evaluating the system dynamics of those financial markets, the proposed methodology was successful at making a distinction between stochastic and chaotic systems. The study provides strong evidence that the specified stock indexes and international exchange rates were stochastic in nature, not chaotic.

Chapter 3

PROPOSED METHODOLOGY

The whole methodology for our proposed chaos-based hybrid prediction model is depicted in Fig A. In our hybrid model, Chaos theory (along with Lyapunov exponent estimation and Takens's Embedding Theorem) is used for phase space construction from bitcoin time series $[x(t)]_{t=1}^{\tau}$ with 'τ' data points. The proposed model is a three-stage prediction model. Each stage proceeds as follows:

Stage-1 (Detection of chaos)

In this stage, we check for the presence or absence of chaos in time series $x(t)$. The Lyapunov exponent (λ) for $x(t)$ is estimated using Saida's method. The positive or negative or negative amplitude of exponent reveals the presence or absence of chaotic dynamics in $x(t)$.

Stage-2 (Phase Space Reconstruction)

If the chaos is present in one-dimensional time series $x(t)$, it will be projected to higher dimensions. The m-dimensional phase space is constructed for $x(t)$ by using Taken's method. As Taken's method uses the time delay L and the minimum embedding dimensions m, we need to determine the appropriate values of L and m first. The time delay L is computed using the Average Mutual Information (AMI) method and the False Nearest Neighbors (FNN) is used to compute embedding dimensions m. The reconstructed higher m-dimensional equivalent of the time series

$x(t)$ is represented by the following equation:

$$X(t) = x(t), x(t+L), x(t+2L), ..., x(t+(m-1)L) \tag{3.1}$$

where t = 1, 2, 3, ... , M and M = τ - (m-1)L.
Thus, the matrix representation for input samples is represented by matrix X as follows:

$$X = \begin{bmatrix} x(1) & x(1+L) & x(1+2L) & .... & x(1+(m-1)L \\ x(2) & x(2+L) & x(2+2L) & .... & x(2+(m-1)L) \\ . & . & . & .... & . \\ . & . & . & .... & . \\ . & . & . & .... & . \\ x(M) & x(M+L) & x(M+2L) & .... & x(M+(m-1)L) \end{bmatrix} \tag{3.2}$$

For target set or output set, numerous previous studies [55] have used the data points [$x(t) = t+1+(m-1)L$; $t = 1$ to M] i.e.

$$y = \begin{bmatrix} x(2+(m-1)L) \\ x(3+(m-1)L) \\ . \\ . \\ . \\ x(M+(m-1)L) \end{bmatrix} \tag{3.3}$$

Many studies [55] have used the concept of multiple time scales for constructing the target set. In this study, for the better prediction of m-dimensional PSR series, we used the concept of multiple time scales to compute a better target set (averaging each row of the input set of $m^t h$ dimension). Data points of the target set, accordion to our approach, are computed by using the following equation:

$$T_i = \frac{1}{m} \sum_{i=1}^{M+(m-1)L} x_i \tag{3.4}$$

Thus, the target or output matrix (Y) according to Eq. 3.4 can be represented as:

$$Y = \begin{bmatrix} \frac{1}{m}[Sum(x(1) & x(1+L) & x(1+2L) & .... & x(1+(m-1)L))] \\ \frac{1}{m}[Sum(x(2) & x(2+L) & x(2+2L) & .... & x(2+(m-1)L))] \\ . & . & . & .... & . \\ . & . & . & .... & . \\ \frac{1}{m}[Sum(x(M) & x(M+L) & x(M+2L) & .... & x(2+(M-1)L))] \end{bmatrix} \tag{3.5}$$

Both of the matrices X and Y are divided into the training set and the test in the ratio 80:20 or 4:1.

Stage-3 (Prediction)

The LSTM model of recurrent neural networks is used for making predictions using different algorithms described in Section 3.3. Mean Squared Error (MSE) is used as a metric for the evaluation of the performance of the model.
The flow chart for the proposed methodology is depicted in Figure 3.1.

3.1 Detection of chaos: Lyapunov exponent computation

The Lyapunov exponent indicates the presence or absence of chaos in a dynamic system. The idea behind the Lyapunov exponent is defined as: **sensitive dependence on initial conditions** is a binary distinction, a system may or may not have sensitive dependence on initial conditions. However, it is reasonable to suspect that some systems are more sensitive to initial conditions than others. The Lyapunov exponent is a way to measure sensitive dependence on the initial conditions of a particular system.
Consider two initial conditions, X_0 and $X_0 + \triangle x_0$, each of them will act as a separate point in a space state. These points are separated by an initial difference $\triangle x_0$. As we iterate forward in time using some dynamical system equations, the initial difference will change in time and generate a path in that state space. The difference between the two orbits is a function of time. The separation between two paths either grows or shrinks with time. If it grows, then there is a sensitive dependence

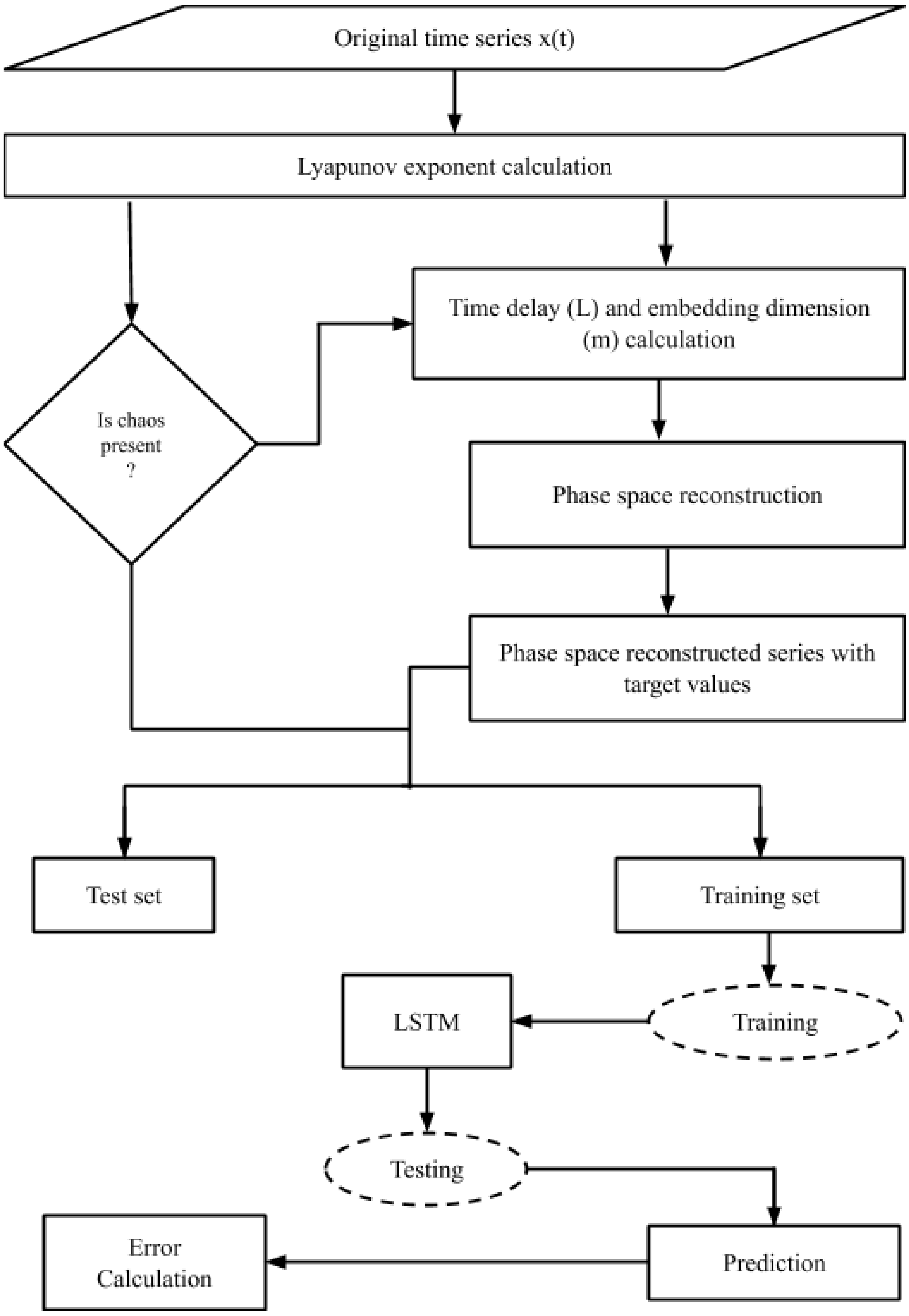

Original time series x(t)
Lyapunov exponent calculation
Is chaos present ?
Time delay (L) and embedding dimension (m) calculation
Phase space reconstruction
Phase space reconstructed series with target values
Test set
Training set
Training
LSTM
Testing
Prediction
Error Calculation

Figure 3.1: Proposed methodology for the hybrid prediction model

on initial conditions and if it shrinks, the system moves towards a stable attractor. Also, consider the path generated by point X_0 as a reference orbit. After some time t, the two paths get separated by some distance which is dependent on the initial position of X_0 and time t. It implies that the difference $\triangle x_0$ at time t is a function of X_0 and t i.e $\triangle x(X_0, t)$. For systems with sensitive dependence on initial conditions, $\triangle x(X_0, t)$ will grow exponentially as:

$$\triangle x(X_0, t) = X_0 2^{\lambda t} \tag{3.6}$$

Such a system is unstable or chaotic i.e. the two paths diverge exponentially with time. If $\triangle x(X_0, t)$ for a system decreases asymptotically with time, then the paths converge towards a fixed point or attractor, making it a stable or non-chaotic system. The trajectories of paths generated in space by initial points X_0 and $X_0 + \triangle x_0$ are depicted in **Figure 3.2**.

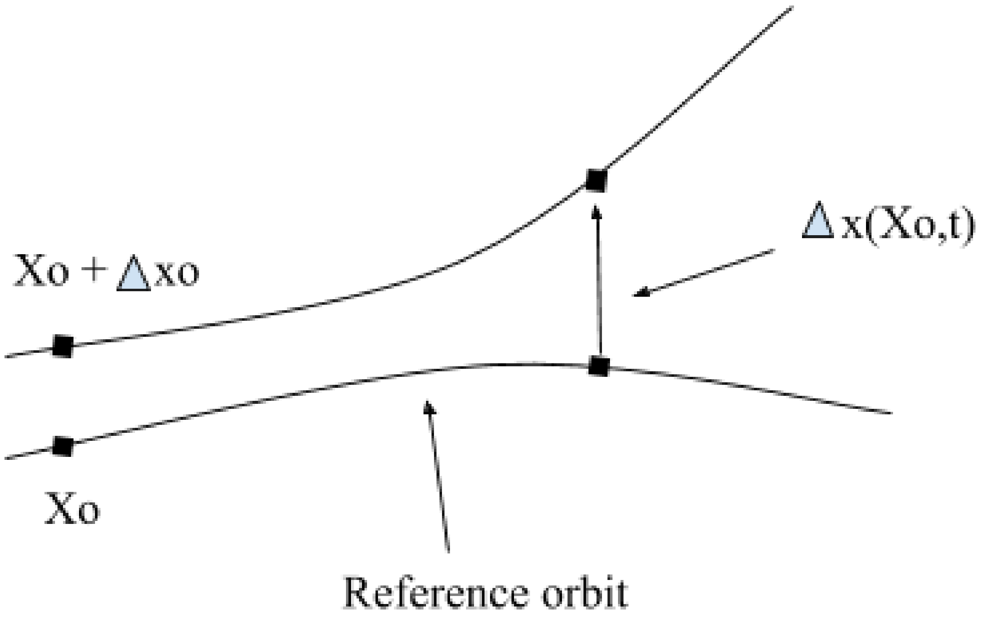


Figure 3.2: Trajectories of paths traced by points X_0 and $X_0 + \triangle x_0$ with time.

The Lyapunov exponent measures the difference $\triangle x(X_0, t)$ between the two trajectories. In time series, the paths are created by the change in the value of a time-variant variable with each time step. For our time bitcoin time series, the paths are created by the movements in the price of bitcoin with time. For a specific period of time, the initial conditions on which the price of bitcoin is sensitively dependent are the information present at the beginning of that period; sentiments from public and investors, prices of the asses that influence bitcoin's price, investor perception towards other cryptocurrencies, perception towards risk, socioeconomic factors, political factors, etc. The sign of the Lyapunov exponent decides whether a dynamical system is chaotic or stochastic. The estimation of the *largest Lyapunov exponent* (Lyapunov exponent henceforth) is done by the methodology developed by Bensaïda in [1]. The advantage of this method is that λ for a system can be estimated without the need for dynamic equations i.e. experimental alone is enough to determine λ.

The Lyapunov exponent λ is distinguished by 3 cases depending upon whether it measures convergence or divergence between trajectories:

- $\lambda > 0$: The positive exponent means a system has chaotic dynamics. The trajectories diverge exponentially with time making the orbit unstable and chaotic. The paths will always diverge by arbitrary difference, no matter how close the points are initially or the distance between initial points is infinitesimally small. The larger the value of λ, the more unstable, chaotic, and complex the system, and the more the sensitive dependence on initial conditions.

- $\lambda = 0$: A system with zero value of exponent has an orbit with a fixed neutral point. This system is near the *transition to chaos.*

- $\lambda < 0$: A system with a negative exponent is a stable system. The trajectories converge asymptotically with time, making the system non-conservative. The orbit of such a system is stable and gets attracted to a fixed point. The more negative the value of λ, the more is the stability. A super-stable system has an infinitely negative value of Lyapunov exponent ($\lambda = -\infty$), e.g. purely random numbers.

Consider a noisy chaotic system with a scalar time series $[x_t]_{t=1}^{\tau}$. The Lyapunov exponent can be determined by conducting a test given the hidden noise on the reconstructed time series:

$$X_t = f(x_{t-L}, x_{t-2L}, x_{t-3L}, ..., x_{t-mL}) + \varepsilon_t \tag{3.7}$$

where L = time delay,

m = embedding dimensions,

ε_t= introduced noise,

f = function to approximate an unknown chaotic map.

The chaotic map is approximated using a neural network. f(.) is basically the activation function of that neural network [56]. The variance of introduced noise ε_t (σ_ε^2) is used to measure the amplitude of noise that is originally present in the time series. For a noise-free dynamical system, the value of variance is zero i.e. $Var(\varepsilon_t) = \sigma_\varepsilon^2 = 0$. As the amount of added noise is increased, the variance also increases. If the noise in a chaotic system is above a certain threshold, it becomes unmanageable, and the detection of chaos becomes very difficult or near impossible as the chaotic dynamics get absorbed by the noise. The chaotic systems with very high levels of noise become stochastic due to the acquisition of property of noise. The threshold for noise is computed by using the skeleton chaotic map [57]. With the increase in the amplitude of the added noise, the Lyapunov exponent decreases and the system tends to acquire stochastic dynamics gradually. It implies that if λ_0 is the Lyapunov exponent of a noise-free skeleton of a chaotic system, the Lyapunov exponent λ_ε of a noisy system will gradually tend to increase towards λ_0 as the amplitude of the noise tends to zero i.e. $lim_{\sigma_\varepsilon^2 \to \infty} \lambda_\varepsilon = \lambda_0$. So λ_ε is always less than or equal to λ_0 i.e. $\lambda_\varepsilon \leq \lambda_0$

In Bensaïda [1], the Lyapunov exponent is estimated using the Jacobian based approach. Using this approach, the equation for computing Lyapunov exponent is given as:

$$\lambda = \frac{1}{2N} ln(v_1) \tag{3.8}$$

where v1= largest eigenvalue of the matrix $(\tau_N G_0)'(\tau_N G_0)$, such that $G_0 = (1, 0, 0, ..., 0)'$, and $\tau_N = \prod_{t=1}^{N-1} J_{N-t}$ is the dot product of Jacobian matrices J_t for the specified range.

$$J_t = \begin{bmatrix} \frac{\partial f}{\partial x_{t-L}} & \frac{\partial f}{\partial x_{t-2L}} & \frac{\partial f}{\partial x_{t-3L}} & \cdots\cdots\cdots & \frac{\partial f}{\partial x_{t-mL+L}} & \frac{\partial f}{\partial x_{t-mL}} \\ 1 & 0 & 0 & \cdots\cdots\cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots\cdots\cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots\cdots\cdots & 0 & 0 \\ . & . & . & \cdots\cdots\cdots & . & . \\ . & . & . & \cdots\cdots\cdots & . & . \\ 0 & 0 & 0 & \cdots\cdots\cdots & 1 & 0 \end{bmatrix} \tag{3.9}$$

The unknown chaotic map f is approximated by using a neural network on a scalar time series $[x_t]_{t=1}^{\tau}$. This neural network has q-dimensional hidden layers and uses a particular activation function to make estimations. It can be:

- **'tanh' (hyperbolic tangent activation):**
  $g(u) = tanh(u), domain = (-1, 1)$
  It is the most powerful and efficient activation function for approximations.

- **'logistic' activation:**
  $g(u) = \frac{1}{1+e^{-u}}$, $domain = (0, 1)$
  It is least efficient for handling numerical estimation problems and not recommended.

- **'sigmoid' activation:**
  $g(u) = \frac{1+|\frac{u}{2}|}{2+|u|+\frac{u^2}{2}}$, $domain = (-1, 1)$

With the increase in the number of hidden layers q, $q \to \infty$, the approximation power of a neural network increases, and it can approximate any noise-free any noise-free and nonlinear functions with some arbitrary accuracy. We have used hyperbolic tangent (tanh) activation as it is the most powerful function for noisy time series and numerical approximations. The chaotic map approximated by the above neural network is given by the following equation:

$$x_t = \alpha_0 + \sum_{i=0}^{q} \alpha_i tanh(\beta_{0.i} + \sum_{j=1}^{m} \beta_{j-1} x_{t-jL}) + \varepsilon_t \tag{3.10}$$

The order vector *(L,m,q)* decides the complexity of the neural network and chaotic map approximated by it. Theoretically, a neural network can estimate any nonlinear function when the number of hidden layers q tend to infinity, but it increases the

computational complexity for approximating chaotic maps. The value of time delay L must be such that it preserves time dependence for the given time series.
For an excessively large value of L, much of the information gets lost and a very small value causes the points in state space to become very close, which decreases the value of Lyapunov exponent, compromising the chaotic dynamics of time series. Low orders *(L,m,q)* degrade the approximating power (to approximated chaotic map) of the neural network, and excessively large orders increase approximation time. [1] developed a strategy to select optimal order *(L,m,q)* to minimize Schwarz Information criteria by selecting the orders as (5,6,5). However, this strategy is inefficient as it tends to ignore high-level chaotic dynamics. In addition, Bensaïda [1] conducted a simulation using the same parameters (5,6,5) and found that it tends to eliminate the chaotic dynamics when the initial conditions are changed.
The wise approach to select the optimal order *(L,m,q)* is to choose all possible combinations of sufficiently high parameters *(L,m,q)* that don't exceed, yet detect the presence of high-order chaotic dynamics, without adding unsolicited coefficients. The next step is to calculate the Lyapunov exponent for all possible orders *(L,m,q)*. The order that gives the highest value of the exponent is kept. Bensaïda [1] suggests that the optimal range for orders is between any of (5,6,5) to (10,12,10).
Before conducting the test for computing λ, we first define two test hypotheses H.
Null Hypothesis $H_0 : \lambda \geq 0$
If H_0 gets accepted, it indicates the presence of chaotic dynamics and its rejection provides the evidence that a system is stochastic (chaotic dynamics are absent).
Alternate Hypothesis $H_1 : \lambda < 0$
It provides evidence of the absence of chaos.
The Lyapunov exponent will be computed at some confidence level α for our bitcoin time-series data to decide whether it is chaotic or stochastic. The positive amplitude of λ will indicate that chaos in the time-series, negative amplitude the absence of chaos, and if time series has very low chaos or near the transition to chaotic dynamics, the value of λ will be zero or almost zero.

3.2 Phase space & Attractor reconstruction

A phase space is a multidimensional representation of a dynamical system whose state changes over time. Every individual point in the phase space represents the state of the system at any particular time. Each axis of the multidimensional phase space represents a parameter (or dimension) of the system. If a system is modeled using only one dimension, it is called a *phase line*. It is called the *phase plane* when modeled using two dimensions. There is a point in the phase space for every possible state of the system. As the system evolves over time, the evolving states trace a path through space, each path following an arbitrary trajectory. A trajectory is a set of the system's states starting from an arbitrary initial condition. It may require a great number of dimensions to construct phase space for a system depending upon the complexity of the system. The phase space of a system may contain as many dimensions as the number of parameters that are required to model the system.
Let us take the example of the bitcoin market and suppose we want to model the relevant information and parameters that describe the dynamics of this system. In the case of the bitcoin market, there are various degrees of freedom or fundamental parameters like the public perception, investor's perception towards risk, price of other cryptocurrencies, prices of related assets, etc. that govern the fluctuations in bitcoin's price. If we want to reconstruct phase space for the price movements of bitcoin's price over the 't' amount of time, it will require as many dimensions as the number of parameters that influence bitcoin's price directly or indirectly over that period of time. With this set of dimensions (or parameters), we have the ability to fully describe the state of the market at any point in time. In this case, by describing the state of the market, we mean that given an initial condition (price) we can compute the price at any point 't' in time in the future. As we move forward in time, the stock price will change with every time step. This evolution of price with time traces a path in the phase space that follows some arbitrary trajectory. The dimensions of the phase space are entrained which is to say there is a strict correlation between them. Although the parameters that influence the price may be somewhat causally connected, there is a high-quality relation between them which decides how the price will change over time, and what path will the movement of price trace in the phase space. The phase space required to model the whole bitcoin

market will be enormously large because it has a vast number of degrees of freedom or the parameters and all the possible states that together describe the dynamics of this system. But since we want to model only how the stock price will change over time, the phase space to model price movements can be constructed by sampling only those dimensions that influence only price movements. It is a very useful thing to do because it will simplify the model a lot. Thus the phase space for only the price movements is finite and bounded if we consider the current scenario of the bitcoin market which is to say that as long as the new parameters that might influence stock price don't emerge, we can model price movements using a finite number of dimensions. But in the future, the new parameters that would affect the price movements might in fact emerge because of the continuous change in public perception about bitcoin, and their increasing awareness and understanding of it. This will make the phase space infinite and unbounded as new kinds of possible variables can emerge inside certain dimensionality. In fact, in principle, one can actually imagine the emergence of entirely new degrees of freedom, entirely new dimensionalities, obviously in the situation of evolution which can happen at any time e.g. the emergence of new strategies, laws, or the rules and regulations that might influence prices in the future. But for the current market scenario, this is out of the scope of this study. The main idea here is that *a complicated system is a system that has a finite and bounded phase space, but a complex system does not. It has an unbounded phase space.*

Now to understand Taken's theorem, let us first understand phase space reconstruction using Lorenz's system.

3.2.1 Lorenz system equations

The Lorenz is an example of a dynamical system consisting of three differential equations where each component depends on the state and the dynamics of the other two components. These equations model convection with only three variables that change over time. The components are actually the state variables or the Cartesian coordinates that form the state space. The Lorenz system has three equations where variables depend only on time. The equations are:

$$\frac{\partial x}{\partial t} = \sigma(y - x) \tag{3.11}$$

$$\frac{\partial y}{\partial t} = \rho x - y - xz \tag{3.12}$$

$$\frac{\partial z}{\partial t} = xy - \beta z \tag{3.13}$$

The values of parameters (σ, ρ, β) for which this system exhibits chaos are (10, $\frac{8}{3}$, 28). The chaos present in a system is usually unpredictable in the short-term. The phase space representation of the above system is depicted in **Figure 3.3**.
Let's try to understand phase space representation in 3-dimensions. While constructing the phase space of a system, the location of the first point (or initial state of the system) is arbitrary, and with each point after that, we step forward in time by calculating the position of the next point based on the position of the previous one. The path traced by those points changes direction at random over time and never arrives at the same point twice (if it did that would be the beginning of a repeating pattern). This type of system is known as a strange attractor or a Lorenz attractor. Suppose we want to construct a phase space using 100 points at random locations by applying equations 1,2, and 3 to them. At first, the trajectory of every individual path started from a respective initial point seems random. Every trajectory that begins at some initial point at some point in time, follows a random path in the beginning, but after some time all these trajectories get attracted to and end up in an attractor. It might seem that the differences in their starting positions no longer matter, but once they reach the attractor, the trajectories diverge, each following its own path indefinitely and never crossing again. So the small differences in their start positions lead to increasing different futures.

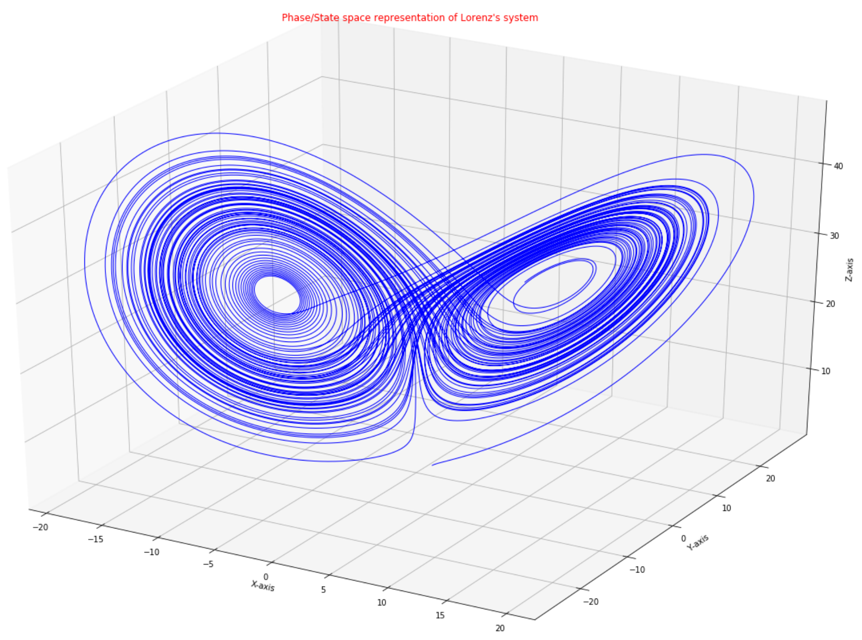


Figure 3.3: Phase space representation of Lorenz time series.

The phase space representation in **Figure 3.3** is generated in Python using 9000 data points of Lorenz time series, generated by equations 3.10, 3.11, and 3.12 with initial conditions *(x,y,z)* as (1,1,1) and values of (σ, ρ, β) as (10,$\frac{8}{3}$,28). This phase space contains three variable or Cartesian coordinates *(X, Y, Z)*. Each point in the orbit of the attractor is at some coordinate *(X, Y, Z)*. As we move forward in time, plotting subsequent points $(X_i, Y_i, Z_i;\ i = 1 \text{ to } N)$ for each time step *t; t=1* to , we get a set of trajectories. These trajectories together constitute an attractor also known as Lorenz's butterfly. When the trajectory of the attractor in phase space is projected on the *X-axis*, we get the chaotic measurement or a time series $[x_t]_{t=1}^{\tau}$. This measurement is chaotic as it never repeats itself as shown by **Figure 3.4**

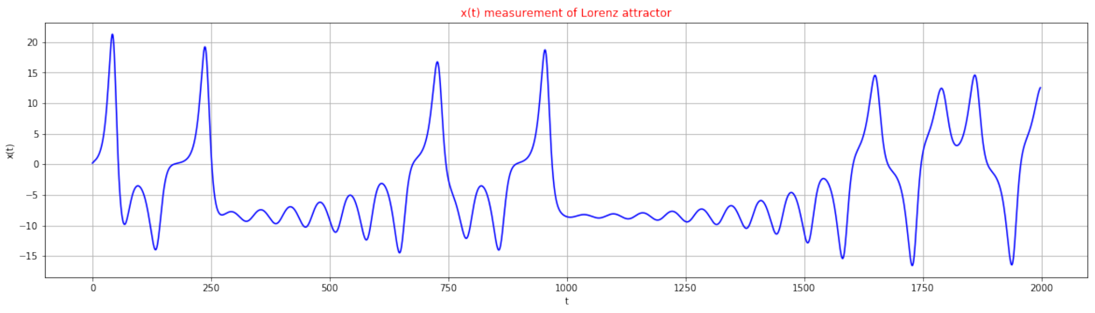


Figure 3.4: Projection of Lorenz attractor in x-axis for 2000 data points

Figure 3.3, the butterfly-like attractor is a manifold. The manifold M consists of the set of all trajectories. Each point in a trajectory is represented by *m(t) = (X(t), Y(t), Z(t))*, and the flow (ϕ) of points on the manifold M is defined by some coupled equations.

We can view a time series then as a projection from manifold M onto a coordinate axis of the phase space. A time series on *X-axis* can be generated by projecting the displacements of points on the manifold to it. This can be repeated on other coordinate axes to generate other simultaneous time series for those axes. So these time series are really just projections of the manifold dynamics onto coordinate axes. Conversely, we can recreate the manifold by projecting the individual time series simultaneously back into the phase space to create the flow. Each of the three time series *X, Y, Z* is the projection of the motion on the manifold *M*.

3.2.2 Taken's Theorem

It is a powerful theorem for non-linear dynamics because it allows having access to other dynamical variables of a system through only one-dimensional measurement. Taken's theorem proven in [58] by Forrest Taken can reconstruct a shadow version of the original manifold simply by using a univariate time series, generated by its projection on a respective coordinate axis. Each point in the m-dimensional reconstruction can be thought of as a time segment with different points capturing different segments of the history of some variable (say X). The reconstructed variable is then the library or collection of the historical behavior of X. The reconstruction preserves essential mathematical properties of the original system such as the topology of the manifold and its Lyapunov exponents. Taken's theorem gives a one-to-one mapping between the original manifold M and the reconstructed shadow manifold (say M_x). It allows recovering the states of the original dynamical system by using lags (time delays) of just a single time series. Now let us discuss phase space reconstruction using Taken's theorem.

3.2.3 Phase space reconstruction

The phase space reconstruction is a very powerful and base method for analyzing chaotic time series. The time series for chaotic non-linear dynamical systems are analyzed to understand the system dynamics by reconstructing the time-delayed phase from them. In unstable chaotic systems, univariate time series can be reconstructed into a multivariate phase space that is just a representation of the original system in higher dimensions. This reconstruction is possible because a univariate time series contains all the information that its multivariate time series equivalent would contain. In phase space, each point represents a state of a chaotic system at some point in time and the trajectory represents how the system fluctuates with time according to different initial conditions.

According to Taken's delay-embedding theorem, an m-dimensional phase space can be unfolded from a one-dimensional chaotic time series. This is accomplished by using an observed scalar time series, $[x_t]_{t=1}^{\tau}$, observed from a chaotic system. The time-delayed version of $[x_t]_{t=1}^{\tau}$ can be utilized as coordinates to reconstruct phase space. The phase space vectors to trace out the orbit of m-dimensional phase space are expressed as:

$$X(t) = x(t), x(t+L), x(t+2L), ..., x(t+(m-1)L) \tag{3.14}$$

where t = 1, 2, 3, ... , M and M = τ - (m-1)L.

Here, is the total number of time steps in one-dimensional time series, m is the number of embedding dimensions of phase space, L is the time delay by which the univariate time series is lagged, and M is the number of points in the reconstructed phase space.

The evolution of the phase space vector X(t) with time is given as $X(t) \rightarrow X(t+1)$. Each observation x(t) is a projection of some point of m-dimensional phase space X(t) onto one dimension. An essential requirement for a phase space reconstruction is that the appropriate values of time delay L and embedding dimension m must be selected. The values of time delay L determines the accuracy of the reconstructed phase space. For a very small value of L, the attractor (manifold) will get plotted along a line, which will make the points undistinguishable in phase space, and a very large L will lead to a distorted structure of the attractor. The two commonly used methods for selecting the appropriate value of time delay L are **Autocorrelation function (ACF)** and **Average Mutual Information (AMI)**. The ACF cannot be used for our nonlinear time series as it is used only for linear time series. The AMI is used for computing time delay L for non-linear time series data. Since our bitcoin time series data is chaotic and non-linear, we used the AMI method. It utilizes the non-linear correlation between the elements of a time series to estimate the value of time delay L.

Remarks. Before determining the minimum embedding dimension, the optimal time delay L must be determined as the value of embedding dimension highly depends on it. There will be different values of minimum embedding dimension for different values of parameter L, especially for a time series from a dynamical system. By choosing an appropriate value of L, the number of embedding dimensions required for phase space reconstruction can be minimized. The time delay L should be chosen in such a way that each vector in the reconstructed phase space is independent.

The average mutual information between two lagged time series X_t and X_{t+L} is calculated by the following equation:

$$I(L) = \sum_{t=1}^{N-L} P(X_t, X_{t+L}).log(\frac{P(X_t, X_{t+L})}{P(X_t).P(X_{t+L})}) \tag{3.15}$$

where $P(X_t)$ = probability density of X_t,

$P(X_{t+L})$ = probability density of X_{t+L},

$P(X_t, X_{t+L})$ = joint probability density of time series X_t and X_{t+L},

I(L) = average mutual information between the delayed and non-delayed time series.

I(L) measures the statistical dependence present between reconstruction variables. To determine the time delay L, either the first local maxima or first local minima can be chosen, depending upon whether the mutual information increases or decreases. The first local minimum is selected as L if delayed and non-delayed time series share the least information i.e. I(L) decreases nonmonotonously. When the mutual information decreases monotonously, either $\frac{I(L)}{I(0)} = \frac{1}{e}$ or $\frac{I(L)}{I(0)} = 0.2$ can be used to identify the suitable value of L.

The optimal value of the embedding dimensions m is computed by the **False Nearest Neighbor (FNN)** method. In this study, the author found that the points on attractor's orbit, due to its dense nature in phase space, acquire neighbors. The equations to predict the evolution of new points on the attractor's orbit are generated by utilizing the information about the evolution of neighborhoods of previous points in the phase space. The neighbors also provide enough information to accurately compute Lyapunov exponents of the chaotic system. The points in a phase space with very few embedding dimensions (not sufficient to unfold the attractor),

may not be the neighbors even if they seem very close to one another, because of the system dynamics. As the attractor is projected down to a smaller space, the points appear as near neighbors though they are far from each other.
If $X^r(t)$is the r^th nearest neighbor of X(t) in an m-dimensional space, the distance between these points is measured as Euclidean distance and its square is given as:

$$E_m^2(t,r) = \sum_{k=0}^{m-1}[x(t+kL) - x^r(t+kL)]^2 \tag{3.16}$$

By embedding $(m+1)^th$ coordinate to each vector of X(t) by time embedding, we can go from dimension m to dimension m+1. This added coordinate is x(t+mL). The Euclidean distance between point X(t) and its r^th nearest neighbor that was determined in m-dimension is given as:

$$E_{m+1}^2(t,r) = E_m^2(t,r) + [x(t+kL) - x^r(t+kL)]^2 \tag{3.17}$$

As we go from dimension *m* to dimension *m+1*, the distance between X(t) and $X^r(t)$ increases and becomes very large. The relative increase in Euclidean distance ∂_t between points X(t) and its neighbor $X^r(t)$ when the dimension of phase space is increased from *m* to *m+1* is computed as:

$$\partial_t = \sqrt{\frac{E_{m+1}^2(t,r) - E_m^2(t,r)}{E_m^2(t,r)}} \tag{3.18}$$

The first criterion used in FNN method for identifying a false nearest neighbor of any point X(t) is:

$$\partial_t > E_{tol} \tag{3.19}$$

where E_{tol} is a constant threshold.
The criterion in Eq. 3.18 as proved in [59] is not sufficient to determine the appropriate value of embedding dimensions *m*. This is because when this criterion is applied to examine very noisy data, it reports error. It erroneously reported that a very small dimensional space is capable of embedding noise. The corollary to the previous statement is that even when a point $X^r(t)$ is at a far distance from X(t), it will appear as a neighbor to X(t). As the number of embedding dimensions *m* is increased, the points in the orbit of attractor get further and further if the attractor is populated uniformly with a fixed number of points. Also, as the number of data

points in a noisy time series is increased, the number of embedding dimensions required to drop the number of false neighbors to almost zero will increase i.e. if the number of data points tends to infinity, the number of embedding dimensions will also tend to infinity. However, in real-world time series data, the number of data points is limited and not infinitely large. In dynamic systems with limited dataset size, if a neighbor to point X(t) is false, then on adding $(m+1)^{th}$ coordinate to the data vectors $[X(t)]_{t=1}^{\tau}$, the resultant Euclidean distance $E_{m+1}(t)$ between points X(t) and $X^r(t)$ on *(m+1)* dimensions will be $E_{m+1}(t) \approx 2E_A$, where E_A is the standard deviation of the attractor in one dimension.

$$E_A = \left(\frac{1}{\tau} \sum_{t=1}^{\tau} [x(t) - \bar{x}]^{\frac{1}{2}} \right) \tag{3.20}$$

where $\bar{x} = \frac{1}{\tau}\sum_{t=1}^{\tau} x(t)$. // The second criterion to identify false nearest neighbors when the attractor is unfolded:

$$\frac{E_{m+1}(t)}{E_A} > A_{tol} \tag{3.21}$$

The pair of metrics defined by Eq. 3.18 and Eq. 3.20 are jointly used to identify a false nearest neighbor. If the test for nearest neighbors fails to satisfy these criteria, those neighbors will be declared as false nearest neighbors. The algorithm is repeated for every embedding dimension until the proportion of false nearest neighbors becomes zero or close to zero and doesn't change from then onwards. The embedding dimension for which there don't exist any false neighbors for points in the phase space is chosen as the minimum embedding dimension i.e. a perfect embedding dimension is the one in which no false nearest neighbors exist in the phase space.

3.3 LSTM model construction

Recurrent neural networks are very versatile as they can be used in a host of applications. RNNs have different architectures for predicting future values on the basis of past values. But as we are using time series data for making predictions, we would only consider the sequence-to-sequence model where both the inputs and outputs are sequences. RNN is widely applied over sequential data. If we want to predict the next sequence we want to know the previous sequence. Traditional neural networks don't retain the previous sequence but RNN will remember the previous sequence. The normal feed-forward neural network predicts output based on the current input whereas a recurrent neural network predicts output based on the current input as well as previous input.

LSTM is the most efficient sequence-to-sequence RNN model because unlike other RNN models that map fixed-sized input vectors to fixed-sized output vectors, it predicts output sequences from input sequences. Data in a sequence can be of variable lengths. Theoretically, these sequences can be infinitely long in length but it is impossible practically. For example, consider a simple recurrent neural network with no hidden units but with a recurrence of some scalar x^0. After n time units, its value will become X^n.

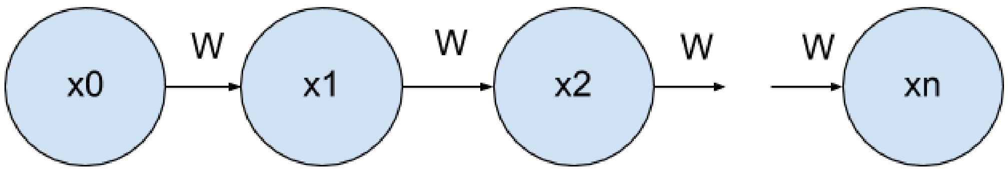


Figure 3.5: Simple recurrent neural network with no hidden units.

$X^{(n)} = W^{(n)}x^{(0)}$; $x^{(i)}$, $W \in R$ ||i $\in [0, n]$

Here, W = scalar weight,

$x^{(0)}$ = initial input scalar,

$X^{(n)}$= scalar at n^{th} time step.

Scalar weights are learned through backpropagation with the help of a time algorithm. For a very large value of n, where $n \to \infty$, the value of $W^n x^{(0)}$ becomes very large if the value of W is slightly greater than 1.

$W^n x^{(0)} \to \infty : W > 1 || 0 : W < 1$

If W is slightly less than 1, then $W^n x^{(0)}$ will become very small and vanish. Due to the vanishing or exploding nature of forward propagated values, the same thing will happen to their gradients (errors).

$\frac{\partial W^n x^{(0)}}{\partial W} \to \infty : W > 1 || 0 : W < 1$

We can generalize this to matrices as well.

$x^{(i)} \in R^D$

$W \in R^{D \times D}$

Where x is an input vector and W is weight matrix transformation. For the entry

values of matrix transformation with $W > 1$, the corresponding eigenvectors of W^n will explode. This means that the values of the input in the direction of eigenvectors will also explode to infinity resulting in the loss of input information. The opposite will happen when the values of W are less than 1. For $W < 1$, corresponding eigenvectors become very small and the components of input in the direction of eigenvectors will vanish which also leads to the loss of input information. This is called the **exploding/vanishing gradient problem**.
RNN has looped connections that allow the network to hold information across inputs. These connections are considered as memory. However, due to the vanishing gradient problem, they cannot retain information for a longer time in memory. When we backpropagate neural network and calculate gradients with respect to the weights, gradients tend to get smaller and smaller as we keep on moving back on the network which means that neurons in the earlier layers learn very slowly as compared to the neuron in the later layers in the network.

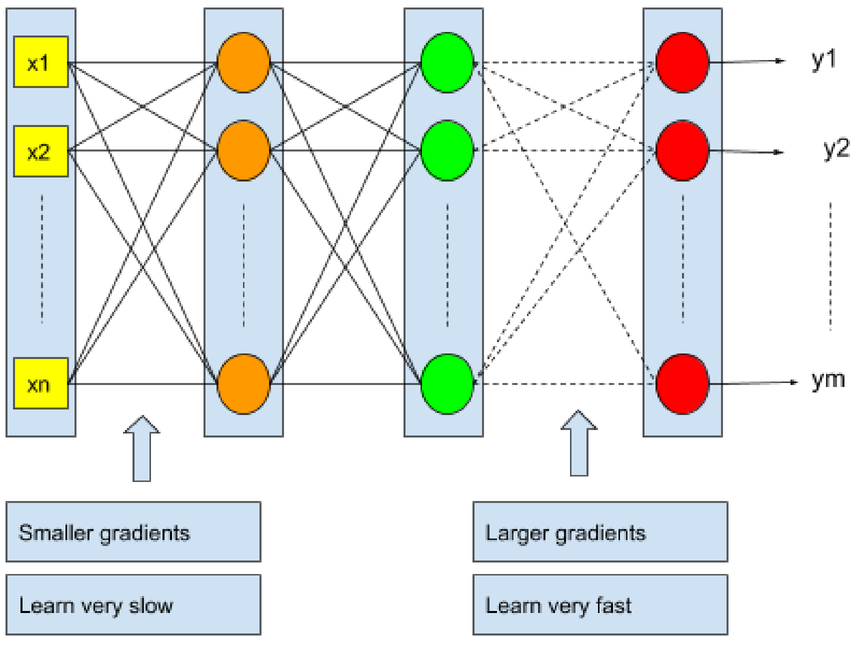


Figure 3.6: Vanishing gradient problem

There are different ways to deal with Exploding/Vanishing gradients:

1. **Introduce skip connections**

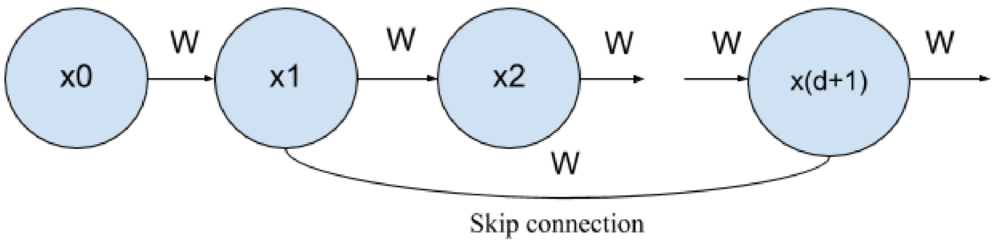


Figure 3.7: Skip connections

Additional edges called skip connection are added to connect states, x^1 to x^{d+1} in the above figure. Here, the current state x^{d+1} is influenced by the previous state x^1 which occurred d time steps ago. Now the gradients will explode or vanish as a function of $\frac{\tau}{d}$ instead of just τ. Here is τ the number of time steps, $\tau \in [1, n]$.

2. **Replace the connections of length 1 with the connections of larger length.**

   In this, the connections of length 1 are replaced by the connection of longer lengths as this forces the network to learn in this modified path only i.e. if the connection is of length 'a' time steps, the network will learn by skipping the 'a' data points.

3. **Leaky recurrent units (learning rate)**

   The learning rate (α) regulates the amount of information the network remembers over time. If the α is close to 1, more memory is retained. If it is closer to 0, the memory of the previous state vanishes i.e. it is deleted or forgotten. But the value α is to be assigned manually at every time step.

Instead of manually assigning a constant value to determine the information to be retained, a set of parameters, one for every time step is used. This set of new parameters act as gates that allow the network to retain and forget information according to their states. The gates constitute a gated recurrent neural network.
Long Short Term Memory (LSTM) is the most efficient gated RNN architecture. LSTMs mitigate the vanishing or exploding gradient problem. Instead of neurons, each LSTM network is formed by the combinations of LSTM cells. A connection, a cell state is present between every consecutive LSTM cell. Each LSTM maintains

a cell state vector (C_t) and a hidden state vector (h_t). In each time-step, an LSTM cell can choose to read, write, or reset the cell using an explicit gating mechanism. This gating mechanism has three gates of the same shape, an input gate, output, and a forget gate. Each gate is a binary gate. The input gate controls the process of updating memory. The equation representing an input gate is given below.

$$i^{(t)} = \sigma\left(U^i x^{(t)} + W^i h^{(t-1)} + b^i\right)$$

where W^i = weight received by input gate at current LSTM cell,
$h^{(t-1)}$ = hidden state of previous time step ‘t-1’,
$x^{(t)}$= input at current time step ‘t’,
b^i = bias of input gate.

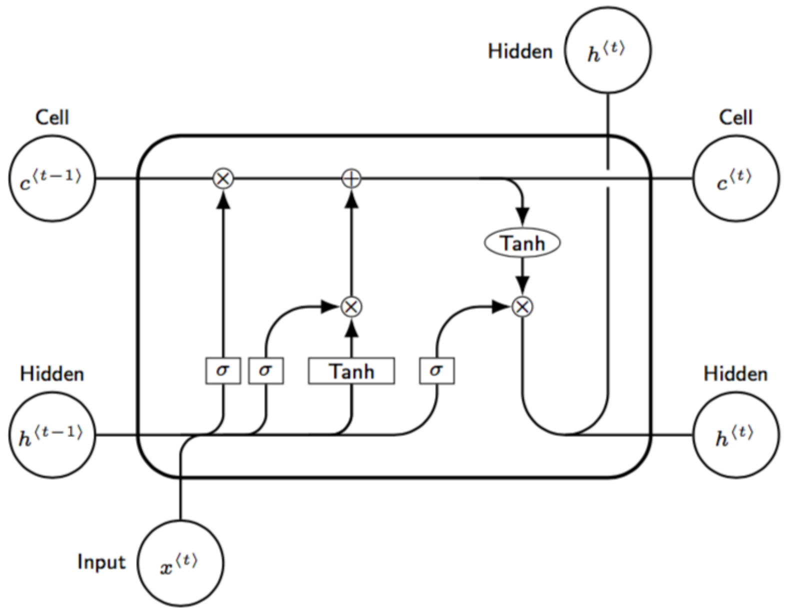


Figure 3.8: Architecture of an LSTM cell.

The forget gate controls if the memory cell is reset to zero. It decides whether the information should be retained or deleted from the memory. If the forget gate is reset to zero, previous information stored in the memory is deleted. The equation for the forget gate is given below.

$$f^{(t)} = \sigma\left(U^f x^{(t)} + W^f h^{(t-1)} + b^f\right)$$

where W^f = weight received by forget gate,
b^f = bias of forget gate.

The output gate controls whether the information of the current cell state is made visible.

$$o^{(t)} = \sigma\left(U^o x^{(t)} + W^o h^{(t-1)} + b^o\right)$$

where W^o = weight received by output gate,
b^o = bias of output gate,
σ = sigma activation funtion.
Every gate has a sigmoid activation. The sigmoid activation constitutes smooth curves in the range [0,1] and helps to keep the model differentiable. Apart from the gates, there is another vector $\bar{C^{(t)}}$ that modifies the cell state.

$$\bar{C^{(t)}} = tanh\left(U^C x^{(t)} + W^C h^{(t-1)} + b^C\right)$$

where $\bar{C^{(t)}}$= new candidate state value at time step 't' that can be applied to the cell state.
It has tanh activation with a zero centered range. tanhdistributes gradients. It allows the cell state information to flow for longer periods of time without vanishing or exploding. Each of the gates takes the hidden state and current input 'x ' as input. They concatenate vectors and apply sigmoid to them.
Now we apply the gates. Since the input gate controls whether the memory cell is updated, it is applied to $\bar{C}$, which is the only vector that can modify the cell state.

$$C^{(t)} = f^{(t)} C^{(t-1)} + i^{(t)} \bar{C^{(t)}}$$

The forget gate controls how much of the old state should be forgotten. The state $C^{(t)}$ is applied to the output gate to get the hidden vector $h^{(t)}$ where,

$$h^{(t)} = tanh\left(C^{(t)}\right) \times o^{(t)}$$

Here we have three gates per LSTM cell, so we have to model a slew of parameters.

3.4 Building LSTM for bitcoin time series

Now let us build our LSTM model to process out bitcoin time-series. Suppose, the dataset is a sequence of data points $[x(t)]_{t=1}^{\tau}$. We will train our model for every time-step to learn to make predictions based on past values. The predicted output at each time step is dependent on previous predictions. The output for $x^{(k+1)}th$ element is dependent on the computations of previous 'k' elements. The results of all those 'k' elements are captured and stored in memory. For every sequence element, the model performs the same computations.

3.4.1 Architecture of the LSTM model

The notation of a simple LSTM is depicted in Figure 3.9:

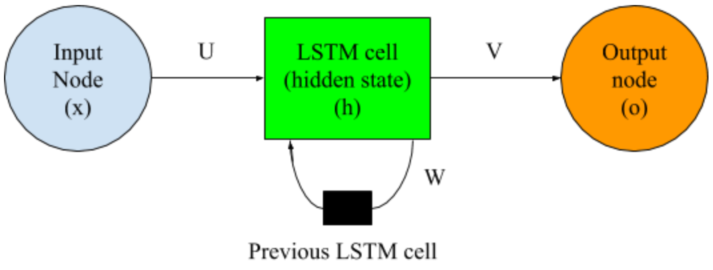


Figure 3.9: Architecture of LSTM for one element of the sequence.

For the complete sequence, we have to unfold (or unroll) our model into a complete graph (collection of LSTM networks). By unfolding our model of the sequence $[x(t)]_{t=1}^{\tau}$, a network is created for every time step i.e. for every data point x(t)where $t \in [1, \tau]$, which results in a graph. For example, if the sequence contains only 5 data points, a 3-layer LSTM network is created for each of the 5 data points. These 3-layer networks are created and connected in the same sequential order in which data points are present in the sequence. The unfolding of the LSTM network into the full graph is shown in Figure.

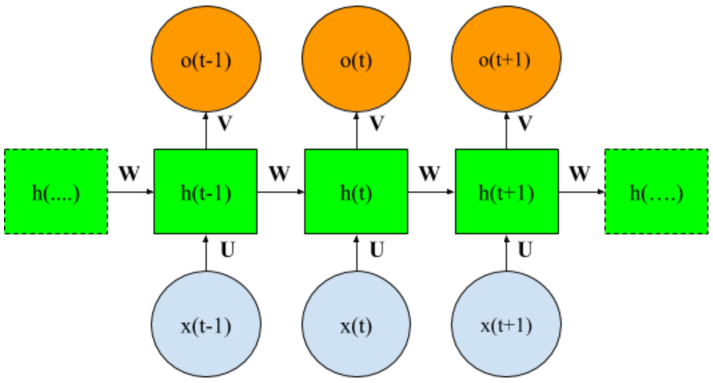


Figure 3.10: Unfolded architecture of LSTM for the complete sequence of elements.

The parameter of each network are:

- **Input ($x^{(t)}$):** At every time step, a different input is given to the network i.e. at times step ‘t’, $x^{(t)}$ is the input.
- **Weights:**
  U = input-to-hidden layer weight layer matrix,
  W = hidden-to-hidden layer weight matrix,
  V = hidden-to-output layer weight matrix.
- **Hidden State ($h^{(t)}$):** The hidden state $h^{(t)}$ represent the memory of the network at time step ‘t’. The hidden state at each time step is calculated as:

  $$h^{(t)} = tanh\left(Ux^{(t)} + Wh^{(t-1)}\right) \tag{3.22}$$

  where $tanh$ = hyperbolic tangent transformation or activation function.

The Eq. 3.21 shows that the hidden state of an LSTM cell at time step ‘t’ is based on the current input, hidden state of previous LSTM cell (at time step t-1).

3.5 Flow chart of LSTM algorithm

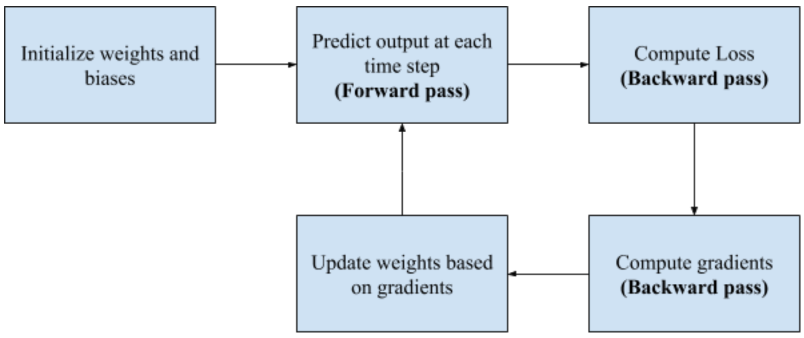

Initialize weights and biases
Predict output at each time step (Forward pass)
Compute Loss (Backward pass)
Compute gradients (Backward pass)
Update weights based on gradients

Figure 3.11: Flow of the LSTM prediction algorithm

3.5.1 Forward Pass

In the forward pass, forward propagation is used to predict output at every time step for a sequence. For a sequence $x(t) = x(1), x(2), x(3), ..., x(\tau)$, flow moves from left-to-right in the network, mapping input at every step to an output value. The predicted output at each time step is represented by the equation below:

$$\hat{y_t} = Vh^{(t)} + b^v \tag{3.23}$$

The predicted output at each time step depends on the hidden-to-hidden output weight matrix, the current hidden state, and the output bias.

3.5.2 Working of forward pass

For each data point $x^{(t)}$ where $t \in [1, \tau]$,the predicted output $\hat{y}_t$ depends on the outputs of the input gate, forget gate, and output gate current LSTM cell, given that the hidden state and cell state of the previous LSTM cell is available. Each LSTM cell gets three inputs, current data point, previous hidden state $h^{(t-1)}$, and cell state $c^{(t-1)}$ from the previous cell.
For every time step $t \in [1, \tau]$, every gate, the input gate, forget gate, and output gate generate its respective output $i^{(t)}$, $f^{(t)}$, $o^{(t)}$ based on its corresponding input-hidden layer weight matrix (U^m where $m \in [i, f, o]$), previous hidden state $h^{(t-1)}$, input data point x(t), and bias b^m where $m \in [i, f, o]$. Each of these three gates uses a sigmoid activation function. After that, a new candidate state value $\bar{c}^{(t)}$ at every time step, which is applied to cell state $c^{(t)}$ to modify it, is calculated. Once the candidate value is determined, the current cell state $c^{(t)}$ is calculated which is a prerequisite for calculating hidden state $h^{(t)}$.
The final predicted output $\hat{(y)}_t$is calculated on the basis of the output layer weight matrix (V) and output bias (b^v). The set of equations that represent the sequential flow of forward pass is given below:

$$i^{(t)} = \sigma \left(U^i x^{(t)} + W^i h^{(t-1)} + b^i \right) \tag{3.24}$$

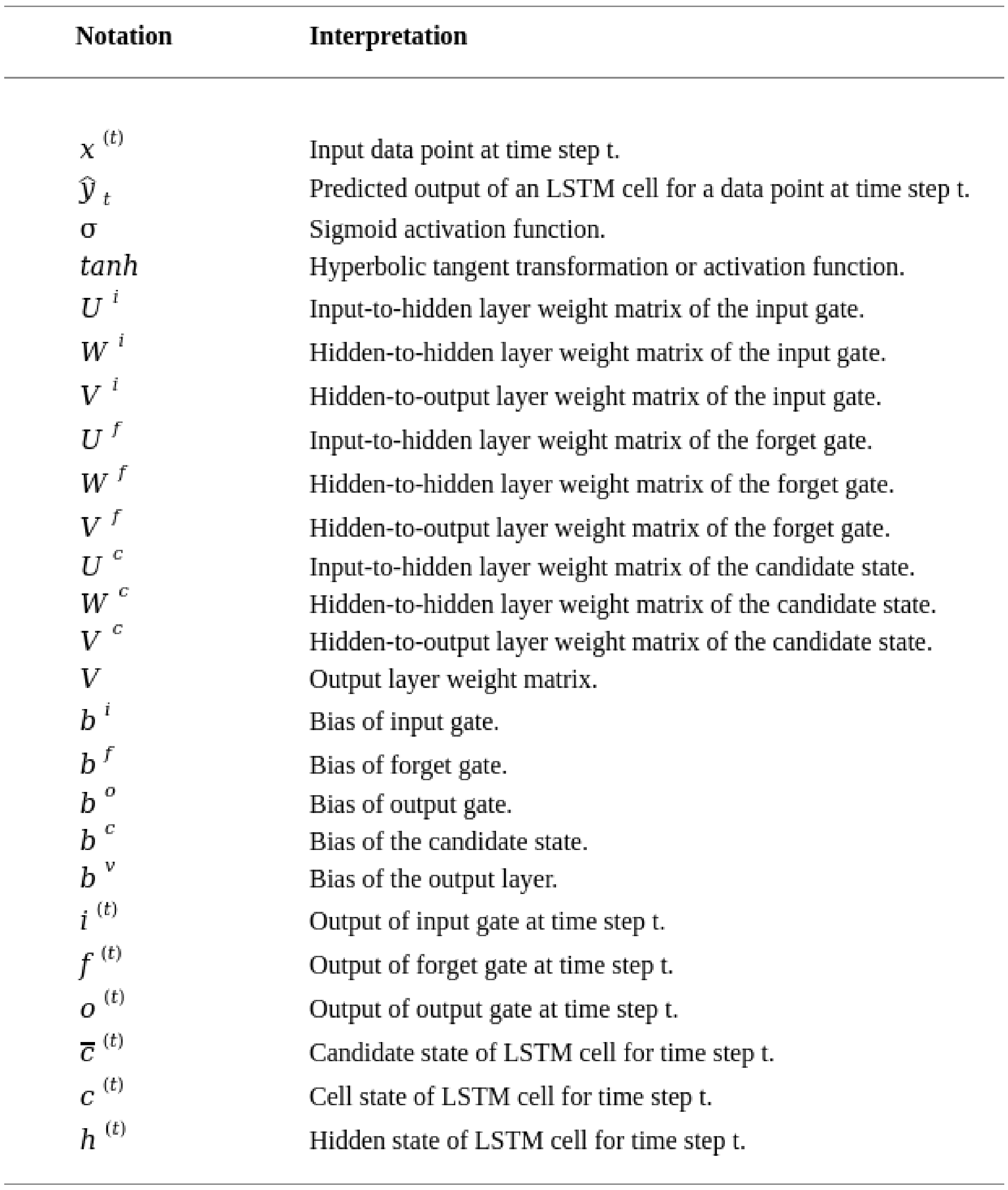
$$f^{(t)} = \sigma \left(U^f x^{(t)} + W^f h^{(t-1)} + b^f \right) \tag{3.25}$$

$$o^{(t)} = \sigma \left(U^o x^{(t)} + W^o h^{(t-1)} + b^o \right) \tag{3.26}$$

$$c^{\bar{(t)}} = tanh \left(U^c x^{(t)} + W^c h^{(t-1)} + b^c \right) \tag{3.27}$$

$$c^{(t)} = f^{(t)} c^{(t-1)} + i^{(t)} c^{\bar{(t)}} \tag{3.28}$$

$$h^{(t)} = tanh \left(C^{(t)} \right) \times o^{(t)} \tag{3.29}$$



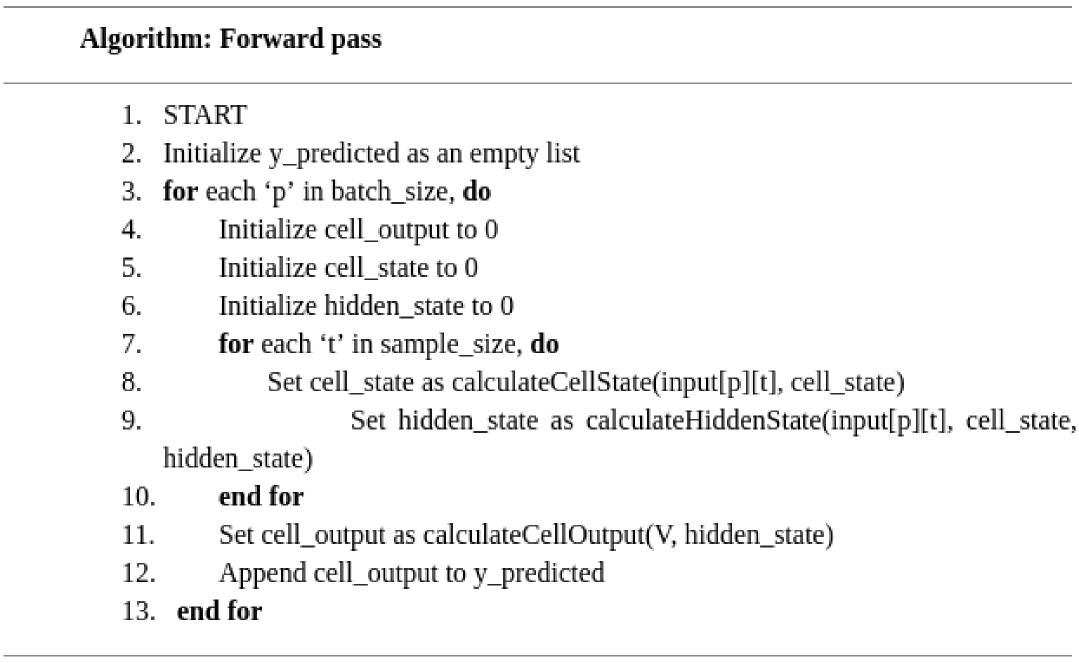
Notation	Interpretation
$x^{(t)}$	Input data point at time step t.
$\widehat{y}_t$	Predicted output of an LSTM cell for a data point at time step t.
σ	Sigmoid activation function.
$tanh$	Hyperbolic tangent transformation or activation function.
U^i	Input-to-hidden layer weight matrix of the input gate.
W^i	Hidden-to-hidden layer weight matrix of the input gate.
V^i	Hidden-to-output layer weight matrix of the input gate.
U^f	Input-to-hidden layer weight matrix of the forget gate.
W^f	Hidden-to-hidden layer weight matrix of the forget gate.
V^f	Hidden-to-output layer weight matrix of the forget gate.
U^c	Input-to-hidden layer weight matrix of the candidate state.
W^c	Hidden-to-hidden layer weight matrix of the candidate state.
V^c	Hidden-to-output layer weight matrix of the candidate state.
V	Output layer weight matrix.
b^i	Bias of input gate.
b^f	Bias of forget gate.
b^o	Bias of output gate.
b^c	Bias of the candidate state.
b^v	Bias of the output layer.
$i^{(t)}$	Output of input gate at time step t.
$f^{(t)}$	Output of forget gate at time step t.
$o^{(t)}$	Output of output gate at time step t.
$\overline{c}^{(t)}$	Candidate state of LSTM cell for time step t.
$c^{(t)}$	Cell state of LSTM cell for time step t.
$h^{(t)}$	Hidden state of LSTM cell for time step t.

Figure 3.12: List of notations and their interpretations

$$\hat{y}_t = Vh^{(t)} + b^v \tag{3.30}$$

3.5.3 Pseudocode for forward pass

The pseudocode for the proposed forward pass algorithm is described below. Before training our model, we divided our data set into equal-sized batches, each batch contains 'n' number of samples (data points). *batch-size* describes the number of batches our dataset is divided into. *sample-size* represents the number of data-points or time steps in a batch. For each batch, the algorithm computes cell state and hidden state for every time step. The variable *input* is used to feed batches as inputs to the LSTM model for processing. Variables *cell-state* and *hidden-state* are used to store cell state and hidden state respectively. *calculateCellState()* and *calculateHiddenState()* functions calculate and return current cell state and hidden state respectively for every time step. *cell-output* variable stores the predicted output value computed by *calculateCellOutput()* function for each data point. *y-predicted* is a list that stores the predicted output of all data points.



Algorithm: Forward pass

```
1.  START
2.  Initialize y_predicted as an empty list
3.  for each 'p' in batch_size, do
4.      Initialize cell_output to 0
5.      Initialize cell_state to 0
6.      Initialize hidden_state to 0
7.      for each 't' in sample_size, do
8.          Set cell_state as calculateCellState(input[p][t], cell_state)
9.              Set hidden_state as calculateHiddenState(input[p][t], cell_state,
    hidden_state)
10.     end for
11.     Set cell_output as calculateCellOutput(V, hidden_state)
12.     Append cell_output to y_predicted
13. end for
```

Figure 3.13: Forward pass algorithm

3.5.4 The complexity of the forward pass algorithm

Since the forward pass is performed for all the timesteps, and the total number of time steps is τ, the runtime complexity of our forward pass algorithm is O(τ). This complexity cannot be reduced by parallel processing of the algorithm for every timestep because the graph is sequential and the output of each time step is dependent on the output of previous timesteps i.e. predicted output for the current time step can be computed only after the predictions for previous time steps are computed. States computed by the forward pass algorithm at each time step must be stored in memory because they are reused as inputs in the backward pass. So the memory cost of this algorithm is also O(τ).

Runtime complexity: O(τ)

Memory cost: O(τ).

3.5.5 Backward pass

The backward propagation is performed by moving from right to left in the computational graph to compute gradients. The backward propagation follows the forward pass. The backward pass uses all the states that were calculated in the forward pass. The gradient computation at each time step depends on the calculations of the current as well as the next time step.

3.5.6 Gradient computation

The first step in gradient computation is the computation of loss. It involves the use of outputs that were predicted during forward propagation to compute respective losses at each time step. The gradients for weight matrices (U, V, W) and biases are computed by using a loss function (L) and updated with a learning rate (α). The loss function calculates loss. To calculate loss at each time step, we can use the mean squared error loss function. The gradient is the measure of change of loss with respect to weight parameters and biases. The loss for each time step is calculated by the following algorithm.

3.5.7 Pseudocode for loss calculation

In the pseudocode below, *len-y-pred* is a variable to store the length of the list *y-predicted* which stores the output predicted in the forward pass for each time step. *mean-squared-error()* is a loss function that calculates and returns loss on the basis of target value and predicted value. *loss-t* variable stores the loss value predicted by *mean-square-error()* function at each time step. loss is a list that stores the losses for all time steps from $t = \tau$ to $t = 1$. Our computation graph includes the sequence of nodes having parameters (U, V, W) and bias (b). Each node is indexed by t for $x^{(t)}$, $h^{(t)}$, $o^{(t)}$, and loss function $L^{(t)}$. The gradient for each node 'n' is computed recursively based on the gradients computed at previous nodes while moving from right to left in the graph. The pseudocode for loss calculation is given in Figure 3.14.

Algorithm: Loss calculation

```
1. START
2. Initialize loss to an empty list
3. for each 't' in len_y_pred, do
4.     Set loss_t to mean_squared_error(target[t], y_predicted[t])
5.     Append loss_t to loss
6. end for
```

Figure 3.14: Algorithm for loss calculation

3.5.8 Gradient clipping

In the backward pass, the gradients propagate from the last time step to the first one which could result in the product of gradients either reducing to 0 (vanishing gradient) or increasing exponentially (exploding gradient). The vanishing gradient problem is automatically handled by LSTM, but to fix the exploding gradient problem, the larger values of gradients must be clipped to smaller values.

3.5.9 Updation of weights

The loss for each weight matrix and biases is updated and minimized with the following equations:

$$U = U - \alpha\frac{\partial L}{\partial U} \tag{3.31}$$

$$W = W - \alpha\frac{\partial L}{\partial W} \tag{3.32}$$

$$V = V - \alpha\frac{\partial L}{\partial V} \tag{3.33}$$

The biases are also updated by doing the same:

$$b = b - \alpha\frac{\partial L}{\partial b} \tag{3.34}$$

Chapter 4

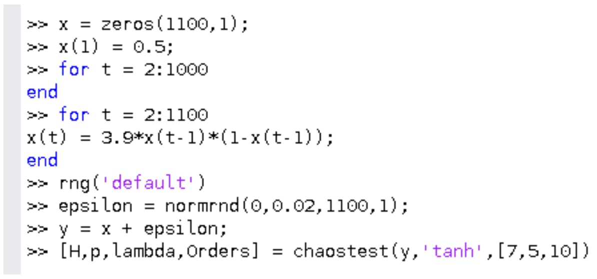
RESULTS AND DISCUSSION

4.1 Chaos detection - Lyapunov Exponent Test

The presence of chaotic dynamics in our dataset is done by performing the Lyapunov exponent test in MATLAB. The test computes the value of the largest Lyapunov exponent (λ) at a specific confidence level α. The inputs to the program are a vector of observations, a string that specifies the activation function to be used by the neural network that determines Lyapunov exponent, the values of orders *(L,m,q)* in the optimal range of any of (5,6,5) to (10,2,10), and the value of confidence level α. The outputs of the program are as follows:

- output the value of hypothesis (H): if H=0 the null hypothesis will be accepted, and will be rejected for H=1,
- p-value: the larger the p-value, the more will be the correctness of the null hypothesis.
- lambda (λ)

To ensure our test accurately determines the value of Lyapunov exponent for any chaotic series, we first performed this test on the actual chaotic system (both with noise and without noise). [60] shown that the logistic map, $x_t = \gamma x_{t-1}(1 - x_{t-1})$, exhibits chaotic dynamics for $\gamma \in [3.57, 4]$, and therefore is produces chaotic times series as we compute equations moving forward in time. In the first case, we first

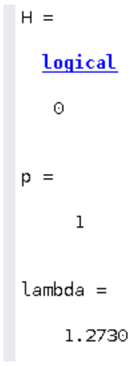


```
>> x = zeros(1100,1);
>> x(1) = 0.5;
>> for t = 2:1000
end
>> for t = 2:1100
x(t) = 3.9*x(t-1)*(1-x(t-1));
end
>> rng('default')
>> epsilon = normrnd(0,0.02,1100,1);
>> y = x + epsilon;
>> [H,p,lambda,Orders] = chaostest(y,'tanh',[7,5,10])
```

Figure 4.1: MATLAB commands for calculating λ for logistic map

performed the test on x_t for 1100 data points, generated by the above equation when $\gamma = 3.9$, and the starting point $x_0 = 0.5$. Next, a very little amount of noise(ε), with its amplitude defined by the standard deviation $\sigma_\varepsilon = 0.02$ to check for the value of λ .

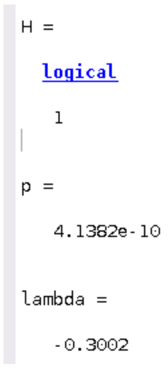
The result obtained from the test for the order (L,m,q) = (7,5,10) is H = 0, p = 1, and lambda (λ) = 1.2730. The value of H = 0, p=1, and the positive value of the exponent ($\lambda > 0$) indicates that the null hypothesis is accepted at a 5% confidence level i.e. the chaos is present.



```
H =

  logical

   0


p =

     1


lambda =

    1.2730
```

Figure 4.2: Outputs for logistic map when $\sigma_\varepsilon = 0.02$

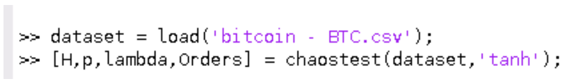
In the next case, the amplitude of the noise for the same data $[x_t]_{t=1}^{1100}$ was increased by increasing the standard deviation to $\sigma_\varepsilon = 0.15$ to check the validity of the statement, “as the noise in a chaotic system increases, it tends to acquire stochastic dynamics”. We found that the statement is indeed correct as the test failed. The result for $\sigma_\varepsilon = 0.15$ for orders (7,6,10) is H=1, p = 4.1382e-10, and lambda (λ)= -0.3002. The value H=1 and the negative value of exponent ($\lambda < 0$) indicate that the null hypothesis is rejected at a 5% confidence level i.e. chaos is absent.



```
H =

  logical

   1


p =

   4.1382e-10


lambda =

   -0.3002
```

Figure 4.3: Outputs for logistic map when $\sigma_\varepsilon = 0.15$

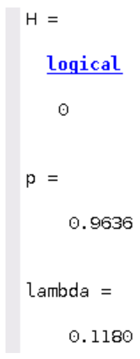
The p-value of the Lyapunov exponent decreased from 1 to 4.1382e-10 just by increasing the amount of noise in the data from 2% to 15%. It shows that as we increase the amount of noise in the data, the chaotic system becomes stochastic, and no longer shows chaotic dynamics.
We further performed the tests for the different values of orders (L,m,q), between (5,6,5) to (10,2,10), at different confidence levels. Every test produced the same results i.e. the increase in the amplitude of noise resulted in the decrease of the p-value of the exponent and the rejection of the null hypothesis. For super noisy systems, such as random variables, the Lyapunov exponent of such a system tends towards minus infinity i.e. $\lambda \to \infty$. So when a system becomes noisier and noisier, the Lyapunov exponent of the system will decrease, and we will not be able to detect chaos because the noise envelops the chaotic dynamics of the system by rendering it stochastic. After confirming the validity of the test for an evidently chaotic system, we tested the application of the test to our real-world bitcoin market times series data. We selected the hourly prices of Bitcoin - collected from **https://coindesk.com/price**. The sample contains 97725 data points from May 2012 to December 2019. Each data point is separated by a one-hour time step.



```
>> dataset = load('bitcoin - BTC.csv');
>> [H,p,lambda,Orders] = chaostest(dataset,'tanh');
```

Figure 4.4: MATLAB commands to load our timeseries dataset λ

The obtained result for our dataset is H = 0, p = 0.9693, lambda (λ) = 0.1180.



```
H =

  logical

   0


p =

    0.9636

lambda =

    0.1180
```

Figure 4.5: Outputs of the test for bitcoin time series.

The positive Lyapunov exponent ($\lambda > 0$) and H =0 marks the acceptance of the null hypothesis. Although the Lyapunov exponent is small (but positive) for our data, the high p-value of the exponent concludes that some amount of chaos is present in our dataset, and therefore it exhibits chaotic dynamics.

4.2 Phase space reconstruction

After detecting the presence of chaos we reconstructed multivariate phase space for our univariate chaotic bitcoin time series using the equation:

$$X(t) = x(t), x(t+L), x(t+2L), ..., x(t+(m-1)L) \tag{4.1}$$

As a reminder, L is the time delay and m is the number of embedding dimensions. To reconstruct phase space, we must first compute the values of the parameters L and m. The L was computed using the **Average Mutual Information (AMI) algorithm**. The variation of average mutual information against the different values of L is plotted in **Figure 4.6** for getting the optimal value of L for bitcoin time series.

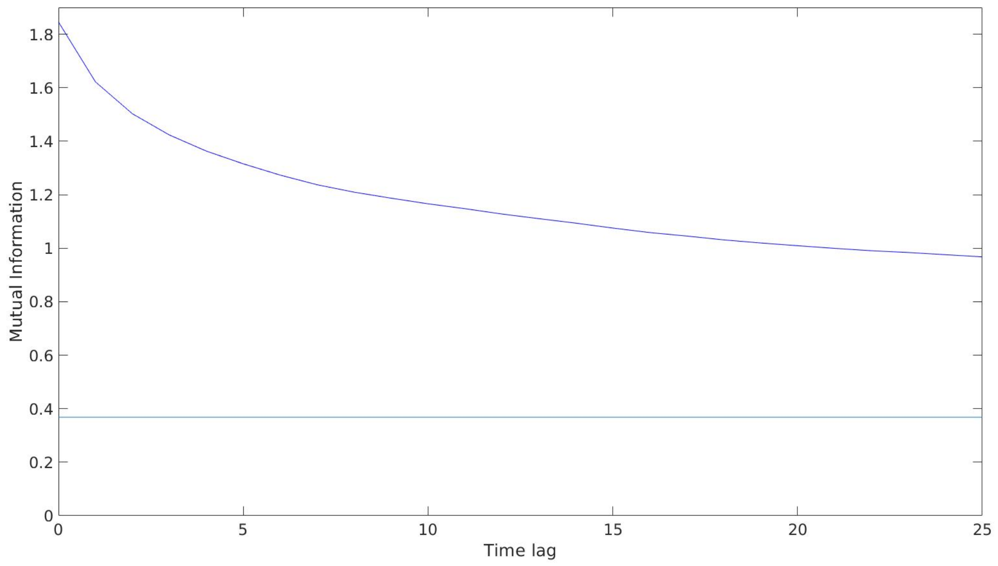
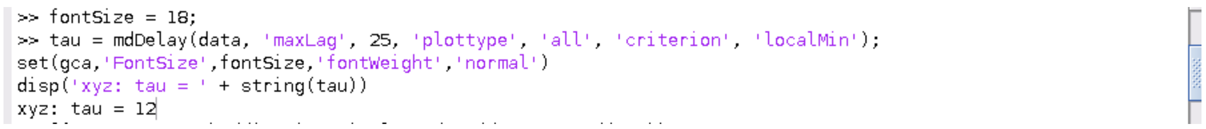


Figure 4.6: Values of average mutual information plotted against the different time delay L.

The presence of chaos in bitcoin prices reveals that the behavior of the bitcoin time series can be described and modeled when its multivariate phase space is reconstructed. The benefit of phase space reconstruction is that we can predict future bitcoin prices even if the source time series is univariate. The optimal value of L was chosen by selecting the value of L at which the first local minima of average mutual information function occurred. The computed optimal value of time delay (L) for our bitcoin time series is 12 as shown in **Figure 4.7**.



```
>> fontSize = 18;
>> tau = mdDelay(data, 'maxLag', 25, 'plottype', 'all', 'criterion', 'localMin');
set(gca,'FontSize',fontSize,'fontWeight','normal')
disp('xyz: tau = ' + string(tau))
xyz: tau = 12
```

Figure 4.7: The optimal value of time delay (L=12) computed by the AMI algorithm.

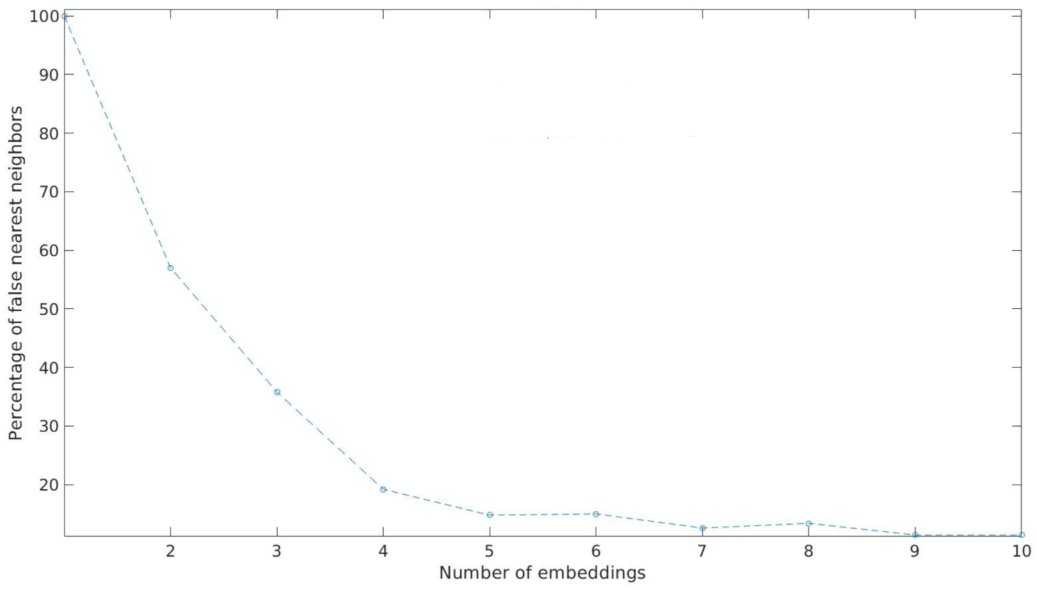


Figure 4.8: Proportion of false nearest neighbors plotted against different embedding dimensions for the time delay L=12.

The optimal value of the minimum embedding dimension (m) was computed using the False Nearest Neighbor (FNN) algorithm. For a given value of L, the FNN computes the proportion of false nearest neighbors for all the points in m-dimensional phase space. The dimension m for which the percentage of the false neighbors becomes zero is selected as the optimal value of the embedding dimension. For the time delay of L=12, the proportions of identified false neighbors are plotted against the different number of embedding dimensions in **Figure 4.8**. If we observe the graph, we can clearly see that the number of false nearest neighbors stopped changing after embedding dimension m=9. Therefore, we chose m=9 as the optimal value of the embedding dimension to reconstruct multivariate phase space. The high value of m shows that our bitcoin time series indeed exhibit complex and chaotic dynamics, and has a large number of degrees of freedom (or factors) that influence its price. We reconstructed phase space for our time series using the computed values of parameters time delay (L=12) and embedding dimension (m=9). This phase space reconstructed time series was used to build the prediction model using the proposed setting. The results of the prediction model are described in the forthcoming section.

4.3 Bitcoin price prediction and model evaluation

This section we discuss the experimental results of the proposed $Chaos + LSTM$ hybrid model and evaluate its performance. Alongside with the proposed model, we also implemented a stand-alone LSTM model to determine the effect of phase space reconstruction (modeling of chaos) on the performance of the model. Both models were trained using the same training data, and the same test data was used for evaluating the performance of respective models.

All the experimentation and implementation of the prediction models was accomplished using a system equipped with Intel i5 3.4 GHz CPU, 8 GB RAM, 2 TB HDD, and an AMD Radeon 530 GPU with 4GB memory. Linux was used as the host operating system and the model training and implementations were done in Jupyter Notebook using Python 3 and Keras deep learning library using the TensorFlow platform. We trained both models for 60-100 epochs and found that the loss doesn't show much variation after 100 epochs, so we trained the proposed model for 100 epochs For building the LSTM part of our proposed $Chaos + LSTM$ hybrid model, we used only one dense layer with linear activation function, root mean squared error loss function with 'adam' as the optimization algorithm. The model is trained using 80% samples of the dataset and then validated on 20% samples. The model's learning rate is 0.01.

The original bitcoin time series that we used for training and testing our models is shown in **Figure 4.9**.

Three prediction models (stand-alone LSTM, and our proposed chaos-based LSTM hybrid, and a chaos+MLP) were built for this study using both traditional and proposed settings (i.e. with and without modeling chaos present in our bitcoin time series). For building the stand-alone LSTM model, we used the original bitcoin time series (chaos not modeled) for both training and testing. However, we trained and tested our proposed hybrid model by using the phase space reconstructed time series (chaos modeled). Chaos+MLP model was also built by using the phase space reconstructed time series for comparison purposes. Each model was used to predict bitcoin prices for the test set. The predictions of every model, which are plotted as target values vs predicted values graph, are depicted in **Figure 4.10**, **Figure 4.11**, and **Figure 4.12**. These figures clearly reveal that the prediction results of the proposed Chaos+LSTM hybrid model are robust in comparison to the results of the stand-alone LSTM and Chaos+MLP model. The results of the stand-alone LSTM model are modest. From **Figure 4.10**, it can be clearly observed that the prediction results are moderate as there is a large gap between predicted and actual prices curve. The **Figure 4.10** depicts the overestimation of prices by the stand-alone LSTM model.

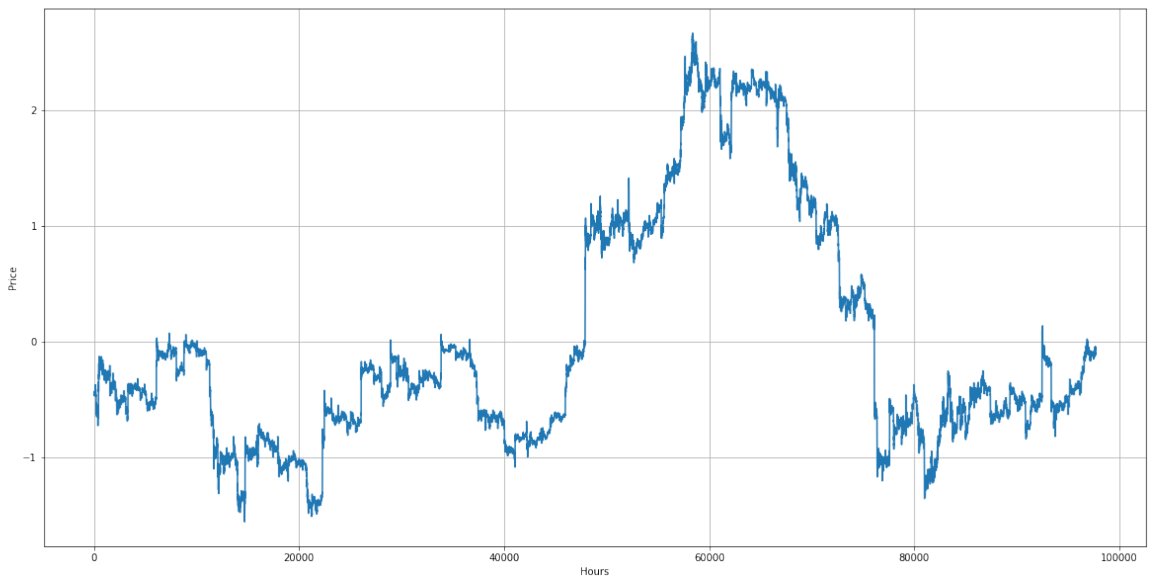


Figure 4.9: Bitcoin time series dataset containing 97725 time steps (or data points).

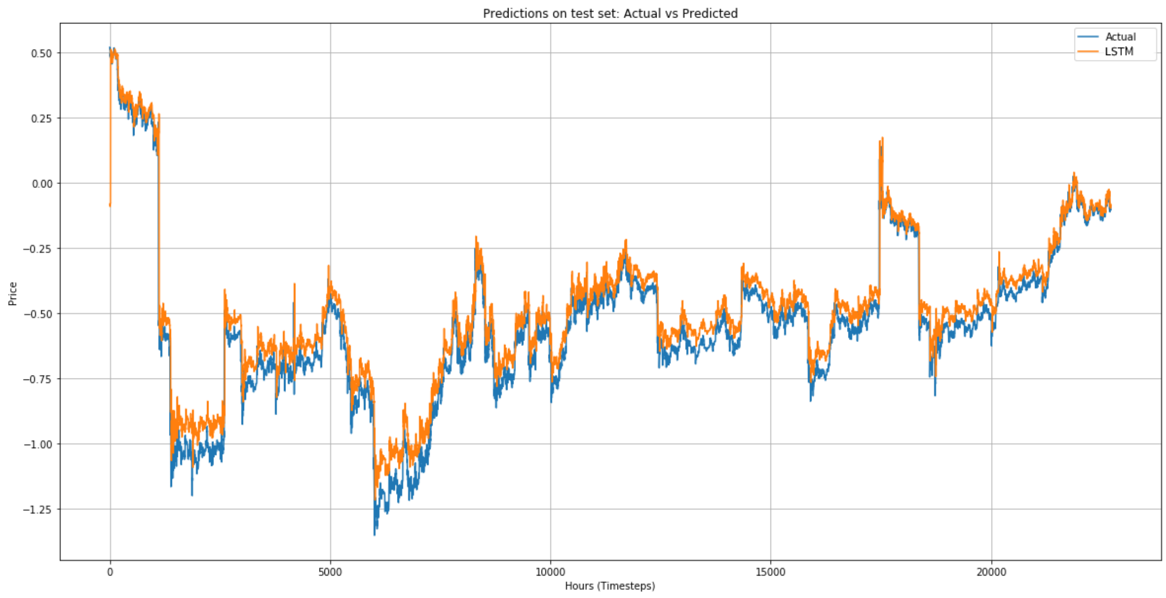


Figure 4.10: Actual and predicted prices using the stand-alone LSTM model.

The reason for this stems from the fact that in the case of the stand-alone LSTM model, we didn't reconstruct phase space of the time series, and trained the model without modeling the chaos present in the bitcoin market. Thus the model was unable to learn the important hidden chaotic dynamics and patterns in time series, which resulted in the overestimation of prices. However, the proposed hybrid model yielded better predictions because it was able to learn hidden and subtle chaotic patterns in the price fluctuations as it was trained using the chaos modeled phase space reconstructed time series. The predicted results of the Chaos+MLP model are depicted in **Figure 4.11.**

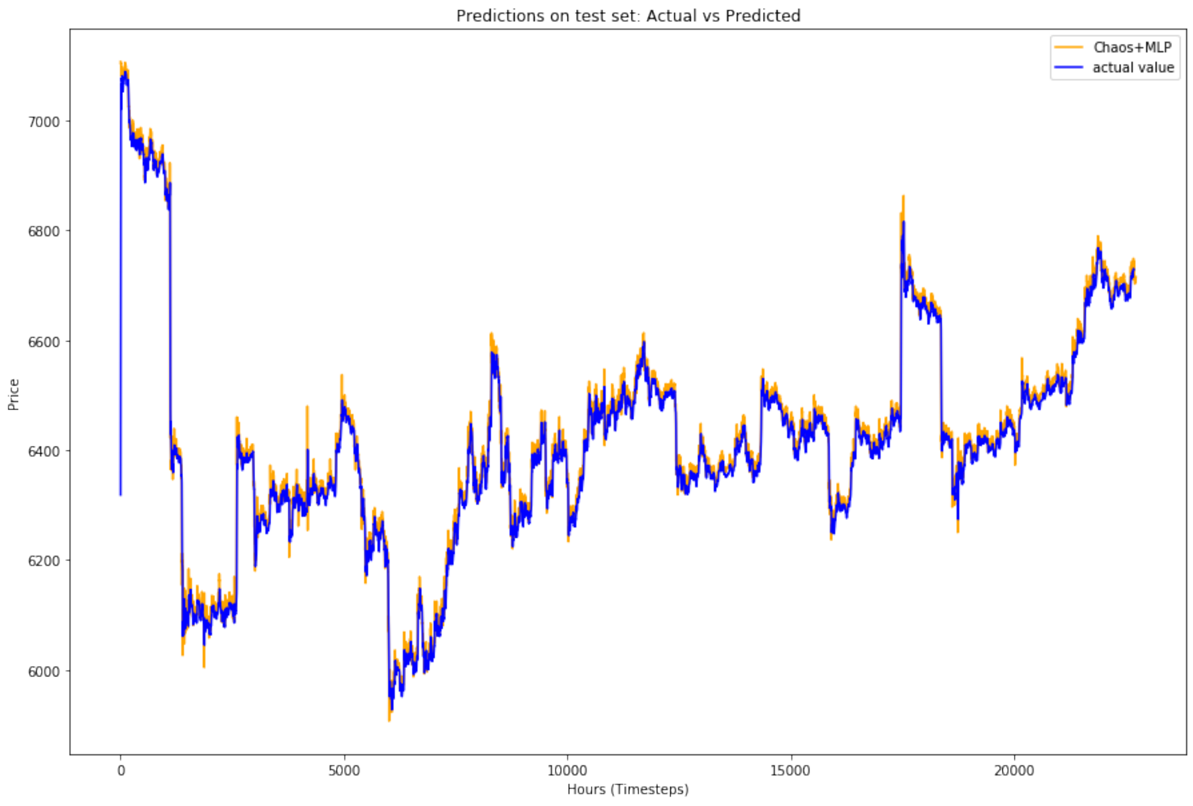


Figure 4.11: The prediction results of Chaos+MLP model.

Although, the Chaos+MLP generates overestimated predictions, we can clearly see that the results are pretty accurate. The Chaos+MLP model outperforms the stand-alone LSTM model. However our proposed hybrid model predicted better results than both stand-alone LSTM and Chaos+MLP model. The evidence can be found by the comparison of Figure 4.11 and Figure 4.12. **Figure 4.12** depicts that the results of the proposed model are highly robust, as there is a perfect overlapping between the predicted and actual prices.

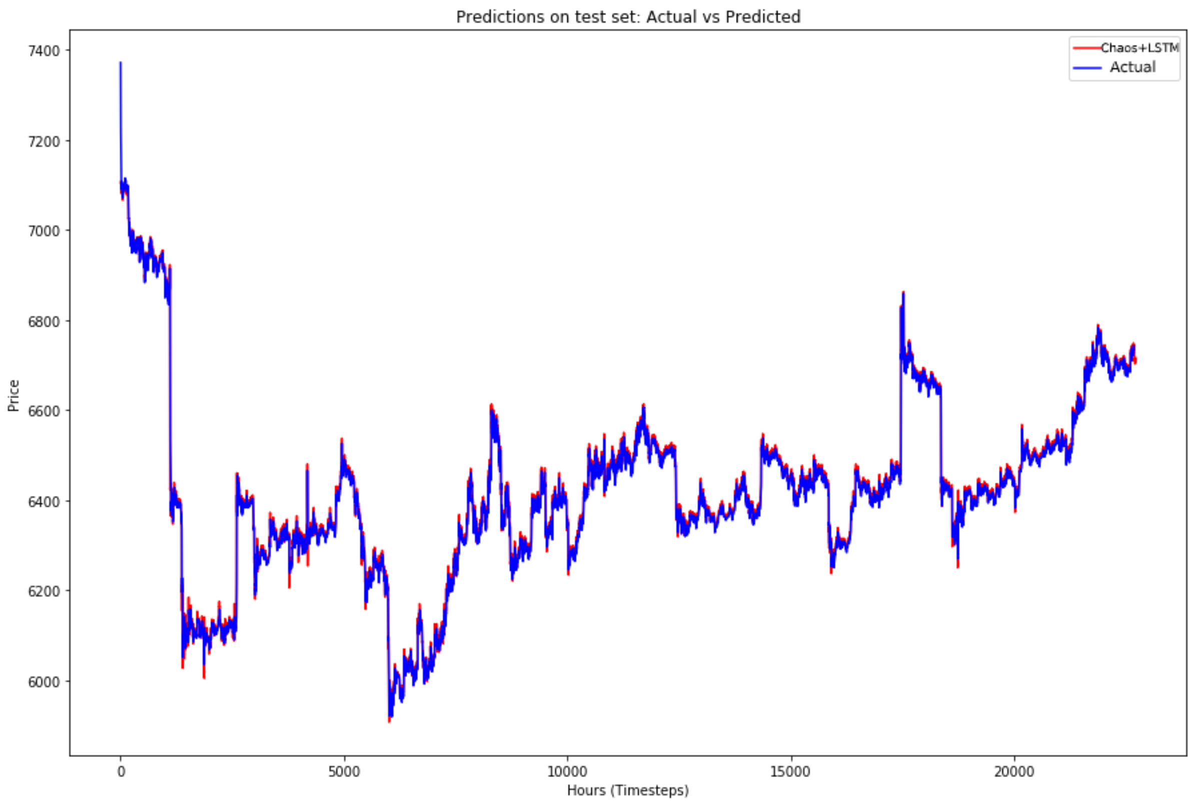


Figure 4.12: Actual (real) and predicted values using the proposed $Chaos + LSTM$ hybrid model.

As there is a perfect overlapping between the predicted and actual price curve in **Figure 4.11** and the two curves are nearly indistinguishable, the zoomed version of the end portion of **Figure 4.11** is depicted in **Figure 4.12**, for the visual ease of the reader.

The models that are used for the comparative study are: Chaos+MLP hybrid model, conventional LSTM and hybrid LSTM with AR(2) model [42], GRU recurrent model [45], ARIMA and Hidden Markov Models based HMM-LSTM hybrid model [40]. These models are preferred over other models present in the literature because these are the state-of-the-art models at the present time. In their work, the authors utilized their respective models to predict bitcoin prices only and they reported corresponding RMSE values for the evaluation of the performance of the models.

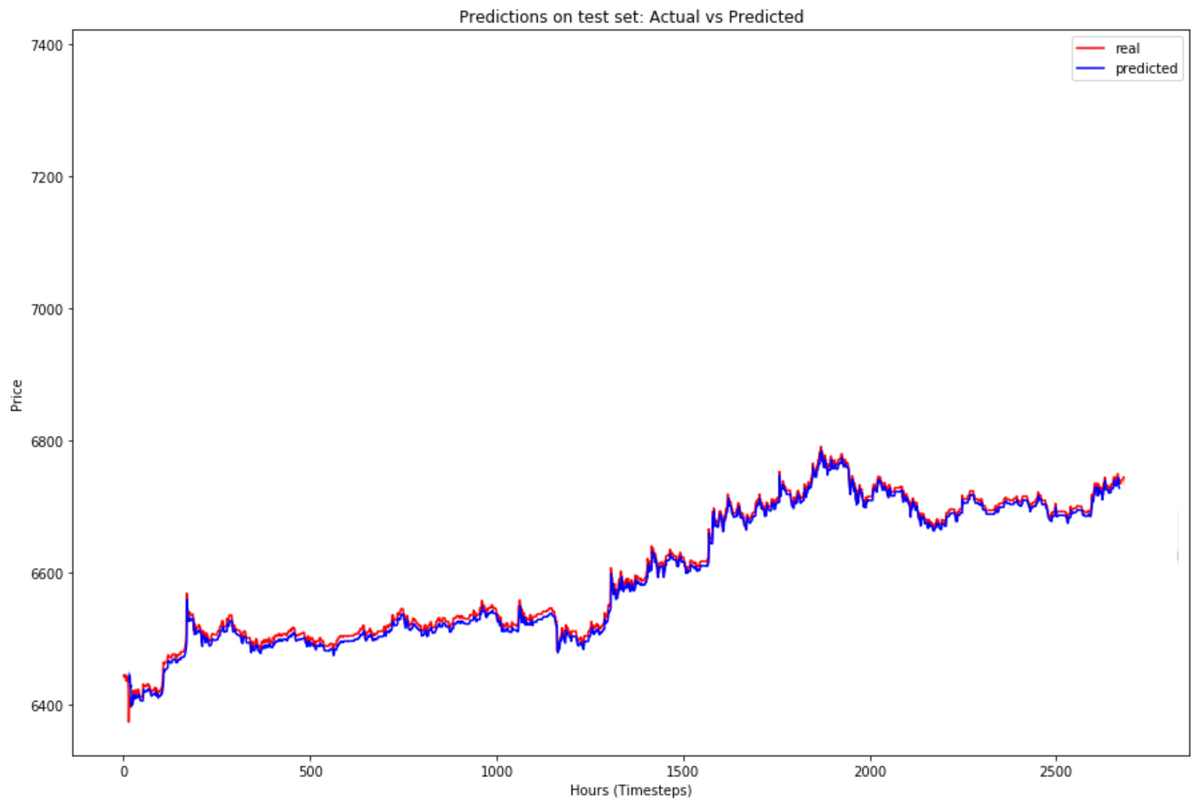


Figure 4.13: Actual (real) and predicted values using the proposed $Chaos + LSTM$ hybrid model.

We used RMSE values to evaluate the performance of the proposed hybrid model. **Figure 4.13** presents the prediction error between the actual and predicted bitcoin prices in terms of RMSE values. The table contains the RMSE values of all the models that are used for comparison of the performance of the proposed hybrid model. As seen from **Figure 4.13**, the Chaos+LSTM model generated an RMSE of 1.6173e-04 while the stand-alone LSTM model generated RMSE of 0.001. The plausible explanation for this difference in RMSE values is the fact that when the LSTM is trained using phase space reconstructed time series, it remembers longer sequences of hidden and subtle chaotic patterns, resulting in less error while making predictions than the LSTM which is trained directly with time series for which chaos is not modeled. Based on the RMSE values, we found that the prediction error of the proposed Chaos+LSTM is the least among all other models. Thus, it outperforms all of these models.

Model	RMSE	Rank
Conventional LSTM	7.006	6
Stand-alone LSTM	0.001	2
Chaos+LSTM	1.6173e-04	1
Chaos+MLP	1.796	4
LSTM with AR(2)	247.33	8
GRU	0.019	3
HMM-LSTM	5.821	5
ARIMA	141.964	7

Figure 4.14: Performance evaluation of the prediction models in terms of RMSE.

Looking at the RMSE values presented in **Figure 4.13**, the closeness of predicted prices to the actual price in **Figure 4.11** and the consistent movement of the prices, we can safely say that the predictions of the proposed Chaos+LSTM are realistic with a relatively lower error rate. The training and validation loss for both stand-alone LSTM and Chaos+LSTM hybrid is depicted by **Figure 4.14** and **Figure 4.15** respectively.

The difference between the training and validation loss can be reduced by adding recurrent dropout to the models and increasing the number of hidden layers but the computational complexity of models will increase as the number of hidden layers increases.

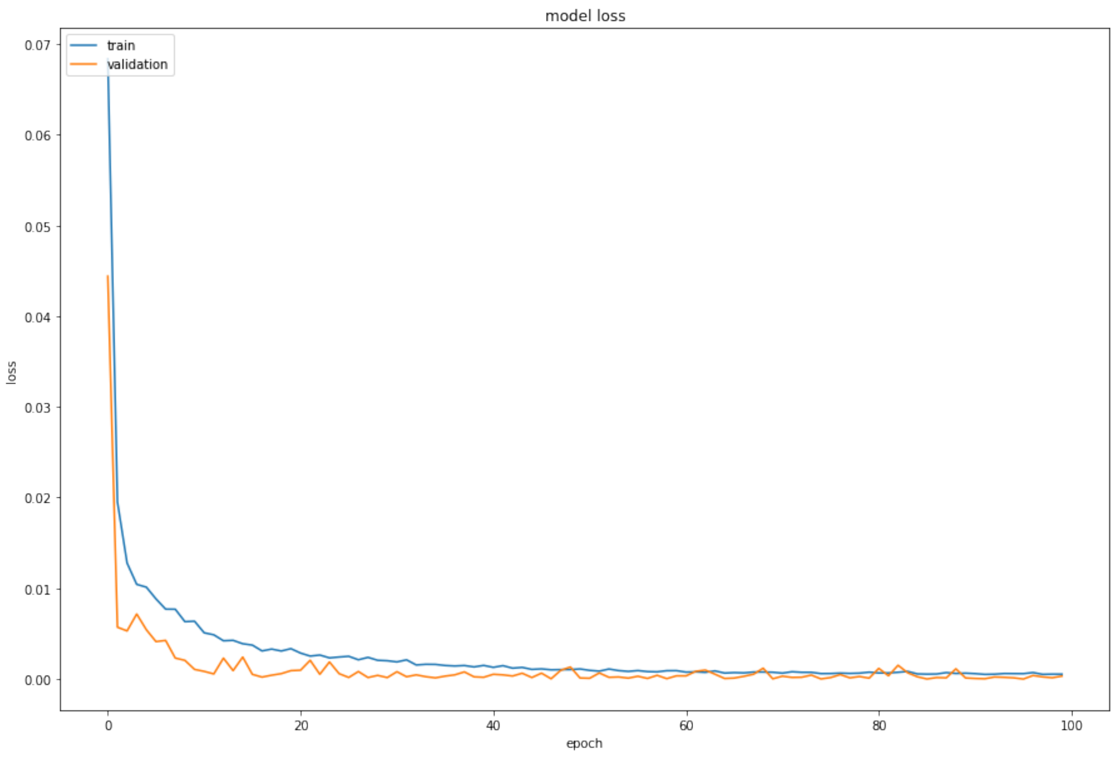


Figure 4.15: Training and validation loss for stand-alone LSTM model.

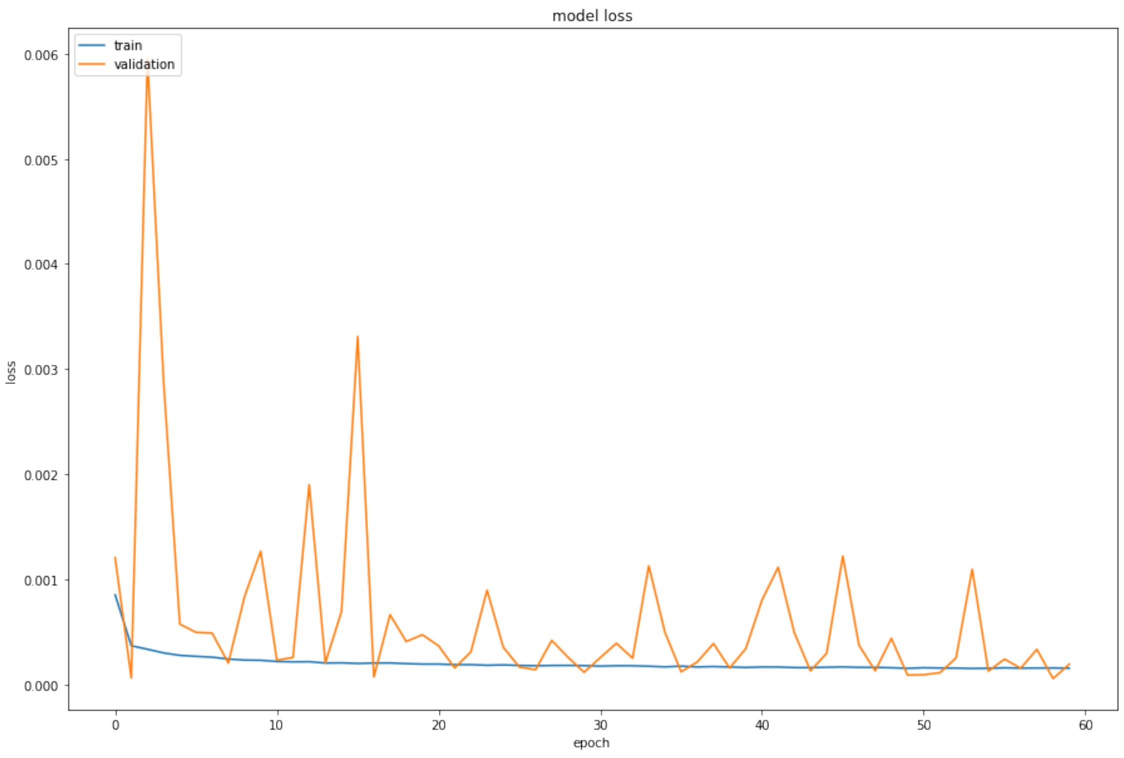


Figure 4.16: Training and validation loss for the Chaos+LSTM model.

Due to the presence of high volatility and complex chaotic dynamics in the

Bitcoin market, the current price prediction models are ineffective at learning the most important factors that are responsible for causing high fluctuations in prices. This lack of understanding of the dynamics of the bitcoin market results in investors losing a lot of money. This study and the proposed model can have implications to unravel those mysterious factors to provide managerial solutions for predicting future prices with more accuracy and therefore minimize investment risks and losses.

Chapter 5

CONCLUSION

The bitcoin market is complex and noisy, and advanced technologies and algorithms are required to analyze its behavior. In the literature, it is not confirmed whether its dynamics are stochastic or dynamic. Most chaos detection tests that were used in the financial markets are usually theoretical in nature and lack the practicality in such markets because they were designed for clean data. Their practical application for the bitcoin market is mitigated and inconclusive as the bitcoin time series is highly noisy. In this study, we utilized the largest Lyapunov exponent test to detect the presence of chaos in the bitcoin market. We tested the methodology described in this study, which was first tested for an actual chaotic system (logistic map in this case) in the presence of heteroskedastic noise to confirm its applicability to a real world time series, to detect chaos in the bitcoin time series. We found strong evidence that the bitcoin market indeed has nonlinear dynamical patterns and thus is chaotic in nature. The results of the conducted simulations reinforced the concept that bitcoin prices exhibit chaotic dynamics, based solely on the amplitude and sign of the Lyapunov exponent. We concluded and answered the skeptics that bitcoin price fluctuations are chaotic in nature and not stochastic. The detection of the presence of chaos in bitcoin prices is the first contribution of this study.

The traditional time series prediction methods generally use unpolished data to develop prediction models to predict the prices of financial assets. But this type of approach has many limitations because, with such unpolished data, the models can't learn complex and subtle trends. In the recent rapid development of artificially intelligent prediction models, LSTM is considered the most advanced model for learning the sequential patterns hidden in time series data. However, in the bitcoin literature, its real prediction potential is still untapped.

The second contribution of this study is that we have proposed a new Chaos+LSTM hybrid model based on time series prediction, that aims at predicting future prices of bitcoin. The instability of the bitcoin market and high fluctuating prices are the outcome of the complex and nonlinear dynamics that may operate on multiple time scales. This study provides a solution to account for the information contained at multiple times scales by modeling nonlinear dynamics using phase space reconstruction. Moreover, our model captures the insightful hidden information and nonlinear patterns by modeling the chaos present in the bitcoin time series and addresses the complex chaotic dynamics of bitcoin prices. Such valuable information is then utilized to implement the LSTM part of the proposed model, as the phase space reconstructed information when integrated with LSTM can drastically improve the performance of the model. We exploited the application of time series phase space reconstruction to model the nonlinear and chaotic dynamics of bitcoin time series. We were able to embed our bitcoin time series into multiple dimensions using an optimal embedding dimension of X which was computed using the False Nearest Neighbors algorithm. *The more is the number of embedding dimensions required to reconstruct phase space to model the behavior of a complex system, the more chaotic are the dynamics of that system.*

We used the phase space reconstructed time series to train our model. The performance of the proposed model was evaluated using the RMSE metric and compared with other state-of-the-art models. The prediction results and RMSE values demonstrated that the proposed Chaos+LSTM hybrid outperformed other models. The Chaos+LSTM model generated better predictions than the stand-alone LSTM model that was developed using the original time series. This provides strong evidence that modeling chaos in the time series can significantly improve the performance of a prediction model. The good prediction results prove the usefulness of the proposed Chaos+LSTM hybrid for predicting bitcoin prices. The proposed approach can also be used for other cryptocurrencies such as Ethereum, Ripple, Bitcoin Cash, Litecoin, etc., but the dynamics of these markets might be different from

the Bitcoin market, so a researcher might have to analyze and model their dynamics differently if needed.
We also conclude that if LSTM undergoes accurate training, it can effectively remember and regulate past sequences of information and learn nonlinear patterns to yield better predictions, given that its hyperparameters are tuned appropriately. Unlike for small datasets, the hyperparameter tuning can be computationally extensive for larger datasets. However, for bitcoin prices, the market data is limited, so, the computational complexity is not an issue.
The results of this study show that when complex deep learning models such as LSTM are trained with data that provides a good understanding of the factors that influence bitcoin prices and by tuning the right hyperparameters, it can be beneficial for assessing investment risks and making financial gains. In the present time, prediction models based on complex deep recurrent neural networks are mostly preferred over other machine learning models for making bitcoin price predictions as they can learn longer sequences of information and retain that information for longer periods of time compared to other machine learning models. Although in the literature, there is enough evidence of these models performing much better than traditional models at predicting future prices, they have their limitations; thus making investments risky. This limitation motivated us to propose a novel hybrid Chaos+LSTM model that can contribute towards reducing financial losses and providing better predictions than conventional models to make investments less risky. Due to the relatively young age of the bitcoin market, the lack of availability of information and previous research on this domain, and the scale of this research and limited time horizon, we were not able to utilize all the factors that govern the dynamics of the bitcoin market. Nonetheless, there is a wide scope to greatly improve the quality of solutions for the bitcoin price prediction problem by doing new research given this study as a baseline. The current study can be significantly improved if more information about the market is available. Last but not least, there is a myriad of research possibilities for future studies. It would be interesting to consider other factors such as bitcoin blockchain information, public sentiments, news, and social media trends, related financial assets, and price fluctuations of Altcoins such as Ethereum, Ripple, Bitcoin Cash, etc. The potential future research can explore

the untapped prediction potential by utilizing all these factors intelligently.

Bibliography

[1] A. Bensaïda and H. Litimi, "High level chaos in the exchange and index markets," *Chaos, Solitons & Fractals*, vol. 54, pp. 90–95, 2013.

[2] E. Bradley and H. Kantz, "Nonlinear time-series analysis revisited," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 25, no. 9, p. 097610, 2015.

[3] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical systems and turbulence, Warwick 1980.* Springer, 1981, pp. 366–381.

[4] S. Lahmiri and S. Bekiros, "Chaos, randomness and multi-fractality in bitcoin market," *Chaos, solitons & fractals*, vol. 106, pp. 28–34, 2018.

[5] G. Çoban and A. H. Büyüklü, "Deterministic flow in phase space of exchange rates: Evidence of chaos in filtered series of turkish lira–dollar daily growth rates," *Chaos, Solitons & Fractals*, vol. 42, no. 2, pp. 1062–1067, 2009.

[6] S. Lahmiri, "Investigating existence of chaos in short and long term dynamics of moroccan exchange rates," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 655–661, 2017.

[7] ——, "A study on chaos in crude oil markets before and after 2008 international financial crisis," *Physica A: Statistical Mechanics and its Applications*, vol. 466, pp. 389–395, 2017.

[8] A. Serletis and M. Shintani, "No evidence of chaos but some evidence of dependence in the us stock market," *Chaos, Solitons & Fractals*, vol. 17, no. 2-3, pp. 449–454, 2003.

[9] S. Lahmiri, “On fractality and chaos in moroccan family business stock returns and volatility,” *Physica A: Statistical Mechanics and its Applications*, vol. 473, pp. 29–39, 2017.

[10] P. Ciaian, M. Rajcaniova, and d. Kancs, “The economics of bitcoin price formation,” *Applied Economics*, vol. 48, no. 19, pp. 1799–1815, 2016.

[11] T. Guo, A. Bifet, and N. Antulov-Fantulin, “Bitcoin volatility forecasting with a glimpse into buy and sell orders,” in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 989–994.

[12] G. Gajardo, W. D. Kristjanpoller, and M. Minutolo, “Does bitcoin exhibit the same asymmetric multifractal cross-correlations with crude oil, gold and djia as the euro, great british pound and yen?” *Chaos, Solitons & Fractals*, vol. 109, pp. 195–205, 2018.

[13] O. Poyser, “Exploring the determinants of bitcoin’s price: an application of bayesian structural time series,” *arXiv preprint arXiv:1706.01437*, 2017.

[14] N. Gandal and H. Halaburda, “Can we predict the winner in a market with network effects? competition in cryptocurrency market,” *Games*, vol. 7, no. 3, p. 16, 2016.

[15] Y. Sovbetov, “Factors influencing cryptocurrency prices: Evidence from bitcoin, ethereum, dash, litcoin, and monero,” *Journal of Economics and Financial Analysis*, vol. 2, no. 2, pp. 1–27, 2018.

[16] M. Muzammal, Q. Qu, and B. Nasrulin, “Renovating blockchain with distributed databases: An open source system,” *Future generation computer systems*, vol. 90, pp. 105–117, 2019.

[17] M. Briere, K. Oosterlinck, and A. Szafarz, “Virtual currency, tangible return: Portfolio diversification with bitcoin,” *Journal of Asset Management*, vol. 16, no. 6, pp. 365–373, 2015.

[18] A. Blundell-Wignall, “The bitcoin question,” 2014.

[19] S. Lo and J. C. Wang, “Bitcoin as money?” 2014.

[20] B. Gu, P. Konana, A. Liu, B. Rajagopalan, and J. Ghosh, “Identifying information in stock message boards and its implications for stock market efficiency,” in *Workshop on Information Systems and Economics, Los Angeles, CA*, 2006.

[21] I. Kaastra and M. Boyd, “Designing a neural network for forecasting financial,” *Neurocomputing*, vol. 10, pp. 215–236, 1996.

[22] H. White, “Economic prediction using neural networks: The case of ibm daily stock returns,” in *ICNN*, vol. 2, 1988, pp. 451–458.

[23] C. Chatfield and M. Yar, “Holt-winters forecasting: some practical issues,” *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 37, no. 2, pp. 129–140, 1988.

[24] L. Alessandretti, A. ElBahrawy, L. M. Aiello, and A. Baronchelli, “Anticipating cryptocurrency prices using machine learning,” *Complexity*, vol. 2018, 2018.

[25] S. Siami-Namini and A. S. Namin, “Forecasting economics and financial time series: Arima vs. lstm,” *arXiv preprint arXiv:1803.06386*, 2018.

[26] A. Chaudhari, “Forecasting cryptocurrency prices using machine learning,” Ph.D. dissertation, Dublin, National College of Ireland, 2020.

[27] M.-Y. Chen, M.-H. Fan, Y.-L. Chen, and H.-M. Wei, “Design of experiments on neural network’s parameters optimization for time series forecasting in stock markets,” *Neural Network World*, vol. 23, no. 4, p. 369, 2013.

[28] A. F. Sheta, S. E. M. Ahmed, and H. Faris, “A comparison between regression, artificial neural networks and support vector machines for predicting stock market index,” *Soft Computing*, vol. 7, no. 8, 2015.

[29] S. McNally, J. Roche, and S. Caton, “Predicting the price of bitcoin using machine learning,” in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 2018, pp. 339–343.

[30] I. Madan, S. Saluja, and A. Zhao, “Automated bitcoin trading via machine learning algorithms,” *URL: http://cs229. stanford. edu/proj2014/Isaac% 20Madan*, vol. 20, 2015.

[31] H. Jang and J. Lee, "An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information," *Ieee Access*, vol. 6, pp. 5427–5437, 2017.

[32] S. Ji, J. Kim, and H. Im, "A comparative study of bitcoin price prediction using deep learning," *Mathematics*, vol. 7, no. 10, p. 898, 2019.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[34] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, vol. 19, pp. 143–155, 1989.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] J. M. Bates and C. W. Granger, "The combination of forecasts," *Journal of the Operational Research Society*, vol. 20, no. 4, pp. 451–468, 1969.

[38] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler, "The accuracy of extrapolation (time series) methods: Results of a forecasting competition," *Journal of forecasting*, vol. 1, no. 2, pp. 111–153, 1982.

[39] E. Pelikan, C. De Groot, and D. Wurtz, "Power consumption in west-bohemia: improved forecasts with decorrelating connectionist networks," *Neural Network World*, vol. 2, no. 6, pp. 701–712, 1992.

[40] I. A. Hashish, F. Forni, G. Andreotti, T. Facchinetti, and S. Darjani, "A hybrid model for bitcoin prices prediction using hidden markov models and optimized lstm networks," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 721–728.

[41] E. Sin and L. Wang, “Bitcoin price prediction using ensembles of neural networks,” in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD).* IEEE, 2017, pp. 666–671.

[42] C.-H. Wu, C.-C. Lu, Y.-F. Ma, and R.-S. Lu, “A new forecasting framework for bitcoin price with lstm,” in *2018 IEEE International Conference on Data Mining Workshops (ICDMW).* IEEE, 2018, pp. 168–175.

[43] N. Indera, I. Yassin, A. Zabidi, and Z. Rizman, “Non-linear autoregressive with exogeneous input (narx) bitcoin price prediction model using pso-optimized parameters and moving average technical indicators,” *Journal of Fundamental and Applied Sciences*, vol. 9, no. 3S, pp. 791–808, 2017.

[44] V. Ravi, D. Pradeepkumar, and K. Deb, “Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms,” *Swarm and Evolutionary Computation*, vol. 36, pp. 136–149, 2017.

[45] A. Dutta, S. Kumar, and M. Basu, “A gated recurrent unit approach to bitcoin price prediction,” *Journal of Risk and Financial Management*, vol. 13, no. 2, p. 23, 2020.

[46] D. Shah and K. Zhang, “Bayesian regression and bitcoin,” in *2014 52nd annual Allerton conference on communication, control, and computing (Allerton).* IEEE, 2014, pp. 409–414.

[47] P. Lukáš and K. Taisei, “Volatility analysis of bitcoin price time series,” *Quantitative Financ and Econ*, vol. 1, pp. 474–485, 2017.

[48] T. Phaladisailoed and T. Numnonda, “Machine learning models comparison for bitcoin price prediction,” in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE).* IEEE, 2018, pp. 506–511.

[49] A. Saxena, T. Sukumar, T. Nadu, and T. Nadu, “Predicting bitcoin price using lstm and compare its predictability with arima model,” *Int. J. Pure Appl. Math*, vol. 119, no. 17, pp. 2591–2600, 2018.

[50] A. Greaves and B. Au, “Using the bitcoin transaction graph to predict the price of bitcoin,” *No Data*, 2015.

[51] P. T. Yamak, L. Yujian, and P. K. Gadosey, “A comparison between arima, lstm, and gru for time series forecasting,” in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, 2019, pp. 49–55.

[52] S. Roy, S. Nanjiba, and A. Chakrabarty, “Bitcoin price forecasting using time series analysis,” in *2018 21st International Conference of Computer and Information Technology (ICCIT).* IEEE, 2018, pp. 1–5.

[53] A. H. de Oliveira Monteiro, A. D. de Souza, B. G. Batista, and M. Zaparoli, “Market prediction in criptocurrency: A systematic literature mapping,” in *Proceedings of the XV Brazilian Symposium on Information Systems*, 2019, pp. 1–8.

[54] T. Guo and N. Antulov-Fantulin, “Predicting short-term bitcoin price fluctuations from buy and sell orders,” *arXiv preprint arXiv:1802.04065*, 2018.

[55] S. A. Ali Shah, W. Aziz, M. S. Ahmed Nadeem, M. Almaraashi, S.-O. Shim, and T. M. Habeebullah, “A novel phase space reconstruction-(psr-) based predictive algorithm to forecast atmospheric particulate matter concentration,” *Scientific Programming*, vol. 2019, 2019.

[56] J.-P. Eckmann and D. Ruelle, “Ergodic theory of chaos and strange attractors,” in *The theory of chaotic attractors.* Springer, 1985, pp. 273–312.

[57] G. Çoban, A. H. Büyüklü, and A. Das, “A linearization based non-iterative approach to measure the gaussian noise level for chaotic time series,” *Chaos, Solitons & Fractals*, vol. 45, no. 3, pp. 266–278, 2012.

[58] G. King and I. Stewart, “Phase space reconstruction for symmetric dynamical systems,” *Physica D: Nonlinear Phenomena*, vol. 58, no. 1-4, pp. 216–228, 1992.

[59] L. Cao, “Practical method for determining the minimum embedding dimension of a scalar time series,” *Physica D: Nonlinear Phenomena*, vol. 110, no. 1-2, pp. 43–50, 1997.

[60] A. BenSaïda, “A practical test for noisy chaotic dynamics,” *SoftwareX*, vol. 3, pp. 1–5, 2015.