University of Essex

Departmental of Mathematical Sciences

MA981 DISSERTATION

Cross-Dataset Time Series Forecasting using ARIMA and LSTM models

Rahul Kithalamane Basavaraj

Supervisor: **Savostyanov, Dmitry V**

November 24, 2023
Colchester

Contents

Abstract

Time series forecasting are used in various fields such as finance, economics, sales etc where data is recorded at equal time intervals of day, month or year with changes in phenomenon over time. Here we study the performance of two popular time series forecasting models: the Auto regressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) neural network model. However, the difference here in this research lies about cross-data set forecasting, where the models are tested on data sets way different from those used in training. In real-world scenarios, these techniques are particularly useful for knowing whether changes in one data set during a certain time period are affecting another data set in a different domain.

Both ARIMA and LSTM models are experimentally configured using a range of time series data sets from various domains. These data sets undergo preprocessing to address missing values, are meticulously configured, and their hyper-parameters are optimized to ensure compatibility with both ARIMA and LSTM models. Cross data set here defines the process of combining two or three different data sets to gain new insights and derive conclusions which usually cannot be answered using a single data set. The primary challenge at hand is to assess the performance of these models across diverse data sets and make meaningful comparisons.

The findings of this research throws light on the advantages and limitations of ARIMA and LSTM models in cross-data set forecasting. It further gives us insights into the capabilities of the model when comparing the trends with respect to two different data sets. The study delves into the literature and as well gives practical guidance for selecting the most suitable models for cross data set applications. These scenarios offers recommendations to select the best suitable model for improving forecasting accuracy in real world scenarios.

CHAPTER 1

Introduction

Time series forecasting (TSF) plays an important role in the field of data analysis. It has critical role in the prediction of various economic and financial cases especially which involves time component [39, 10]. However, this task is becoming very challenging due to the dynamic nature of data set trends in recent years.To address these challenges we have developed various forecasting methods right from the traditional statistical model such as Auto Regressive Integrated Moving Average (ARIMA) to the advanced machine learning algorithms like Long Short-Term Memory (LSTM) [31]. In this report we take a look into the performance of ARIMA and LSTM models in time series forecasting with a focus on economic and financial data set.

The Economic and financial time series data which we are trying to analyze here are characterized by non-linear stationary nature and are continuously subjected to volatility from external factors. Auto Regressive Integrated Moving Average (ARIMA) and Auto-Regression (AR) have been the long standing traditional statistical models in time series forecasting to model stationary data but these models too face challenges while dealing with large non-linear and complex data sets [30, 40]. The development of deep learning (DL) has transformed time series forecasting significantly. DL techniques such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks have gained importance in their ability to capture complex sequential dependencies in time series data [35, 55]. LSTMs, with their four key gates the input, output, forget, and candidate, have proven effective in modeling sequential data and outperforming traditional methods [41].

Recent advancement in the field of machine learning, in particular deep learning have introduced new approaches to time series forecasting. But their is a lack of empirical evidence with respect to LSTM's model out performing with respect to economic and financial time series data as compared to the traditional econometric methods like ARIMA even though they have shown enough promise with respect to various domains [37]. The primary research question addressed here is to assess the performance of traditional forecasting models such as ARIMA with deep learning based algorithms specifically LSTM across diverse data sets and make meaningful comparisons [48].

The real world systems are filled with a combination of both linear and non-linear behaviors. Thus to handle such data sets we don't rely solely on one modeling technique. This is where a concept of cross data set time series forecasting comes into play. Our aim here is to combine ARIMA's linear approach with LSTM's non-linear capabilities to create an accurate forecasting model [45] [25] . By this approach we can help overcome the limitations inherent in each individual models and also provides solutions for the real world time series data problems.

In this research we have proposed a two phase hybrid model [34, 20] . In the first phase we use ARIMA to identify the linear structures present in the data. ARIMA is particularly used when the time series data set is stationary and it contains no missing values. Autoregressive (AR) and moving average (MA) are the two main models here created to represent the linear relationship. By preprocessing the data with ARIMA, we find out the linear patterns present in them [14, 42].

In the second phase of the model we have used LSTM, a non linear neural network. This model is capable of handling the preprocessed data, identify the complex non linear relationships and dependencies present in the data. Whenever we employ an extension of LSTM known as deep LSTMS's, then they operate each block in the hierarchical architecture at different time scales [45]. By combining both the models we find out the capabilities in each of these models to identify the linear and non-linear capabilities so as to find out a comprehensive solution for time series forecasting data set [4].

The approaches of both the models in the time series data have various applications such as photovoltaic power forecasting, anomaly detection, predict wind speed, audio signal classification, indoor temperature prediction, traffic flow forecasting, weather

prediction etc.

In summary, as we proceed through the paper we will demonstrate the challenges present in various time series data, the potential in the combination of ARIMA and LSTM models to increase the accuracy and reliability in time series forecasting [57, 33]. Along with traditional methods like ARIMA, TSF has done significant advancement in integrating deep learning techniques particularly LSTM based models to address the challenges of various types of time series data [31]. We aim to contribute to the ever evolving forecasting techniques and provide a robust and reliable predictions in a wide range of applications.

CHAPTER 2

Literature Review

Time series forecasting is significant to the process of extracting meaningful statistical insights from diverse domains such as economics, finance, sales and more. Traditional methods like auto regressive integrated moving average (ARIMA) and long short-term memory (LSTM) neural network models face certain challenges with respect to complexity and reliability of the data and the model [57]. What sets this research apart is it's focus on cross data set time series forecasting to assess whether changes in one data set during a specific time period can impact another data set of a different domain.

The significance of this research lies in its potential to shed light on the advantages and limitations of ARIMA and LSTM models in the context of cross-data set forecasting. Here we even compare the performance of the model to make sure that error rates are reduced [31]. The emergence of deep learning models have revolutionized time series forecasting techniques that are performing well in capturing complex sequential data [26, 25]. However we also know that the deep learning techniques such as LSTM requires a large data in time series forecasting and to address this issue we used transfer learning techniques of machine learning that involves taking a pre trained model and reusing it as a starting point for the related task [28].

The primary objective for this literature review aims to address the understanding of capabilities between the traditional forecasting models such as ARIMA, compared to deep learning models such as LSTM. This in turn will guide us in selecting the most suitable model for enhancing accuracy and reliability in real-world applications. We even evaluate the strengths and limitations of the model, examine the capabilities of

deep learning models, investigate the existing empirical evidence, explore the concept of cross data set time series forecasting and also providing insights into the practical applications of these models [40].

ARIMA, introduced by Box and Jenkins in 1976 [34, 53, 20, 14], has been a long reliable choice for modeling and predicting time series data . It combines various components such as auto regressive (AR), integrated (I), and moving average (MA) to characterize time series. AR models the influence of past observations, MA controls noise in the data, and the integration term ensures it's stationary [31]. Despite ARIMA being robust, it has certain limitations in distinguishing between stationary and non-stationary processes and constructing confidence intervals, especially for the MA component. Moreover, the challenge is with respect to computational complexity, particularly while dealing with large and complex data sets [48].

LSTM, introduced by Hochreiter and Schmidhuber in 1997 [30, 31], was designed in such a way to overcome the vanishing gradient problem in traditional RNNs . LSTM introduces it's key components which includes the forget gate, input gate, output gate, and cell state update [38]. These elements captures and remembers the very intricate sequential dependencies while working at the same moment, making LSTM highly effective in modeling time series data. The forget gate checks what information should be retained, the input gate regulates when the addition of new information takes place, the output gate determines what information should be used for the hidden state, and the cell state update combines and updates the cell state. LSTM's architecture has the capability to capture long-range of dependencies in data set, making it a very powerful tool for time series forecasting [34].

LSTM has found applications in various domains such as natural language processing, speech recognition and more even including time series analysis. In the context of LSTM, we need to further explore whats the precise number of epochs which are needed to better influence the performance of deep learning models so that a comprehensive time series forecasting methods are found out in the context of economics and finance [31]. In a research the author introduces an innovative transfer learning [50] framework for it's utility in real world applications of time series prediction, which addresses data limitations and cold start issues. This innovative transfer learning framework used here determines which hidden layers can be transferred between the source domain

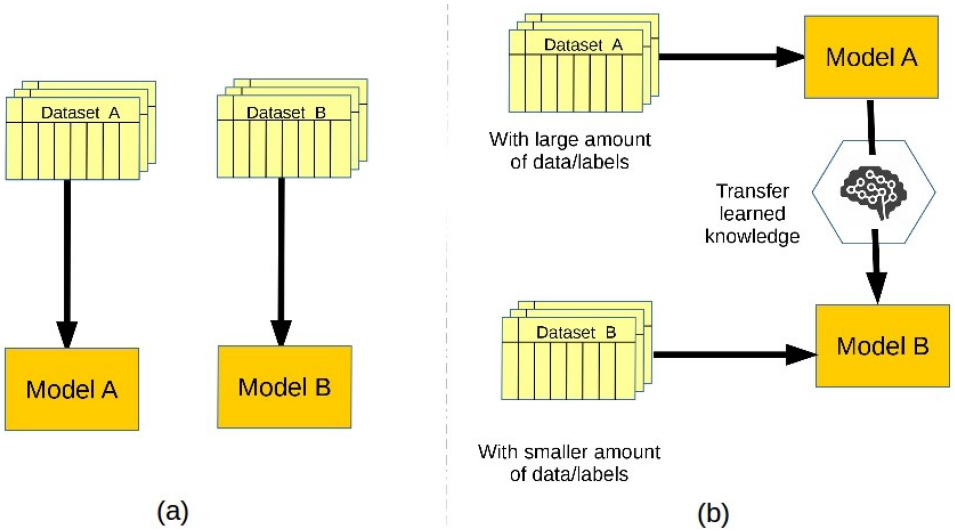

Figure 2.1: Representation of (a) traditional machine learning (b) transfer Learning [48]

and the target domain [56]. In case the data lacks sufficient clean training data from the target domain the author propose the relationship-aligned transfer learning (RATL) algorithm which transfers knowledge from the source domain to the target domain in turn improving forecasting performance [54].

The study finds that LSTM has a reduced error rate of 84–87% when compared to ARIMA. While the number of training epochs has no significant impact on the model's performance but it outperforms ARIMA in every prediction [30]. The development of a hybrid model which integrates two well established techniques captures both trend and seasonal components, improved accuracy and flexibility of the model [34]. The architecture of the proposed CNN model which employees stacked dilated convolutions, ReLU activation functions and 1×1 convolutions for forecasting discusses the use of weighted optimization method to minimize mean absolute error [23]. The author in one of his research presents valuable exploration of ANNs for non linear time series forecasting but has left further research into the model selection, comparative analysis and normalization of data [11].

In the context of TSF, the long-interval time series data sets have been developed and addressed by LSTM-based models, such as deep LSTM (DLSTM) [51]. The DLSTM architecture is applied to the auto encoder model to forecast future data points based on historical sequential data [48]. Both the encoder and decoder layers which are used to capture the intricate temporal patterns in the input sequence and output sequence are replaced with deep LSTM's. DLSTMs handle input at different time scales, with

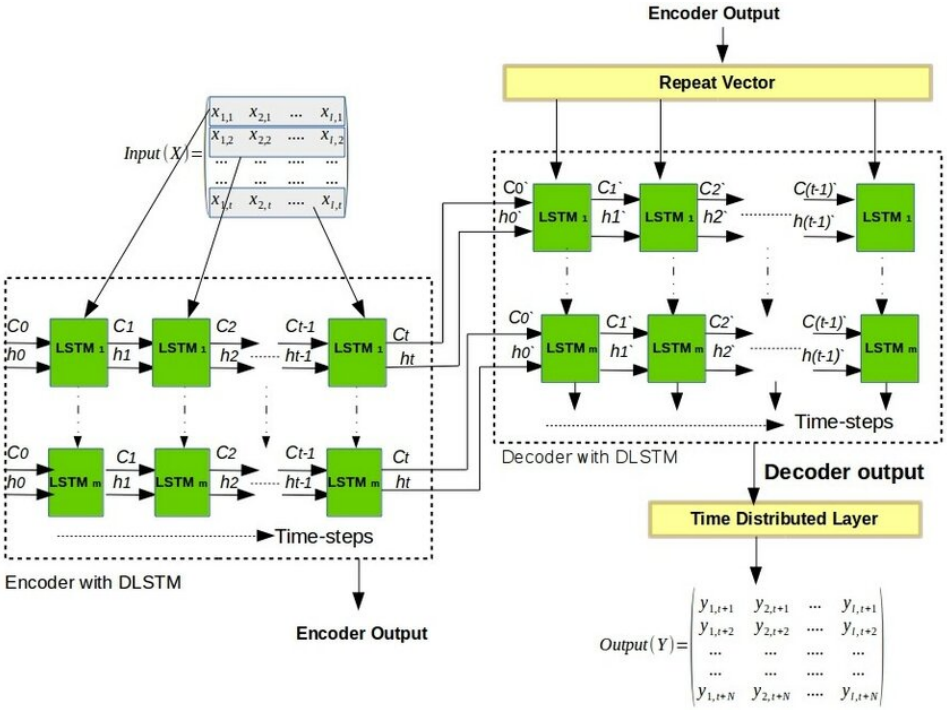

Figure 2.2: DLSTM-based Auto-Encoder architecture [48]

prediction at each step using a dense independent layer and a stacked LSTM hierarchical design. The LSTM networks are a type of recurrent neural network (RNN) which are known for their ability to capture long term dependencies in sequential data, disentangle variations of input data with the help of hierarchical approach, which in turn offers enhanced accuracy and reliability in time series forecasts leading to dynamic predictions [48].

In conclusion, the literature review provides valuable insights into the strengths and limitations of traditional and deep learning models for time series forecasting. It emphasizes the potential of deep learning models like LSTM and introduces concepts like transfer learning [50] to address data limitations. The research suggests that LSTM-based models outperform ARIMA in terms of forecasting accuracy, particularly for complex time series data. Further research is much needed to fine-tune these models and explore their real-world applications in various domains.

CHAPTER 3

Methodology and Analysis

Time series forecasting plays an important role while dealing with wide range of problems. The number of techniques which are available for implementing time series forecasting, comes with a unique range of characteristics. Some times the data can be continuous or discrete[55]. The main objectives of time series analysis are summarizing data with statistics and plots, then finding a suitable statistical model along with the estimation of future values of the series by means of control [10]. The primary goal is to construct a model, observe the patterns present in the time series data and make future predictions and enhance the prediction accuracy based on this data. Traditionally time series forecasting are carried out by ARIMA models, a method introduced and generalized by Box and Jenkins in 1970 [34, 53, 20, 14].

The research of this paper is to build these two time series forecasting models (LSTM and ARIMA), determine the performance of these time series forecasting models across diverse data sets and make meaningful comparisons. The chosen methodology involves various stages in the process of evaluating the model, first understanding the problem, then with data gathering and data preprocessing, exploratory data analysis, selecting libraries and training models using various metrics [57]. The limitations of ARIMA models such as non-linear relationships between variables and assumption of a constant standard deviation in the model's errors can be addressed with a generalized auto regressive conditional heteroskedasticity (GARCH) model [30]. LSTM (Long Short-Term Memory) is a specific type of recurrent neural network (RNN) which was introduced by Hochreiter and Schmidhuber in 1979 [30, 31]. New techniques which are

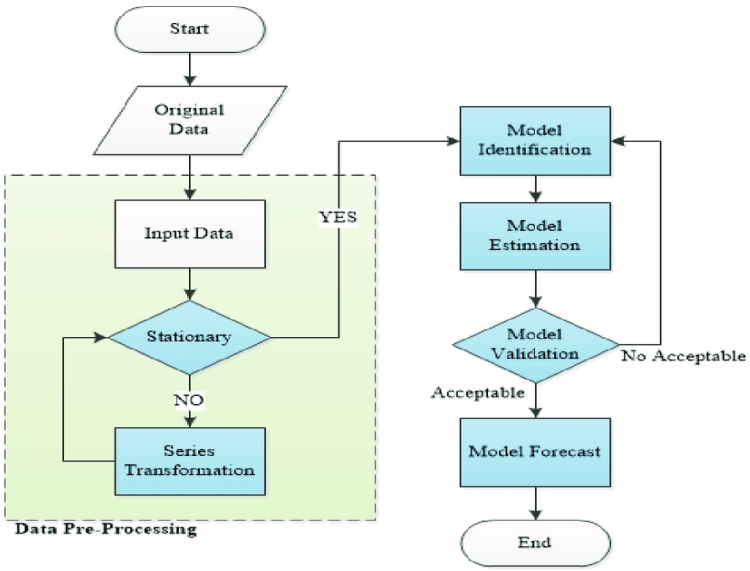

Figure 3.1: Flow Chart of Time Series Methodology [33]

emerging in deep learning are focused on tackling the limitations present in the model.

3.1 Data Collection

Historical stock prices are an important starting point to undertake cross-data set time series forecasting, as their data sets are accessible through various sources. One such widely used source to download the data especially in .csv format is yahoo finance. We utilized the historical NVDA (NVIDIA) data spanning from September 17, 2014, to November 10, 2023, totaling 2305 data points, to compare it with the historical BTC-USD (bitcoin) transaction data obtained from crypto spanning from September 17, 2014, to November 10, 2023, totaling 3344 data points. The shape of NVDA.csv data set is (2305, 7) and that of BTC-USD.csv data set is (3344, 7). These historical stock prices also referred to as technical indicator's, are the foundation to any company's financial conditions. We collected data set which includes timestamp of each day and a corresponding variables to each day which clearly presents each days opening, high, closing prices along with the total volume of stocks traded each day. To study momentum, volume, volatility, and cycle-based indicators we incorporated diverse range of technical indicators where these indicators are tailored to focus on different aspect of market activity. Investopedia defined that these indicators are derived from the price activity that evaluates aspects such as price levels, directions, momentum, and more [25, 1].

3.1.1 Data Description

The several features included in the data set, with each having a specific purpose are:

- **Date:** The date when the data was recorded.
- **Open:** The opening trading value on a given day.
- **High:** The highest trading value reached during the day.
- **Low:** The lowest trading value observed on the same day.
- **Close:** The most recent trading value recorded.
- **Adj Close:** The modified closing price index that reflects accurate stock value.
- **Volume BTC:** The total trading volume in bitcoin (BTC) for the day.
- **Volume USD:** The total trading volume in US dollars (USD) for the day.

3.1.2 Data Preprocessing

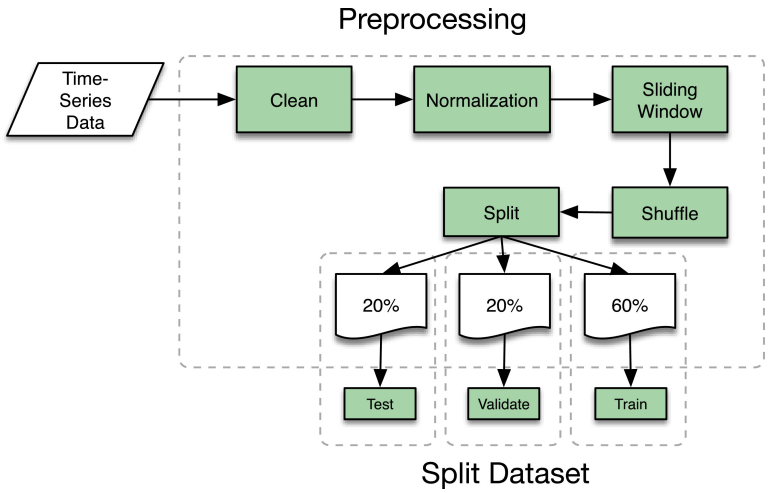

Figure 3.2: Data Preprocessing [42]

In the first step of our process we examined the data collected for missing or null values, checking for duplicates, outliers or inconsistencies, checking for duplicates, dropping unnecessary columns, checking whether time stamps are taken in proper chronological order, normalizing the column as well as displaying and saving the

```
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2305 non-null   object
 1   Open       2305 non-null   float64
 2   High       2305 non-null   float64
 3   Low        2305 non-null   float64
 4   Close      2305 non-null   float64
 5   Adj Close  2305 non-null   float64
 6   Volume     2305 non-null   int64
```

Figure 3.3: Number of missing values in NVDA.csv

```
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       3344 non-null   object
 1   Open       3344 non-null   float64
 2   High       3344 non-null   float64
 3   Low        3344 non-null   float64
 4   Close      3344 non-null   float64
 5   Adj Close  3344 non-null   float64
 6   Volume     3344 non-null   float64
```

Figure 3.4: Number of missing values in BTC-USD.csv

preprocessed data [57]. This step is crucial in addressing data quality issues so that the data is ready for analysis or model training.

To conduct the preprocessing steps, we used simple python libraries such as pandas and sklearn. In the first step of our analysis we checked the data for missing values. We found that the sum of missing values in each column were equal to 0. In the next step we converted the date column to date time format so that it gets easier for manipulation and analysis of temporal patterns. This column is then sorted chronologically. In time series analysis this becomes an important step. The date column is always set as an index in time series analysis so that it becomes easier to perform date-based indexing and slicing. Sometimes the data might be containing duplicate values and to ensure meaningful analysis we removed the duplicate values and then we dropped those unnecessary columns to ensure our focus only on adj close prices. The adj close prices were then normalized using MinMaxScaler of sklearn. This method was performed so as to ensure the values range between 0 and 1. The preprocessed data was then

saved on to a new csv file ready for exploratory data analysis (EDA) and subsequent modeling.

3.2 Exploratory Data Analysis

Exploratory data analysis is an important step in the process of analysing and summarizing the most important relationships between various variables present in the raw data. This analysis leads us to visualization of both structured and unstructured data by means of both statistical graphics and techniques used in them [57]. In the first step of analysis the basic information about the data set were displayed, this included each column data types and their non null count displayed in fig 3.3 and fig 3.4. We then displayed the sample of first few rows of the data set and provided the summary statistics (count, mean, std, min,max, etc) for the numerical columns as referenced to in fig 3.5,3.6, 3.7 and 3.8.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	4.7725	4.8325	4.7500	4.7875	4.578554	17358400
1	2014-09-18	4.8175	4.8650	4.8025	4.8600	4.647889	21960400
2	2014-09-19	4.8750	4.8800	4.7625	4.7700	4.561820	60131600
3	2014-09-22	4.8675	4.8675	4.7175	4.7225	4.516392	22146400
4	2014-09-23	4.6975	4.7550	4.6800	4.7025	4.497263	19614400

Figure 3.5: Display of first few rows NVIDIA Data

	Date	Open	High	Low	Close	Adj Close
0	17-09-2014	465.864014	468.174011	452.421997	457.334015	457.334015
1	18-09-2014	456.859985	456.859985	413.104004	424.440002	424.440002
2	19-09-2014	424.102997	427.834991	384.532013	394.795990	394.795990
3	20-09-2014	394.673004	423.295990	389.882996	408.903992	408.903992
4	21-09-2014	408.084991	412.425995	393.181000	398.821014	398.821014

Figure 3.6: Display of first few rows BTC Data

In the next step of analysis data was checked for missing values. It showed no signs of missing values. To visualize how our target column i.e, adj close is distributed over a period of time, we did distribution plotting of the column using histogram as evidenced in the fig 3.9 and 3.10. The plot shows that both the data sets were right skewed meaning the maximum price distribution were during the first half of the period

	Open	High	Low	Close	Adj Close
count	2305.000000	2305.000000	2305.000000	2305.000000	2305.000000
mean	102.553175	104.449378	100.600690	102.611079	102.337142
std	111.170118	113.222123	108.998550	111.177143	111.228515
min	4.232500	4.325000	4.192500	4.197500	4.014305
25%	24.642500	24.985001	24.350000	24.684999	24.363161
50%	56.750000	57.587502	55.549999	56.560001	56.004223
75%	151.080002	153.524994	148.467499	151.300003	151.162582
max	502.160004	502.660004	489.579987	493.549988	493.509338

Figure 3.7: Summary statistics NVIDIA Data

	Open	High	Low	Close	Adj Close
count	3344.000000	3344.000000	3344.000000	3344.000000	3344.000000
mean	14192.888679	14528.193925	13830.341463	14202.511578	14202.511578
std	15992.967844	16382.993808	15553.120416	15993.051558	15993.051558
min	176.897003	211.731003	171.509995	178.102997	178.102997
25%	891.026764	904.273514	852.265259	894.441254	894.441254
50%	8141.181640	8273.189942	7911.633056	8145.025879	8145.025879
75%	22948.430177	23373.049802	22563.162110	22957.069340	22957.069340
max	67549.734380	68789.625000	66382.062500	67566.828130	67566.828130

Figure 3.8: Summary statistics BTC-USD Data

selected. We then created a correlation matrix between numerical variables and the display indicates that most of them have a perfect positive correlation coefficient of 1 meaning a linear relationship exist between the two variables. As one variable increases the other variable also increases proportionally.

We then visualized the adj closing prices over time so as to observe the trends and patterns present in the data. The distribution of adj close prices was visualised for each month using a box plot as referenced in fig 3.11 and 3.12, and the display found that the maximum values were obtained in the months of march, april, and may for the BTC data set, but the largest values of adj closing prices were obtained in the month of july for the NVDA data set. Adjusted close prices with 30 day rolling mean and standard deviation were plotted to identify trends and volatility. The main aim of this preliminary analysis is to understand the structure of the data, handle missing values, calculate basic statistical measure, correlation analysis, identify any outliers which are present in the data and also explore the relationship between different features. The key steps in this stage of analysis are to ensure data's quality and integrity [57].

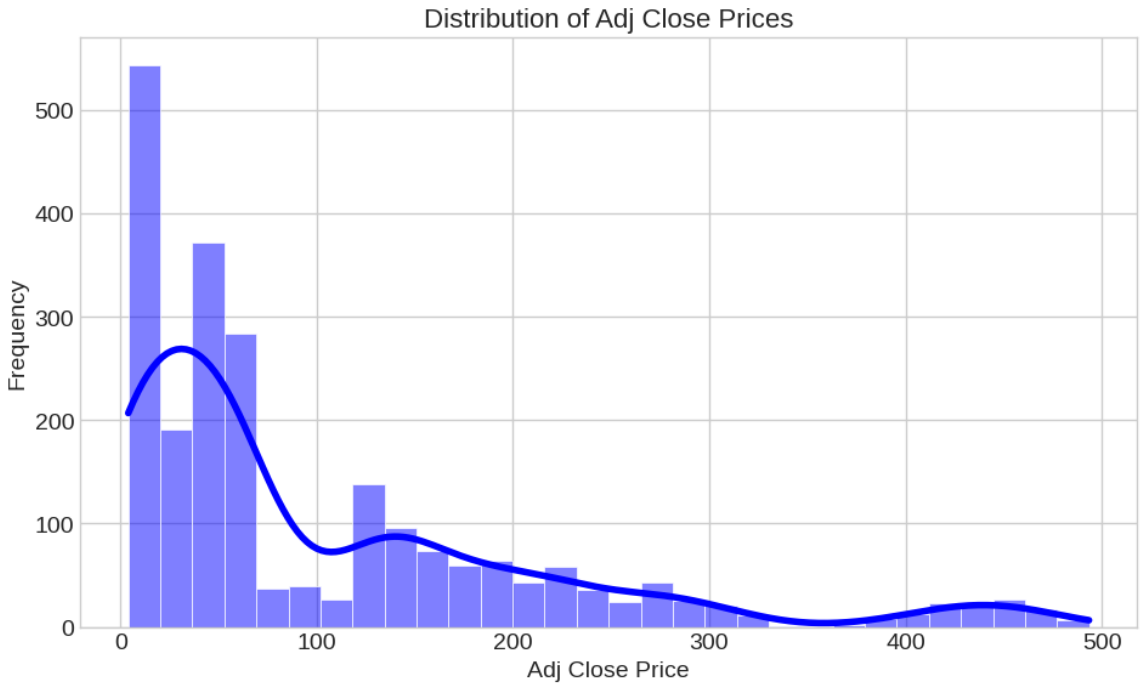

Figure 3.9: NVDA Data

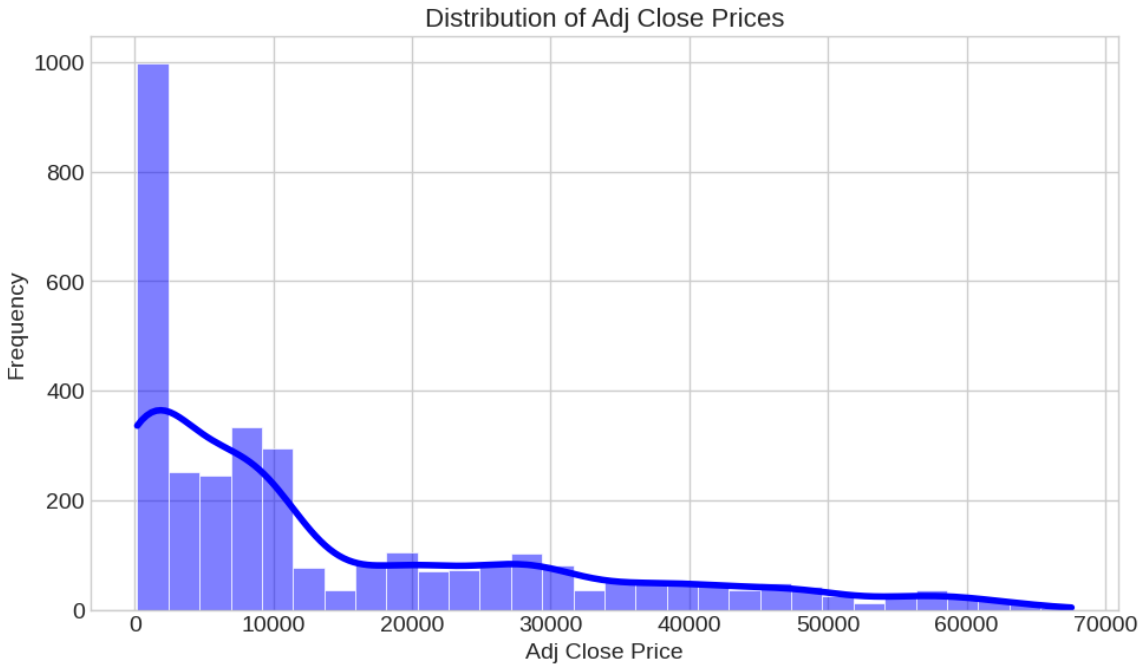

Figure 3.10: BTC-USD Data

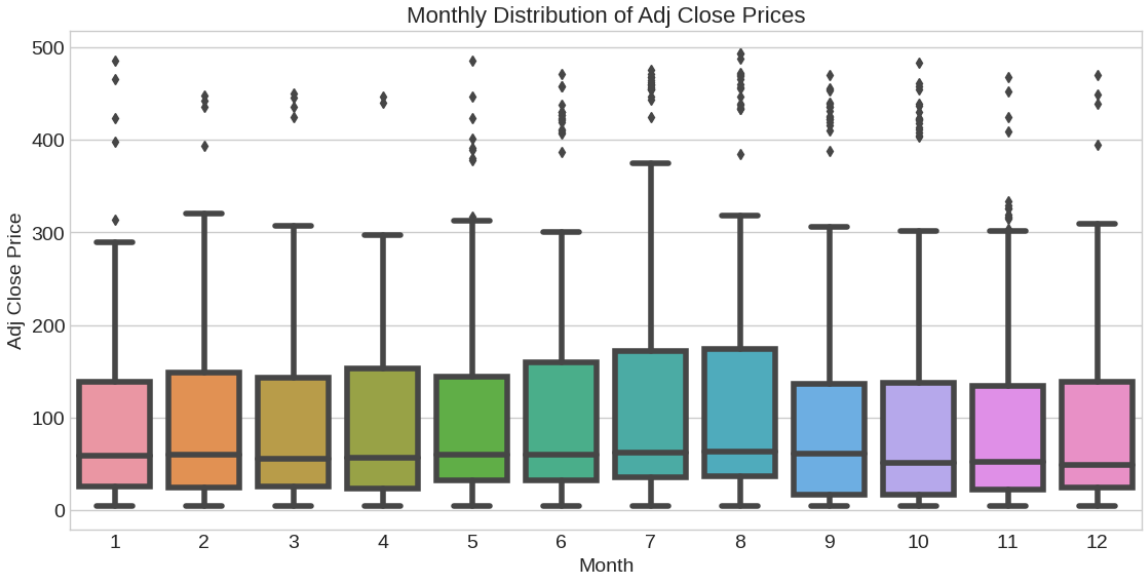

Figure 3.11: NVDA Data

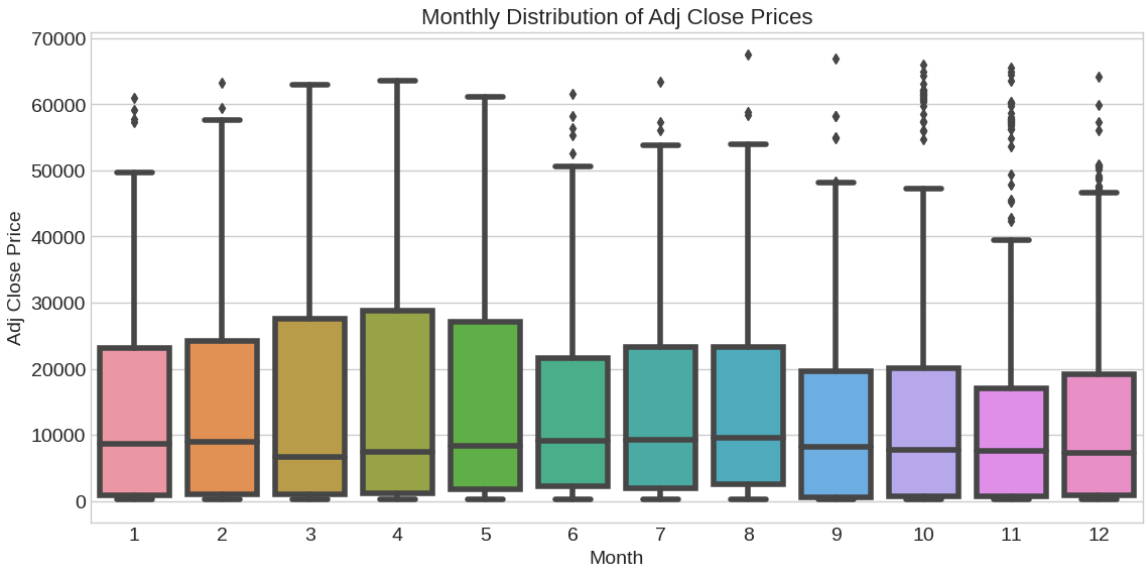

Figure 3.12: BTC-USD Data

3.2.1 Choosing Libraries and Training Models

For model building and prediction, we used python in-built libraries that deals with machine learning and deep learning problems. We used the ARIMA and LSTM models in our time series forecasting methodology. However, before we begin the training process, we perform hyper parameter optimisation to ensure that the model provides the best accuracy. Each of the python libraries contains certain specific functions which are then put together in a single file so as to assist us in doing the obvious steps [57]. Different domains have their own specific set of libraries and the problem we are trying to solve here is of time series forecasting which is a type of machine learning and deep learning problem. The libraries which we are going to use to solve the above problem are sklearn, tensorflow, numpy, pandas, matplotlib, seaborn etc [57]. A typical deep learning model model contains input layer, multiple hidden layers and the outer layer. We choose rules and embed them into the problem type so that these models can serve specific purposes [57].

3.2.2 Evaluating Models

Model evaluation is a crucial step which involves measurement of various parameters. In time series forecasting, basic parameters include accuracy, precision, recall, r2 Score, mean square error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and median absolute error average and percentage (MDAPE). Accuracy, precision and recall provides general measures of correctness of predictions, significant consequences and capture positive instances. In time series forecasting r2 score helps us in understanding how well the model captures the variability in the data. Mean squared error measures the average squared deviation between predicted and actual values while the mean absolute error measures the absolute deviation of predictions from actual values. Mean absolute error, mean absolute percentage error and mean absolute error average and percentage helps us understand the absolute deviation of predictions, percentage measure of average relative error and the difference between them. We used these metrics collectively which helped in the assessment of the model's reliability and the accuracy of its predictions.

3.3 ARIMA Model: Auto regressive Integrated Moving Average

Time series analysis is an important technique in the analysis of historical data, forecasting future values and then extract meaningful patterns and characteristics [10]. The predicted value in this analysis is not just by historical data but also by additional variables. In the case of stock market data which is discussed here, closing value serves as a prediction variable and date is an additional variable which is considered in this model.

One of the prominent model which is used in our study is auto regressive modeling, specifically auto regressive integrated moving average (ARIMA). This model was introduced by Box and Jenkins in 1976 [34, 53, 20, 14]. ARIMA model remains very robust to capture correlation through lagged linear relations, handle non-stationary time series data by differencing the series and also predict time series data. ARIMA models combine auto regressive (AR) and moving average (MA) components, and and Box-Jenkins methodology [34, 53, 20, 14] utilizes the maximum likelihood method for parameter estimation. The auto regressive model of order p, denoted as AR(p), mathematically expresses this concept through an equation which involves lagged observations and a random error term. This model has drawn attention among the researchers and has also been widely applied in various fields where the statistical concepts are constant over time for predictive modeling and data analysis.

The accuracy of a forecasting model is measured by adjusting its parameters to minimize the difference between the predicted and actual values. The expectation is that any well trained models are able to predict values outside the data set and among them auto regressive integrated moving average models are highly effective in diverse applications even while predicting the stock market data.

The statistical properties in these models remains constant throughout and these models are designed in such a way they capture the correlation under assumption that time series is stationary. In case the model accounts for significant seasonal patterns [7] in the data with specific to auto regressive and moving average components, then this model is called as seasonal ARIMA (SARIMA) and it is also an extension to ARIMA model since it incorporates the model with seasonal terms only. The sequence of data

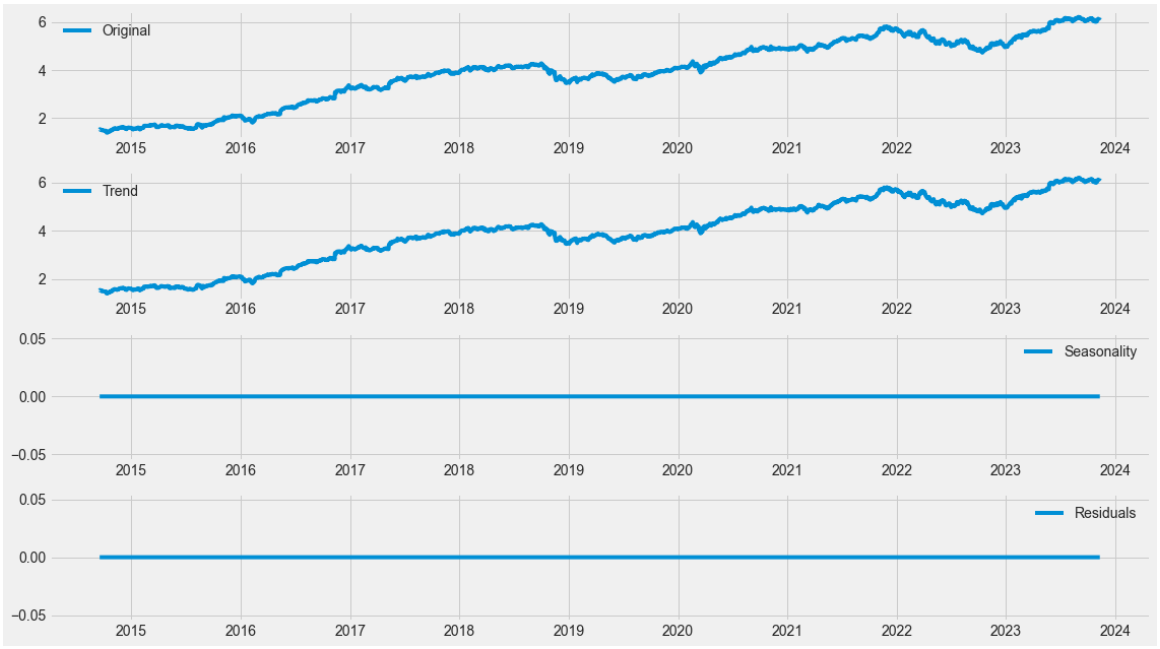

Figure 3.13: Stationarity, Seasonality and Trend

points which are collected at successive points of time are considered stationary if the statistical properties such as mean and variance remains constant throughout. For visual inspection of these statistical test we conduct dickey-fuller test. The visualisation inspection also gives out long term increase or decrease of data over time, checks if the patterns are repeating at regular intervals. These visualization patterns represents trends and seasonality in the data [7]. If the Box-Jenkins [34, 53, 20, 14] method are further extended by Box-Tiao method then we get introduced to a new model which is ARIMAX model. ARIMAX model mathematically includes auto regressive, moving average components and along with that some external factors are included into the model which is also known as covariates.

The methodology employed in this section of time series forecasting is auto regressive integrated moving average (ARIMA) model which is a combination three components mainly auto regressive (AR), integrated (I), and moving average (MA). The primary goal of this model is to predict future values based on historical data, where the input data is stationary. In order to streamline our modeling process, we understood the correlation between these feature so we divided both the data sets into a training set and a test set, following a ratio of 70% for training and 30% for testing [34]. However for forecasting and making informed decisions and predictions we focused on the adjusted close feature column. This division resulted in a training set consisting of 2340 data

points and a test set containing 1004 data for crypto and a training set consisting of 1613 data points and a test set containing 692 data points for NVIDIA data.

Box-Jenkins Method: ARIMA Model

The Box Jenkins [34, 53, 20, 14] method extends auto regressive modeling to ARIMA where it has played a fundamental role in time series forecasting for over half a century. In the context of this ARIMA model which are denoted by (p, d, q), the predicted variable is a function of linear patterns which are based on past observations and random errors. The general form of ARIMA model is expressed as:

$$1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)(1 - B)\, dY_t = \theta_0 + \theta_1 a_t + \theta_2 a_{t-1} + \ldots + \theta_q a_{t-q} \tag{3.1}$$

where,

Y_t : Actual value at time t,

a_t : Random error,

ϕ_i, θ_j : Model parameters,

B : Backward shift operator,

p, q : Orders of the model, and

d : Order of differencing.

a_t : Random errors, identically distributed with a mean of zero and a constant variance of $2\sigma^2$.

The Box Jenkins [34, 53, 20, 14] methodology comprises of three key steps which are sequential in nature: model identification (I), parameter estimation (E), and diagnostic checking (D). The prerequisite for ARIMA modeling is that it needs to be stationary over time, then a time series maintains constant statistical characteristics. In model identification process, the theoretical auto correlation properties are matched with empirical ones using tools like the auto correlation function and partial auto correlation function. The parameter estimation of ARIMA model is done by analyzing the patterns in these three key steps and then by utilizing selection methods such as Akaike's information criterion (AIC) and the minimum description length (MDL) we observe

that time series exhibits trends and heteroscedasticity. To eliminate and stabilize the trends and variance before fitting an ARIMA model we employ differencing and power transformation.

Forecasting a model to predict the value of an observation x_t based on previous observations in a statistical approach involves building up of stochastic models [39]. However, ARIMA models are constrained by the conditions of stationarity and invertibility, meaning time series should be time-invariant, stable, and exhibit independent and identically distributed residuals. A prerequisite for ARIMA modeling is stationarity in time series which has constant statistical properties over time. In the identification phase, it is necessary to perform data transformations to achieve stationarity. If we observe that time series exhibits trend and heteroscedasticity then we apply differencing and power transformation so that we can remove the trend and stabilize the variance before fitting the model.

The ARIMA model is represented as: The parameters (p, d, q) must be identified by analyzing time series stationarity. The Box-Jenkins [34, 53, 20, 14] method extends auto regressive modeling to ARIMA, where 'I' stands for integrated, implying differencing to achieve stationarity. The ARIMA model is expressed as:

$$\phi_p(B)(1-B)^d Y_t = \phi_0 + \theta_q(B)\varepsilon_t \tag{3.2}$$

where:

Y_t is the time series forecast variable,

B is the back shift operator, i.e., $BY_t = Y_{t-1}$,

d is the differencing operator,

$\phi_p(B)$ is the auto regressive polynomial of order p,

$(1-B)^d$ is the differencing operator of order d to achieve stationarity,

$\theta_q(B)$ is the regular moving average polynomial of order q,

ε_t is the error term.

This equation here represents the ARIMA model, where the time series has undergone differencing in order achieve stationarity. The notation ARIMA(p, d, q) is commonly used, where p is the number of lag observations, d is the quantity of differencing, and q is the size of the moving average window. The effectiveness of the ARIMA

model lies upon its ability to capture and represent various components of time series data, such as trends, seasonality, and stationarity. However, the choice of parameters (p, d, q) is critical and often requires iterative model building and diagnostic checking to ensure suitability. In addition to this, the ARIMA model may face challenges with regards to making accurate long-term predictions, as observed in studies analyzing both the data sets price trends over extended periods. Advanced techniques, such as online ARIMA algorithms where the data becomes available over time, have been proposed to enhance the efficiency and scalability of the model, especially for large-scale data.

Identification, Estimation, and Diagnostic Checking:

For effective evaluation of ARIMA model we must require the time series data to be stationary. The requirements for time series forecasting are characterized by constant mean, variance and covariances over time having weak stationarity. The evaluation of the ARIMA model is based on the assessment of the sum of squared correlations of estimated residuals, suitably scaled. Once the model is identified then our objective would be find the appropriate order of the ARIMA model. Next step the parameters of the ARIMA model are estimated using nonlinear optimization procedure to minimize the overall errors. The last step in this process is the assessment of ARIMA model adequacy.

Various statistical analysis and plots of residuals are performed on the data so that we can check how good the model is performing. The underlying patterns in the ARIMA model can be identified accurately by this three step iterative process. If the model is found to be unfit we then deploy the process of finding an alternative model in this analysis. Then the process key steps such as parameter estimation, identification and diagnostic checking are also repeated. The Box-Jenkins method involves the following steps:

- **Identification (I):** Determine the orders of differencing (d), auto regressive (p), and moving average (q) components by analyzing the auto correlation function (ACF) and partial auto correlation function (PACF) plots [57]. The auto correlation function measures the correlation between time series and its lagged values whereas partial auto correlation function measures the same after removing the intervening lags. These lag values are used for identifying ARIMA model pa-

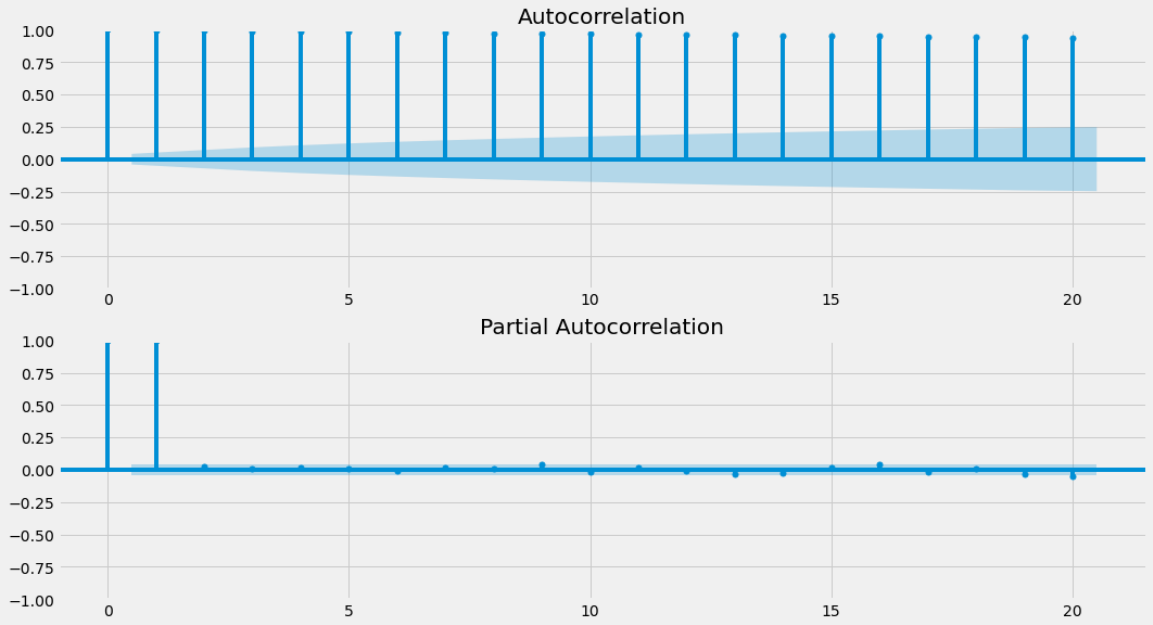

Figure 3.14: Auto correlation (ACF) and Partial auto correlation (PACF) graph

rameters. Identify trends and seasonality through time series plot examination. Apply statistical criteria such as Akaike information criterion (AIC) or Bayesian information criterion (BIC) for order selection. Consider domain knowledge and external factors. Determine stationarity through differencing.

- **Estimation (E):** Apply statistical methods to estimate the parameters of the ARIMA model such as AR and MA coefficients based on the identified orders. Iteratively refine the parameter estimates of the ARIMA model [33].

- **Diagnostic Checking (D):** Validate the adequacy of the ARIMA model. Examine the residuals to check whether they are uncorrelated with zero mean. Check for auto correlation based on statistical test on residuals. Identify patterns or deviations through plots. In case of inadequacies refine the model. Consider additional factors such as outliers.

The determination of suitable values for the parameters p and q in our ARIMA model was guided by an examination of the auto correlation and partial auto correlation plots. Multiple combinations of p, d, and q values were systematically tested based on these plots. Results obtained from the dickey-fuller test indicated that we cannot reject the null hypothesis. In order to address the non-stationarity, we employed the ARIMA model with differencing. The order of differencing was strategically selected to

transform our time series data into a stationary form. We ran multiple combinations of p,d and q values (for example: (1,1,1),(2,1,20),(1,1,20) etc) to find the best fit for our model.

3.3.1 ARIMA Model and it's Components

ARIMA (Auto regressive Integrated Moving Average) is a widely used time series forecasting model which are specifically designed to address both auto correlation and non stationarity in the data.The ARIMA model structure is defined by three key components: p, d, and q, representing the orders of the auto regression (AR), differencing (I) , and moving average (MA) components, respectively [46]. The three fundamental components of the model are:

- Auto regressive terms (AR) model process correlation between the current and and previous information .
- Integrated terms (I) model the differences required to make time series stationary.
- Moving Average (MA) model the relationship between the current observation and a residual error from a previous observation.

Auto regressive Models: AR(p) Formulation

An auto regressive (AR) is a type of time series model in which the current value depends on its own past observations. The general form of an auto regressive model of order p (AR(p)) is expressed as:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t \tag{3.3}$$

Y_t : Prediction variable (stock price),

ε_t : White noise with mean 0 and variance σ_w^2,

$\phi_1, \phi_2, \ldots, \phi_p$: Model parameters.

The concept of White noise ε_t is fundamental to time series analysis and stochastic processes. Each random variables are drawn from the same probability distribution

having constant variance. The key characteristics in a white noise sequence are that it is independent to each other meaning their is no relationship between the values at different time points, the expected value in the sequence of random variable is zero and is constant throughout the time. Deviations from white noise behaviour indicates presence of patterns or trends which are of interest to analyst [7].

3.3.2 Moving Average Component

The Moving Average (MA) component is a statistical calculation to determine the trends and patterns over a specific time frame [46]. In mathematical terms it calculates the simple average during a specific time frame and divides it by the total number of time frame. This component becomes particularly useful in time series analysis to smoothing short term fluctuations and highlighting the long term trends making it easier to identify the underlying trends or patterns. The formula for moving average at time t and with a window size q is as follows:

$$X_t = \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} = \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \tag{3.4}$$

θ_j : MA coefficients at time t,

ε_{t-j} : Data points within that specified window,

q : Size of the interval.

Moving averages comes in different forms such as simple moving average, weighted moving average and exponential moving average. Each of these forms have their unique way to assign weights to data points. Moving averages always comes in combination with other time series models which in turn increases the accuracy of forecasting.

3.3.3 Integrated Component

The integrated component, denoted by d, represents the measure of differences at different times,in order to make the time series stationary [46]. To achieve the stationarity in the series, the statistical properties such as mean and variance should be constant throughout. This process can be expressed as:

$$X_t^* = X_t - X_{t-1} - \ldots - X_{t-d} \tag{3.5}$$

X_t : The value of the time series variable at time $t,$ representing the current observation.

X_{t-1} : The value of the time series variable at time $t-1,$ the previous time step.

$t-d$: t is the current time point and d is the order of differencing.

In most of the instances, when $d = 1$ then the model achieves stationarity. When d = 0 then the model is typically represented as ARMA. The integrated component in a model plays an important role in addressing trends or seasonality within the time series data, thereby increasing the model's potentiality to make precise predictions.

3.3.4 Evaluation Parameters

To assess the performance of the ARIMA model, several evaluation parameters are considered:

- **R2 Score:** It is a statistical measure to find out the difference between the sum of squares of both dependent and independent variables in a prediction model. It also evaluates how well the testing data fits into that of the trained regression model [57].
- **MSE (Mean Square Error):** The mean square of differences between the trained model values and tested values. It increases the weight of larger values by removing the negative sign [57].
- **RMSE (Root Mean Square Error):** It is obtained by taking the square root of MSE. The root square is added to ensure consistency [34].
- **MAE (Mean Absolute Error):** The mean of the absolute difference between observed and predicted values. It measures how large is the difference between the actual and predicted values. Smaller the MAE value better is the model performance and vice versa [33].

- **MAPE (Mean Absolute Percentage Error):** Measures the percentage of MAE to the real values. Lower the MAPE value better is the model performance [33, 14, 34].

- **MdAPE (Median Absolute Percentage Error):** A modification of MAPE, using the median of the MAE values [57].

The evaluation parameters provide a comprehensive assessment of the model's performance.

Seasonal ARIMA (SARIMA) Model:

SARIMA is an extension to ARIMA for data which exhibits strong seasonal components. Whenever the time series data exhibits repeating seasonal patterns or trends at regular intervals then in that particular case it becomes very useful. The SARIMA model are designed in such a way that they can capture both seasonal and non seasonal components of time series with periodic patterns [34]. The SARIMA model is given by:

$$\phi_p(B)\Phi_P(B_s)(1-B_s)^D(1-B)^d Y_t = \phi_0 + \theta_q(B)\Theta_Q(B_s)\varepsilon_t \tag{3.6}$$

Where:

$\Phi_P(B_s)$ and $\Theta_Q(B_s)$ are the auto regressive and moving average seasonal polynomials,
D is the seasonal differencing operator,
B_s is the back shift operator applied to the seasonal component.

The resulting parameters in the SARIMA model can often involve analysis of auto correlation and partial auto correlation functions and the order selection criteria can be AIC or BIC (Bayesian Information Criterion). The notation for SARIMA is as follows:

$$X_t = SARIMA(p,d,q)(P,D,Q)_m \tag{3.7}$$

Here, P, D, and Q represent the number of seasonal auto-regressive, integrated, and moving average terms, respectively. The periodicity of the seasonal phenomenon (e.g., daily, weekly) is determined by m.

The computational complexity of SARIMA modeling process is intensive and it involves multiple steps. The mistake in any of the below modeling process steps can cause significant errors and as well as it can become time consuming.

- Estimating the trends and seasonality and isolating them.
- Performing white noise tests.
- Conducting unit root tests to assess stationarity.
- Calculating model parameters.
- Estimating model coefficients.

Box-Tiao Method: ARIMAX Model

The Box-Tiao method [33] is an incorporation of exogenous variables, known as ARIMAX. The key difference from the ARIMA model is the term $\Theta(B)X_t$, where $\Theta(B)$ is the polynomial operator of the exogenous variable X_t. The ARIMAX model is given by:

$$\phi_p(B)(1-B)^d Y_t = \phi_0 + \Theta(B)X_t + \theta_q(B)\varepsilon_t \tag{3.8}$$

$\phi_p(B)$: Auto regressive (AR) polynomial of order p.

$(1-B)^d$: Differencing operator of order d to achieve stationarity.

Y_t : Actual value at time t.

ϕ_0 : Constant term.

$\Theta(B)$: Operator for exogenous variable(s) X_t.

X_t : Exogenous variable(s).

$\theta_q(B)$: Moving Average (MA) polynomial of order q.

ε_t : Error term at time t.

In the context of Box-Tiao method the external factors may influence the time series data. The ARIMAX model involves an iterative process such as model identification, parameter estimation, and diagnostic checking. This models becomes useful in time series analysis when it deals with both auto correlation and the influence of external factors.

3.3.5 Limitations of the Model

The ARIMA (Auto Regressive Integrated Moving Average) model is a widely used powerful tool for time series forecasting and analysis. The limitations of the ARIMA model are as follows:

- The model does not perform well when the data set is complex and non-linear in nature.
- The statistical properties of the model do not change with time since the model assume stationarity, thereby resulting in information loss.
- The model cannot predict accurately if the data set contains outliers.
- The trends in the model cannot be captured due to fixed window size.
- The model struggles with complex seasonal variations in the data.
- The model is not suitable for making long term predictions.
- High computational complexity in the modeling process.

3.3.6 Flowchart

The various steps involved in the flowchart of a working ARIMA model are as follows:

- **Time Series Selection:** Time series of a data is denoted by $Y = Y1 + Y2 + \ldots + Yn$. To evaluate the critical property such as stationarity and flow diverges in time series analysis we use a technique which is known as differencing
- **Stationarity Check:** In this step we can proceed to estimate the model's criteria like AIC, BIC or R2 score in the case of time series data being stationary, we do not apply differencing. If the time series data is not stationary then we determine how many past observations need to be considered to make it stable through differencing.
- **Model Estimation and Validation:** Here we estimate the order of ARIMA model parameters like auto regression (p), differencing (d), and moving average (q) based on the model's criteria like AIC, BIC or R2 score. If the residuals follow a normal

distribution pattern then the model may not need validation, else the model undergoes continuous iterative process until a suitable model is obtained and then proceeded to evaluation [33].

- **Evaluation:** If the models evaluation metrics and criteria is suitable for future forecasting then that model is considered as the best fit model, else the process of selection and analysis is repeated to refine the model [33, 40].

- **Model Forecasting:** If the model's evaluation metrics and criteria are best fitted for forecasting, then it should be proceeded further for generating future time points. Else loop back for further selection, analysis and refinement [10].

In conclusion the flowchart illustrates step by step process for time series forecasting with the automated ARIMA tool. The significant steps in the process were finding out time series stationarity, model selection and analysis along with the validation for best fit. The robustness of this model can be captured using residual analysis and iterative validation steps. Overall it is a systematic and repetitive process to build a suitable and reliable forecasting model.

Furthermore, the graphical analysis in fig 3.16 and 3.17 captures the patterns of time series data. It evaluates how good the model's performance is with respect to capturing both data sets patterns. The difference, lags, significance of the coefficients, goodness-of-fit measures (AIC, BIC), RMSE and diagnostic tests are crucial for evaluating the model's appropriateness. The data captured during this analysis provides an estimate into the average deviation of the model's predictions from the actual values.

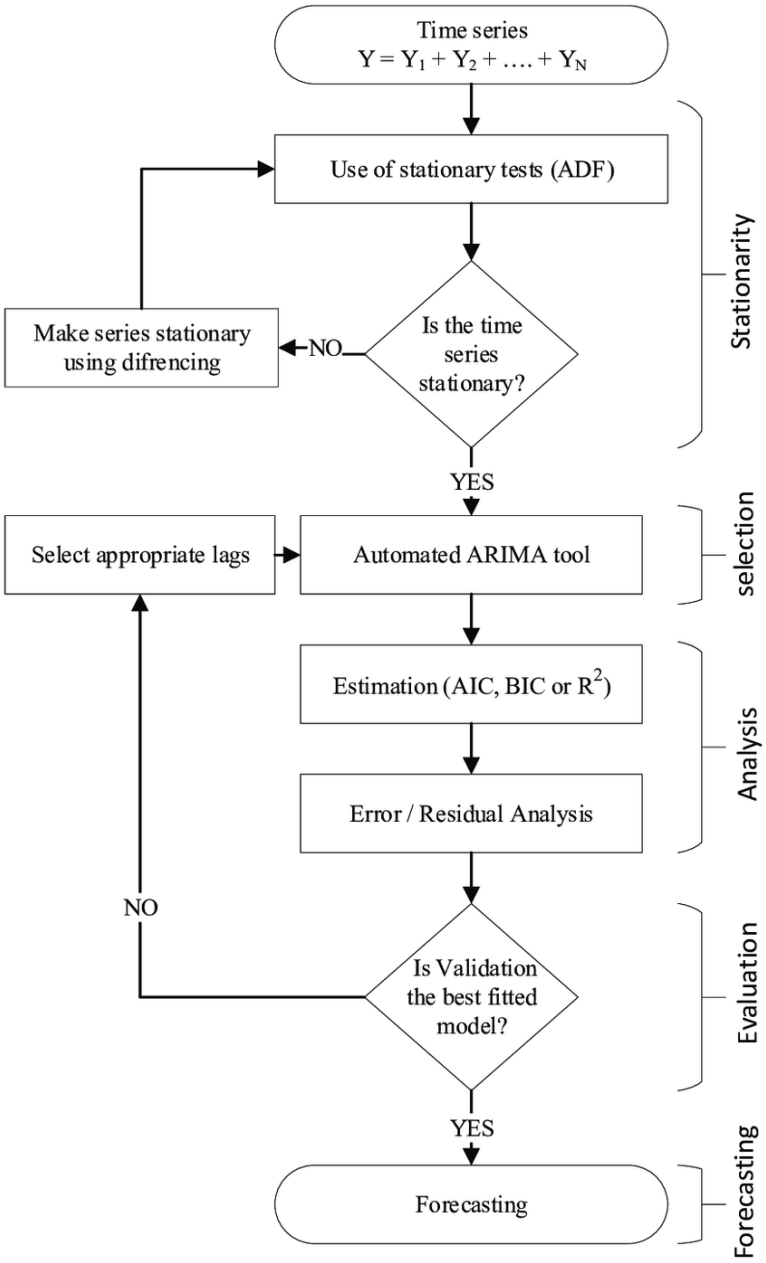

Figure 3.15: Flowchart for ARIMA working [47]

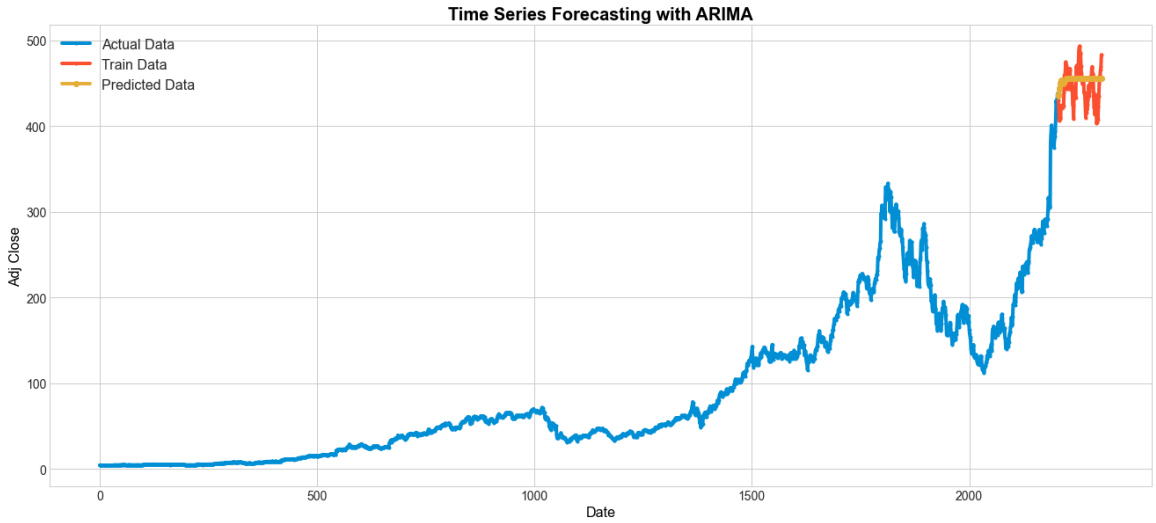

Figure 3.16: Time Series Forecasting NVDA Data

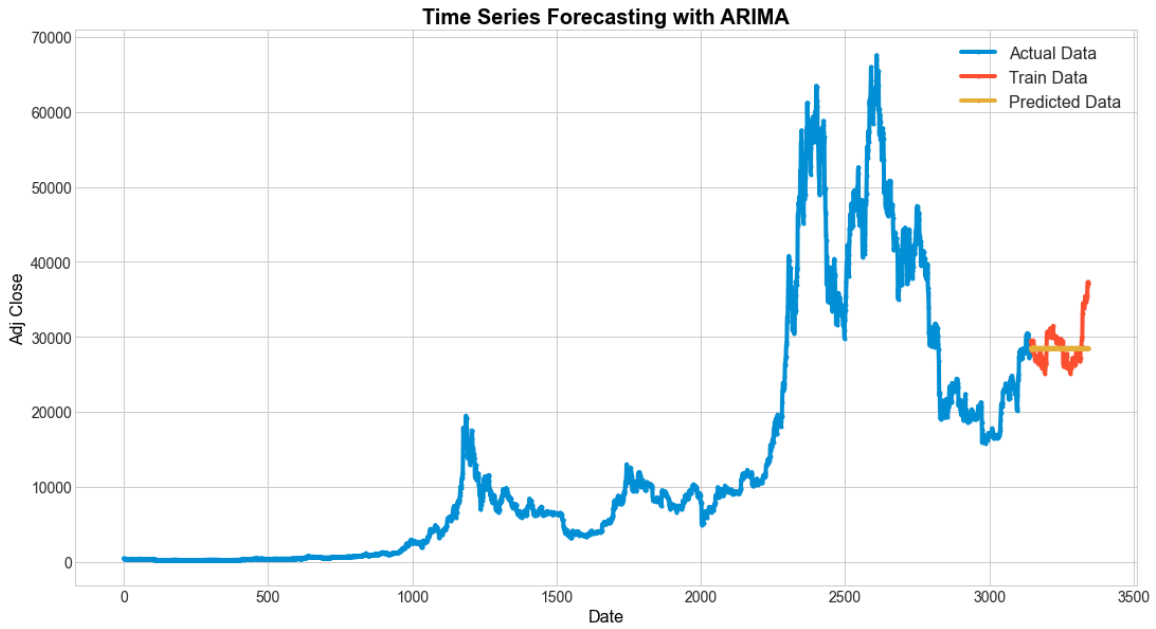

Figure 3.17: Time Series Forecasting BTC-USD Data

3.4 LSTM Model: Long Short-Term Memory

3.4.1 Introduction

LSTM (Long Short-Term Memory) [30] is a specialized type of recurrent neural network (RNN) designed to address the challenges of learning and remembering long term dependencies in sequential data, problem of vanishing gradients in traditional RNNs. It was introduced by Sepp Hochreiter and Jurgen Schmidhuber in 1997 [8] and has become a widely used architecture in the field of deep learning. To understand LSTM [57], we will first delve into the concepts of artificial neural networks (ANN), convolutional neural networks (CNN) and recurrent neural networks (RNN). These are all types of neural networks used in machine learning and deep learning [25, 41]. Each of them have their unique strengths and are suitable for specific type of task only.

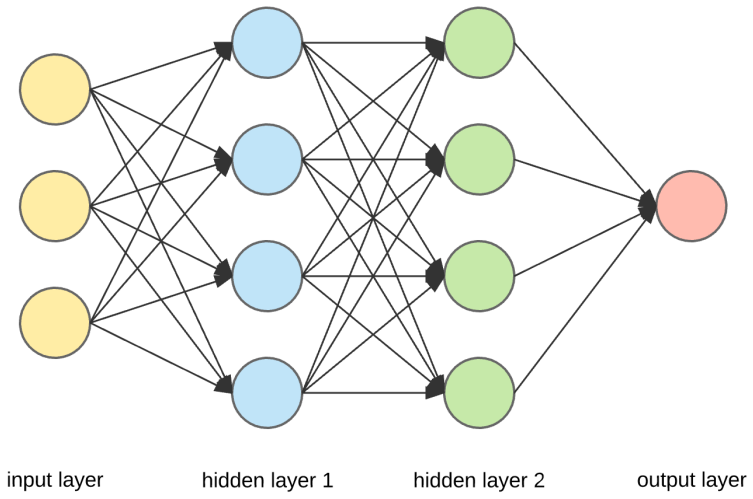

Figure 3.18: Deep neural network architecture [58]

Artificial neural networks (ANN)

Artificial neural networks (ANN) [20, 14]are computational models which are inspired by the structure and functioning of the human brain. Just like how human brain functions each neuron in ANN takes one or more input, performs mathematical operations and then produces an output. They are also a subset of machine learning algorithms which are designed to recognize patterns, make predictions and learn from data. It comprises of three main types of layers an input layer, hidden layers and an output

layer. Input layer (O), output layer (L), and hidden layers are given by ($l = 1, ..., L-1$) [26]. The nodes(neurons) which are present in the input layer represents features, these nodes are connected through synapses which carry weights in hidden layers. Synapses act as decision makers. The strength of connection between neurons are associated with weights. Each node which is present in the hidden layers act as an activation function (e.g, sigmoid, $\tanh$) to transform weighted inputs. Sigmoid activation (σ) function controls the flow of information passing through the gates, allowing the squashing of values to take place between 0 and 1. Hyperbolic tangent ($\tanh$) helps regulate the overall information that is stored in the network and then balance out the values in the memory. Output of the i-th neuron in layer 1: $Z_i^{(1)} = g(a_i^{(1)})$, where $a_i^{(1)}$ is the activation level. Net input $u_i^{(1)}$ of neuron i in layer 1: $u_i^{(1)} = \sum_j w_{ij}^{(1)} \cdot g'(a_j^{(1-1)})$, where $w_{ij}^{(1)}$ is the weight, g' is the derivative of the activation function, and $a_j^{(1-1)}$ is the output of neuron j in the previous layer. Activation function is given by $g(y) = -\frac{1}{1+e^{-y}}$ (sigmoid function).These non linear activation functions enable ANNs to learn about complex relationships in the data [11]. The weights are adjusted during the training process so that the difference between the expected and predicted value is minimum and this is done by using functions like soft max. Weight change in layer 1 at time v is given by the equation:

$$\Delta w_{ij}^{(1)}(v) = \eta \cdot \delta_i^{(1)} \cdot g(a_j^{(1-1)}) + \alpha \cdot \Delta w_{ij}^{(1)}(v-1) \tag{3.9}$$

where

η: Learning rate, a hyper parameter controlling the step size during weight updates.

α: Momentum, a hyper parameter that adds a fraction of the previous weight update to the current update

$\delta_i^{(1)}$: Error term associated with neuron i in layer 1.

$g(a_j^{(1-1)})$: Output of neuron j in the preceding layer, transformed by the activation function.

If $Z_j^{(1-1)} < 0$, then decrease $w_{ij}^{(1)}$ and vice versa [12]. The final layer produces the networks output which could be classification, regression or any other outcome. Feed forward is a process of passing input data to generate output data and back propagation method is applied from the outer layer to the hidden layer for weight adjustments.

The ANNs are very versatile in nature and this method can be applied to wide range of problem solving tasks which contains classification, regression and pattern recognition. The main strength of ANNs [21] are that they are good for general purpose task and forms the basic foundation to specialized neural network architectures. The main goal of ANNs is to minimize the loss during the training process [14]. Overfitting usually happens with respect to new data and is a challenge which we are addressing during training. Hyperparameter tuning takes place before training process begins so as to adjust according to the model. ANNs are a powerful tool having complex model in the field of machine learning. These models have wide range of applications including image and speech recognition, natural language processing, and time series forecasting etc. Going further we have different types of neural networks such as feed forward neural networks (FNN), recurrent neural networks (RNN) and convolutional neural networks (CNN).

Convolutional neural networks (CNN)

Here when we try to do time series forecasting on the stock market data we understand that primary factors influencing the stock prices are demand and supply. Stock markets are known for its volatility and unstable nature. Yan LeCunn's work on LeNet-5 (a 7-Level CNN) in 1998 has been influential. Researchers have used CNNs to model short and long term influences to predict stock price changes based on time series plots. Convolutional neural networks (CNN) [41] are designed in such a way that they process structured grid data, which becomes highly effective for image and video analysis. The other task CNNs can do is that they use convolutional layers present in them to learn hierarchical representations of the data and make them useful for important task where spatial relationships are important like object detection's and image classifications. The strength of this network is that they are excellent at capturing local patterns and hierarchical features in the data due to their arrangement in having overlapping regions [25].

The main steps for CNN modelling are data transformation, networking setup and model exploration. In the initial process the data is prepared in a format accepted by the CNN network. For a one-dimensional input $x = (x_t)_{t=0}^{N-1}$ and no zero padding, the output feature map from the first layer is represented here mathematically. Convolution

output width without zero padding is represented by $N_l = N_{l-1} - k + 1$ where the filter size parameter k controls the receptive field for each output node [23]. This network is setup using tensorflow's CNN framework based on softmax regression. It performs various permutations and combinations in the model to fine tune the process and increase its accuracy. One of the research papers introduced conditional time series forecasting using adaptation of deep convolutional wave net architecture. While applying multiple convolutional filters they used ReLU activation function. This method was tested on financial time series data and was compared to auto regressive and long-short term memory (LSTM) networks. LSTM networks worked the best for time series forecasting.

CNNs [25] are exceptional with respect to classification problems, but here with its limited research on financial time series forecasting other networks are outperforming in comparison to CNN. The CNNs does have advantage over recurrent networks due to their convolutional structure. In this research they presented structures which are inspired by wave net model. This has made them successful in forecasting financial time series of limited length. They even out perform linear and recurrent model. They even achieved better accuracy on non linear and noisy forecasting task [45].

Recurrent neural networks (RNN)

Recurrent neural networks (RNN) [51, 41] objectives are to predict the next step in the sequence of observations based on the previous steps. The sequential learning which takes place from the previous stages and observations, helps us in forecasting future trends. The hidden layers present in this network act as an internal storage for capturing information from earlier stages in sequential data. They become helpful in predicting the future data based on the past information. Even though they perform same task for each element in sequence, yet the RNNs struggle while remembering long sequence of data. A generic RNN [1, 34] can be written as

$$(H_t = f(H_{t-1}, X_t)) \tag{3.10}$$

and

$$(\hat{Y}_{t+h|t} = g(H_t)) \tag{3.11}$$

Here H_t is the hidden state.

Feed-forward neural networks (FNN)

Feed forward neural network [19] also known as multi layer perceptrons (MLPs) are a fundamental type of neural network architecture which consists of different layers each layer having hidden nodes in them. The network takes an input $x(1), ..., x(t)$ and produces an output $\hat{x}(t+1)$ for the next time step in series. The key components of this network are:

- Input Layer: $x(1), ..., x(t)$ are input variables at time steps 1 to t.
- First Layer: Linear combinations are calculated for M_1 nodes ($a_1(i)$). $a_1(i) = \sum_{j=1}^{t} w_1(i,j)x(j) + b_1(i)$, where w_1 is the weight matrix and b_1 is the bias vector. The outputs $a_1(i)$ are then transformed using a non linear activation function $h(\cdot)$ to get $f_1(i)$. $f_1(i) = h(a_1(i))$
- Subsequent Layers: For layers $l = 2, ..., L-1$, similar linear combinations and non linear transformations occurs: $f_l(i) = h\left(\sum_{j=0}^{M_{l-1}} w_l(i,j) f_{l-1}(j) + b_l(j)\right)$ here w_l is the weighted matrix, b_l is the bias vector.
- Output Layer: In the final layer L, the forecasted value $\hat{x}(t+1)$ is computed as $\hat{x}(t+1) = h\left(\sum_{j=0}^{M_{L-1}} w_L(j) f_{L-1}(j) + b_L\right)$

The simple nature of this networks architecture is that it forms no loops or cycles and uses activation functions to introduce non linearity into the model. They can be versatile for many task which contains static images, tabular data etc but cannot be efficient while processing grid like data as CNNs. Some of the limitations with respect to feed forward neural network are that they have limited memory, fixed input size, requires large amounts of data, prone to over fitting and are computationally intensive to perform large scale models. The acknowledgement of deep learning models [52] outperforming in every aspect of research has sparked in academic interest and specially for time series prediction these methods mentioned above have become the preferred choice since they are outperforming every other approaches.

The structure of LSTMs [57] are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. These networks contain memory

cells, allowing them to store, manipulate and retrieve information over long periods of time. The flow of information to each cell is controlled using gates(forget gate, memory gate, output gate) having sigmoidal layers [55]. Forget gate here determines what information needs to be removed from the cell's memory. Memory gate represents new data that is to be stored in the cell. The output gate gives output based on cell state and added data. The use case of these models are common for natural language processing tasks such as speech recognition, machine translation and language modeling where understanding the context of this sequence is crucial. These models address the vanishing gradient problem which occurred in traditional RNNs. They are effective for tasks involving sequential data and are capable of capturing long term dependencies over extended time intervals [34].

3.4.2 LSTM Architecture

The LSTM architecture contains a memory cell which regulates information flow into the model through gates [31, 18]. These models try to capture and remember information over long sequences. They are a variant of RNN designed to address the vanishing gradient problem. This model is composed of several layers. The core components of an LSTM model includes the forget gate, input gate, output gate, and cell state update. The details of each component are explained below:

Forget Gate (F_t)

The forget gate [30] is responsible for determining what information from the previous cell state (c_{t-1}) should be retained and what should be discarded from the LSTM memory. This gate uses sigmoid function to make decision based on previous hidden state and current input state. It is computed using the following equation:

$$F_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.12}$$

Where:

F_t: Forget gate value at time t.

σ: Sigmoid activation function.

W_f: Weight matrix.

U_f: Weight matrix.

x_t: Input at time t.

h_{t-1}: Previous hidden state.

b_f: Bias term.

The forget gate produces values between 0 and 1, indicating the degree to which each element in the cell state should be completely forgotten or remembered. The updated cell state after preserving information from the previous cell state and current input computes the hidden state for the current time step.

Input Gate (I_t)

The input gate [30] in an LSTM network operates on the same signal as that of the forget gate. The objective of this input gate is to decide which new information should be added to the LSTM memory and which shouldn't be added. The output of this gate is a fraction between 0 and 1. It consists of the following equation:

$$I_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.13}$$

Where:

I_t : Input gate value at time t.

σ : Sigmoid activation function.

W_i : Weight matrix for input gate.

U_i : Weight matrix for input gate.

x_t : Input at time t.

h_{t-1} : Previous hidden state.

b_i : Bias term for input gate.

The input gate, through the sigmoid function, determines which parts of the cell state should be updated. The tanh function creates a vector of new values which must be added to the previous state. This mechanism in the LSTM allows new information to be updated through an input gate while forgetting the irrelevant data through forget gate.

Output Gate (O_t)

The output gate [30, 46] in an LSTM unit controls what information from the cell state should be used to determine the values in the hidden state. The output gate is a matrix of 0s and 1s and it helps us in deciding which output we are using. The equation is given as follows:

$$O_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.14}$$

Where:

O_t : Output gate value at time t

σ : Sigmoid activation function

W_o : Weight matrix for the output gate

U_o : Weight matrix for the input and previous hidden state

x_t : Input at time t

h_{t-1} : Previous hidden state

b_o : Bias term

The output gate, by using sigmoid layer, determines how much of the cell state information should be used to compute the hidden state. It is the decision making layer of the cell. A tanh function is a non-linear function which maps values between -1 and 1.

Cell State Update (c_t)

The cell state in the LSTM memory is updated by combining information from the forget gate and the input gate [45]. This updated information is given by the equation:

$$c_t = F_t \odot c_{t-1} + I_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.15}$$

Where:

c_t: Cell state at time t.

F_t: Forget gate value.

I_t: Input gate value.

$\odot$: Element-wise multiplication.

$\tanh$: Hyperbolic tangent activation function.

W_c: Weight matrix.

U_c: Weight matrix.

x_t: Input at time t.

h_{t-1}: Previous hidden state.

b_c: Bias term.

This equation allows LSTM [58] to regulate the flow of information that should be retained (determined by the forget gate) and the new information (determined by the input gate) to update the cell state.

Hidden State (h_t)

The hidden state in LSTM [45] is a crucial component which is produced by applying the output gate values to the cell state after passing it through the hyperbolic tangent function. This state encodes the information we provide and controls the input, forget, output gates in LSTM so that they keep time dependencies in check. The equation of the hidden state is given by:

$$h_t = O_t \odot \tanh(c_t) \tag{3.16}$$

Where:

h_t: Hidden state at time t.

O_t: Output gate value.

$\tanh$: Hyperbolic tangent activation function.

c_t: Updated cell state.

The hidden state is the output of the LSTM at each time step. The output depends not only on the most recent input data but also the data it has seen in the past.

3.4.3 Applications of LSTM

Long Short-Term Memory (LSTM) networks are predominantly used in wide range of task such as sequence prediction problems, natural language processing, speech recognition, time series analysis, and more [56, 59]. They have the ability to learn, process and classify sequential data because of long-range dependencies between time steps of data. Some of the applications of LSTMs are:

- LSTMs are used in natural language processing task such as language translation and text generation like chatbots, content creation etc.
- These networks are effective in speech recognition tasks like google assistant, siri and others.
- LSTMs capture long term term dependencies in the data and therefore becomes suitable for time series prediction task such as stock price prediction, financial market trends, weather prediction, energy consumption and others.
- LSTMs become more suitable for computer vision applications since they can recognize and predict complex sequences of data.
- LSTMs are employed for healthcare related task like disease prediction, patient health monitoring over time.
- LSTMs have wide applications in video analysis, music composition, fraud detection, robotics and other domains where understanding and predicting sequential patterns of data become essential.

3.4.4 Limitations of LSTM

The Long Short-Term Memory (LSTM) networks are powerful in handling complex and challenging data sets but these networks also have certain limitations in them [60]:

- LSTMs are computationally resource intensive making them difficult for resource constrained tasks.

- They require more time to process large data sets, due to models architectural complexity.
- Like many deep learning models LSTMs are prone to overfitting while dealing with small data sets.
- LSTMs requires training of large data sets so that they can learn complex patterns and relationships effectively.
- LSTM models are so complex that it becomes very difficult to interpret and take decisions on the internal working model decisions.
- They have various hyper parameters, and determining the best parameters requires several experiments.
- Due to sequential computations over time steps, training LSTM models gets slower.

3.4.5 Flowchart

The various steps involved in the flowchart of a working LSTM model [57] are as follows:

- **Start Node:** The first step in the process begins with start node and this node indicates data prediction workflow.
- **Processing training data set:** Data preprocessing is an important task in the preparation of training data sets for the LSTM models. This process includes many tasks such as normalization, handling missing values, and feature engineering so that the data becomes suitable for training the model.
- **Train data set:** In this step preprocessed data are used to train the LSTM model. They even capture the temporal dependencies within the input sequence of this phase.
- **Train, test, selection of column with respect to data:** In this step we select specific column with respect to date in the time series data so that training and testing can happen with respect to data.

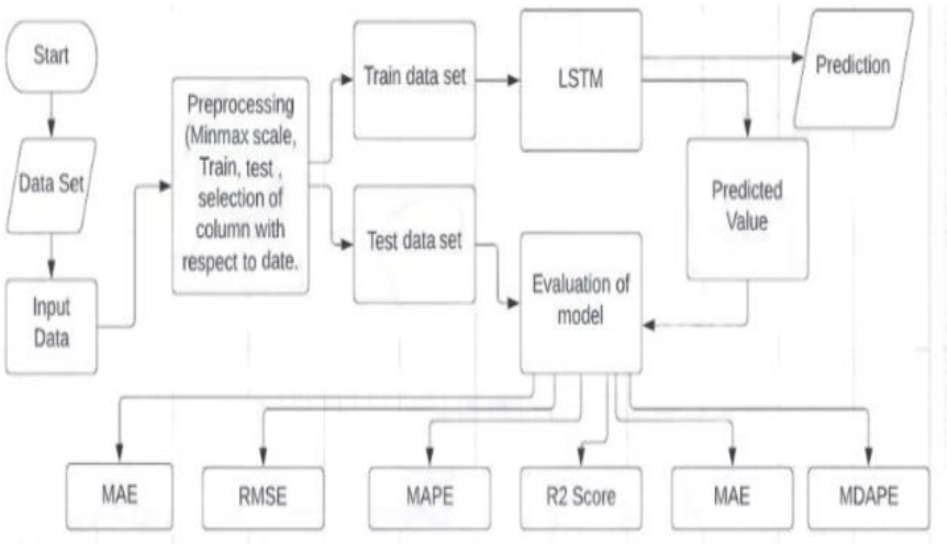

Figure 3.19: Flowchart for LSTM working [57]

- **Test data set:** It is that separate data, distinct from that of training data, used to test the LSTM model.
- **Evaluation of model:** Model evaluation is a critical step in the process of LSTM model. Here we evaluate the model based on various metrics such as root mean square error (RMSE), mean absolute error (MSE), r2 score, mean absolute percentage error (MAPE) and mean deviation (MDAPE). These metrics are obtained comprehensively in the evaluation of the model [36, 34].
- **Predicted value:** In the final step of this working model is the prediction. Here the step after model evaluation is to use the trained LSTM model representing the input to make predictions on fresh or unknown data as an output.

3.4.6 Conclusion

In conclusion, LSTM is a robust architecture able to handle large sequential data. Its unique design to capture long term dependencies in the stock price data, improve models performance by hyper parameters adjustments, make architecture choices and manage the flow of information from each of its memory cell makes the model powerful. LSTM has found applications in a wide range of domains and it continues to be a valuable asset in the field of machine learning and artificial intelligence.

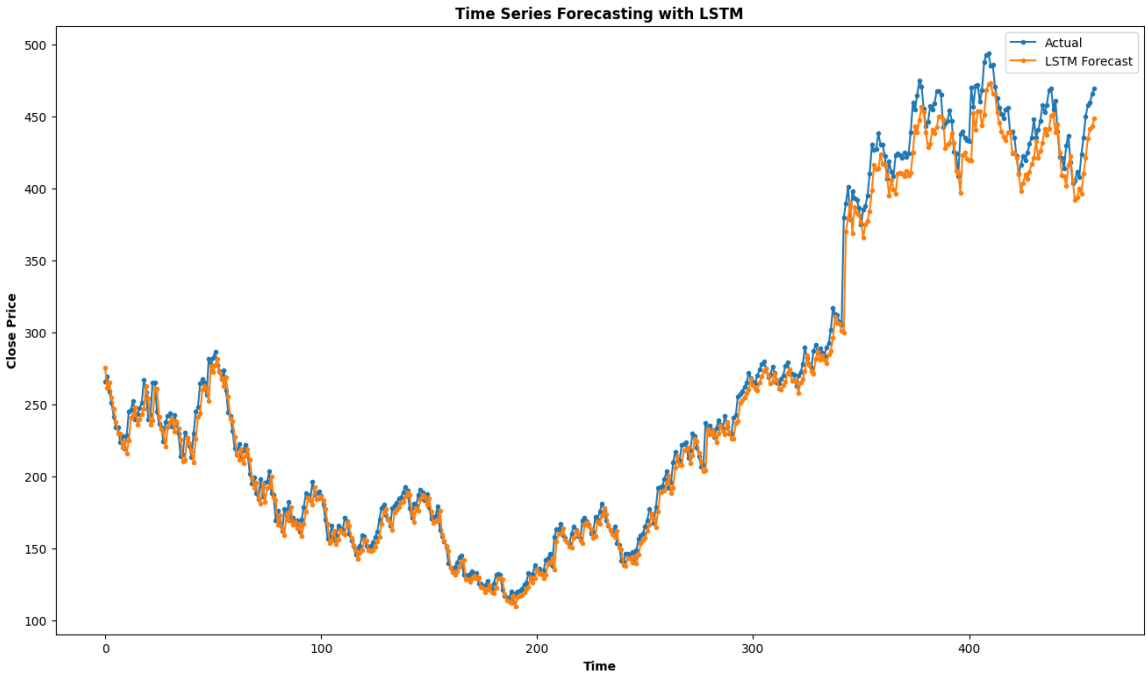

Figure 3.20: Time Series Forecasting with LSTM NVDA Data

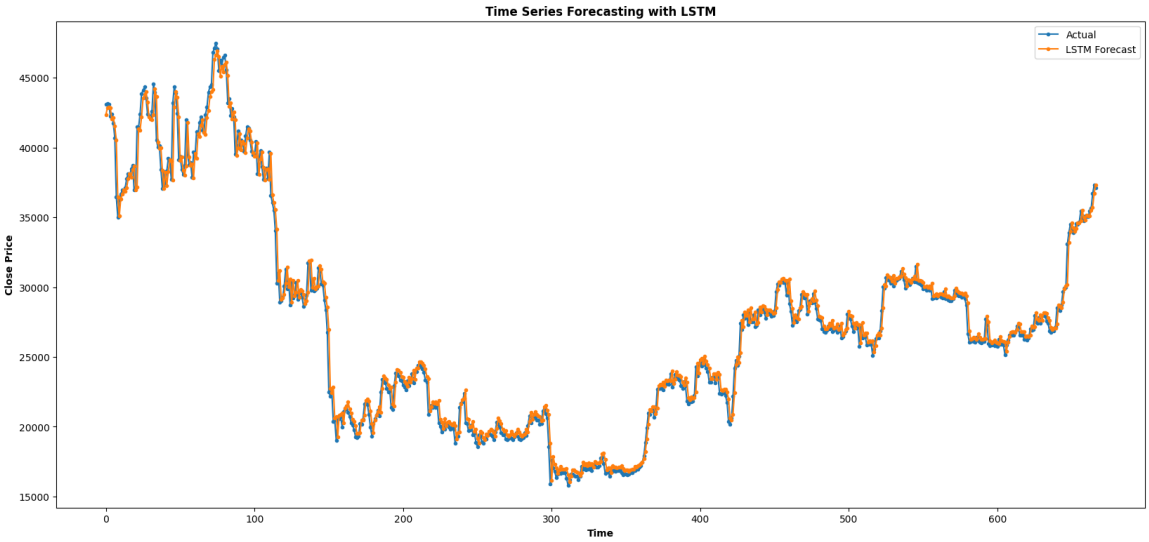

Figure 3.21: Time Series Forecasting with LSTM BTC-USD Data

CHAPTER 4

Result and Discussions

In this research we conducted cross data set time series forecasting on two diverse data sets: NVIDIA stock prices (NVDA) and Bitcoin to USD exchange rates (BTC-USD). The evaluation metrics were analyzed based on the performance of two prominent methodologies ARIMA (Auto regressive Integrated Moving Average) and LSTM (Long Short Term Memory). In order to capture the temporal patterns in both the data sets we divided them in ratio of 80% for training and 20% for testing. We then conducted dickey fuller test for both NVDA and BTC-USD data sets so as to assess their stationarity. From fig 4.1 and 4.2 it was evident that the p-values obtained for both data sets were higher than the common significance level of 0.05. This indicated that the time series data was not stationary. In order to make the data stationary differencing was applied. We ran multiple combinations of p,d and q values for the ARIMA model and found that for both data sets ARIMA$(2, 1, 20)$ is the best fit model having the least RMSE value. In the case of LSTM model we trained both the data sets to 100 epochs and updated the weights based on the error calculated during training process. The performance of both ARIMA and LSTM models were then calculated based on RMSE, R2, MAE, MAPE, and MDAPE evaluation metrics.

From table 4.1 and 4.2 it is evidenced that LSTM outperforms ARIMA when compared to both the data sets RMSE values. The lower RMSE values indicates lower prediction errors with respect to LSTM model. The coefficient of determination which is the r2 score represents goodness of fit. Here LSTM (0.98) scores slightly superior to that of ARIMA, hence both the models performs similar. With respect to NVDA data, LSTM

(8.25) exhibits lowers absolute errors as compared to ARIMA. Likewise with respect to BTC-USD data, LSTM (697.61) outperforms ARIMA in minimizing absolute errors. LSTM displays lower percentage of mean absolute error for both NVDA (3.11%) and BTC-USD (2.59%) data sets. This indicates the model has better accuracy as compared to ARIMA. The LSTM model even outperforms ARIMA with respect to median absolute percentage errors, which indicates its robust predictions.

```
Results of Dickey-Fuller Test:
Test Statistic                  -0.805797
p-value                          0.817413
#Lags Used                       8.000000
Number of Observations Used    2296.000000
Critical Value (1%)             -3.433201
Critical Value (5%)             -2.862800
Critical Value (10%)            -2.567441
dtype: float64
```

Figure 4.1: Dickey-Fuller Test NVDA

```
Results of Dickey-Fuller Test:
Test Statistic                  -0.874410
p-value                          0.796283
#Lags Used                       0.000000
Number of Observations Used    3343.000000
Critical Value (1%)             -3.432308
Critical Value (5%)             -2.862405
Critical Value (10%)            -2.567230
dtype: float64
```

Figure 4.2: Dickey-Fuller Test BTC-USD

In comparison to both ARIMA and LSTM models in time series forecasting, LSTM outperforms and demonstrates superior performance across various evaluation metrics. The dickey fuller test suggests an additional effort is required to make the time series data stationary. The lower value of RMSE, higher R2 score, and reduced absolute percentage errors highlight's LSTM ability to capture complex temporal patterns and long-term dependencies in time series data, especially in dynamic financial data sets like NVDA and BTC-USD. Therefore, based on the comprehensive time series analysis, LSTM emerges as the preferred model, for accurate and reliable model forecasting in these financial domains. The model's limitations is with respect to overfitting, which

	NVDA	BTC-USD
RMSE	536.71	11888075.31
R2 Score	-0.1	-0.0027
MAE	18.95	2827.62
MAPE	4.36	9.47
MDAPE	3.56	9.76

Table 4.1: Evaluation Metrics for ARIMA Forecasting

	NVDA	BTC-USD
RMSE	11.42	966.06
R2 Score	0.98	0.98
MAE	8.25	697.61
MAPE	3.11	2.59
MDAPE	3.11	2.59

Table 4.2: Evaluation Metrics for LSTM Forecasting

can be overcome by utilising a large enough sample size. Finally with respect to cross data set forecasting, the models were trained on NVDA adj closing price index and it was applied to BTC-USD adj closing price index. Based on the analysis it was found that mean square error was found to be 421755500.77 and 15488680.64. The results of the cross-data set time series forecasting experiments indicated greater prediction errors. ARIMA model performs the best when the underlying data adheres to linear trends or seasonality. Cryptocurrency markets are always volatile and the data exhibits non-linearity. It is always desirable to obtain lower MSE values. Lower MSE value indicates better model performance. The correlation between NVDA and BTC-USD close prices are not straight forward and both the models are not able to capture the additional factors influencing over one another. We had found that LSTM models are very robust and they can capture the complex patterns in the data while performing time series forecasting. But here with respect to cross data set time series forecasting it encompasses the complexities and the challenges a highly robust model can face with respect to predicting a value, solely based on financial instrument's prices.

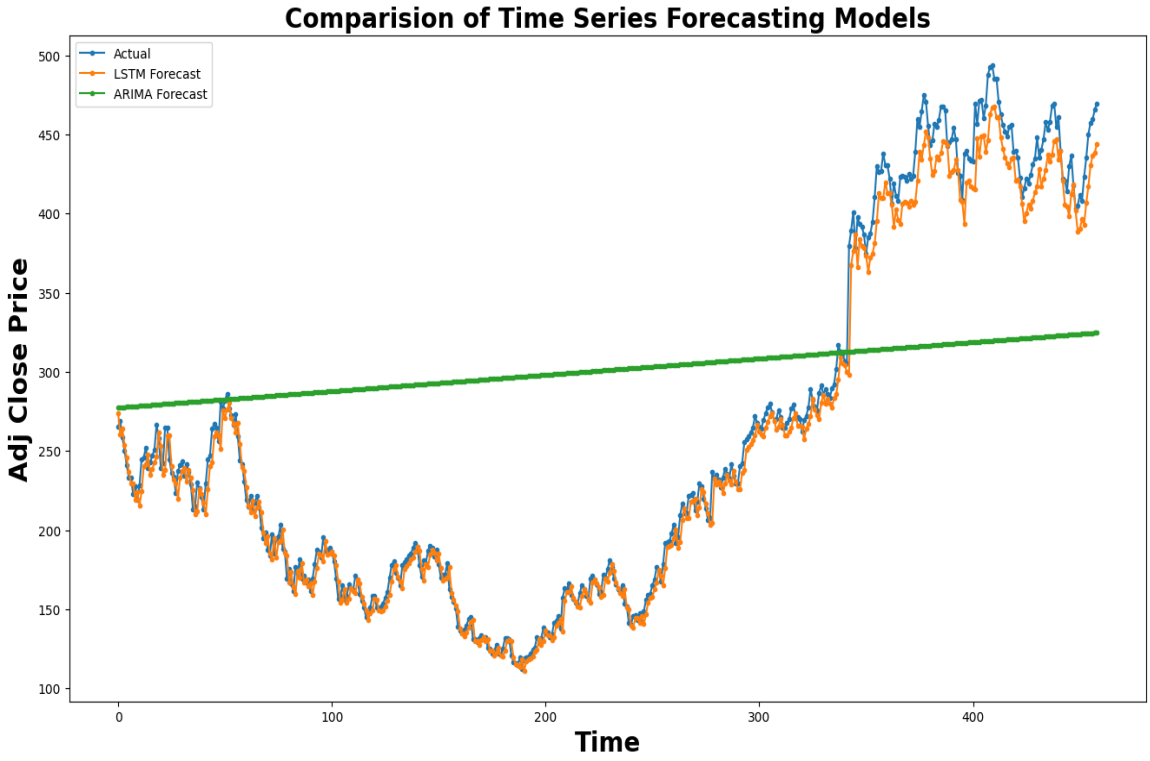

Figure 4.3: NVIDIA Data

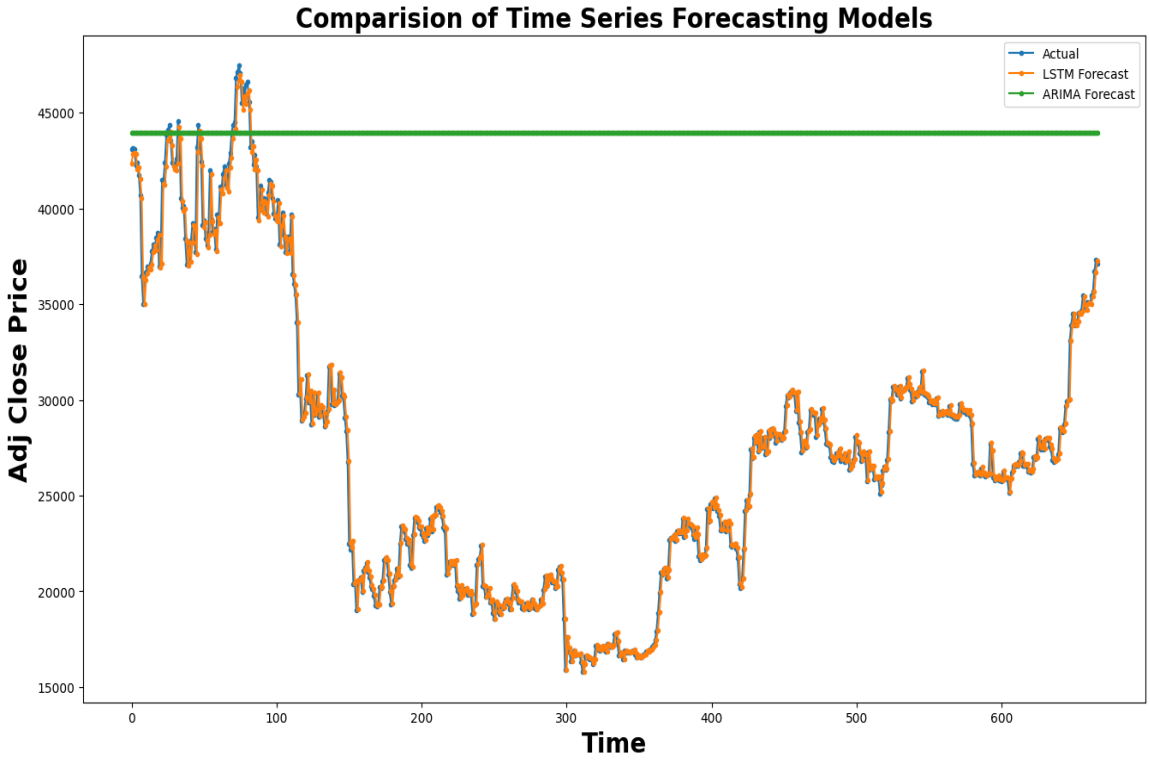

Figure 4.4: BTC Data

CHAPTER 5

Conclusions and Future Work

The purpose of this study is to predict one data set based on the information from another specifically between NVDA (NVIDIA) and BTC-USD (Bitcoin) data set. The results of these evaluation metrics in both models, ARIMA and LSTM have noteworthy findings. They shed light on the interdependence of these financial instruments. The evaluation metrics which highlighted MSE, R2 Score, MAE, MAPE, and MDAPE provided comprehensive view on model's performance. In the analysis of time series forecasting, LSTM models demonstrated superior ability and were able to capture complex patterns and trends in data. The successful prediction of one data set with respect to another has a huge practical implications in the financial sector with respect to making decisions, mitigating risks and maximizing opportunities.

The future work in this field of analysis is to explore multivariate time series modeling techniques. These techniques are explored in greater depth to understand the additional factors and relationships which lie between financial instrument prices. There can be additional research done to incorporate relevant features into the forecasting models so as to improve their prediction capacity. Dynamically updating the model with the latest information can built resilience into model and can also enhance their evaluation metrics. More research in the field of machine learning ensemble techniques can lead to robust forecasting framework. While ARIMA is robust when the data is linear, LSTM promises to capture complex patterns across data sets and with more research and innovation we can improve the prediction capabilities of these models.

Bibliography

[1] Ken-ichi Kamijo and Tetsuji Tanigawa. "Stock price pattern recognition-a recurrent neural network approach". In: *1990 IJCNN international joint conference on neural networks*. IEEE. 1990, pp. 215–221.

[2] Francis S. Wong. "Time series forecasting using backpropagation neural networks". In: *Neurocomputing* 2.4 (1991), pp. 147–159.

[3] Kanad Chakraborty et al. "Forecasting the behavior of multivariate time series using neural networks". In: *Neural networks* 5.6 (1992), pp. 961–970.

[4] Thomas Kolarik and Gottfried Rudorfer. "Time series forecasting using neural networks". In: *ACM Sigapl Apl Quote Quad* 25.1 (1994), pp. 86–94.

[5] James W Denton. "How good are neural networks for causal forecasting?" In: *The Journal of Business Forecasting* 14.2 (1995), p. 17.

[6] Gerson Lachtermacher and J David Fuller. "Back propagation in time-series forecasting". In: *Journal of forecasting* 14.4 (1995), pp. 381–393.

[7] Richard Webby and Marcus O'Connor. "Judgemental and statistical time series forecasting: a review of the literature". In: *International Journal of forecasting* 12.1 (1996), pp. 91–118.

[8] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[9] Michael Nelson et al. "Time series forecasting using neural networks: Should the data be deseasonalized first?" In: *Journal of forecasting* 18.5 (1999), pp. 359–367.

[10] Chris Chatfield. *Time-series forecasting*. CRC press, 2000.

[11] G Peter Zhang, B Eddy Patuwo, and Michael Y Hu. "A simulation study of artificial neural networks for nonlinear time-series forecasting". In: *Computers & Operations Research* 28.4 (2001), pp. 381–396.

[12] Héctor Allende, Claudio Moraga, and Rodrigo Salas. "Artificial neural networks in time series forecasting: A comparative analysis". In: *Kybernetika* 38.6 (2002), pp. 685–707.

[13] Kyoung-jae Kim. "Financial time series forecasting using support vector machines". In: *Neurocomputing* 55.1-2 (2003), pp. 307–319.

[14] G Peter Zhang. "Time series forecasting using a hybrid ARIMA and neural network model". In: *Neurocomputing* 50 (2003), pp. 159–175.

[15] Chris Chatfield. "Time-series forecasting". In: *Significance* 2.3 (2005), pp. 131–133.

[16] G Peter Zhang and Douglas M Kline. "Quarterly time-series forecasting with neural networks". In: *IEEE transactions on neural networks* 18.6 (2007), pp. 1800–1814.

[17] A. E. Clark and C. G. Troskie. "Time Series and Model Selection". In: *Communications in Statistics - Simulation and Computation* (2008). DOI: 10.1080/03610910701884153.

[18] Nesreen K Ahmed et al. "An empirical comparison of machine learning models for time series forecasting". In: *Econometric reviews* 29.5-6 (2010), pp. 594–621.

[19] *Feedforward neural network*. 2011. URL: https://en.wikipedia.org/wiki/Feedforward_neural_network.

[20] Mehdi Khashei and Mehdi Bijari. "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting". In: *Applied soft computing* 11.2 (2011), pp. 2664–2675.

[21] Weizhong Yan. "Toward automatic time-series forecasting using neural networks". In: *IEEE transactions on neural networks and learning systems* 23.7 (2012), pp. 1028–1039.

[22] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. "Machine learning strategies for time series forecasting". In: *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2* (2013), pp. 62–77.

[23] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. "Conditional time series forecasting with convolutional neural networks". In: *arXiv preprint arXiv:1703.04691* (2017).

[24] Arden Dertat. *Applied Deep Learning - Part 1: Artificial Neural Networks*. 2017. URL: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6 (visited on 08/08/2017).

[25] Alexiei Dingli and Karl Sant Fournier. "Financial time series forecasting-a deep learning approach". In: *International Journal of Machine Learning and Computing* 7.5 (2017), pp. 118–122.

[26] John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017).

[27] Jason Brownlee. *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[28] Hassan Ismail Fawaz et al. "Transfer learning for time series classification". In: *2018 IEEE international conference on big data (Big Data)*. IEEE. 2018, pp. 1367–1376.

[29] Nikolay Laptev, Jiafan Yu, and Ram Rajagopal. "Reconstruction and regression loss for time-series transfer learning". In: *Proceedings of the Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) and the 4th Workshop on the Mining and LEarning from Time Series (MiLeTS), London, UK*. Vol. 20. 2018.

[30] Sima Siami-Namini and Akbar Siami Namin. "Forecasting economics and financial time series: ARIMA vs. LSTM". In: *arXiv preprint arXiv:1803.06386* (2018).

[31] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "A comparison of ARIMA and LSTM in forecasting time series". In: *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2018, pp. 1394–1401.

[32] Rui Ye and Qun Dai. "A novel transfer learning framework for time series forecasting". In: *Knowledge-Based Systems* 156 (2018), pp. 74–99.

[33] Julio Barzola-Monteses et al. "Time series analysis for predicting hydroelectric power production: The ecuador case". In: *Sustainability* 11.23 (2019), p. 6539.

[34] Oussama Fathi. “Time series forecasting using a hybrid ARIMA and LSTM model”. In: *Velvet Consulting* (2019), pp. 1–7.

[35] Qi-Qiao He, Patrick Cheong-Iao Pang, and Yain-Whar Si. “Transfer learning for financial time series forecasting”. In: *PRICAI 2019: Trends in Artificial Intelligence: 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26–30, 2019, Proceedings, Part II 16*. Springer. 2019, pp. 24–36.

[36] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. “A comparative analysis of forecasting financial time series using arima, lstm, and bilstm”. In: *arXiv preprint arXiv:1911.09512* (2019).

[37] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. “The performance of LSTM and BiLSTM in forecasting time series”. In: *2019 IEEE International conference on big data (Big Data)*. IEEE. 2019, pp. 3285–3292.

[38] Peter T Yamak, Li Yujian, and Pius 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.

[39] Srihari Athiyarath, Mousumi Paul, and Srivatsa Krishnaswamy. “A comparative study and analysis of time series forecasting techniques”. In: *SN Computer Science* 1.3 (2020), p. 175.

[40] Vitor Cerqueira, Luis Torgo, and Igor Mozetič. “Evaluating time series forecasting models: An empirical study on performance estimation methods”. In: *Machine Learning* 109 (2020), pp. 1997–2028.

[41] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. “Financial time series forecasting with deep learning: A systematic literature review: 2005–2019”. In: *Applied soft computing* 90 (2020), p. 106181.

[42] Zhipeng Shen et al. “A novel time series forecasting model with deep learning”. In: *Neurocomputing* 396 (2020), pp. 302–313.

[43] Qitao Gu and Qun Dai. “A novel active multi-source transfer learning algorithm for time series forecasting”. In: *Applied Intelligence* 51 (2021), pp. 1326–1350.

[44] Qitao Gu et al. “Integrating multi-source transfer learning, active learning and metric learning paradigms for time series prediction”. In: *Applied Soft Computing* 109 (2021), p. 107583.

[45] Bryan Lim and Stefan Zohren. “Time-series forecasting with deep learning: a survey”. In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200209.

[46] Apoorva Mahadik, Devyani Vaghela, and Amrapali Mhaisgawali. “Stock price prediction using lstm and arima”. In: *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE. 2021, pp. 1594–1601.

[47] Muhammad Ali Musarat et al. *Kabul River Flow Prediction Using Automated ARIMA Forecasting: A Machine Learning Approach*. 2021. URL: https://www.researchgate.net/figure/Methodology-flowchart-321-Automated-ARIMA-Forecasting-The-automatic-model-selection_fig1_354860118 (visited on 09/27/2021).

[48] Alaa Sagheer, Hala Hamdoun, and Hassan Youness. “Deep LSTM-based transfer learning approach for coherent forecasts in hierarchical time series”. In: *Sensors* 21.13 (2021), p. 4379.

[49] José F Torres et al. “Deep learning for time series forecasting: a survey”. In: *Big Data* 9.1 (2021), pp. 3–21.

[50] Manuel Weber et al. “Transfer learning with time series data: a systematic mapping study”. In: *Ieee Access* 9 (2021), pp. 165409–165432.

[51] Rui Ye and Qun Dai. “Implementing transfer learning across different datasets for time series forecasting”. In: *Pattern Recognition* 109 (2021), p. 107617.

[52] Konstantinos Benidis et al. “Deep learning for time series forecasting: Tutorial and literature survey”. In: *ACM Computing Surveys* 55.6 (2022), pp. 1–36.

[53] Shanoli Samui Pal and Samarjit Kar. “Fuzzy transfer learning in time series forecasting for stock market prices”. In: *Soft Computing* 26.14 (2022), pp. 6941–6952.

[54] Rui Ye and Qun Dai. “A relationship-aligned transfer learning algorithm for time series forecasting”. In: *Information Sciences* 593 (2022), pp. 17–34.

[55] Wei Emma Zhang et al. “Time Series Visualization and Forecasting from Australian Building and Construction Statistics”. In: *Applied Sciences* 12.5 (2022), p. 2420.

[56] Xiaofeng Zhou et al. “Time series prediction method of industrial process with limited data based on transfer learning”. In: *IEEE Transactions on Industrial Informatics* (2022).

[57] Khulood Albeladi, Bassam Zafar, and Ahmed Mueen. “Time Series Forecasting using LSTM and ARIMA”. In: *International Journal of Advanced Computer Science and Applications* 14.1 (2023).

[58] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. “Machine learning advances for time series forecasting”. In: *Journal of economic surveys* 37.1 (2023), pp. 76–111.

[59] *Applications of LSTM*. URL: https://www.knowledgehut.com/blog/web-development/long-short-term-memory.

[60] Prudhviraju Srivatsavaya. *LSTM â Implementation, Advantages and Diadvantages*. URL: https://medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb.