Bachelor of Science in

Computer Science

Project Report on Stock Market Forecasting using Time Series Data Analysis

A Report submitted as a partial fulfilment in the requirement for the degree of **Bachelor of Science in Computer Science (Hons.)** at

WEST BENGAL STATE UNIVERSITY

Submitted By

Chitranku Sarkar	**Subhodeep Saha**	**Avrodeep Pal**
Roll No: 6241107-07171	Roll No: 6241107-07174	Roll No: 6241107-07164
Reg. No: 1032111400447	Reg. No: 1032111600471	Reg. No: 1032111600442

Under The Supervision of

Sri. Phulen Mahato

Assistant Professor, Department of Computer Science

Barrackpore Rastraguru Surendranath College

July 2024

DECLARATION

Apart from the efforts of team, the completion of any inter-disciplinary project depends upon cooperation, co-ordination and combined efforts of several sources of knowledge. We are eternally grateful to our teacher and as well as project supervisor **Sri. Phulen Mahato** for his even willingness to give us valuable advice and direction under which we executed this project. His constant guidance and willingness to share his vast knowledge made us understand this project and helped us to complete the assigned tasks.

This report is submitted as a partial fulfillment of the requirements for the degree of **B.Sc in Computer Science (CBCS System)** for the academic year **2021-2024**, at **Barrackpore Rastraguru Surendranath College**, affiliated to **West Bengal State University, Barasat**.

We also declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct, and the results presented in this report or parts of it have not been presented for the award of any other degree.

Chitranku Sarkar

Roll No: 6241107-07171

Reg. No: 1032111400447 ______________________________

Subhodeep Saha

Roll No: 6241107-07174

Reg. No: 1032111600471 ______________________________

Avrodeep Pal

Roll No: 6241107-07164

Reg. No: 1032111600442 ______________________________

CERTIFICATE

This is to certify that the project titled: "**Stock Market Forecasting using Time Series Data Analysis**" is completed by **Chitranku Sarkar** (Roll No: 6241107-07171), **Subhodeep Saha** (Roll No: 6241107-07174), **Avrodeep Pal** (Roll No: 6241107-07164) during the academic year **2021-2024**. This work is a genuine and conducted as part of the requirements for the degree of **B.Sc. in Computer Science** at **Barrackpore Rastraguru Surendranath College**, affiliated with **West Bengal State University**. This project is submitted in fulfillment of the partial requirements for the aforementioned degree and reflects the student's dedication and knowledge in the field of Computer Science.

(*Sri. Phulen Mahato*)

Supervisor

Department of Computer Science

Barrackpore Rastraguru Surendranath College

(*Sri. Phulen Mahato*)

Head of the Department

Department of Computer Science

Barrackpore Rastraguru Surendranath College

External Examiner(s)

ACKNOWLEDGEMENT

Undertaking this dissertation work as an essential component of our academic curriculum has been an enriching and scientifically progressive experience. This research effort has not only widened our knowledge horizons but also propelled us towards a greater understanding of our chosen field. In the course of preparing this dissertation and shaping its form, we acknowledge with profound gratitude the invaluable contributions of several individuals and institutions. Our heartfelt appreciation goes to **Dr. Monojit Roy** (Principal, Barrackpore Rastraguru Surendranath College), **Sri. Phulen Mahato** (Head of Department, Computer Science), and **Dr. Anirban Chakraborty** (Department of Computer Science) for their valuable guidance, constant support, and continuous motivation throughout this effort.

Special thanks are extended to our dedicated supervisor, **Sri. Phulen Mahato**, whose expertise and encouragement have been instrumental in shaping the direction of our project.

We also express our sincere gratitude to all the faculty members, lab assistants, and administrative staff of the university for their generous assistance and timely help whenever needed.

In conclusion, we would like to acknowledge the contributions of all those who played a part, big or small, in making this dissertation work a reality.

Table of Contents

Chapter 1
Introduction

1.1 Motivation

In recent years, Artificial Intelligence (AI) has been a significant technology that is being applied in making driverless cars, intelligent robots, image and speech recognition, automatic translations, and medical assistants [4]. A stock is an investment in an institution where it represents ownership in a company. The stock market is a place where those stocks are purchased. Purchasing a stock of a company is owning a small share of an institution Stock market prediction means forecasting the current trends of a company and predicting the value of stocks, whether they're going up or down. Consequently, it has made researchers think to come up with reliable predictive models over the decade [2]. The urge to predict stock prices more accurately is making researchers work for the betterment of current predictive machine learning models. Because of the unpredictable nature of the stock market, no certain models of machine learning can precisely forecast the stock market, which is the inspiring factor for us to research and build a better predictive system. Various methods of machine learning have been applied for forecasting the stock market throughout recent years. Among them, models such as Support Vector Regression (SVR), Artificial Neural Networks (ANNs), Bayesian Neural Network (BNN), and so on have been exploited to improve time series forecasting [7]. **We are predicting stock prices using a machine learning algorithm to develop a model that forecasts stock prices effectively based on current market trends.**

We have used LSTM recurrent neural networks to predict stock prices accurately. LSTMs are very important as they are very powerful in sequence prediction problems because they can store previous or past information. This is crucial in stock prediction as we need to store and read previous stock information to forecast stock prices accurately in the future.

1.2 Background Study

Time series analysis is a statistical method that analyses and manipulates time series data. Time series is made of data points collected at constant time intervals. Time series analysis unlike regression analysis is very useful in order to get the important characteristics and statistics of a time series data. Time series analysis has the features to help us understand the underlying factors that lead to a specific trend in time series data points and thus help us predict data points. The first step towards time series analysis is to check if the time series data is stationary. The two main reasons behind non-stationarity are:

- **Trend:** The mean of a time series is variable over time. For example, the average number of car users is growing over time.
- **Seasonality:** Variations at a particular time-interval such as, people might buy cars in a particular month because of salary increment or festival.

To check if a time series is stationary, the following tests are performed:

- **Plotting Rolling Statistics:** Plots the moving average or variance and check if it varies with time. This is more like a visual representation.
- **Dickey-Fuller Test:** Unlike the first one this a statistical test to check stationarity. In this test, null hypothesis considers time series as non-stationary. Results are the test-statistic and some critical-values for different confidence levels. The series is said to be stationary if the null hypothesis gets rejected when the test-statistic becomes less than the critical-value. The main purpose is to reduce these features from the time series by estimating the trend and seasonality in the series.

The following techniques can be useful to model or estimate trend and seasonality:

- **Aggregation:** Considers averages for a time period like month/week.
- **Smoothing:** Considers rolling averages.
- **Polynomial Fitting:** Fits a regression model.

According to different problem solving, any of the techniques can be used. After the estimation, trend and seasonality can be reduced by using the following methods:

- **Differencing:** It is the most common way to reduce non-stationary features by taking the difference of observation between a particular instant with a previous instant. Trend and seasonality reduction can be improved by changing the order of differencing.
- **Decomposing:** It separately models the trend and seasonality of the time series and the rest of it is returned so that the residuals can be modelled.

After these steps, forecasting techniques can be applied on the non-stationary series. In final step, trend and seasonality constraints are applied back to convert the predicted values into the original scale.

1.3 Objective

Prediction is a procedure to make assumptions about the future in light of existing information. The more exact the prediction, the simpler it could be to settle on a decision for the future. As we discussed before, predicting the stock market accurately is a difficult task due to the dynamic nature of the stock market. If the stock market rises, then a country's financial development would be high and vice-versa. In recent years, we can use a huge amount of data and analyse it due to the development of computer technology. There are two approaches to stock market prediction. One approach focuses on historical data, while the other focuses on data aside from historical data. Regarding the first approach, there are two types of methods: fundamental and technical. The technical method uses historical data. We can use various types of technical methods to predict the stock market, such as Neural Networks, Evolutionary Algorithms, Support Vector Machines, Neuro-Fuzzy systems, Hidden Markov models, and decision trees.

In this paper, we use **Long Short-Term Memory (LSTM) networks to predict how the stock market will behave in the upcoming thirty days by using some well-known companies' last twenty (approximately) years of historical data.** Stock prices can be treated as a discrete time series model, which is an arrangement of well-defined numerical data items collected at successive points at regular intervals of time. Since it is fundamental to distinguish a model with the end goal of analysing trends of stock prices with adequate information for decision making, it is recommended that using LSTM networks is a better algorithmic approach than forecasting directly. LSTM models are particularly well-suited for this task as they can store previous or past information, which is crucial in stock prediction. This ability to retain information over long sequences makes LSTM networks highly effective for capturing long-term dependencies and patterns in financial time series data.

Chapter 2
Literature Survey

2.1 Machine Learning Models

Stock price prediction can be predicted using AI and machine learning models in machine learning fields. Using the SVM model for stock price prediction. SVM is one of the machine learning algorithms which works on classification algorithms. It is used to get a new text as an output. Applying Multiple Linear Regression with predict the trend in stock prices (Osman Hegazy et al. 2013 [2]; V Kranthi Sai Reddy, 2018 [8]; a Banerjee et al. 2020 [5]; Lufuno Ronald Marwala [8]). Random Walk Hypothesis which is proposed by Horne, j. C et al 1997 [3] which is used to predict stock prices, Horne j.c [4] said that the stock values are changes random and the past price values are not dependent on current values. EMH (Efficient Market Hypothesis) is different from the Random walk hypothesis but the EMH works mainly on Short term patterns for predicting stock prices.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) have been significantly utilized due to their ability to handle sequential data, which is essential for time series forecasting. RNNs maintain a form of memory through hidden states, making them suitable for capturing temporal dependencies in stock market data. However, traditional RNNs suffer from issues like vanishing and exploding gradients, which can hinder their performance on long sequences. Researchers have applied RNNs to predict stock prices with varying degrees of success, highlighting the need for models that can better manage long-term dependencies. To address these limitations, Long Short-Term Memory (LSTM) networks were developed as an extension of RNNs.

2.3 Long Short Term Memory

Long Short-Term Memory (LSTM) networks, an extension of RNNs, address these limitations by incorporating gates that control the flow of information, effectively managing long-term dependencies and mitigating gradient issues. LSTMs have shown significant promise in financial time series forecasting due to their ability to remember information over extended periods. Studies by Xiongwen Panget al. [2] and Hyeong Kya Choi [2] have demonstrated the superior performance of LSTM networks in predicting stock prices, emphasizing their capability to handle the complexities of financial data. LSTMs incorporate gates that control the flow of information, effectively managing long-term dependencies and mitigating gradient issues. Additionally, LSTMs have been effectively combined with other models, such as Convolutional Neural Networks (CNN) and ARIMA, to enhance predictive accuracy and robustness.

Chapter 3
Time Series Analysis

3.1 Definition

Time Series (TS) is a series of data points indexed in time order. Most commonly, a TS is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete time data. It is mathematically defined as a set of vectors x(T), T = 0,1,2,... where T represents the time elapsed we can denote the observations by $Y_1,Y_2,...Y_T$. The variable x(T) is treated as a random variable. A TS is uni-variate when it contains records of a single variable. If records of more than one variable are considered, then it is called multivariate. Again TS can be continuous or discrete. If observations are measured at every instances of time then it is called a continuous TS, whereas a discrete time series contains observations measured at discrete points of time. For example flow of river, concentration of a chemical process, temperature reading etc. can be recorded as a continuous TS. On the other hand population of a country, production of a company, exchange rates between two different currencies may present discrete TS. The consecutive calculations are usually recorded in a discrete TS at equally spaced time intervals such as hourly, daily, weekly, monthly or yearly time separations. The variable observed in a discrete time series is assumed to be measured by the real number scale as a continuous variable. By merging data together over a specified time interval we can easily transform continuous TS to a discrete one. There are a number of important interests in a TS such as Smoothing, Modeling, Forecasting, Control.

- **Smoothing:** The observed Yt are assumed to be the result of "noise" values t additively contaminating a smooth signal t
  $$Y_t = \eta_t + \varepsilon_t$$
  We may wish to recover the values of the underlying η_t
- **Modelling:** We may wish to develop a simple mathematical model which explains the observed pattern of $Y_1,Y_2,...Y_T$. This model may depend on unknown parameters and these will need to be estimated.

- **Forecasting:** On the basis of observations $Y_1,Y_2,...Y_T$, we may wish to predict what the value of Y_{T+L} will be $(L>1)$, and possibly to give an indication of what the uncertainty is in the prediction.
- **Control:** We may wish to intervene with the process which is producing the Y_t values in such a way that the future values are altered to produce a favorable outcome.

3.2 Components of a Time Series

A time series in general is supposed to be affected by four main components, which can be separated from the observed data. These components are: Trend, Cyclical, Seasonal and Irregular components. A brief description of these four components is given here. The general tendency of a time series to increase, decrease or stagnate over a long period of time is termed as Secular Trend or simply Trend. Thus, it can be said that trend is a long term movement in a time series. For example, series relating to population growth, number of houses in a city etc. show upward trend, whereas downward trend can be observed in series relating to mortality rates, epidemics, etc. Seasonal variations in a time series are fluctuations within a year during the season. The important factors causing seasonal variations are: climate and weather conditions, customs, traditional habits, etc. For example sales of ice-cream increase in summer, sales of woolen cloths increase in winter. Seasonal variation is an important factor for businessmen, shopkeeper and producers for making proper future plans. The cyclical variation in a time series describes the medium-term changes in the series, caused by circumstances, which repeat in cycles. The duration of a cycle extends over longer period of time, usually two or more years. Most of the economic and financial time series show some kind of cyclical variation. For example a business cycle consists of four phases such as i) Prosperity ii) Decline iii) Depression iv) Recovery. A typical business cycle is shown in Fig. 3.1

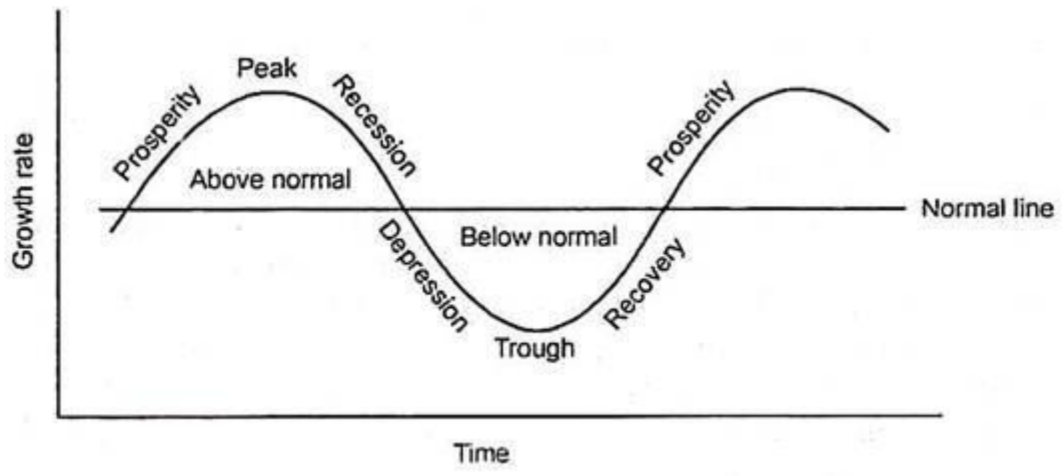

Fig. 3.1: Four phase Business Cycle

Irregular or random variations in a time series are caused by unpredictable influences which are not regular and also do not repeat in a particular pattern. These variations are caused by incidents such as war, strike, earthquake, flood, revolution, etc. There is no defined statistical techniques for measuring random fluctuations in a time series. Considering the effect of these four components, two different types of models are generally used for a time series known as Multiplicative and Additive models.

Multiplicative Model: $Y(t) = T(t) \times S(t) \times C(t) \times I(t)$.

Additive Model: $Y(t) = T(t) + S(t) + C(t) + I(t)$.

Here $Y(t)$ is the observation and $T(t)$, $S(t)$,$C(t)$ and $I(t)$ are respectively the trend, seasonal, cyclical and irregular variation at time t. Multiplicative model is based on the assumption that the four components of a time series are not necessarily independent and they can affect one another; whereas in the additive model it is assumed that the four components are independent of each other.

3.3 Time Series Visualization

Plots of the raw sample data can provide valuable diagnostics to identify temporal structures like trends, cycles, and seasonality that can influence the choice of model. There are 6 different types of visualizations that we can use on our time series data. They are:

i) Line Plot: The first, and perhaps most popular, visualization for time series is the line plot. In **Fig. 3.2**, time series is shown on the X-axis with observation values along the Y-axis. This is an example of visualizing the Daily Temperature over Time data-set directly as a line plot. Also the Maximum and Minimum temperature is shown.

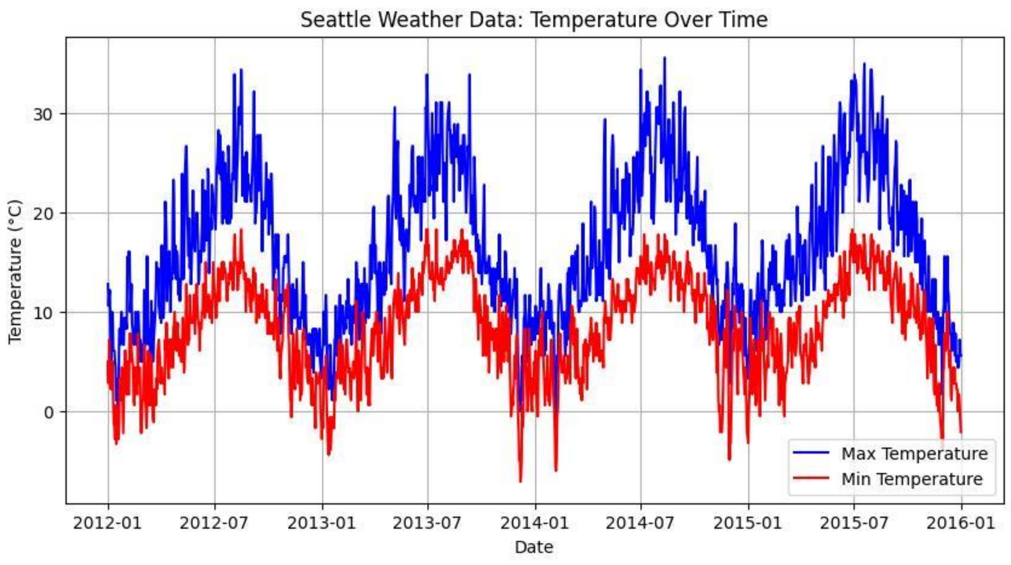

Fig. 3.2: Line Plot

ii) Histogram and Density Plot: This means a plot of the values without the temporal ordering. Some linear time series forecasting methods assume a well-behaved distribution of observations. This can be explicitly checked using tools like statistical hypothesis tests. But plots can provide a useful first check of the distribution of observations both on raw observations and after any type of data transform has been performed. **Fig. 3.3** shows a histogram plot of the observations in the Daily Maximum and Minimum Temperatures data-set. A histogram groups values into bins, and the frequency or count of observations in each bin can provide insight into the underlying distribution of the observations.

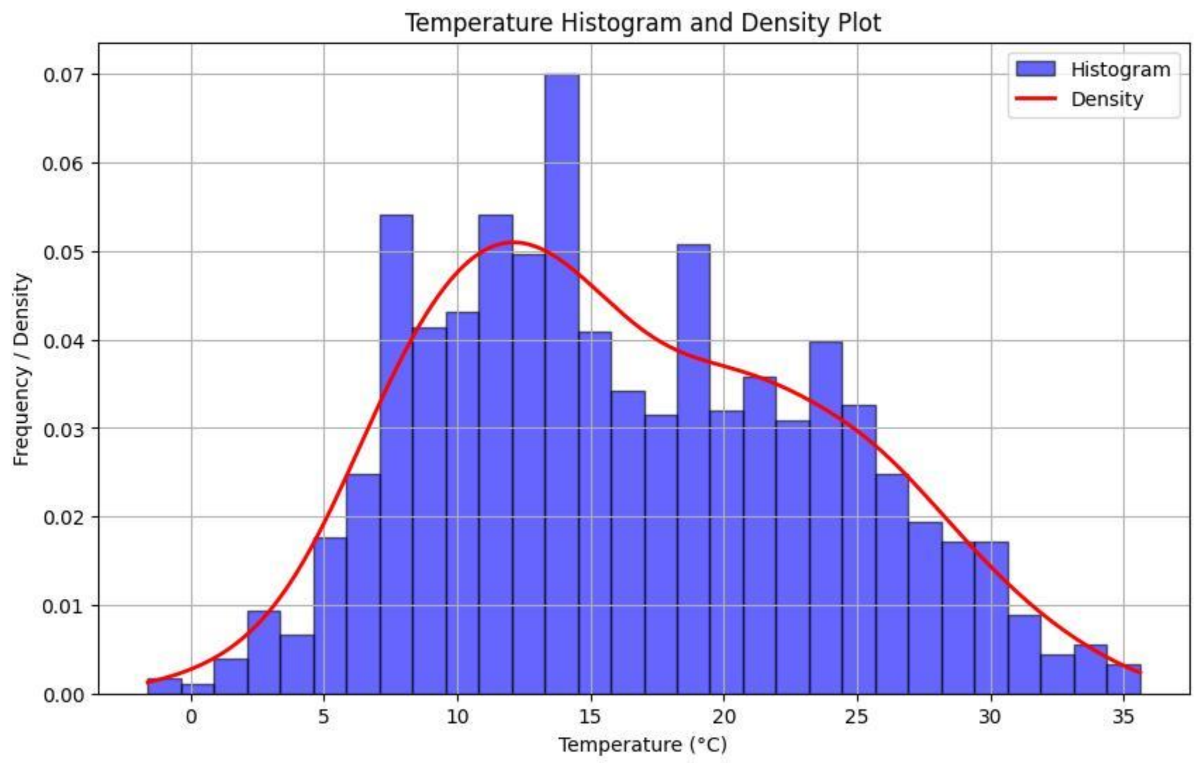

Fig. 3.3: Histogram and Density Plot

iii) Scatter Plot: Time series modelling assumes a relationship between an observation and the previous observation. Previous observations in a time series are called lags, with the observation at the previous time step called lag1, the observation at two time steps ago lag2, and so on. A useful type of plot to explore the relationship between each observation and a lag of that observation is called the scatter plot. **Fig. 3.4** shows f a lag plot for the Minimum Daily Temperatures dataset.

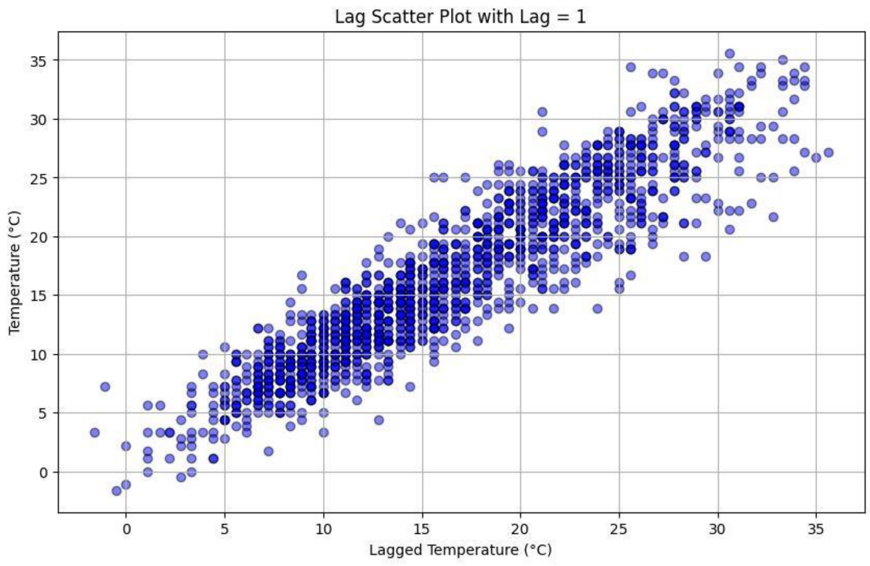

Fig. 3.4: Lag Scatter Plot

Auto-correlation Plot: We can quantify the strength and type of relationship between observations and their lags. In statistics, this is called correlation, and when calculated against lag values in time series, it is called auto-correlation (self-correlation). A correlation value calculated between two groups of numbers, such as observations and their lag1 values, results in a number between -1 and 1. The sign of this number indicates a negative or positive correlation respectively. A value close to zero suggests a weak correlation, whereas a value closer to - 1 or 1 indicates a strong correlation. Correlation values, called correlation coefficients, can be calculated for each observation and different lag values. Once calculated, a plot can be created to help better understand how this relationship changes over the lag. This type of plot is called an auto-correlation plot and Pandas provides this capability built in, called the auto-correlation plot. For Maximum Temperature we have shown the auto co-relation plot.

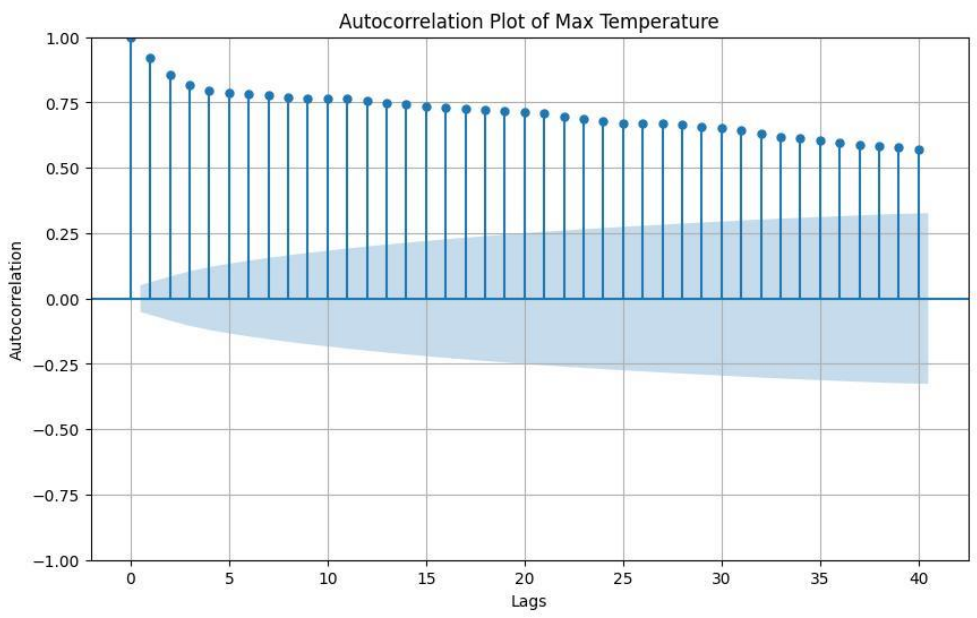

Fig. 3.5: Time Series Auto Correlation Plot

3.4 LSTM Networks for Time Series Prediction

Time series analysis involves studying sequential data points collected over time to identify patterns, trends, and seasonal variations. This analysis is crucial in predicting future values based on historical data, especially in stock market prediction where past price movements can inform future trends. LSTM networks, a type of Recurrent Neural Network (RNN), are well-suited for this task as they can capture long-term dependencies and patterns in time series data. LSTMs have memory cells and gates (input, forget, and output gates) that regulate the flow of information, enabling them to maintain and learn from long-term sequences. For stock market prediction, the sequential nature of the data is leveraged by normalizing the stock prices, creating time-based sequences, and splitting the data into training and testing sets. The LSTM model then learns the temporal dependencies in the stock prices through its architecture, which includes multiple layers and units with activation functions. Training involves minimizing a loss function like Mean Squared Error (MSE) using an optimizer, with techniques like dropout and early stopping to prevent overfitting. The trained model's performance is evaluated using metrics such as RMSE, MAE, and MAPE, ensuring it effectively captures the time series patterns to predict future stock prices.

Chapter 4
Data Collection

4.1 Data Sources

For experimental study, we downloaded historical datasets named TESLA, BankNifty, Google etc. from the website named "Kaggle" (https://www.kaggle.com/datasets)

Attribute Name	Min	Max
Open	3.228	1234.41
Close	3.16	1229.91
High	3.326	1243.49
Low	2.296	1217

Table 4.1: TESLA

Attribute Name	Min	Max
Open	0.049665	182.63
Close	0.049107	182.01
High	0.049665	182.94
Low	0.049107	179.12

Table 4.2: Apple

Sample Input:

Date	Open	High	Low	Close	Adj Close	Volume
29-06-2010	3.8	5	3.508	4.778	4.778	93831500
30-06-2010	5.158	6.084	4.66	4.766	4.766	85935500
01-07-2010	5	5.184	4.054	4.392	4.392	41094000
02-07-2010	4.6	4.62	3.742	3.84	3.84	25699000
06-07-2010	4	4	3.166	3.222	3.222	34334500
07-07-2010	3.28	3.326	2.996	3.16	3.16	34608500
08-07-2010	3.228	3.504	3.114	3.492	3.492	38557000
09-07-2010	3.516	3.58	3.31	3.48	3.48	20253000
12-07-2010	3.59	3.614	3.4	3.41	3.41	11012500

Table 4.3: Sample Input Snapshot

❖ As there are multiple options (stocks) available to forecast the closing price we are choosing the 'TESLA' stock to predict the output.

4.2 Data Types and Features

In this project, we utilize several types of data to enhance the accuracy of our stock market predictions. The primary data types include historical stock price data, macroeconomic indicators, and sentiment data from financial news sources.

i) Historical Stock Price Data:

- **Open Price:** The price at which a stock starts trading when the market opens.
- **Close Price:** The price at which a stock ends trading when the market closes.
- **High Price:** The highest price at which a stock traded during the trading day.
- **Low Price:** The lowest price at which a stock traded during the trading day.
- **Volume:** The total number of shares traded during the trading day.
- **Adjusted Close Price:** The closing price adjusted for dividends and stock splits.

ii) Macroeconomic Indicators:

- **Gross Domestic Product (GDP):** A measure of the economic performance of a country.
- **Inflation Rate:** The rate at which the general level of prices for goods and services is rising.
- **Unemployment Rate:** The percentage of the labour force that is unemployed and actively seeking employment.
- **Interest Rates:** The amount charged by lenders to borrowers for the use of money expressed as a percentage of the principal.

iii) Sentiment Data:

- **News Sentiment:** Analysis of sentiment from financial news articles using natural language processing techniques.
- **Social Media Sentiment:** Sentiment analysis of social media posts (e.g., Twitter) related to stocks and market trends.
- **Analyst Ratings:** Sentiment derived from financial analysts' ratings and reports.

iv) Feature Engineering:

- **Moving Averages**: Calculation of moving averages (e.g., 50-day, 200-day) to identify trends.
- **Relative Strength Index (RSI)**: A momentum oscillator that measures the speed and change of price movements.
- **Bollinger Bands**: A volatility indicator that plots standard deviation levels above and below a moving average.
- **Lagged Values**: Previous day's/week's/month's stock prices to capture temporal dependencies.
- **Technical Indicators**: Various technical analysis indicators such as MACD (Moving Average Convergence Divergence), Stochastic Oscillator, etc.

4.3 Data Pre-processing

The code begins by loading and resampling Tesla's adjusted close prices to a weekly frequency, followed by seasonal decomposition into trend, seasonal, and residual components. Each component is then scaled using MinMaxScaler. The data is split into training and testing sets, and sequences are created to feed into LSTM models. It's split into training and testing sets, with sequences prepared for LSTM input. Separate LSTM models are trained for each component to forecast future values, and their predictions are combined to estimate the final price.

The results are evaluated by comparing the predicted prices to actual values and visualizing the validation loss. Each LSTM model, designed to predict the next week's value, is trained separately for trend, seasonal, and residual components. The predictions for these components are combined to estimate the final price, and the results are evaluated and plotted.

Chapter 5
Methodologies

5.1 LSTM Algorithm

LSTM uses the RNN approach which has the ability to memorize. Each LSTM cell has three gates i.e. input, forget and output gates. While the data that enters the LSTM's network, the data that is required is kept and the unnecessary data will be forgotten by the forget gate. LSTM can be used in many applications such as for weather forecasting, NLP, speech recognition, handwriting recognition, time-series prediction, etc.

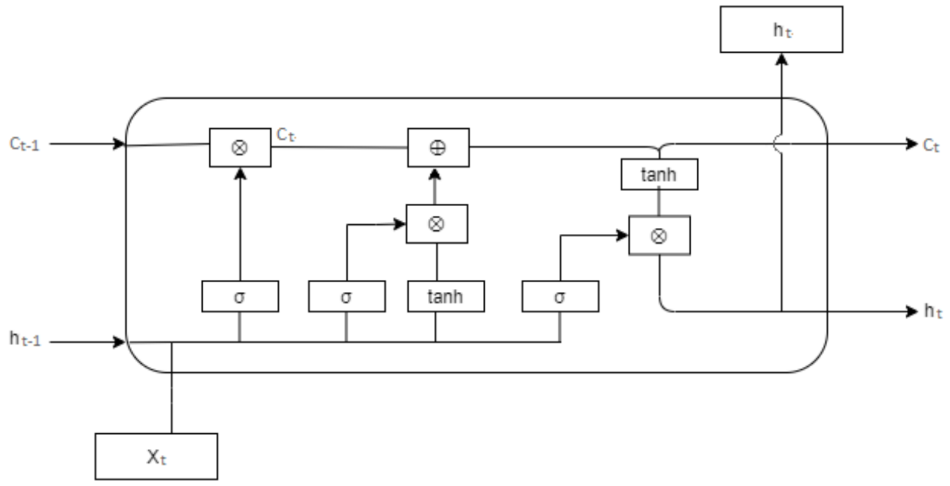

Fig. 5.1: LSTM Architecture

As shown in **Fig. 5.1**, the inputs to the current cell state (C_t) is the previous hidden state (h_{t-1}), previous cell state (C_{t-1}) and present input (X_t). The cell consists of three gates i.e. forget gate, input gate and output gate.

i) Forget Gate: A forget gate will remove unnecessary data from the cell state.

- The information that is less important or not required for the LSTM to understand things is removed by performing multiplication of hidden state by a sigmoid function.
- This step is necessary to optimize the performance of the model.
- It takes two inputs i.e., $h_{(t-1)}$ and x_t , where $h_{(t-1)}$ is the previous cell hidden state output and x_t is the current cell input.
- $F_t = \sigma (W_{fx} * X_t + W_{fh} * h_{t-1} + b_f)$

- **ii) Input Gate:** This cell is responsible for regulating the data that is added to the cell from the input. Forget gate is used to filter some input.
- A vector is created by adding all the possible values from the previous cell hidden state $h_{(t-1)}$ and current cell input X_t by using the tanh function. The output of the tanh function in the ranges of [-1, 1].
- Finally, the outputs of sigmoid and tanh functions are multiplied and the output is added to the cell state.
- $I_t = \sigma (W_{ix} * X_t + W_{hh} * h_{t-1} + b_i) + \tanh(W_{cx} * X_t + Wch * h_{t-1} + b_i)$

iii) Output Gate:

- Tanh function is applied to the cell state to create a vector with all possible values.
- Sigmoid function is applied to previous cell hidden state $h_{(t-1)}$ and current cell input x_t to filter necessary data from the previous cell.
- Now, the outputs of sigmoid and tanh functions are multiplied and this output is sent as a hidden state of the next cell.
- $Ot = \sigma (W_{ox} * X_t + W_{hh} * h_{t-1} + W_{oc} * C_{t-1} + b_i)$

Intermediate cell state (C_t) is obtained by the multiplication of Forget gate (F_t) with previous cell state (C_{t-1}). Then this intermediate state is added to the output of the input gate.

$$C_t = F_t * C_{t-1} + I_t$$

Current hidden/output state is obtained by multiplying output gate and tanh of cell state.

$$h_t = O_t * \tanh(C_t)$$

5.2 System Architecture

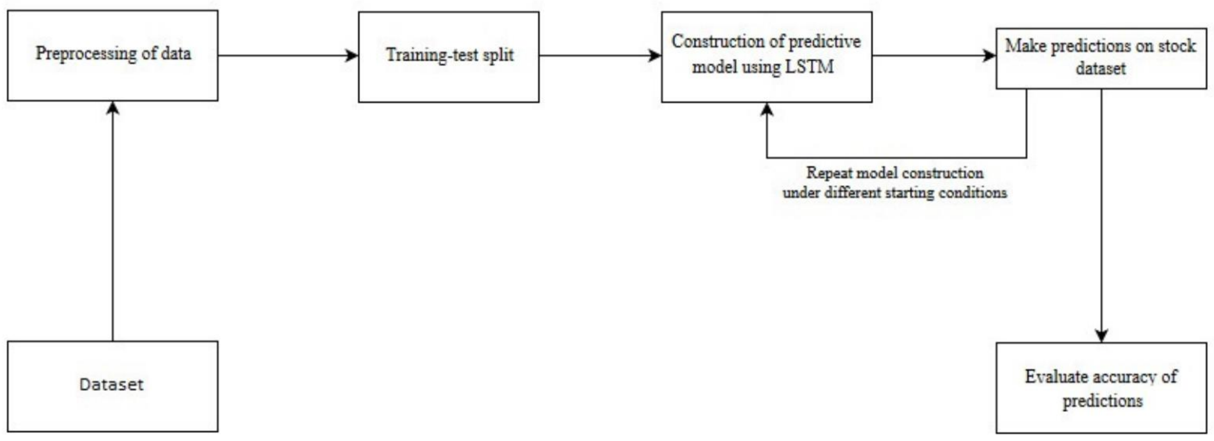
Preprocessing of data
Training-test split
Construction of predictive model using LSTM
Make predictions on stock dataset
Repeat model construction under different starting conditions
Dataset
Evaluate accuracy of predictions

Fig. 6.1: System Architecture

Data Selection: The first step is to select data for an organization and split the data into training and testing. We have used 80% for training and 20% for testing purposes.

Pre-processing of data: In pre-processing, we are selecting attributes required for the algorithm and the remaining attributes are neglected. The selected attributes are Trade Open, Trade High, Trade Low, Trade Close, Trade Volume. In pre-processing, we are using normalization to get values in a particular range.

Prediction using LSTM: In this system, we are using the LSTM algorithm for predicting stock values. Initially, the training data is passed through the system and train the model. Then in the testing phase, the predicted values are compared with the actual values.

Evaluation: In the evaluation phase we are calculating the Accuracy, Mean Square Error (MSE) and Root Mean Square Error (RMSE) values for comparison.

5.3 Model Implementations and Requirements

5.3.1 Implementation: The model implementation begins with loading and pre-processing stock market data, specifically Tesla's adjusted close prices. The data is resampled to a weekly frequency and decomposed into trend, seasonal, and residual components using seasonal decomposition. These components are then normalized using MinMaxScaler. The dataset is split into training and testing sets, with sequences created for LSTM input. Three separate LSTM models are built, each trained on one of the decomposed components (trend, seasonal, and residual). Each model is trained for 150 epochs, using validation data to monitor performance. After training, predictions are made for each component, and these are combined to generate final stock price predictions. The performance of the models is evaluated by plotting predicted versus actual prices and validating loss curves. This approach leverages the strengths of LSTM networks in capturing temporal dependencies and patterns in time series data.

5.3.2 Software Requirements:

- **Google Colab:** Google Colab is a popular cloud-based platform that allows us to write and execute Python code in a Jupyter notebook environment. It's particularly useful for machine learning and data analysis tasks because it provides free access to GPUs and TPUs, which can significantly speed up computations.
- **Python:** Python is a key language in machine learning due to its simplicity and extensive libraries. It supports data handling and pre-processing with tools like pandas and numpy, enabling efficient data manipulation and preparation. For model building, Python offers libraries such as scikit-learn for traditional algorithms, and TensorFlow and PyTorch for deep learning.

5.3.3 Hardware Requirements:

- **Windows:** intel i5 10th generation, 8GB RAM, 512 GB SSD
- **Macbook:** Air M1, RAM: 8GB, SSD: 128 GB
- **Linux**: intel i3 10th generation, 8GB RAM, 512 GB SSD.
- Stable Internet Connection.

❖ Computers equipped with a intel i3 10th gen processor or higher, the minimum RAM requirement would be 4GB and free Hard Drive Space of 100GB (Solid State Drives are preferred). The Optimal RAM & Internal Storage would be 8GB & 128GB and a suitable OS with stable internet connection.

Chapter 6
Implementation & Result

6.1 Libraries used

We have used python as the programming language to implement the interface, by using the various libraries provided by python. The libraries used are pandas, sklearn, numpy, matplotlib, statsmodel and tensorflow which has been imported in the program to implement the forecasting. These libraries allow us to perform:

- **pandas:** pandas is used for data manipulation and analysis. It provides data structures and functions needed to handle time-series data, such as reading the stock data from a CSV file, converting dates, setting the date as the index, and resampling the data to weekly frequency.
- **Scikit-learn (sklearn):** Scikit-learn is utilized for data preprocessing. Specifically, the MinMaxScaler from sklearn is used to normalize the data, which is essential for ensuring that the LSTM model performs optimally by scaling the input features to a range between 0 and 1.
- **Numpy:** Numpy is employed for numerical operations. It is crucial for creating datasets, reshaping arrays etc. It provides support for arrays and matrices, which are extensively used in machine learning algorithms.
- **Matplotlib:** Matplotlib is used for plotting and visualizing data. It helps in creating various plots, such as the original adjusted close prices, seasonally decomposed components, and the predicted versus actual prices, which are vital for understanding data trends and model performance.
- **Statsmodels:** Statsmodels is utilized for statistical modeling and analysis. The seasonal_decompose function from statsmodels is used to perform seasonal decomposition of the time series data, which separates the data into trend, seasonal, and residual components.

- **TensorFlow:** TensorFlow, specifically its Keras API, is used for building and training the LSTM models. The LSTM layers are designed to capture the temporal dependencies in the time-series data. The models are built with sequential layers including LSTM, LeakyReLU activation, Dropout, and Dense layers, optimized using the Nadam optimizer.

6.2 Code Snippet

The code snippet for Time Series Data Analysis of Stock Market for forecasting / predicting using LSTM network is mentioned below:

```
import pandas as pd

from sklearn.preprocessing import MinMaxScaler

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import seasonal_decompose

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout, LeakyReLU

from tensorflow.keras.optimizers import Nadam


# Load data

data = pd.read_csv('TSLA.csv')

data['Date'] = pd.to_datetime(data['Date'])

data.set_index('Date', inplace=True)


# Filter data to include only up to 2020 to avoid covid effect(noise)

data = data[data.index <= '2020-12-31']
```

```
# Resample the data to weekly frequency, using the mean of the daily adjusted close prices
data = data.resample('W').mean()

# Perform seasonal decomposition
result = seasonal_decompose(data['Adj Close'], model='multiplicative', period=52)
data['trend'] = result.trend
data['seasonal'] = result.seasonal
data['residual'] = result.resid

# Drop NaN values created by the decomposition
data.dropna(inplace=True)

# Scale data
scaler = MinMaxScaler()
scaled_trend = scaler.fit_transform(data['trend'].values.reshape(-1, 1))
scaled_seasonal = scaler.fit_transform(data['seasonal'].values.reshape(-1, 1))
scaled_residual = scaler.fit_transform(data['residual'].values.reshape(-1, 1))

# Split the data
train_size = len(scaled_trend) - 104  # Last 24 months for testing
train_trend, test_trend = scaled_trend[:train_size], scaled_trend[train_size:]
train_seasonal, test_seasonal = scaled_seasonal[:train_size], scaled_seasonal[train_size:]
train_residual, test_residual = scaled_residual[:train_size], scaled_residual[train_size:]

# Prepare dataset
```

```
n_input = 3  # number of previous weeks to use for prediction
n_output = 1  # predicting the next week

def create_dataset(data, n_input, n_output):
    X, y = [], []
    for i in range(len(data) - n_input - n_output + 1):
        X.append(data[i:i+n_input])
        y.append(data[i+n_input:i+n_input+n_output])
    return np.array(X), np.array(y)

X_train_trend, y_train_trend = create_dataset(train_trend, n_input, n_output)
X_test_trend, y_test_trend = create_dataset(test_trend, n_input, n_output)

X_train_seasonal, y_train_seasonal = create_dataset(train_seasonal, n_input,
n_output)
X_test_seasonal, y_test_seasonal = create_dataset(test_seasonal, n_input,
n_output)

X_train_residual, y_train_residual = create_dataset(train_residual, n_input,
n_output)
X_test_residual, y_test_residual = create_dataset(test_residual, n_input,
n_output)

#  y_train and y_test are reshaped correctly for inverse scaling
y_train_trend = y_train_trend.reshape(-1, 1)
y_test_trend = y_test_trend.reshape(-1, 1)

y_train_seasonal = y_train_seasonal.reshape(-1, 1)
y_test_seasonal = y_test_seasonal.reshape(-1, 1)
```

```
y_train_residual = y_train_residual.reshape(-1, 1)
y_test_residual = y_test_residual.reshape(-1, 1)

# Define LSTM model
def create_lstm_model():
    model = Sequential()
    model.add(LSTM(150, return_sequences=True, input_shape=(n_input, 1)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(Dropout(0.3))
    model.add(LSTM(150, return_sequences=True))
    model.add(LeakyReLU(alpha=0.1))
    model.add(Dropout(0.3))
    model.add(LSTM(150))
    model.add(LeakyReLU(alpha=0.1))
    model.add(Dropout(0.3))
    model.add(Dense(n_output))
    model.compile(optimizer=Nadam(), loss='mae')
    return model

# Train models for each component
model_trend = create_lstm_model()
history_trend = model_trend.fit(X_train_trend, y_train_trend, epochs=100,
validation_data=(X_test_trend, y_test_trend), verbose=1)

model_seasonal = create_lstm_model()
history_seasonal = model_seasonal.fit(X_train_seasonal, y_train_seasonal,
epochs=100, validation_data=(X_test_seasonal, y_test_seasonal), verbose=1)
```

```
model_residual = create_lstm_model()
history_residual = model_residual.fit(X_train_residual, y_train_residual,
epochs=100, validation_data=(X_test_residual, y_test_residual), verbose=1)

# Make predictions
train_predictions_trend = model_trend.predict(X_train_trend)
test_predictions_trend = model_trend.predict(X_test_trend)

train_predictions_seasonal = model_seasonal.predict(X_train_seasonal)
test_predictions_seasonal = model_seasonal.predict(X_test_seasonal)

train_predictions_residual = model_residual.predict(X_train_residual)
test_predictions_residual = model_residual.predict(X_test_residual)

# Inverse transform the predictions and actual values to original scale
trend_scaler = MinMaxScaler()
seasonal_scaler = MinMaxScaler()
residual_scaler = MinMaxScaler()



scaled_trend = trend_scaler.fit_transform(data['trend'].values.reshape(-1, 1))
scaled_seasonal =
seasonal_scaler.fit_transform(data['seasonal'].values.reshape(-1, 1))
scaled_residual =
residual_scaler.fit_transform(data['residual'].values.reshape(-1, 1))

train_predictions_trend =
trend_scaler.inverse_transform(train_predictions_trend)
y_train_trend_inv = trend_scaler.inverse_transform(y_train_trend)
```

```
test_predictions_trend =
trend_scaler.inverse_transform(test_predictions_trend)
y_test_trend_inv = trend_scaler.inverse_transform(y_test_trend)


train_predictions_seasonal =
seasonal_scaler.inverse_transform(train_predictions_seasonal)
y_train_seasonal_inv = seasonal_scaler.inverse_transform(y_train_seasonal)
test_predictions_seasonal =
seasonal_scaler.inverse_transform(test_predictions_seasonal)
y_test_seasonal_inv = seasonal_scaler.inverse_transform(y_test_seasonal)


train_predictions_residual =
residual_scaler.inverse_transform(train_predictions_residual)
y_train_residual_inv = residual_scaler.inverse_transform(y_train_residual)
test_predictions_residual =
residual_scaler.inverse_transform(test_predictions_residual)
y_test_residual_inv = residual_scaler.inverse_transform(y_test_residual)


# Combine the components to get final predictions
train_predictions = train_predictions_trend + train_predictions_seasonal +
train_predictions_residual
test_predictions = test_predictions_trend + test_predictions_seasonal +
test_predictions_residual


# Plot predicted vs actual values for the test set
plt.figure(figsize=(10, 6))
plt.plot(data.index[-len(y_test_trend_inv):], data['Adj Close'][-
len(y_test_trend_inv):], label='Actual Prices')
plt.plot(data.index[-len(y_test_trend_inv):], test_predictions, label='Predicted
Prices')
plt.title('Predicted vs Actual Prices (up to 2020)')
plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()
```

- **Code Snippet for the calculation of Mean Absolute Error, Mean Squared Error, Root mean squared error**:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, 
mean_absolute_percentage_error

import numpy as np

# Calculate Mean Absolute Error (MAE)

mae = mean_absolute_error(y_test_trend_inv, test_predictions)

print(f"Mean Absolute Error (MAE): {mae}")

# Calculate Mean Squared Error (MSE)

mse = mean_squared_error(y_test_trend_inv, test_predictions)

print(f"Mean Squared Error (MSE): {mse}")

# Calculate Root Mean Squared Error (RMSE)

rmse = np.sqrt(mse)

print(f"Root Mean Squared Error (RMSE): {rmse}")

# Calculate Mean Absolute Percentage Error (MAPE)

mape = mean_absolute_percentage_error(y_test_trend_inv, test_predictions)

print(f"Mean Absolute Percentage Error (MAPE): {mape * 100:.2f}%")

# Calculate simple accuracy percentage

accuracy = np.mean(np.abs((y_test_trend_inv - test_predictions) / 
y_test_trend_inv) < 0.1)

print(f"Accuracy: {accuracy * 100:.2f}%")
```

6.3 Experimental Results

Seasonally Decomposed Components:

- **Trend:** The trend component shows the long-term movement of the stock prices, filtered to remove seasonal and residual variations.
- **Seasonal:** The seasonal component captures repeating patterns or cycles in the stock prices at a weekly frequency.
- **Residual:** The residual component represents the noise or irregular fluctuations in the stock prices after removing trend and seasonal effects.

Predicted vs Actual Prices: This graph compares the actual adjusted close prices with the predicted prices obtained from the combined trend, seasonal, and residual components. It demonstrates the effectiveness of the LSTM model in capturing the overall patterns and making accurate predictions.

Validation Loss: The validation loss graphs for the trend, seasonal, and residual models show the mean squared error (MSE) loss over the epochs during training. This helps in understanding the model's performance and convergence during training.

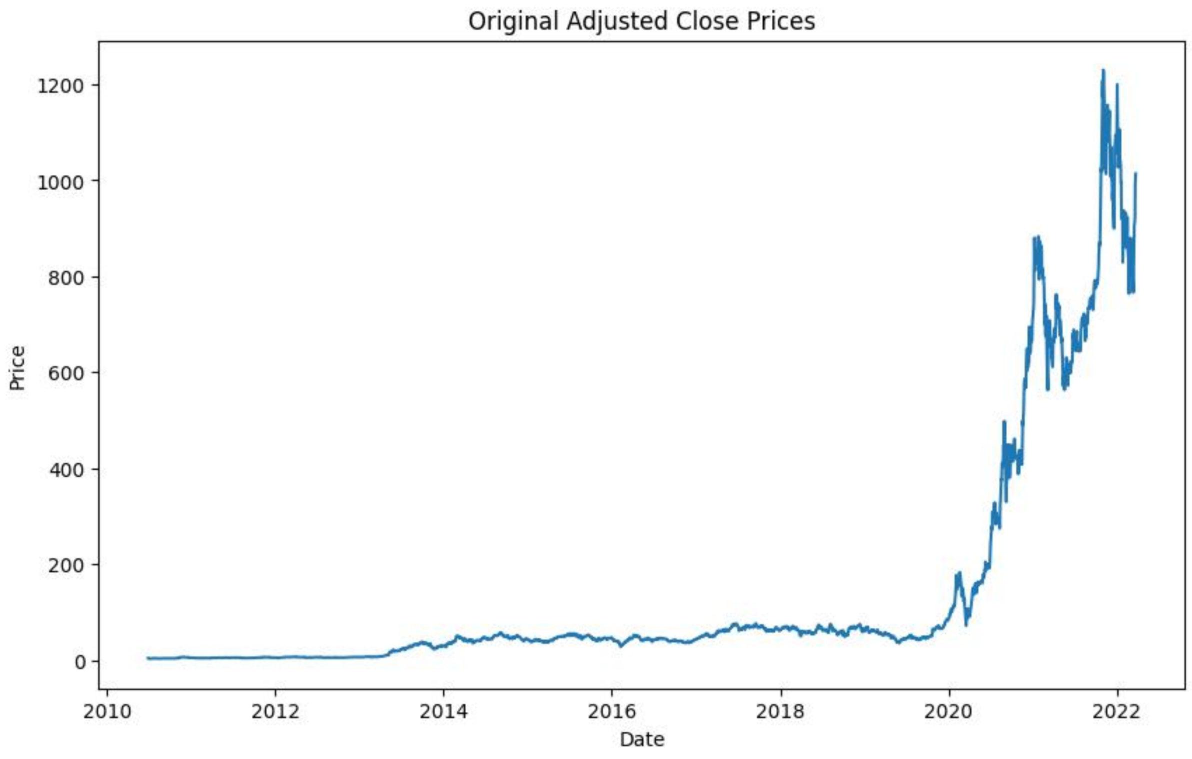

Fig. 6.1: Original Adjusted Close Prices Graph

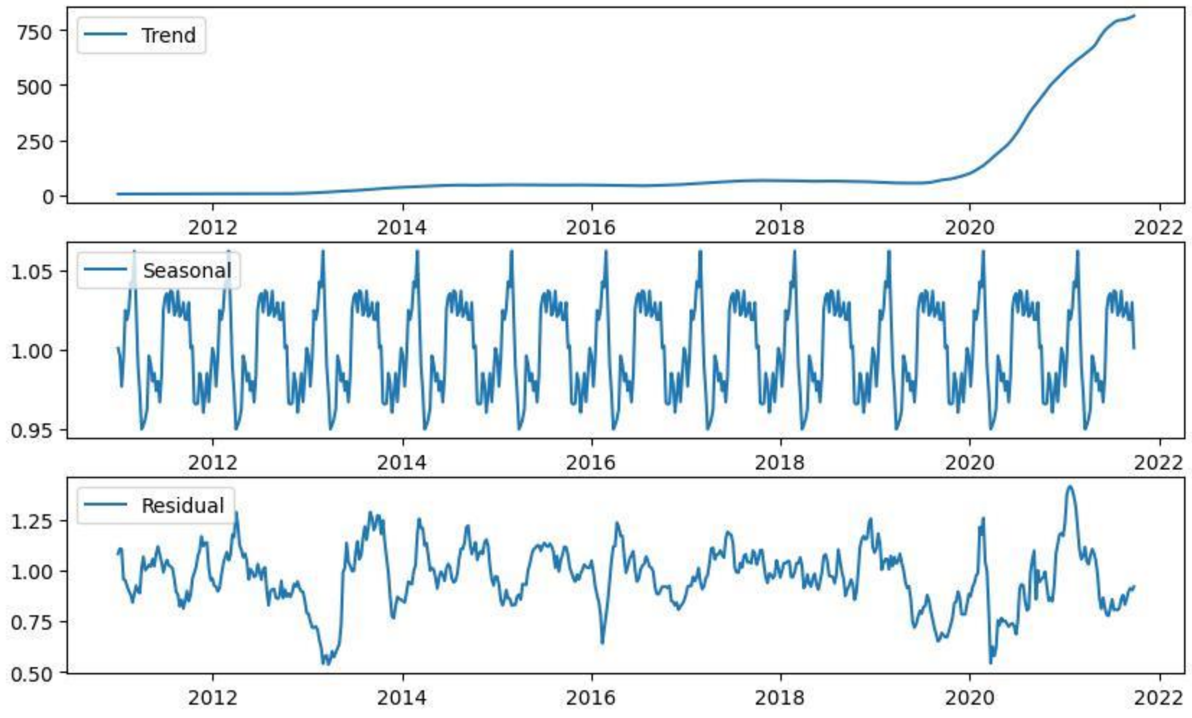

Fig. 6.2: Trend, Seasonal and Residual Decompose Graph

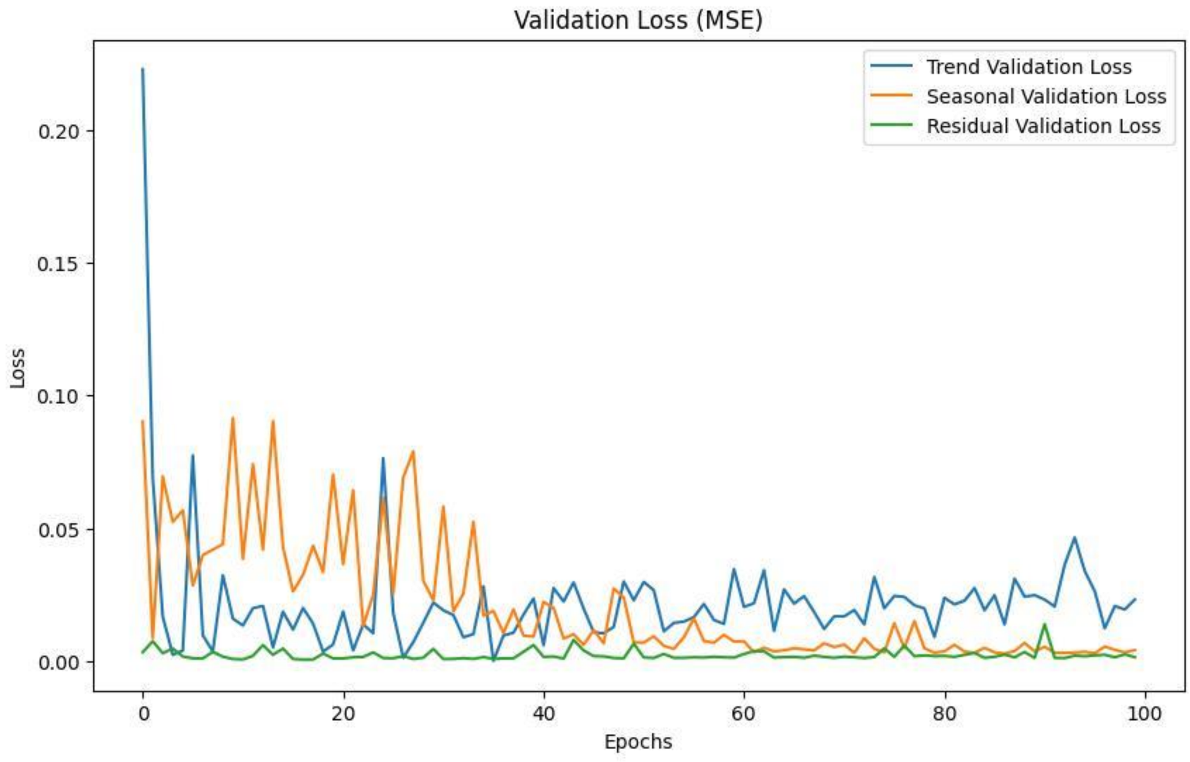

Fig. 6.3: Validation Loss Graph

```
Epoch   Trend Validation Loss    Seasonal Validation Loss        Residual Validation Loss
1       0.222865                 0.090186                0.003251
11      0.013341                 0.038465                0.000555
21      0.018600                 0.036452                0.000927
31      0.019068                 0.058189                0.000704
41      0.005907                 0.022225                0.001524
51      0.029744                 0.006909                0.001336
61      0.020337                 0.007389                0.002611
71      0.016771                 0.006152                0.001600
81      0.023774                 0.003589                0.001872
91      0.023108                 0.005301                0.013828
```

Fig 6.4: Validation Loss in Tabular Form

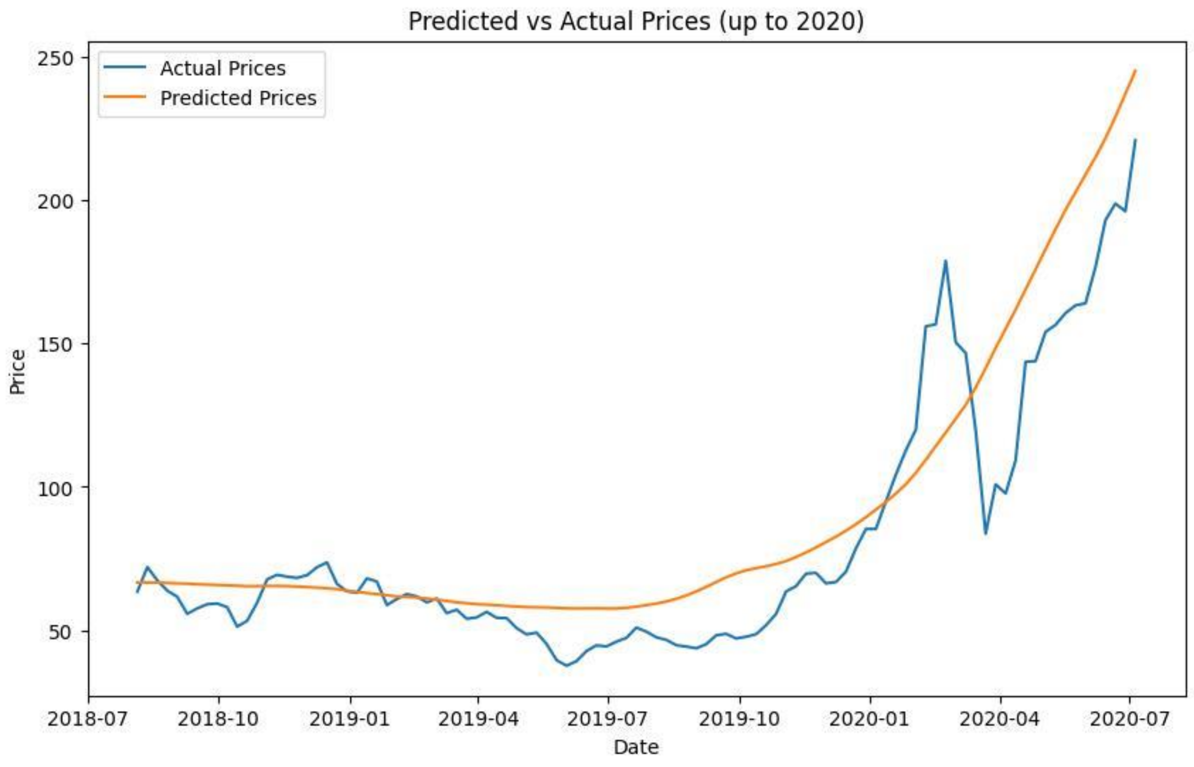

Fig 6.5: Actual and Predicted Chart (Visual Comparison)

<u>**Evaluation Metrics (Output)**</u>

Mean Absolute Error (MAE): 8.396965011062937

Mean Squared Error (MSE): 154.63776142685674

Root Mean Squared Error (RMSE): 12.435343237195214

Mean Absolute Percentage Error (MAPE): 7.38%

Accuracy: 75.25%

Chapter 7
Conclusion & Future Scope

7.1 Challenges Faced

i) Data Quality and Pre-processing:

- **Missing Data:** The original dataset had missing values which required handling to prevent biases and inaccuracies in the model training.
- **Noise in Data:** Stock market data can be highly volatile, with significant noise that needed to be managed through pre-processing techniques like resampling and seasonal decomposition.
- **Time Frame Selection:** Selecting an appropriate time frame (excluding data beyond 2020 to avoid the COVID-19 effect) was crucial to ensure the model's predictions were not influenced by atypical market behaviour.

ii) Model Selection and Training:

- **Model Complexity:** Building and training an LSTM model involves selecting the right architecture and hyper parameters, which can be complex and time-consuming.
- **Overfitting:** Ensuring the model generalizes well to unseen data required careful tuning of the model and regularization techniques like dropout layers.
- **Computational Resources:** Training deep learning models, especially LSTMs, can be computationally intensive, necessitating the use of powerful hardware and cloud resources like Google Colab.

iii) Combining Model Outputs:

- **Component Integration:** Combining predictions from trend, seasonal, and residual models to form a final prediction required careful handling and scaling to ensure accuracy.
- **Inverse Transformation:** Reversing the scaling transformation to interpret the model outputs in the

original scale of the data involved ensuring the transformations were correctly applied.

iv) Evaluation and Validation:

- **Evaluation Metrics:** Choosing appropriate metrics (e.g., mean squared error) to evaluate the model's performance was critical for assessing accuracy.
- **Validation Strategy:** Implementing a robust validation strategy to prevent overfitting and ensure the model's predictions were reliable and generalizable.

7.2 Conclusion

As modern science and technology advance, more knowledge is being applied in the field of finance to predict future circumstances, aiming to minimize loss and maximize profit. To address these forecasting problems, various algorithms have been developed and tested. However, it is unfortunate that no model can guarantee successful stock market predictions, as numerous factors and dependencies play crucial roles in determining future stock prices.

In this paper, we have tried to forecast the stock prices using a commonly employed time series model, Long Short-Term Memory (LSTM). LSTM is particularly suited for sequential data due to its ability to capture long-term dependencies. For this study, we collected stock price data in CSV format and applied various pre-processing techniques, including decomposing and differencing, to ensure stationarity.

The stock price data was then used to train the LSTM model. The model was trained on different components—trend, seasonal, and residual—extracted through seasonal decomposition. After the training, the model was utilized to forecast stock prices, which were visualized using graphs. Despite the power of LSTM in handling time series data, predicting stock prices remains a challenging task due to the complexity and dependencies inherent in the data, which can sometimes lead to erroneous predictions.

In conclusion, as the stock market is a critical sector, comparing different time series models, including advanced models like LSTM, can be helpful in making informed decisions on whether to buy or sell stocks. This crucial purpose can be served through careful and detailed time series analysis of stock market predictions.

7.3 Future Work

Stock market is one of the most unpredictable field of finance. A large number of factors such as raw materials, suppliers, people's sentiment towards a company have a big impact on the stock prices of a company. In future works, we would like to concentrate more on the public sentiments of a company to forecast the stock prices. Sentiments can be analyzed and manipulated in order to determine people's opinions regarding a company. These public opinions can be analyzed from Twitter or Facebook and financial news. A hybrid model which combines historical data and sentiments can be developed in order to predict stock prices more accurately. Other factors like environmental factors such as flood, storm etc. can analyzed and combine with sentiment to predict the stock prices and increase the accuracy. As stock market is a huge and random sector more work can be done to find better and accurate predictive times models. In the future, we can extend this application for predicting cryptocurrency trading and also, we can add sentiment analysis for better predictions.

References

[1] Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose1, Giridhar Maji, Narayan C. Debnath, Soumya Sen.

[2] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman - Stock price prediction using LSTM, RNN and CNN-sliding window model - 2017.

[3] Murtaza Roondiwala, Harshal Patel, Shraddha Varma, “Predicting Stock Prices Using LSTM” in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.

[4] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, “An innovative neural network approach for stock market prediction”, 2018

[5] Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur , INDIA - Stock Market Prediction Using Machine Learning.

[6] Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.

[7] Anurag Sinha Department of computer science, Student, Amity University Jharkhand Ranchi, Jharkhand (India), 834001 - Stock Market Prediction Using Machine Learning.

[8] V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India - Stock Market Prediction Using Machine Learning. Mukt Shabd Journal Volume X.