Stock Market Prediction using LSTM and Markov Chain Models: A Case Study of Royal Bank of Canada Stock

by

Amer Kumar
B.Sc., Ghulam Ishaq Khan Institute of Engineering, Science and Technology, 2018

A Master's Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

Masters of Engineering

in the Department of Electrical and Computer Engineering

Stock Market Prediction using LSTM and Markov Chain Models: A Case Study of Royal Bank of Canada Stock

by

Amer Kumar
B.Sc., Ghulam Ishaq Khan Institute of Engineering, Science and Technology, 2018

Supervisory Committee

Prof. Fayez Gebali, PEng, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mohamed Watheq El-Kharashi, Co-Supervisor
(Department of Electrical and Computer Engineering)

ABSTRACT

Stock price prediction is one of the most important aspects of financial investment. This research aims to provide insights into the dynamics of stock prices, enabling more informed decision-making in financial investments by combining these two modeling approaches. Using a four-layer long short-term memory (LSTM) architecture and the Root Mean Square Error (RMSE) as the loss function, we aim to capture temporal dependencies and patterns to predict closing prices. Furthermore, we employ a three-state Markov chain to estimate the transition matrix, and metrics like steady-state distribution and mean hitting times have been used to calculate the matrix. The preliminary results indicate that this approach shows promising results for stock market prediction as LSTM has predictive power that caters more to long-term temporal trends while Markov Chain provides probabilistic values for staying and transitioning to states. The findings of the study highlight the effectiveness of combining LSTM and Markov Chain in capturing the intricate dynamics of the stock market data and predicting stock market prices.

Contents

List of Tables

List of Figures

List of Acronyms

BPTT Backpropagation through Time

CNN Convolutional Neural Network

LSTM Long Short Term Memory

MACD Moving Average Convergence and Divergence

MAE Mean Absolute Error

MSE Mean Squared Error

RMSE Root Mean Square Error

RNN Recurrent Neural Network

RY Royal Bank of Canada

TSX Toronto Stock Exchange

ACKNOWLEDGEMENTS

I would like to thank:

my parents, siblings, friends, for their unwavering support, understanding, and encouragement throughout this journey.

Dr. Feyaz Gebali, for invaluable guidance, support, and expertise throughout the completion of this report. His unwavering commitment to academic excellence, and his continuous availability for discussions and consultations.

Dr. Seyed Mehdi Hazrati Fard, for his valuable insights and contributions to this report. His expertise in neural networks and his collaborative approach have greatly enriched my understanding of the subject matter and provided valuable perspectives for this project.

We are what we repeatedly do. Excellence, then, is not an act, but a habit.
Aristotle

DEDICATION

I dedicate this project to my beloved brother, Omesh, and my dear friend, Tanveer, whose unwavering support and belief in my abilities inspired me throughout this academic journey. To Omesh, thank you for being my pillar of strength and pushing me to reach my highest aspirations. And to Tanveer, thank you for standing by me through the ups and downs of this educational pursuit, your friendship has been invaluable.

Chapter 1

Introduction

The main goal of this master of engineering project is to answer the following research question: How can LSTM and Markov Chain models be used to predict the future states and prices of a selected stock? LSTM (Long Short-Term Memory) is a type of recurrent neural network that can capture temporal dependencies and patterns in sequential data, such as stock prices. Markov Chain is a mathematical model that describes the probability of transitioning from one state to another in a stochastic process, such as stock price movements. By combining these two models, this research aims to provide valuable insights into the dynamics of stock prices, enabling more informed decision-making in financial investments.

The Royal Bank of Canada (RY) stock has been selected for this analysis due to its prominence as one of the leading financial stocks listed on the Toronto Stock Exchange. The dataset used encompasses the daily closing prices of RY shares spanning from March 1st, 2019, to March 31st, 2023.

To predict the closing prices, a four-layer LSTM architecture is employed, and the Root Mean Square Error (RMSE) is utilized as the loss function to quantify the disparity between the actual and predicted values. 80 percent of data has been chosen for training (which is until June 8th, 2022), and after this, the remaining 20 percent has been used for testing. Finally, we have applied RMSE to compare the actual with predicted values. Similarly, a three-state Markov chain is constructed to estimate the transition matrix, and metrics like steady-state distribution and mean hitting times have been used to calculate the matrix. Different thresholds for classifying in Markov chain have been used to calculate different transition matrices. These probabilities provide insights into the likelihood of remaining in a specific state over the predicted period.

The preliminary results indicate that this approach shows promising results for stock market prediction as LSTM has predictive power that caters more to long-term temporal trends while Markov chain provides probabilistic values for staying and transitioning to states. Using both of these approaches gives us two-fold results that are effective in stock price and overall market prediction. Therefore, this report demonstrates the feasibility and effectiveness of applying LSTM modeling and Markov Chain analysis for stock market prediction.

1.1 Related Work

Investing in the stock market means trying to predict if the prices of stocks will go up or down in the future. It's not the same as gambling. However, it can be difficult to predict how stocks will perform because it requires knowledge and understanding of how stocks have performed in the past and other factors that may impact their future performance. To make good investment decisions, many investors study and research stocks' historical performance. This can be a challenging task, as it requires a lot of knowledge and expertise in the field of corporate finance. Even experts in the field often find stock price forecasting to be a difficult task [1].

The selection of efficient stocks for the construction of investment portfolios remains a prevalent challenge faced by fund managers and investors. In the process of portfolio construction and management, the initial step necessitates the examination of the performance of stocks or funds, followed by the grouping of said stocks or funds into a stock fund, otherwise known as a portfolio [2].

A study by Kaya and Karsligil revolves around the determination of stock prices for any given business entity extends beyond an analysis of the financial position of the company alone, and necessitates an examination of the flow of information pertaining to the company, as well as an evaluation of the economic performance of the sector and the country in which the company operates. As a result, the forecasting of stock prices has become increasingly complex as it involves myriad factors [3].

When building a portfolio, portfolio managers usually use two different strategies to choose the securities or funds to include. These strategies are called quantitative and qualitative analysis. Quantitative analysis uses numbers and statistics to evaluate the performance of an asset before it is included in the portfolio. On the other hand, qualitative analysis looks at the management of the company or fund, which can give important insights into the potential performance of the asset. However,

this approach takes more time to research and it is not always easy to find all the information needed [4].

Several researchers over the years [5] [6] have developed various methodologies for predicting future share market prices using either deterministic statistical approach or stochastic approach. However, there are little studies on the use of stochastic approach in stock selection for the Toronto stock market/exchange. The goal of this project is to apply Markov chain techniques to the predicted closing stock prices using steady states for investment consideration.

Many researchers have used Markov chains model to analyze and predict the behavior of the stock market. Studies have shown that this model is effective in predicting how stocks will move. For example, Choji et al. used the Markov chain model to predict the future performance of two banks - Guarantee Trust Bank of Nigeria and First bank of Nigeria. They used data from 2005 to 2010 and used a transition probability matrix and probability vector to make predictions about the future share prices of the banks, whether they would increase, decrease, or stay the same [7].

D. Zhang and X. Zhang used the Markov chain model to predict the stock market trend in China. They found that this model is good for analyzing and predicting stock market index and closing stock prices. However, the model does not take into account any past events and it is suggested that the results from the Markov chain model should be combined with other factors that may affect stock market variations in making decisions [8].

The study conducted by Adil Moghar and Mhamed Hamiche aims to build a model using Recurrent Neural Networks (RNN) and especially the Long-Short Term Memory model (LSTM) to predict future stock market values. The main objective of this paper is to see which precision a Machine learning algorithm can predict and how much the epochs can improve the model. The authors use daily stock exchange rates of NASDAQ and S&P 500 from January 4, 2010, to January 30, 2020, to construct and test their model. They use two evaluation measures, the coefficient of determination R2 and the Root Mean Squared Error (RMSE), to judge the relevance of the results. They compare their model with other models such as ARIMA, SVM, and MLP. They find that their model outperforms the other models in terms of accuracy and error rate. They also find that increasing the number of epochs improves the performance of their model [9].

The paper by Fischer and Krauss1 aims to apply Long Short-Term Memory

(LSTM) networks, a type of Recurrent Neural Networks (RNN), to predict the directional movements of the constituent stocks of the S&P 500 index from 1992 to 2015. The authors compare their model with other methods such as random forest (RAF), deep neural net (DNN), and logistic regression (LOG). They find that LSTM networks outperform the other methods in terms of accuracy and Sharpe ratio. They also identify some common patterns among the stocks selected for trading by the LSTM model, such as high volatility and short-term reversal return profile. They further analyze the sources of profitability and risk exposure of the LSTM model and compare them with the benchmark models [10].

The paper by Kim and Kim present a novel model for forecasting stock prices using both stock time series and stock chart images as inputs. The model, called the feature fusion LSTM-CNN model, consists of two components: a long short-term memory (LSTM) network and a convolutional neural network (CNN). The LSTM network extracts temporal features from the stock time series, such as price and volume. The CNN network extracts image features from the stock chart images, such as trend and pattern. The features from both networks are then concatenated and fed into a fully connected layer to generate the final prediction. The authors evaluate their model on the SPDR S&P 500 ETF data, using four types of stock chart images: line chart, bar chart, candlestick chart, and point-and-figure chart. They compare their model with two baseline models: a single LSTM model that uses only the stock time series as input, and a single CNN model that uses only the stock chart images as input. They use three metrics to measure the performance: mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). They also conduct an ablation study to analyze the contribution of each feature type to the prediction. The results show that the feature fusion LSTM-CNN model outperforms the single models in all metrics and for all types of stock chart images. The authors also find that the candlestick chart is the most suitable image type for forecasting stock prices, as it contains more information than the other types, such as open, close, high, and low prices [11].

The paper titled "Deep learning for stock prediction using numerical and textual information" proposes a novel deep learning approach that uses both numerical and textual information to forecast stock prices as time series. The paper addresses the limitations of previous works that use either Bag-of-Words model or single type of input or single company data. The paper introduces two deep learning models, Paragraph Vector and Long Short-Term Memory (LSTM), to capture the complex

relationships between news events and opening prices. Paragraph Vector is a model that converts news articles into distributed representations that preserve the semantic and syntactic information. LSTM is a model that can learn the long-term dependencies and temporal effects of past events on future prices. The paper evaluates the performance of the proposed approach on real-world data of fifty companies listed on Tokyo Stock Exchange and shows that it outperforms the baseline methods [12].

These studies collectively contribute to the exploration of LSTM and Markov Chain techniques in stock market prediction, showcasing the potential of these models in capturing intricate dynamics, incorporating textual information, and improving prediction accuracy. By considering and building upon the insights gained from these papers, we aim to propose an innovative approach that combines LSTM and Markov Chain for efficient financial stock selection and portfolio construction.

1.2 Report Organization

The report is organized as follows:

Chapter 1 provides an introduction to the report, outlining its goals and presenting an overview of the related work in the field.

Chapter 2 offers a comprehensive overview of LSTM and Markov Chain, which are the key methodologies utilized in the subsequent sections of the report.

Chapter 3 delves into the methodology employed in the report. The first section explains the implementation and utilization of LSTM, providing a detailed description of the approach adopted. The second section focuses on the application of Markov Chain, outlining how it has been incorporated into the analysis.

Chapter 4 showcases the experiments conducted in the report, highlighting the specific methodology described in Chapter 3 and discussing the data used and experimental setup.

Chapter 5 contains the conclusion of the report, summarizing the key findings and discussing their implications. Additionally, it enumerates potential avenues for future research and development of the concept and its applications.

Chapter 2

LSTM and Markov Chain Overview

2.1 Introduction

In this chapter, we will provide a brief overview of neural networks, with a focus on the Long Short-Term Memory (LSTM) model, and introduce the Markov Chain approach. LSTM has emerged as a powerful technique for capturing long-term dependencies and intricate temporal patterns in sequential data, making it highly suitable for stock market prediction. We will discuss the basic structure, variants, and applications of LSTM, highlighting its significance in financial analysis. Additionally, we will introduce the Markov Chain approach, which enables us to model the probabilistic transitions between different states in the stock market. By combining LSTM and Markov Chain techniques, we aim to enhance stock market analysis and prediction accuracy and effectiveness.

2.2 Long Short Term Memory

The subsequent sections provide a brief overview of neural networks, including RNNs and LSTM, which is a specialized type of RNN.

2.2.1 Neural Networks

Neural networks are a type of machine learning that uses algorithms and mathematics to process data and recognize relationships without any task-specific rules. Neural

networks are inspired by the structure and function of the biological neurons and synapse in the human brain. Neural networks can learn from data and adapt to new situations, making them suitable for solving complex problems in various domains, such as computer vision, natural language processing, speech recognition, and more.

A neural network consists of a collection of nodes or units, called artificial neurons, that are connected by weighted links. Each node receives some input from the previous layer or the external source, performs some computation based on the input and its activation function, and produces some output to the next layer or the final destination. The activation function is a non-linear function that determines whether the node should fire or not, based on a threshold value. The weights are the parameters that control the strength and direction of the connection between nodes. The weights are learned from data through a process called training, which involves adjusting the weights to minimize the error between the predicted output and the desired output.

Different types of neural networks are based on their architecture, such as feedforward neural networks, recurrent neural networks, convolutional neural networks, etc. In this report, we will focus on recurrent neural networks (RNNs), a special kind of neural network that can process sequential data and learn long-term dependencies.

2.2.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a type of neural network that have feedback loops or cycles in their structure. This allows them to store and update their internal state over time, which can represent context or memory information. RNNs can handle variable-length input and output sequences, such as text, speech, or time series data. RNNs can also model complex temporal dynamics and non-linear relationships among sequential data.

The basic structure of an RNN is shown in Figure 2.1. It consists of an input layer, a hidden layer, and an output layer. The hidden layer contains one or more recurrent units or cells, which have a self-loop that connects their output to their input. The hidden state of each cell is updated at each time step based on the current input and the previous hidden state. The output layer produces an output based on the current hidden state at each time step.

RNNs can be trained using a technique called backpropagation through time (BPTT), which is an extension of the standard backpropagation algorithm for feedfor-

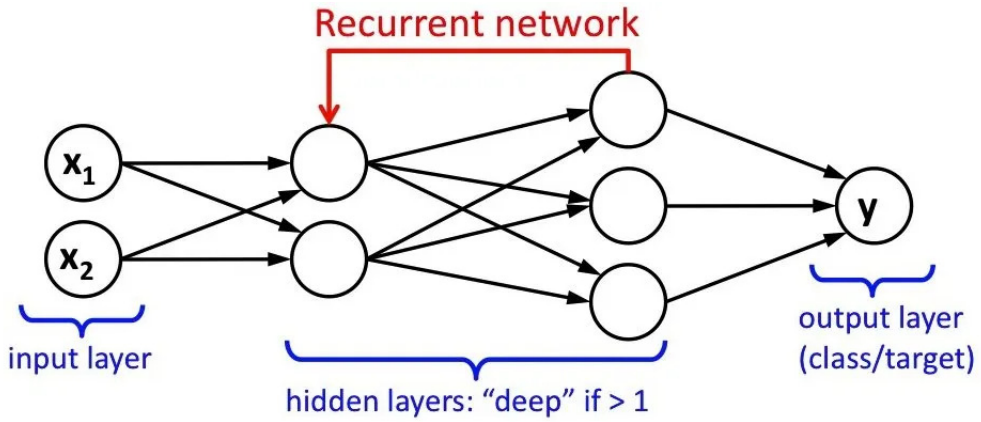

Figure 2.1: RNN Basic Structure

ward neural networks. BPTT involves unfolding the RNN into a feedforward network with multiple copies of the same cell, one for each time step. The error is then propagated backward from the output layer to the input layer through all the time steps, and the weights are updated accordingly.

2.2.3 RNN Advantages and Disadvantages

RNNs have many advantages, such as:

1. They can model sequential data of variable length and capture temporal dependencies and patterns.
2. They can share parameters across different time steps, which reduces the number of parameters and improves generalization.
3. They can generate output at each time step or only at the end of the sequence, depending on the task.

However, RNNs also have some limitations, such as:

1. They are prone to vanishing and exploding gradients, which makes them difficult to train and optimize.
2. They have a limited memory capacity and tend to forget long-term dependencies over time.

3. They are not efficient at parallelizing computation due to their sequential nature.

4. To overcome these limitations, various extensions and modifications of RNNs have been proposed, such as Long Short-Term Memory (LSTM), which is the main technique used in this report.

2.2.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network that can process sequential data and learn long-term dependencies. LSTM was proposed by Hochreiter and Schmidhuber in 1997 to address the problems of vanishing and exploding gradients that affect conventional RNNs. LSTM is designed to remember information for long periods of time by using gates to regulate the information flow.

2.2.5 LSTM Architecture

The basic structure of LSTM is composed of a chain of memory blocks, each containing a cell and three gates: input gate, forget gate and output gate. The cell is responsible for storing and updating the internal state of the block over time. The gates are sigmoid neural networks that control the flow of information into and out of the cell. The gates learn to selectively remember or forget the previous state, and to regulate the output of the current state.

The basic structure of LSTM is shown in Figure 2.2. It consists of an input layer, a hidden layer, and an output layer. The hidden layer contains one or more LSTM blocks, which have a self-loop that connects their output to their input. The hidden state of each block is updated at each time step based on the current input, the previous hidden state, and the previous cell state. The output layer produces an output based on the current hidden state at each time step.

2.2.6 Traning & Testing

To train an LSTM model, we need to provide a sequence of input data and a sequence of desired output data. The model will learn to map the input sequence to the output sequence by adjusting its weights through backpropagation through time (BPTT). BPTT involves unfolding the LSTM into a feedforward network with multiple copies of the same block, one for each time step. The error is then propagated backward

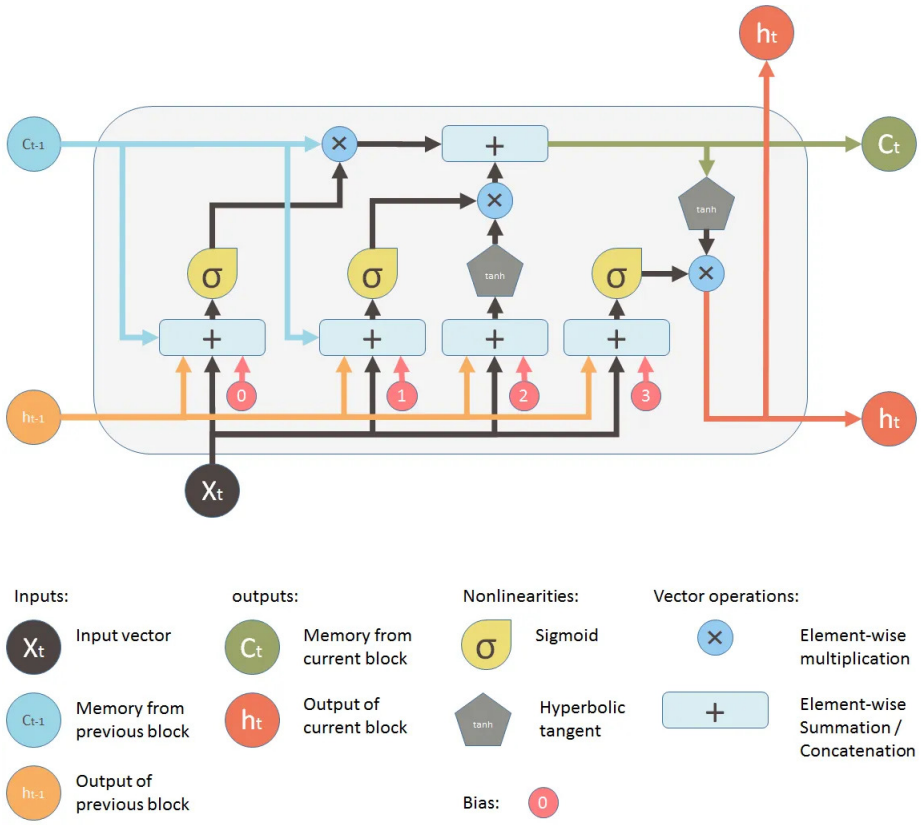

Figure 2.2: LSTM structure

from the output layer to the input layer through all the time steps, and the weights are updated accordingly.

To test an LSTM model, we need to provide a new sequence of input data and compare the predicted output sequence with the actual output sequence. We can use various metrics to evaluate the performance of the model, such as accuracy, precision, recall, F1-score, etc.

2.2.7 Applications of LSTM

LSTM can be used for various applications that involve sequential data, such as:

- Natural language processing: LSTM can be used for tasks such as text generation, sentiment analysis, machine translation, text summarization, question answering, etc.
- Speech recognition: LSTM can be used for tasks such as speech-to-text conversion, speaker identification, speech synthesis, etc.

- Time series forecasting: LSTM can be used for tasks such as stock market prediction, weather forecasting, traffic prediction, etc.

2.2.8 Why LSTM for Stock Prediction?

LSTM is suitable for stock prediction because it can:

- Handle variable-length input and output sequences, such as stock prices over different periods.
- Capture temporal dependencies and patterns among sequential data, such as trends, cycles, seasonality, etc.
- Remember or forget information selectively, depending on its relevance for prediction.
- Handle complex and non-linear relationships among sequential data, such as interactions between different stocks or factors.
- Generate output at each time step or only at the end of the sequence, depending on the task.

2.3 Markov Chain

The subsequent sections give the brief overview of Markov Chain.

2.3.1 What is a Markov Chain?

A Markov chain is a fundamental component of our analysis, and it is essential to understand the concept of a stochastic process in order to grasp its significance.

A stochastic process refers to the evolution of a random variable over time. Time can be categorized into two distinct forms: discrete and continuous. Discrete-time is countable, while continuous time encompasses an uncountable set.

In our study, we focus on the discrete-time stochastic process, which allows us to observe the system's state at specific time intervals. This simplification enables us to employ the power of a discrete-time stochastic process in constructing our Markov chain model.

In essence, a Markov chain is a specific type of discrete-time stochastic process that adheres to certain assumptions and definite probabilistic rules. It entails random variables transitioning from one state to another, relying solely on the present state, and governed by the Markov property.

Now, a discrete-time stochastic process is a Markov chain, if for t=0,1,2,3... and all states

$$P(X_{t+1} = S|X_t = S_t, X_{t-1} = S_{t-1}, ..., X_0 = S_0) = P(X_{t+1} = S|X_t = S_t) \qquad (2.1)$$

Here it can be seen from the above-mentioned eq.(2.1) that the future state depends only on the present state.

In summary, a Markov chain is a stochastic process characterized by random variables transitioning from one state to another based on specific assumptions and well-defined probabilistic rules, while adhering to the Markov property.

The Markov property is a simple yet powerful concept that states that the future outcomes of the process are independent of the past, given the present state. This property is closely associated with a memoryless process, as it relies solely on the current state and the randomness involved in transitioning to subsequent states. The left-hand side of the equation in the figure above illustrates this property, showcasing the conditional probability of future outcomes given the current state at time t.

2.3.2 Markov Chain Model

A Markov chain model is a stochastic model that represents random variables in a way that adheres to the Markov property. The primary objective of our model is to predict future states, with the only requirement being knowledge of the current state.

To construct a Markov chain, there are three essential components that we need to consider:

States: In the context of the dynamical system represented by eq.(2.2) below, "states" encompass all possible occurrences or configurations within the state-space 's'.

$$S = [s_1, s_2, s_3, \ldots, s_n]^T \qquad (2.2)$$

Initial state distribution: The initial state distribution, denoted by the column vector 'q' as shown in eq.(2.3), represents the probability distribution of the starting

state.

$$Q = [q_1, q_2, ...q_s]^T \tag{2.3}$$

State transition probabilities: The state transition probabilities, represented by the s-by-s matrix 'P' as depicted in eq.(2.4), quantify the likelihood of transitioning from one state to another within the system. Each element of the matrix corresponds to the transition probability between two states.

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \dots & \dots & \dots & \dots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{bmatrix} \tag{2.4}$$

By defining these components, we can establish a Markov chain model that allows us to analyze and make predictions based on the probabilistic transitions between states.

2.3.3 Properties of Markov Chain

- State space: The set of all possible states that the Markov chain can take.
- Initial distribution: The probability distribution of the initial state of the Markov chain.
- Transition probability: The probability of moving from one state to another in one step.
- Transition matrix: The matrix that contains all the transition probabilities between all pairs of states.
- Stationary distribution: The probability distribution of the state that does not change over time.
- Ergodicity: The property that ensures that the Markov chain will eventually reach a stationary distribution regardless of the initial distribution.
- Irreducibility: The property that ensures that every state can be reached from any other state in a finite number of steps.

- Periodicity: The property that measures how often a state repeats itself in a cycle.
- Recurrence: The property that measures how likely a state will be visited again after leaving it.
- Absorption: The property that measures whether a state can be left once entered.

2.3.4 Applications of Markov Chain

Markov chains have many applications in various fields, such as:

1. Natural language processing: Markov chains can be used to generate text or speech based on statistical models of language, such as n-grams or hidden Markov models.
2. Computer science: Markov chains can be used to design algorithms for optimization, simulation, sampling, ranking, etc., such as PageRank, Metropolis-Hastings, Gibbs sampling, etc.
3. Economics: Markov chains can be used to model economic processes that involve uncertainty and risk, such as stock prices, exchange rates, consumer behavior, etc.

2.3.5 Why Markov Chain for Stock Prediction?

Markov chain is suitable for stock prediction because it can:

1. Handle variable-length input and output sequences, such as stock prices over different periods.
2. Capture temporal dependencies and patterns among sequential data, such as trends, cycles, seasonality, etc.
3. Predict future states or events based on past states or events with probabilistic models.
4. Handle uncertainty and risk in stock market dynamics with stochastic models.

5. Generate output at each time step or only at the end of the sequence, depending on the task.

Chapter 3

Methodology

The report utilizes time series data tracking daily price changes for the Royal Bank of Canada stock, which is listed on the Toronto Stock Exchange (TSX). This dataset covers the daily closing price of Royal Bank of Canada (RY) stock by market value from 1^{st} March 2019 to 31^{st} March 2023. This dataset has been downloaded from public available web source called **Yahoo Finance**. In particular, the paper draws upon a snapshot of this dataset for analysis.

Table 3.1: Stock Price Data

Date	Open	High	Low	Close	Adj Close	Volume
2019-03-01	78.32	78.53	77.75	77.81	66.418854	963500
2019-03-04	77.82	78.17	77.31	77.74	66.359108	865000
2019-03-05	77.74	77.89	77.4	77.75	66.367645	584500
2019-03-06	77.72	77.88	77.06	77.23	65.923759	663000
2019-03-07	77.2	77.31	76.28	76.67	65.488449	1042300

The following is the description of above columns:

1. Date: This column represents the date on which the sample of the stock market data is extracted. The date corresponds to the day on which the different values of the stock sample are recorded.

2. Open: This column represents the opening price of the stock on a particular trading day. The stock market has fixed starting and closing times during which stocks are traded.

3. High: This column represents the highest value that the stock reaches during a trading day. The stock price can fluctuate greatly during the trading day, and

the high value represents the peak price for that day.

4. Low: This column represents the lowest value that the stock reaches during a trading day. As with the high value, the low value reflects the fluctuation in the stock price during the trading day.

5. Close: This column represents the closing price of the stock on a particular trading day. Similar to the opening price, the closing price is fixed by the stock market's starting and closing times.

6. Adj Close: This column represents the adjusted closing price of the stock after accounting for any dividends and stock splits. The adjusted closing price is an important metric for financial analysis as it reflects the true economic value of the stock. Further explanation of the calculation of the adjusted closing price can be found in the appendix.

7. Volume: This column represents the total number of shares traded on a particular trading day. The volume metric provides a measure of the level of market activity for a particular stock, and can be used to identify trends and patterns in stock price movements.

In this study, we will employ the above-mentioned dataset in figure 3.1 of daily closing prices of Royal Bank of Canada (RY) shares from March 1st, 2019, to March 31st, 2023. Our objective is to investigate the effectiveness of LSTM and Markov Chain models in forecasting future states and prices of the RY stock. By utilizing a four-layer LSTM architecture and the Root Mean Square Error (RMSE) as the loss function, we aim to capture temporal dependencies and patterns to predict closing prices. Furthermore, we will construct a three-state Markov Chain to estimate the transition matrix, leveraging metrics such as steady-state distribution and mean hitting times. This research seeks to provide valuable insights into stock price dynamics and enhance decision-making in financial investments by combining these modeling approaches. Preliminary results indicate promising outcomes, demonstrating the potential of LSTM modeling and Markov Chain analysis in stock market prediction.

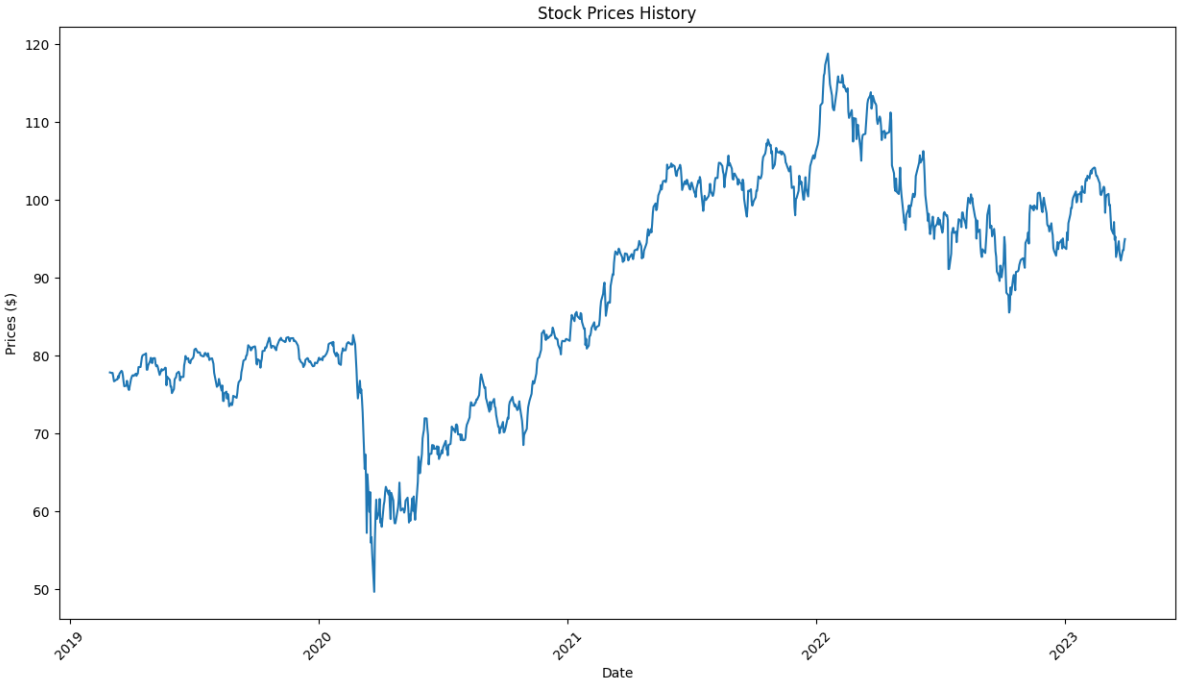

Figure 3.1: RY Daily Closing Price Trend (Mar 2019 - Mar 2023)

3.1 Part 1: LSTM Modeling

3.1.1 Data Preprocessing

The dataset used in this study is considered to be mostly clean, with no missing values present. However, in order to improve convergence, achieve better performance, and avoid numerical instabilities, it is necessary to normalize the stock data. To this end, we have employed the MinMaxScaler normalization technique, which scales the closing stock prices in the dataset to a range of 0 to 1.

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

The above Eq. (3.1) is used by the MinMaxScaler technique to normalize the data, where x represents the original feature value, $x_{\min}$ denotes the minimum value of the feature across the dataset, and $x_{\max}$ represents the maximum value of the feature across the dataset. The resulting x_{scaled} value falls within the range of 0 and 1, ensuring that all features are scaled consistently.

3.1.2 Feature Selection

This step encompasses the selection of pertinent features to be utilized as inputs for the LSTM model. In light of the provided data, we have opted to employ the daily closing stock price as the input data. Given the closing price being a singular value, to enhance the accuracy of our closing price predictions, we have employed a sliding window approach spanning 60 days for each sample. Additionally, we have designated the value corresponding to the following day as the output label in our model.

3.1.3 Train/Test Split

In the train-test split process of our LSTM model, we have employed a partitioning strategy to effectively allocate the available data. Specifically, the dataset consisting of stock closing prices spanning from 1st March 2019 to 31st March 2023 has been divided into two sets: a training set and a testing set. The training set encompasses the initial 80 percent of the data, extending until 7th June 2022, and is utilized to train the model. By training the model on this substantial portion of the data, it can learn the underlying patterns and relationships within the training set. The remaining 20 percent of closing prices, covering the period from 8th June 2022 to 31st March 2023, form the testing set, which is used to assess the model's performance and evaluate its predictive accuracy. By evaluating the model on independent testing data, we can gain valuable insights into its ability to generalize to unseen data and make accurate predictions in real-world scenarios. This evaluation process allows us to measure the model's effectiveness and determine its predictive capabilities.

3.1.4 Model Architecture Design

In the LSTM architecture design phase, we aim to construct an effective model that can analyze and capture temporal patterns in the data. The chosen architecture involves a sequential model, which is a linear stack of layers specifically designed for sequential data analysis.

To begin, we add an LSTM layer to the model. This layer is crucial for learning complex relationships within the sequential data. By including 100 network units in this layer, we provide the model with the capacity to capture intricate temporal dependencies. Additionally, we set the parameter related to preserving temporal information to ensure that the LSTM layer generates an output sequence of the

same length as the input. This preservation of temporal information is essential for retaining the sequential context and enabling the subsequent layers to make informed decisions based on the learned dependencies.

We introduce another LSTM layer with the same configuration of 100 network units to further enhance the model's understanding of the data. However, this time, we set the parameter related to preserving temporal information to false. By doing so, this layer focuses on summarizing and consolidating the learned information from the previous LSTM layer, providing a compressed representation of the sequential data.

Next, we include a densely connected neural network layer with 25 network units. This layer serves as a bridge between the LSTM layers and helps the model capture higher-level representations and abstract features. The densely connected layer enables the model to extract more meaningful information from the outputs of the LSTM layers.

Finally, we add a densely connected layer with a single network unit as the output layer. This layer provides the final prediction or output of the LSTM model. By utilizing these layers in the architecture, we create an LSTM model that can effectively analyze and model sequential data, facilitating accurate predictions or classifications based on the provided inputs. Following table shows the model configurations:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 100)           40800

 lstm_1 (LSTM)               (None, 100)               80400

 dense (Dense)               (None, 25)                2525

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 123,751
Trainable params: 123,751
Non-trainable params: 0
```

3.1.5 Training LSTM Model

In this step, we trained the LSTM model to make accurate predictions. Before diving into the training process, let's briefly discuss the environment we utilized for this task.

Table 3.2: Environment Details

System	Windows
Release	10
Version	10.0.22621
Machine	AMD64
Python Version	3.10.11
Python Implementation	CPython
Python Compiler	MSC v.1929 64 bit (AMD64)

Moving on to the training process, one crucial aspect is the selection of an appropriate optimizer and loss function. The optimizer determines how the model parameters (weights and biases) are updated during training, while the loss function measures the discrepancy between the model's predictions and the actual values.

Among the various optimizer options available, we chose Adaptive Moment Estimation (Adam) due to its advantages in terms of adaptive learning rates, momentum optimization, and bias correction. Adam is widely used and offers fast convergence, robustness to hyperparameters, and improved optimization performance.

As for the loss function, we opted for Mean Squared Error (MSE). MSE calculates the average squared difference between the predicted and actual values, allowing us to assess the model's performance based on the magnitude of these errors.

With the optimizer and loss function selected, we commenced the training process by fitting the model with the training set. During training, we utilized a batch size of 1 and trained the model for 3 epochs.

Using a batch size of 1 implies that the model processed one training example at a time, allowing it to focus on the specific characteristics of each individual example. Training for 3 epochs involved iterating over the entire dataset three times. In each epoch, the model made predictions, calculated the loss, and updated its parameters based on each individual example. This iterative process was repeated for three

complete passes through the dataset, enabling the model to learn and improve its performance over time.

By carefully selecting the optimizer, loss function, batch size, and number of epochs, we aimed to optimize the training process and enhance the LSTM model's ability to make accurate predictions.

3.1.6 Model Evaluation

In the final step, we evaluate the performance of our trained LSTM model by applying it to the test set and utilizing the root mean square error (RMSE) metric.

To begin, we predict the values of the test set by applying our model. Subsequently, we employ the inverse_transform method to denormalize the stock prices. This denormalization is achieved using the following eq.(3.2):

$$\text{original\_value} = \text{scaled\_data} \times (\text{max\_val} - \text{min\_val}) + \text{min\_val} \tag{3.2}$$

This denormalization is achieved using the above-mentioned eq.(3.2) where scaled_data represents the normalized data, while max_val and min_val denote the maximum and minimum values of the original stock prices, respectively.

Finally, we calculate the RMSE on the predicted values to quantify the difference between the predicted prices and the actual values.

3.2 Part 2: Markov Chain Modeling

3.2.1 Markov Chain Modeling of Stock Price States

To model the dynamics and dependencies of stock price states over time, we employ a Markov chain, a stochastic process that describes state transitions based on probabilities. The Markov chain assumes that the future state of the system depends solely on the current state, disregarding past states, known as the Markov property or memorylessness.

Applying a Markov chain to stock price prediction involves two steps: state discretization and transition matrix estimation.

3.2.2 State Discretization of Stock Price

In state discretization, we simplify the prediction task by categorizing continuous price values into three states: high, low, and stable. By doing so, we reduce noise and uncertainty, enabling us to focus on the general price trend. To establish the state boundaries, various methods can be used. One approach involves defining thresholds based on percentage changes from the previous day's closing price. For instance, if the percentage change lies within a specified threshold, such as 1 percent, the price is classified as stable. Values exceeding 1 percent are categorized as high, while those below 1 percent are labeled as low. Alternatively, thresholds can be determined using mean plus/minus one standard deviation or other technical indicators such as Bollinger Bands or Moving Average Convergence Divergence (MACD). The choice of thresholds depends on preferences and domain knowledge.

As we have used the stock price data of Royal Bank of Canada (RY) from March 2019 to March 2023, employing daily closing prices as input data. Percentage thresholds are utilized for our Markov chain modeling.

$$\text{High state} = \left(\frac{\text{current price} - \text{previous price}}{\text{previous price}}\right) > \text{threshold} \tag{3.3}$$

The eq.(3.3) represents a condition to determine whether the system is in a "High state." It compares the percentage change in the current price relative to the previous price with a given threshold value. If the percentage change is greater than the threshold, the condition evaluates to true, indicating that the system is in a "High state." Otherwise, it evaluates to false.

$$\text{Low state} = \left(\frac{\text{current price} - \text{previous price}}{\text{previous price}}\right) < -\text{threshold} \tag{3.4}$$

The eq.(3.4) represents a condition to determine whether the system is in a "Low state." Similar to the previous equation, it calculates the percentage change in the current price relative to the previous price. However, in this case, it compares the percentage change with the negative of the given threshold value. If the percentage change is less than the negative threshold, the condition evaluates to true, indicating that the system is in a "Low state." Otherwise, it evaluates to false.

$$\text{Stable state} = \left(\frac{\text{current price} - \text{previous price}}{\text{previous price}}\right) < \text{threshold}$$
$$\text{and} \quad \left(\frac{\text{current price} - \text{previous price}}{\text{previous price}}\right) > -\text{threshold} \tag{3.5}$$

The eq.(3.5) represents a condition to determine whether the system is in a "Stable state." Similar to the previous equations, it calculates the percentage change in the current price relative to the previous price. In this case, it compares the percentage change and check that it is within given threshold value in either direction. If the percentage change is within the threshold, the condition evaluates to true, indicating that the system is in a "Stable state." Otherwise, it evaluates to false.

By utilizing these thresholds, we convert the continuous price values into discrete state values. For instance, the first ten prices and their corresponding states are presented in Table 3.2. This process is repeated for all prices in the dataset to obtain a sequence of states representing the discretized stock price data. Here we have assumed a threshold of 0.25 percent.

Table 3.3: State Discretization of Stock Price Using Stock Price

Date	Close	Percentage Change	State
2019-03-01	77.81	N/A	N/A
2019-03-04	77.74	-0.08	Stable
2019-03-05	77.75	0.012	Stable
2019-03-06	77.23	-0.66	Low
2019-03-07	76.67	-0.72	Low
2019-03-08	76.72	0.06	Stable
2019-03-11	76.91	0.26	High
2019-03-12	76.91	-0.01	Stable
2019-03-13	77.33	0.55	High
2019-03-14	77.11	-0.29	Low

We can repeat this process for all the prices in our data set and obtain a sequence of states representing our discretized stock price data.

3.2.3 Transition Matrix Estimation

To model the dynamics and dependencies of the stock price states over time, we need to estimate the transition probabilities between the states using a Markov chain. A

Markov chain is a stochastic process that describes how a system changes from one state to another, based on some probabilities. The main assumption of a Markov chain is that the future state of the system only depends on the current state, and not on the past states. This is known as the Markov property or memorylessness.

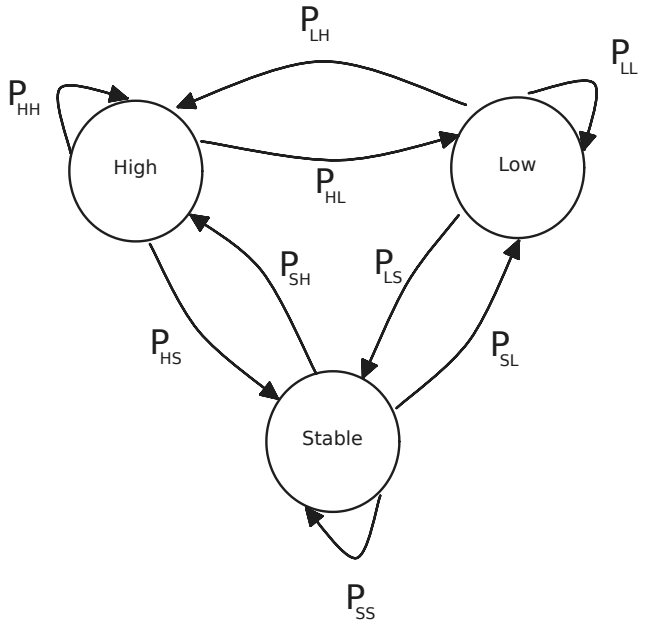

Figure 3.2: Markov Chain States Transitions

To estimate the transition probabilities between the states using a Markov chain, we need to count how many times each state is followed by another state in our sequence of states. For example, we can count how many times a high state is followed by a low state, or how many times a medium state is followed by a high state. Then, we can divide these counts by the total number of occurrences of each state, to get the probability of going from one state to another. For example, we can divide the number of times a high state is followed by a low state by the number of times a high state occurs, to get the probability of going from high to low. We can repeat this process for all possible pairs of states, and obtain a matrix of transition probabilities. This matrix represents the Markov chain that models the stock price states.

To illustrate this step, we will use the sequence of states that we obtained from our discretized stock price data. We will count how many times each pair of states occurs in our sequence and divide them by the total number of occurrences of each state. For example, we can count how many times S is followed by H, L or S in our

sequence and divide them by the total number of S in our sequence. We can do this for all pairs of states and obtain the following matrix:

Considering the same threshold of 0.25%. We will get the following transition probabilities if we only use them until the training data date(7th June,2022).

$$P = \begin{bmatrix} P_{HH} & P_{HL} & P_{HS} \\ P_{LH} & P_{LL} & P_{LS} \\ P_{SH} & P_{SL} & P_{SS} \end{bmatrix} = \begin{bmatrix} 0.66 & 0.34 & 0.35 \\ 0.25 & 0.58 & 0.34 \\ 0.09 & 0.08 & 0.31 \end{bmatrix} \tag{3.6}$$

This matrix shows the transition probabilities between the three states: high (H), low (L) and stable (S). For example, $P_{HH} = 0.66$ means that if the current state is high, there is a 66% chance that the next state will be high as well. Similarly, $P_{LS} = 0.34$ means that if the current state is stable, there is a 34% chance that the next state will be low.

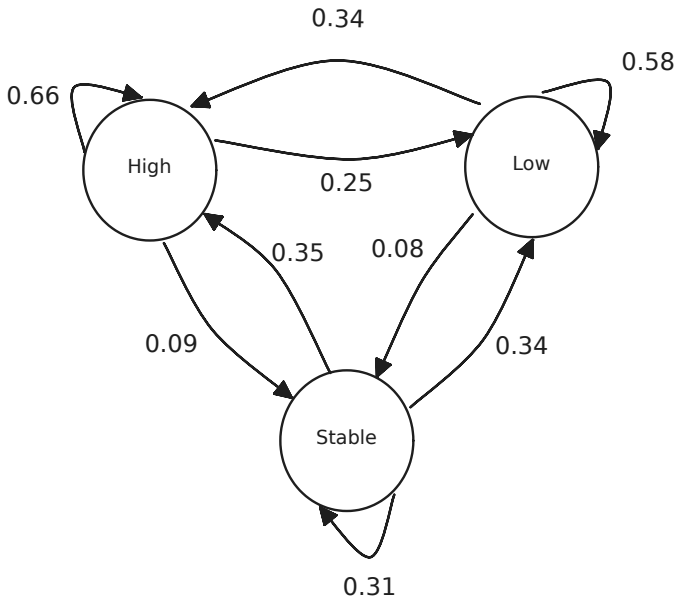

Figure 3.3: Markov Chain States Transition Probabiliites

Chapter 4

Experiments & Results

4.1 LSTM

After training the LSTM model and evaluating its performance on the test set, we obtained the following results.

The model achieved an RMSE value of 0.596, indicating the average difference between the predicted stock prices and the actual values. A lower RMSE value signifies a better predictive accuracy of the model.

The predicted stock prices were then denormalized using the inverse_transform method to obtain the original values. The denormalization equation eq. (4.1) utilized was:

$$\text{original\_value} = \text{scaled\_data} \times (\text{max\_val} - \text{min\_val}) + \text{min\_val} \tag{4.1}$$

By denormalizing the predicted values, we transformed them back to their original scale, providing a more meaningful representation of the stock prices.

To visually assess the performance of the LSTM model, we plotted the predicted stock prices against the actual values on a line chart. The chart depicted the trends and patterns captured by the model and allowed for visual comparison with the real stock prices.

The line chart(4.1) showcases the model's ability to capture the underlying trends and patterns in stock prices. The predicted values exhibit a remarkable similarity to the actual values, indicating the model's capability to make accurate predictions. The closeness of the green line to the orange line signifies the model's effectiveness in capturing the variations and dynamics of the stock prices.

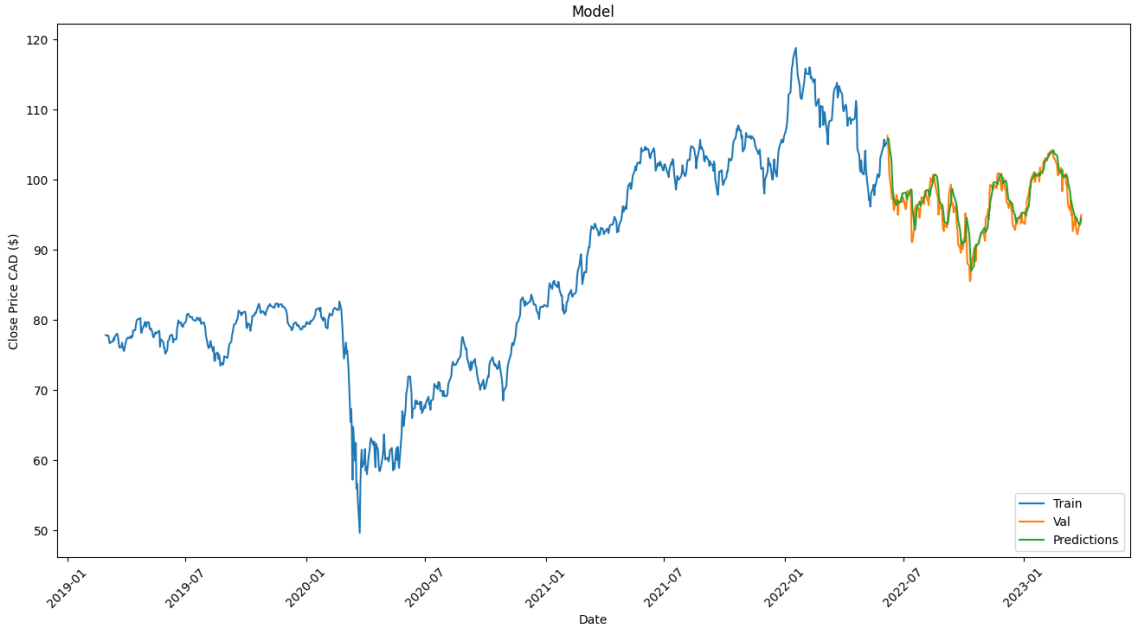

Figure 4.1: RY Predicted Daily Closing Price Trend (Mar 2019 - Mar 2023)

Furthermore, we analyzed the residuals, which are the differences between the predicted and actual values. The residual plot provided insights into the model's ability to capture the underlying variations in the stock prices. A random distribution of residuals around zero indicated that the model captured the patterns effectively, while any systematic deviations could highlight areas for improvement.

The following table represents the residuals that we have calculated based on the trained model.

Table 4.1: Residual Statistics

Statistic	**Value**
Count	205
Mean	0.59
Standard Deviation	1.65
Minimum	-3.51
25% Percentile	-0.35
50% Percentile (Median)	0.44
75% Percentile	1.47
Maximum	7.35

Based on the calculated residuals, we obtained the following statistical measures: The dataset consists of 205 residuals, representing the discrepancies between the

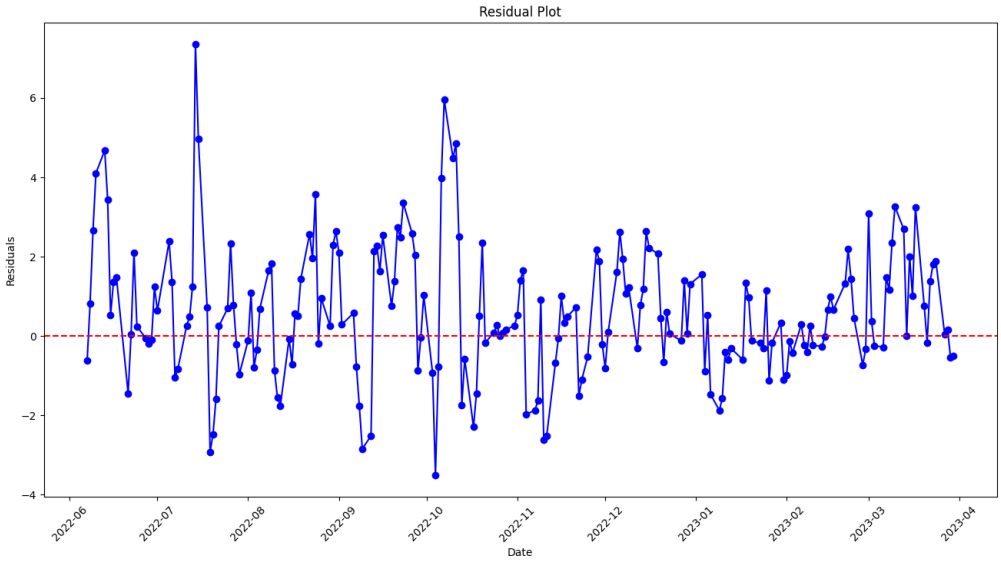

Figure 4.2: RY Daily Closing Price Residual Trend

model's predictions and the actual values. The average residual value is 0.59, indicating relatively accurate predictions. The residuals exhibit moderate variability with a standard deviation of 1.655. The minimum and maximum residuals are -3.51 and 7.35, respectively, reflecting the largest deviations from the actual values. The 25th percentile and 75th percentile values of -0.35 and 1.47 provide insight into the lower and upper ranges of deviations. Overall, the residuals capture the model's prediction errors and their statistical characteristics.

By examining the residual plot and the statistical measures, we can infer several observations. The random distribution of residuals around zero suggests that the model has successfully captured the underlying variations in the stock prices. The close-to-zero mean indicates that, on average, the model's predictions are aligned with the actual values. However, the moderate standard deviation indicates that there are some instances where the model deviates from the actual values, indicating areas where further improvement could be made.

Overall, the LSTM model demonstrated promising performance in predicting the stock prices based on the selected features and the employed architecture. However, further analysis and comparison with alternative models or techniques may be necessary to assess the model's competitiveness and determine its practical utility in real-world scenarios.

Other metrics, such as mean absolute error (MAE), can also be considered to

evaluate the model's performance. MAE provides a measure of the average absolute difference between the predicted and actual values, offering additional insights into the accuracy of the model's predictions.

In addition to evaluating the model's performance, it is important to consider its limitations and potential sources of error. These may include the assumption of linearity in the data, sensitivity to hyperparameters, potential overfitting or underfitting, and the influence of external factors or unforeseen events on stock price dynamics.

Further research and analysis could involve optimizing the model's hyperparameters, exploring alternative architectures, incorporating additional features or indicators, or testing the model on different datasets or time periods. Such investigations can contribute to refining the model and enhancing its predictive capabilities in the field of stock price prediction.

4.2 Markov Chain

In this section, we present the results obtained from our experiments using the proposed Markov chain modeling approach for stock price prediction. We utilize the transition matrix to calculate the steady-state distribution matrix and mean hitting times.

The probabilities within our transition matrix are dependent on the criteria we employ for counting state transitions. In our methodology, we have chosen to classify states based on the percentage change from the previous day's closing price. This percentage-based approach provides us with the flexibility to employ various thresholds for calculating the transition matrix, but in our experiments, after working with multiple thresholds, we resorted to 0.1% as we want to get a clear view of transition probabilities. The higher the threshold, the more chance for transition probabilities to stay within a stable state. In our experiments, we investigate three thresholds: 0.1%, 0.5%, and 1%. Considering that we utilize daily closing prices, employing a 1% threshold proves sufficient as it captures the majority of state changes as stable.

In the following subsection, we present the transition matrices, steady-state distribution matrices, and mean hitting times for the different thresholds.

Threshold 0.1%

When applying a threshold of 0.1%, the resulting transition matrix is as follows:

$$P = \begin{bmatrix} 0.67 & 0.35 & 0.47 \\ 0.28 & 0.60 & 0.38 \\ 0.05 & 0.05 & 0.15 \end{bmatrix} \tag{4.2}$$

Utilizing this transition matrix (4.2), we observe the convergence of probabilities in the steady-state distribution matrix. The plot below depicts the number of iterations required to reach convergence. Notably, after approximately five iterations, the probabilities stabilize. The convergence pattern is evident: the red line representing the high state converges to around 0.5%, the blue line representing the downstate converges to around 0.43%, and the stable state, denoted by the green line, converges to 0.05%.

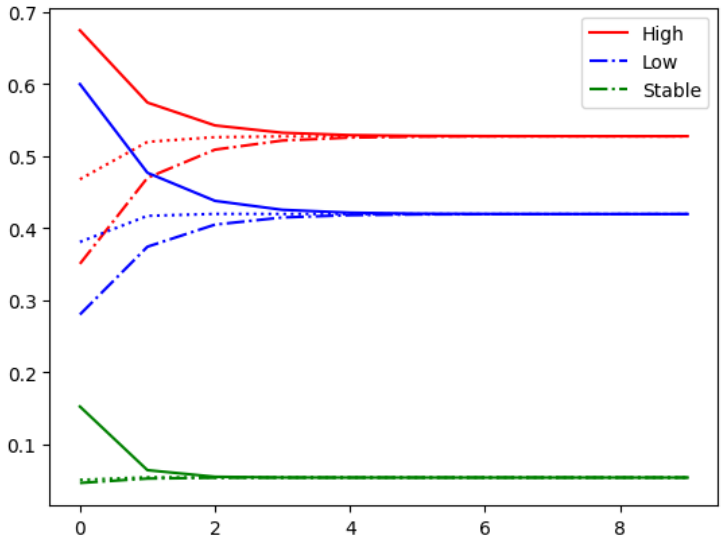

Figure 4.3: Convergence of Steady-State Transition Probabilities (0.1% Threshold)

Furthermore, we analyze the mean hitting times associated with this threshold. Upon examination, we find that regardless of the initial state (high, low, or stable), the stock tends to remain in the same state throughout the day. This observation is supported by the mean hitting times matrix (4.3), which reveals that all states have mean hitting times greater than 1.

$$M = \begin{bmatrix} 1.16 & 0.44 & 0.71 \\ 0.35 & 1.15 & 0.60 \\ 0.07 & 0.08 & 1.06 \end{bmatrix} \tag{4.3}$$

Based on these results, we can infer that when employing a 0.1% threshold, the stock predominantly exhibits stability throughout the trading day, rarely experiencing significant upward or downward movements.

Threshold 0.5%

When utilizing a threshold of 0.5%, the resulting transition matrix is as follows:

$$P = \begin{bmatrix} 0.63 & 0.29 & 0.22 \\ 0.23 & 0.56 & 0.18 \\ 0.14 & 0.15 & 0.60 \end{bmatrix} \tag{4.4}$$

Examining this transition matrix(4.4), we observe the probabilities associated with transitioning between states. The matrix reveals that the stock tends to exhibit a higher probability of remaining in the same state, particularly in the stable state, while the probabilities of transitioning to other states are relatively lower.

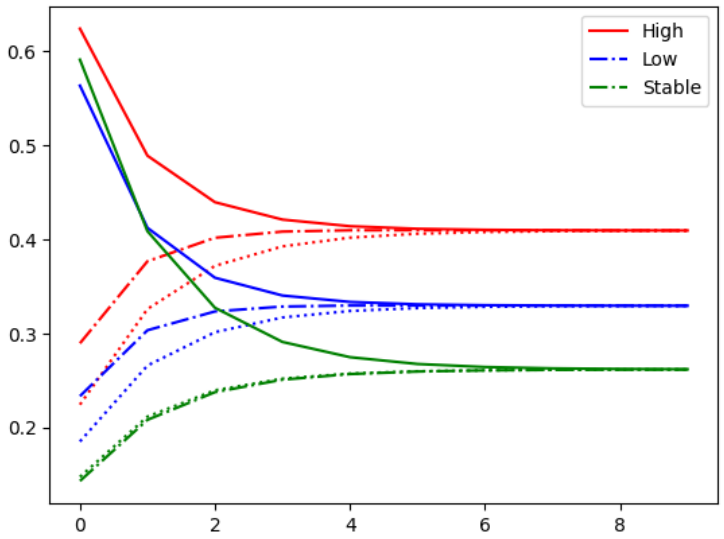

Figure 4.4: Convergence of Steady-State Transition Probabilities (0.5% Threshold)

To visualize the convergence of probabilities, we present a plot showcasing the number of iterations required for the steady-state distribution matrix to converge.

In this case, we observe that the probabilities reach convergence after approximately six iterations. The convergence pattern can be observed in the plot, with the red line representing the high state stabilizing around 0.41%, the blue line representing the downstate stabilizing around 0.33%, and the green line denoting the stable state remaining at approximately 0.26%.

Additionally, we analyze the mean hitting times associated with the 0.5% threshold. The mean hitting times matrix (4.5) provides insights into the average number of steps required for the stock to transition between different states. For this threshold, the mean hitting times matrix is as follows:

$$M = \begin{bmatrix} 1.13 & 0.37 & 0.32 \\ 0.30 & 1.12 & 0.27 \\ 0.20 & 0.21 & 1.08 \end{bmatrix} \tag{4.5}$$

Threshold 1%

Using a threshold of 1%, the resulting transition matrix is as follows:

$$P = \begin{bmatrix} 0.60 & 0.28 & 0.09 \\ 0.30 & 0.53 & 0.08 \\ 0.10 & 0.19 & 0.83 \end{bmatrix} \tag{4.6}$$

Analyzing this transition matrix(4.6), we observe the probabilities associated with transitioning between states. The matrix reveals that when employing a 1% threshold, the stock exhibits a higher tendency to remain in the same state, particularly in the stable state, as the probabilities for transitioning to other states are relatively low.

To visualize the convergence of probabilities, we present a plot depicting the number of iterations required for the steady-state distribution matrix to converge. Upon examination, we find that the probabilities stabilize after more than ten iterations. The convergence pattern is evident in the plot: the red line representing the high state converges to around 0.29%, the blue line representing the downstate converges to around 0.26%, and the green line representing the stable state stabilizes at approximately 0.45%.

Comparing these steady-state probabilities with the 0.1% and 0.5% thresholds, we observe that as the threshold increases, the probabilities for each state decrease gradually. This implies that employing a higher threshold, such as 1%, reduces the likelihood of state transitions and price fluctuations compared to the lower thresholds.

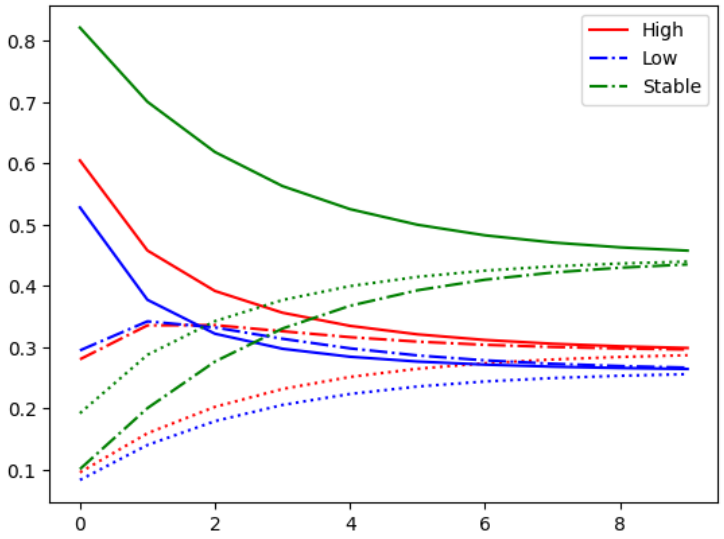

Figure 4.5: Convergence of Steady-State Transition Probabilities (1% Threshold)

Additionally, we analyze the mean hitting times associated with the 1% threshold. The mean hitting times matrix(4.7) provides insights into the average number of steps required for the stock to transition between different states. For this threshold, the mean hitting times matrix is as follows:

$$M = \begin{bmatrix} 1.11 & 0.33 & 0.13 \\ 0.34 & 1.12 & 0.12 \\ 0.18 & 0.25 & 1.03 \end{bmatrix} \tag{4.7}$$

Comparing these mean hitting times with the 0.1% and 0.5% thresholds, we observe that the mean hitting times for each state are not that drastically different.

In summary, when employing a 1% threshold, the stock demonstrates a higher probability of staying in a particular state, particularly the stable state, with relatively lower state transition probabilities and reduced mean hitting times. It makes more sense to go with the lower thresholds i-e 0.1% as we want to visible state transitions rather than staying in a stable state.

4.3 Combining LSTM and Markov Chain

Having obtained promising results from both the LSTM model and the Markov chain approach for stock price prediction, it is valuable to explore the potential benefits of combining these two methodologies. By integrating the strengths of both approaches,

we can potentially improve the accuracy and reliability of the predictions.

There are multiple ways in which Markov chain can be combined with LSTM. Here we are presenting two following approaches:

1. Probabilistic long-term probabilities of LSTM Predictions Using Markov Chain
2. Next-Day Probabilities Using Markov Chain for LSTM Predictions

4.3.1 Probabilistic long-term probabilities of LSTM Predictions Using Markov Chain

The LSTM model captures complex temporal dependencies and patterns in the data, allowing it to make accurate predictions based on historical information. On the other hand, the Markov chain approach provides insights into the probabilistic behavior and transitions between different states of stock prices.

To combine the LSTM and Markov chain results, we followed the approach outlined in the previous sections for each methodology. Firstly, we obtained predicted values from the LSTM model. These predicted values were then utilized in conjunction with the Markov chain method, using a threshold of 0.1% for state transitions.

By training the Markov chain on the predicted values from the LSTM model, we obtained the following transition matrix(4.8):

$$P = \begin{bmatrix} 0.69 & 0.34 & 0.45 \\ 0.30 & 0.61 & 0.45 \\ 0.01 & 0.04 & 0.09 \end{bmatrix} \tag{4.8}$$

This matrix(4.8) shows the transition probabilities between the three states: high (H), low (L) and stable (S). For example, $P_{HH} = 0.69$ means that if the current state is high, there is a 69% chance that the next state will be high as well. Similarly, $P_{LH} = 0.30$ means that there is a 34% chance that the next state will be low given current state is high. Now after obtaining the transition matrix, we can utilize this matrix in two ways:

Subsequently, we performed matrix multiplication multiple times to obtain the steady-state matrix(4.9), yielding the following result:

$$S = \begin{bmatrix} 0.53 & 0.53 & 0.53 \\ 0.44 & 0.44 & 0.44 \\ 0.03 & 0.03 & 0.03 \end{bmatrix} \tag{4.9}$$

The convergence of the steady-state transition probabilities, trained on the predicted LSTM model data using the 0.1% threshold, is visualized below:

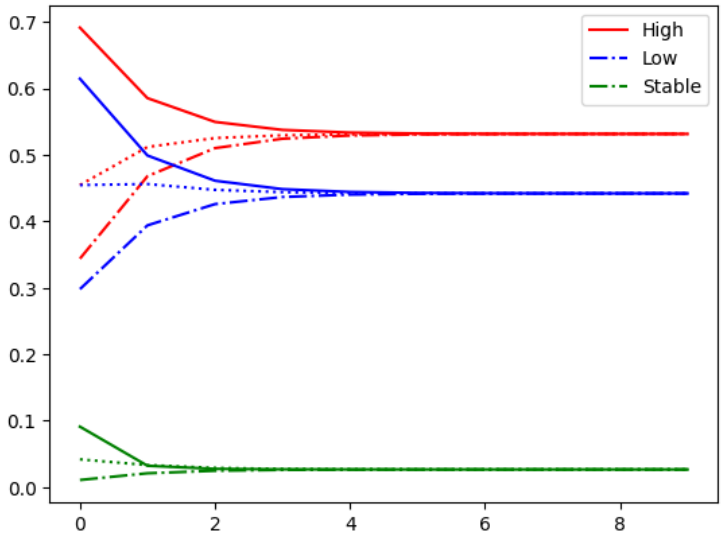

Figure 4.6: Convergence of Steady-State Transition Probabilities Trained on Predicted LSTM Model Data

Upon careful examination of the steady-state transition probabilities, a significant observation emerges: there is a notable probability (53%) of remaining in the high state for consecutive time steps. This finding suggests that the stock prices, as predicted by the LSTM model, tend to exhibit an overall upward trend. Probabilistically, this implies that more than half of the days are likely to experience a positive change of more than 0.1% in the stock prices.

For example, we can utilize the LSTM-predicted stock prices to calculate the percentage changes between consecutive time steps. These percentage changes can then be used to determine the appropriate state transitions and update the transition matrix of the Markov chain accordingly. This integration enables the Markov chain to reflect the temporal dynamics captured by the LSTM model and adjust its probabilistic behavior accordingly.

4.3.2 Next-Day Probabilities Using Markov Chain for LSTM Predictions

We can also use the earlier transition matrix(4.2) with 0.1% threshold obtained using the historical dataset to predict the next-day probability which will give us the

probabilities of staying in the particular state for the next day only.

The reason for this is that the matrix itself predicts the next time step. P^2 gives us the probability of two-time steps in the future. P^3 gives the probability of three-time steps in the future, and so on. It is this constant self-matrix multiplication that produces the future outcomes of the Markov model. However, we must remember that we ought to multiply it by the column vector q denoting the initial conditions as either high, low, or stable. Assuming, the current day is in high state, represented by q vector(4.10).

$$q = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{4.10}$$

Now using the transition matrix(4.2) and initial state vector(4.10), and multiplying these together give us the probabilities of the next day.

$$\begin{bmatrix} 0.67 & 0.35 & 0.47 \\ 0.28 & 0.60 & 0.38 \\ 0.05 & 0.05 & 0.15 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\intercal = \begin{bmatrix} 0.67 & 0.28 & 0.05 \end{bmatrix}^\intercal \tag{4.11}$$

This vector shows that if the current state is high, there is a 67% chance that the next state will remain as high, a 28% chance that it will be low, and a mere 5% chance that it will go to a stable state which is there will change of 0.25% in either direction. The is only valid for the next day as we are using the transition matrix as it is. Similarly, if we multiply the transition matrix two times, it will give us the probability of the third day. Using the same above example.

$$\begin{bmatrix} 0.67 & 0.35 & 0.47 \\ 0.28 & 0.60 & 0.38 \\ 0.05 & 0.05 & 0.15 \end{bmatrix}^2 \times \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\intercal = \begin{bmatrix} 0.55 & 0.34 & 0.11 \end{bmatrix}^\intercal \tag{4.12}$$

We can repeat this process for multiple steps ahead to get a sequence of predicted states for the future stock price.

Now we can use these probabilities in conjunction with predictions of LSTM which will give us both ends of stock price movement and confidence of staying in the particular state using two distinct approaches which focus on two different aspects.

Furthermore, by combining the LSTM and Markov chain results, we can generate more reliable predictions by taking advantage of the strengths of each methodology. The LSTM model can capture the intricate patterns and trends in the stock prices,

while the Markov chain provides a framework for modeling the probabilistic behavior and state transitions.

Chapter 5

Conclusion & Future Work

In this study, we investigated the application of LSTM and Markov chain modeling approaches for stock price prediction.We conducted experiments using historical stock price data and evaluated the performance of each method individually.Furthermore, we explored the potential benefits of combining the results obtained from both approaches.

The LSTM model demonstrated promising performance in capturing the complex temporal dependencies and patterns in the stock price data. It achieved an RMSE value of 0.596, indicating its ability to make accurate predictions. The line chart depicting the predicted stock prices against the actual values showcased the model's capability to capture the underlying trends and patterns in the data. The residual analysis further confirmed the model's effectiveness in capturing the variations in the stock prices. However, the moderate standard deviation of the residuals suggests the presence of some deviations between the predicted and actual values, highlighting areas for further improvement.

On the other hand, the Markov chain modeling approach provided insights into the probabilistic behavior and transitions between different states of the stock prices. We calculated transition matrices, steady-state transition matrices, and mean hitting times for different thresholds. The results showed that the probabilities in the transition matrices were sensitive to the chosen criteria for state transitions. By considering various thresholds, we gained flexibility in classifying different states based on percentage changes. The Markov chain approach allowed us to model the probabilistic behavior of the stock prices and provided valuable information about state transitions and steady-state distributions.

Combining the results from LSTM and Markov chain modeling presented an op-

portunity to leverage the strengths of both methodologies. By incorporating LSTM predictions into the Markov chain model, we introduced an element of dynamicity and adaptability. This integration aimed to enhance the accuracy and reliability of the predictions by considering both the temporal dynamics captured by LSTM and the probabilistic behavior captured by the Markov chain. Further research is needed to explore the best integration strategies and assess the effectiveness of the combined approach.

In conclusion, this study contributes to the field of stock price prediction by evaluating and comparing the performance of LSTM and Markov chain modeling approaches. The LSTM model demonstrated its ability to capture complex patterns, while the Markov chain approach provided insights into probabilistic behavior. The integration of both methodologies holds promise for improving prediction accuracy. However, it is crucial to consider the limitations and potential sources of error associated with each method and the combined approach.

Future work in this area may include:

- Further refining and optimizing the LSTM model by exploring alternative architectures, hyperparameter tuning, and feature selection techniques. Additionally, testing the model on different datasets and time periods can provide insights into its robustness and generalizability.

- Investigating other variations of LSTM, such as long short-term memory networks with attention mechanisms or hybrid models that combine LSTM with other deep learning techniques, to further enhance predictive performance.

- Exploring advanced techniques for combining LSTM and Markov chain results, such as ensemble methods or Bayesian frameworks, to leverage the strengths of each methodology more effectively.

- Conducting comparative studies with other predictive models, such as support vector machines, random forests, or neural networks, to evaluate the competitiveness and relative performance of LSTM and Markov chain approaches.

- Considering additional features or indicators, such as volume, news sentiment, or macroeconomic variables, to enrich the predictive models and capture more comprehensive information about stock price dynamics.

- Investigating the application of the combined LSTM and Markov chain approach in real-world scenarios and assessing its practical utility by considering transaction costs, trading strategies, and risk management aspects.
- Exploring the use of alternative evaluation metrics, such as directional accuracy, profit-loss analysis, or risk-adjusted performance measures, to provide a more comprehensive assessment of the predictive models' effectiveness.
- Conducting studies on different financial markets, sectors, or specific stocks to assess the generalizability of the proposed models.

Appendix A

Additional Information

A.0.1 LSTM Code

Listing A.1: Python code for LSTM model

```
import math
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from matplotlib.dates import YearLocator, DateFormatter

# Specify the file path
csv_file_path = 'RY.csv'

# Read the CSV file into a DataFrame
stock_data = pd.read_csv(csv_file_path)

close_prices = stock_data['Close']

values = close_prices.values

training_data_len = math.ceil(len(values) * 0.8)
print(training_data_len)

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(values.reshape(-1, 1))
```

```
train_data = scaled_data[0:training_data_len, :]

x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i - 60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],
    1))

test_data = scaled_data[training_data_len - 60:, :]
x_test = []
y_test = values[training_data_len:]

for i in range(60, len(test_data)):
    x_test.append(test_data[i - 60:i, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

model = keras.Sequential()
model.add(layers.LSTM(100, return_sequences=True, input_shape=(
    x_train.shape[1], 1)))
model.add(layers.LSTM(100, return_sequences=False))
model.add(layers.Dense(25))
model.add(layers.Dense(1))
model.summary()

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=1, epochs=3)
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(predictions - y_test) ** 2)
rmse
```

A.0.2 Markov Chain Code

Listing A.2: Python code for Markov Chain Model

```
1
2  #there are total of  trading days.
3  df = stock_data[:training_data_len]
4  up = 0
5  up_list = []
6  down = 0
7  down_list = []
8  same = 0
9  same_list = []
10 prev_price = df['Close'].values[0]
11
12 threshold = 0.01
13
14 for index in range(1, df['Close'].count()):
15     current_price = df.loc[index, 'Close']
16     price_movement = (current_price - prev_price) / prev_price
17
18     state = 0
19     if price_movement > threshold:
20         up+=1
21         up_list.append(current_price)
22         state = 1
23     elif price_movement < -(threshold):
24         down+=1
25         down_list.append(current_price)
26         state = -1
27     else:
28         same+=1
29         same_list.append(current_price)
30         state = 0
31
32     prev_price = current_price
33
34 print(up, down, same)
35
36 transition_matrix = []
37
38 #calcualting transition from state up to up,down, same
39
40 up_up = 0
41 up_down = 0
```

```
42 up_same = 0
43 for index in range(0, len(up_list) - 1):
44     current_price = up_list[index]
45     next_value = up_list[index + 1]
46     price_movement = (next_value - current_price) / current_price
47 
48     if price_movement > threshold:
49         up_up+=1
50     elif price_movement < -(threshold):
51         up_down+=1
52     else:
53         up_same+=1
54 
55 up_length = len(up_list) - 1
56 
57 transition_matrix.append(up_up/up_length)
58 transition_matrix.append(up_down/up_length)
59 transition_matrix.append(up_same/up_length)
60 
61 
62 #calcualting transition from state down to up,down, same
63 
64 down_up = 0
65 down_down = 0
66 down_same = 0
67 for index in range(0, len(down_list) - 1):
68     current_price = down_list[index]
69     next_value = down_list[index + 1]
70     price_movement = (next_value - current_price) / current_price
71     if price_movement > threshold:
72         down_up+=1
73     elif price_movement < -(threshold):
74         down_down+=1
75     else:
76         down_same+=1
77 down_length = len(down_list) - 1
78 transition_matrix.append(down_up/down_length)
79 transition_matrix.append(down_down/down_length)
80 transition_matrix.append(down_same/down_length)
81 
82 #calcualting transition from state down to up,down, same
83 same_up = 0
```

```
same_down = 0
same_same = 0
for index in range(0, len(same_list) - 1):
    current_price = same_list[index]
    next_value = same_list[index + 1]
    price_movement = (next_value - current_price) / current_price
    if price_movement > threshold:
        same_up+=1
    elif price_movement < -(threshold):
        same_down+=1
    else:
        same_same+=1
same_length = len(same_list) - 1
transition_matrix.append(same_up/same_length)
transition_matrix.append(same_down/same_length)
transition_matrix.append(same_same/same_length)

print(transition_matrix)

def matrix_power(A, n):
  A = np.reshape(A, (3,3))
  # Check if A is a valid 3x3 matrix
  if A.shape != (3, 3):
    print("Invalid matrix shape")
    return None
  # Check if n is a valid integer
  if not isinstance(n, int):
    print("Invalid power")
    return None
  # If n is zero, return the identity matrix
  if n == 0:
    return np.eye(3)
  # If n is positive, use numpy matrix power function
  if n > 0:
    return np.linalg.matrix_power(A, n)
  # If n is negative, use the inverse of the matrix and the absolute
      value of n
  if n < 0:
    return np.linalg.matrix_power(np.linalg.inv(A), -n)

print(matrix_power(transition_matrix,10)) #steady state matrix
```

Bibliography

[1] F. O. Mettle, E. N. B. Quaye, and R. A. Laryea, "A methodology for stochastic analysis of share prices as markov chains with finite states," *Springerplus*, vol. 3, no. 1, pp. 1–11, 2014.

[2] H. Markowitz, "The utility of wealth," *Journal of Political Economy*, vol. 60, no. 2, pp. 151–158, 1952.

[3] M. I. Kaya and M. E. Karsligil, "Stock price prediction using financial news articles," in *2010 2nd IEEE International Conference on Information and Financial Engineering*, Chongqing, 2009, pp. 478–482.

[4] A. Lendasse, E. de Bodt, V. Wertz, and M. Verleysen, "Nonlinear financial time series forecasting- application to the bel 20 stock market index," *European Journal of Economics and Social Systems*, vol. 14, no. 1, pp. 81–91, 2008.

[5] Z. Liu and S. Wang, "Decoding chinese stock market returns: three-state hidden semi-markov model," *Pacific-Basin Finance Journal*, vol. 44, pp. 127–149, 2017.

[6] A. Shmilovici and I. Ben-Gal, "Predicting stock returns using a variable order markov tree model," *Studies in Nonlinear Dynamics & Econometrics*, vol. 16, no. 5, 2012.

[7] D. N. Choji, S. N. Eduno, and G. T. Kassem, "Markov chain model application on share price movement in stock market," *Journal of Computer Engineering and Intelligent Systems*, vol. 4, no. 10, 2013.

[8] D. Zhang and X. Zhang, "Study on forecasting the stock market trend based on stochastic analysis method: international," *Journal of Business and Management*, vol. 4, no. 6, 2009.

[9] A. Moghar and M. Hamiche, "Stock market prediction using lstm recurrent neural network," *Procedia Computer Science*, vol. 170, pp. 1168–1173, 2020.

[10] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[11] T. Kim and H. Kim, "Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data," *PLoS One*, vol. 14, no. 2, p. e0212320, 2019.

[12] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, "Deep learning for stock prediction using numerical and textual information," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. IEEE, 2016, pp. 1–6.