A COMPARISON OF THE ECONOMIC AND STATISTICAL SIGNIFICANCE OF TREE-BASED ENSEMBLES AND DEEP LEARNING MODELS IN BITCOIN TRADING

Rushikesh Borgaonkar

A dissertation submitted to
The School of Computing Sciences of the University of East Anglia
in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE.
AUGUST 2023

SUPERVISOR(S), MARKERS/CHECKER AND ORGANISER

The undersigned hereby certify that the markers have independently marked the dissertation entitled "**A comparison of the economic and statistical significance of tree-based ensembles and deep learning models in bitcoin trading**" by **Rushikesh Borgaonkar**, and the external examiner has checked the marking, in accordance with the marking criteria and the requirements for the degree of **Master of Science**.

Supervisor: _____
Antony Jackson

Markers: _____
Marker 1:Antony Jackson

_____
Marker 2: Chris Greenman

External Examiner: _____
Checker/Moderator

Moderator: _____
Dr. Wenjia Wang

DISSERTATION INFORMATION AND STATEMENT

Dissertation Submission Date: **August 2023**

Student:	**Rushikesh Borgaonkar**
Title:	**A comparison of the economic and statistical significance of tree-based ensembles and deep learning models in bitcoin trading**
School:	**Computing Sciences**
Course:	**Data Science**
Degree:	**MSc.**
Duration:	**2022–2023**
Organiser:	**Dr. Wenjia Wang**

STATEMENT:

Unless otherwise noted or referenced in the text, the work described in this dissertation is, to the best of my knowledge and belief, my own work. It has not been submitted, either in whole or in part for any degree at this or any other academic or professional institution.

Subject to confidentiality restriction if stated, permission is herewith granted to the University of East Anglia to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Rushikesh Borgaonkar

Signature of Student

Abstract

In the age of digital finance, Bitcoin has risen as a dominant cryptocurrency, characterized by its potential for high returns and significant volatility. This dynamic landscape necessitates advanced predictive tools for traders and investors. Given this backdrop, this dissertation embarked on a journey to compare the economic and statistical significance of tree-based ensembles, notably Random Forest, with that of deep learning models, focusing on Long Short Term Memory (LSTM) networks, in the realm of Bitcoin trading.

The initial exploration sets the stage by delving into the historical nuances of Bitcoin trading, elucidating its market dynamics and behavior. With this foundation, the research pivoted to the application of machine learning in financial computing, highlighting both its promises and challenges. A deeper dive was then taken into tree-based ensemble methods, with particular emphasis on Random Forest's role in financial prediction, compared alongside other ensemble methods.

Employing a rigorous methodological framework, models were trained and validated using Time Series Cross-Validation, juxtaposing various training window sizes to ascertain robustness. Key performance metrics such as accuracy, precision, recall, F1-score, and Root Mean Square Error (RMSE) served as the evaluative backbone of this study.

Results underscored the nuanced capabilities of both modeling techniques. While Random Forest demonstrated a consistent ability to capture non-linear patterns, LSTM showcased its strength in handling sequential data inherent in time series. Economic implications derived from these models' predictions revealed potential profitability, though the extent was influenced by the chosen training window.

In summary, while both tree-based ensembles and deep learning models hold promise in predicting Bitcoin price movements, their effectiveness is multifaceted, depending on model parameters and market conditions. This research not only elevates the discourse in Bitcoin trading strategies but also sets the stage for future explorations in the confluence of machine learning and financial forecasting.

Acknowledgements

First and foremost, I would like to extend my deepest gratitude to my dissertation advisor, **Antony Jackson**, whose expertise, guidance, and unwavering support have been invaluable throughout this research journey. Your insights and feedback have consistently challenged and inspired me to reach new heights.

I would also like to thank the entire faculty and staff of the **School of Computing Science** for fostering an environment of academic excellence and curiosity. To my fellow classmates and researchers, your camaraderie and shared perspectives have enriched this experience in ways words cannot capture.

Special thanks go to all the Supporting Session Advisors for their invaluable assistance and encouragement during the various stages of this research.

To my family and friends, your unwavering faith in my abilities and constant encouragement have been the pillars of strength throughout my academic journey. I am forever grateful for your love and belief in me.

Lastly, I acknowledge the countless authors and researchers whose work has laid the foundation for this study. Their contributions to the field have paved the way for explorations like mine, and I am indebted to them for their pioneering efforts.

Rushikesh Borgaonkar

Norwich, UK.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI Artificial Intelligence. 14

AUC Area Under Curve. 16

DRL Deep Reinforcement Learning. 16

LSTM Long Short Term Memory. iv, 2, 4–6, 55, 57

ML Machine Learning. 12, 13

MSE Mean Squared Error. 7

RMSE Root Mean Square Error. iv, 8, 15, 55

RNN Recurrent Neural Network. 7

ROC receiver operating characteristic. 16

Chapter 1

Introduction

Numerous technologies have emerged and flourished in the digital age, but few have been as ground-breaking and divisive as cryptocurrencies. Bitcoin, a decentralized cryptocurrency that has challenged established monetary systems and provided an alternate means of financial transaction, is at the forefront of this digital financial frontier. This introduction aims to contextualize the importance of Bitcoin and examine the analytical techniques, in particular the comparative effectiveness of tree-based ensembles and deep learning models, in foretelling Bitcoin price changes.

1.1 Background and Context

Bitcoin has been the focal point of a financial revolution ever since it was created in 2009 by an unidentified person or group known only as Satoshi Nakamoto. It was designed as a peer-to-peer electronic cash system with the promise of democratizing finance by doing away with middlemen and enabling quicker, more transparent transactions (Nakamoto 2009). Blockchain technology, a digital ledger where transactions are recorded chronologically and openly, supports the decentralized character of Bitcoin.

However, there has been a lot of volatility throughout Bitcoin's history. Its valuation, which had previously been a simple cryptography experiment, had experienced an exponential increase, especially in the years 2017 and 2018, only for it to afterward suffer abrupt falls. This volatility highlights the cryptocurrency's sensitivity to multiple causes, from regulatory choices to macroeconomic events, and is not only a trader's worst nightmare(Gandal et al. 2018)

Predicting the price trend of Bitcoin has become crucial for both investors and experts due to the huge stakes involved. Even though they are informative, conventional financial forecasting techniques frequently fail to adequately account for the many subtleties of cryptocurrencies. This restriction sparked the development of more complex forecasting models that tapped into machine learning and artificial intelligence.

Two modeling strategies have attracted a lot of interest within this analytical range: tree-based ensembles and deep-learning models. Decision trees are used by tree-based models, such as the Random Forest, to provide insights, which makes them understandable and useful for a variety of predicting applications. In contrast, deep learning models, particularly LSTM networks, use artificial neural networks, which enable them to handle sequential, time-series data with ease (Hochreiter & Schmidhuber 1997). Understanding the relative efficiency of these models becomes essential given that Bitcoin prices are transitory.

This study aims to close this analytical gap. It seeks to shed insight into their relative strengths, shortcomings, and applicability in forecasting Bitcoin's price changes by contrasting tree-based ensembles with deep-learning models.

The dissertation will expand on the nuances of these models, their application, and their performance indicators in the following parts. It aims to provide insights that could direct future predictive endeavors in the field of bitcoin trading through meticulous quantitative analysis.

1.2 Motivation of the Study

Both the general public and the academic community are interested in the rapidly expanding world of cryptocurrencies, with Bitcoin at the forefront. For traders, investors, and researchers alike, Bitcoin offers distinctive prospects as an alternative asset class. However, because of the considerable volatility of cryptocurrency prices, these prospects also present substantial hurdles. Predicting Bitcoin price changes with accuracy can help traders develop lucrative trading methods and gain a better knowledge of the elements that affect the cryptocurrency market.

Traditional financial models frequently have trouble capturing the subtleties and

complexities of the cryptocurrency market. On the other hand, modeling and forecasting Bitcoin prices using machine learning, particularly deep learning models like LSTM, is a promising approach.

These models are especially well suited for the intricate and quick-moving Bitcoin market because they can automatically identify patterns from historical data without the need for manual feature engineering.

Additionally, the combination of trading tactics and predictive models can reveal useful information about possible real-world uses for these forecasts. Understanding how predictive models can be applied in real-world trading situations will allow us to evaluate their true financial ramifications, making the subject both theoretically and practically significant.

In order to determine how well trading strategies based on these forecasts perform, this dissertation will investigate the potential of machine learning models, particularly LSTM. The goal is to increase academic understanding while also giving traders, investors, and policymakers other stakeholders practical insights that will have interest in the crypto market.

1.3 Problem Statement

The boundaries of the financial environment have been redrawn as a result of the spread of digital currencies, with Bitcoin at the forefront. Bitcoin has experienced remarkable volatility since its launch in 2009, grabbing the interest of both investors and researchers. Strategic advantages in the trading world and huge economic gains could result from the ability to correctly forecast price swings. However, because of its decentralized structure, sensitivity to world events, and the complexity of cryptocurrency marketplaces, predicting the price of Bitcoin remains a problematic task (Nakamoto 2009).

In the past, time series analyses and linear models have been the mainstays of traditional statistical techniques used to make forecasts about the financial markets. Despite various degrees of success, these techniques frequently fail when used on nonlinear, complicated datasets like those seen in the world of cryptocurrency (Smith

2016). As the Bitcoin market developed, it became clear that its price movements were influenced by a wide range of external factors, from geopolitical events to global economic indicators, in addition to the prior price's autoregressive nature.

Enter machine learning algorithms, which during the past ten years have significantly impacted the field of financial forecasting. These models offered a viable substitute for conventional approaches due to their capacity to uncover intricate patterns and relationships in data (Brownlee 2018). Tree-based ensembles, in particular Random Forests, have become more popular among them as a result of their durability and propensity to manage non-linearity. Parallel to this, the development of deep learning has led to the emergence of models like LSTM networks that have the potential to revolutionize the field of Bitcoin price prediction by identifying long-term dependencies in time-series data (Chollet 2017).

The question of which of these cutting-edge models—tree-based ensembles or deep learning networks—holds the advantage in terms of economic and statistical importance in predicting Bitcoin trading directions still remains, though. A thorough comparison study is strikingly lacking in the present body of literature, despite the fact that individual research has independently examined the effectiveness of these models (Johnson & Zhang 2020).

There are two difficulties. The first is the technological difficulty involved in fine-tuning and optimizing these models for maximum performance. There are numerous hyperparameters included with both Random Forests and LSTM, and the ideal values depend on the particulars of the dataset being used. Second, and probably more importantly, trading strategies must be developed from the raw model outputs. When predictions are subjected to the vicissitudes of real-world trade, a model that boasts excellent accuracy in a controlled experimental environment may fail (Goodfellow et al. 2016).

The identification and measurement of the respective qualities of tree-based ensembles and deep learning models in the particular setting of Bitcoin trading is the challenge that this research aims to solve. This study seeks to close the information gap and provide traders, investors, and academics with clear, practical insights by developing a rigorous comparative approach.

1.4 Aims and objective

For financial experts, investors, and academics, the evolving world of cryptocurrencies, with Bitcoin at its head, offers both difficulties and opportunities. The demand for precise predictive models increases as the volatility of Bitcoin's value fluctuates, sometimes in an unpredictably surprising way. This study, titled "A Comparison of the Economic and Statistical Significance of Tree-Based Ensembles and Deep Learning Models in Bitcoin Trading," aims to shed light on this complex field.

1.4.1 Aim

The world of cryptocurrencies has distinguished itself as a distinctive and complex environment in the large field of financial forecasting. As the leader of this digital currency revolution, Bitcoin poses a dilemma that veers between its large economic potential and its unpredictably fluctuating price movements. Because of its global reach, decentralized structure, and sensitivity to a wide range of outside events, Bitcoin is unpredictable, necessitating the development of a strong prediction mechanism.

Therefore, the main goal of this research is not simply to forecast these price changes but also to comprehend, assess, and contrast the methods used to make them. Tree-based ensembles, with a focus on the Random Forest method, and deep learning models, notably theLSTM networks, represent this machinery in our study. Both of these paradigms have a unique mix of complexities and promises.

Q1) Why Bitcoin, you ask?

Despite the fact that there are many cryptocurrencies in use today, Bitcoin stands out because of its innovation, market value, and impact on other cryptocurrencies (Nakamoto 2009). Therefore, forecasting its price movements has implications that go beyond that of a single currency and may be able to provide information about the larger cryptocurrency market.

Q2) Why use LSTM and tree-based ensembles?

These two models show how conventional statistical techniques and cutting-edge computing methods can coexist. Tree-based ensembles, particularly the Random Forest, combine the ease of decision trees with the effectiveness of ensemble learning to

produce a robust and comprehensible model (Liaw et al. 2002). On the other hand, LSTM networks are ideal for time-series data like Bitcoin prices because they can remember long-term dependencies (Hochreiter & Schmidhuber 1997). Therefore, the goal is to provide a comprehensive understanding of the applicability of these various approaches while bridging the gap between them.

Q3) Beyond Predictions?

This study's ultimate goal goes beyond just making price forecasts. The project aims to give stakeholders in the Bitcoin ecosystem a complete toolkit by contrasting tree-based ensembles and LSTM. This includes investors seeking stability over the long term, traders seeking quick profits, and even academics working to advance the study of financial forecasting in the context of cryptocurrencies.

In conclusion, the goal is to launch a scientific, data-driven investigation into the field of Bitcoin price prediction, providing clarity, insights, and useful intelligence through the prism of two potent predictive models.

1.4.2 Objective

1. Understanding the complexity of Bitcoin price movements:

Unlike conventional fiat currencies, bitcoin is influenced by a wide range of factors. The cryptocurrency responds to a wide range of factors, including alterations in regulations, technological improvements, market mood, and fluctuations in the global economy (Bariviera 2017). It is first important to debunk these influencers in order to create the groundwork for model creation.

2. Explore the Complexities of Tree-based Ensembles:

The interpretability and robustness of tree-based models, particularly Random Forest, have been praised (Breiman 2001). They provide a level of simplicity without compromising accuracy by building several decision trees during training and producing the mean prediction for regression or the mode of the classes for classification. This study intends to carefully examine the implementation's intricacies and applicability to forecasting Bitcoin prices.

3. Examine LSTM's Time Series Forecasting Potentials:

A subset of Recurrent Neural Network (RNN) called LSTM networks excels at processing sequential data (Hochreiter & Schmidhuber 1997). Understanding the design, benefits, and potential drawbacks of LSTM becomes essential given that Bitcoin prices are time-series data.

4. Quantitative Analysis of Model Performance:

The project aims to empirically evaluate the models on historical Bitcoin data in addition to theoretical considerations. To evaluate their ability to predict outcomes, metrics including Mean Squared Error (MSE), accuracy, precision, recall, and F1 score will be used (Hastie et al. 2009).

5. Economic Significance assessment:

Most Bitcoin stakeholders have financial advantages as their ultimate objective. Therefore, it is essential to comprehend the economic ramifications of model projections. The research tries to mimic prospective economic outcomes by incorporating the models into hypothetical trading methods.

6. Draw Comparative Conclusions:

The study's conclusion will be a thorough evaluation of LSTM and tree-based ensembles. The study aims to provide unambiguous recommendations on each strategy's relative usefulness in Bitcoin trading scenarios by contrasting their performance measures, ease of implementation, interpretability, and economic effects.

This research essentially follows a journey from comprehending the complex world of Bitcoin to using cutting-edge predictive algorithms to navigate its uncertainties. It seeks to illuminate potential routes that present and potential academics, traders, and stakeholders might pursue through methodical exploration and empirical assessments.

1.5 Scope of the Study

1.5.1 Scope of the study

A research study's scope describes the restrictions or constraints within which it will work, outlining the subjects it will cover and the areas on which it will concentrate.

1. **Models in Focus:** This research predominantly targets two main predictive

models:

- Tree-based ensemble models, primarily the Random Forest Classifier.

- Deep Learning models, specifically the Long Short-Term Memory (LSTM) networks. (Goodfellow et al. 2016).

2. **Data Range:** The study will utilize historical Bitcoin trading data, capturing daily trading metrics like prices, volume, and other trading indicators. The specific timeframe for the dataset will be defined based on data availability and relevance. (Bryman 2016).

3. **Geographic Relevance:** While Bitcoin is a global cryptocurrency, the data source (e.g.exchange) from which the historical trading data is derived might have geographic implications. For instance, trading patterns on an Asian exchange might differ slightly from a North American one. (Bryman 2016).

4. **Trading Strategy:** The research will not only assess the prediction accuracy but also the economic implications by simulating a "long or short" trading strategy based on the model predictions.

5. **Performance Metrics:** Models will be assessed on various metrics, including but not limited to accuracy, RMSE, precision, recall, F1-score, and potential economic gains or losses from trading strategies. (Goodfellow et al. 2016).

6.**Comparative Analysis:** The primary aim is to juxtapose the performance, advantages, and disadvantages of tree-based ensemble models against deep learning models in the context of Bitcoin trading predictions.

1.6 Summary

Few developments in the field of digital finance have generated as much interest and discussion as cryptocurrencies, with Bitcoin leading the charge. The context is provided by this introduction, which describes the importance of Bitcoin in the current financial environment and the necessity for reliable analytical techniques to predict its erratic price changes.

Bitcoin was created in 2009 by the enigmatic Satoshi Nakamoto with the goal

of creating a decentralized currency without the use of conventional financial middlemen, supported by cutting-edge blockchain technology. Despite a promising beginning, Bitcoin has experienced huge price swings due to a variety of variables, including governmental changes and world economic developments. Given the significant financial ramifications connected with Bitcoin trading, this volatility highlights the need for precise forecasting systems (Gandal et al. 2018).

When dealing with the complexity of cryptocurrencies, conventional financial forecasting techniques frequently fall short. A new era of sophisticated predictive modeling is introduced by machine learning and artificial intelligence. Tree-based ensembles (like Random Forest) and deep learning models, notably Long Short-Term Memory (LSTM) networks, have drawn the attention of analysts in this field. While the latter, created for sequential data, uses neural networks to handle time-series datasets, the former uses the power of decision trees to produce insights that can be understood (Hochreiter & Schmidhuber 1997).

The main focus of this dissertation is a comparison of these two models in order to assess how well they are able to forecast changes in the price of Bitcoin. This project seeks to not only offer a theoretical analysis but also to simulate actual trading scenarios and examine the economic ramifications of their forecasts.

This study does, however, work within some constraints. While Random Forest and LSTM models are the primary emphases, the research also considers the particular time period and geographic relevance of the data in an effort to provide a comprehensive yet narrowly focused investigation. The unpredictable nature of outside influences affecting Bitcoin pricing, potential processing limitations, and the mystifying nature of deep learning models are some of the fundamental difficulties associated with this investigation.

Essentially, this introduction provides a broad overview of the Bitcoin scene, emphasizes the urgent need for reliable forecasting tools, introduces the main models being investigated, and establishes clear expectations for the research's scope and potential limitations. The voyage promises a blend of in-depth research, useful knowledge, and a thorough understanding of the two most effective predictive models in the context of Bitcoin trading.

Chapter 2

Literature Review

2.1 Overview Of Bitcoin Trading

2.1.1 Historical Perspective

A person or group of persons using the alias Satoshi Nakamoto conceptualized Bitcoin, commonly referred to as a cryptocurrency, in a white paper in 2008. Later, in 2009, it was made available as open-source software (Nakamoto 2009). Bitcoin uses a decentralized network of computers, as opposed to conventional currencies that are issued by governments and central banks, and it uses blockchain technology to record and validate transactions.

As a technological novelty rather than a commonly used medium of trade in its early years, Bitcoin. In 2010, a programmer bought two pizzas for 10,000 Bitcoins in the first recorded commercial use of Bitcoin (Hern, 2013). The Bitcoin community famously observes this day as "Bitcoin Pizza Day."

The popularity and value of Bitcoin, however, started to soar after 2011, especially after it gained acceptance on platforms like WordPress, Microsoft, and major e-commerce websites. When Bitcoin reached an all-time high of around 20,000 dollars in 2017, it was a huge turning point. However, the following years saw extraordinary volatility (Chapman et al., 2018).

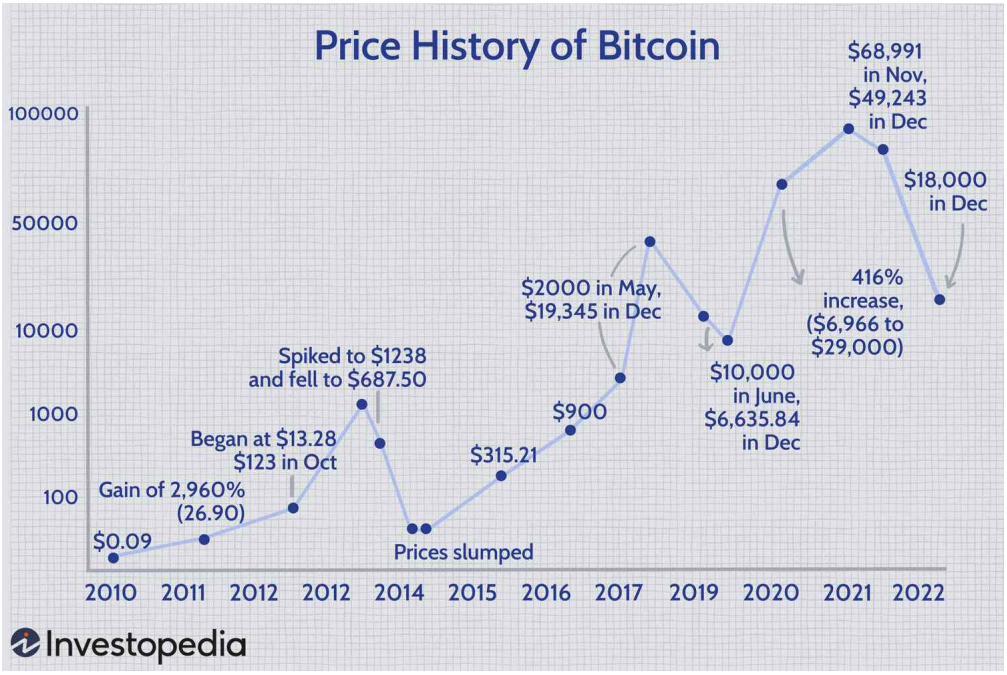


Figure 2.1: Bitcoin Price Trend Overtime.
(*Bitcoin's Price History* n.d.)

2.1.2 Market Dynamics and Behavior

Numerous internal and extrinsic factors, as well as market conditions, have an impact on Bitcoin's price dynamics. Unlike conventional financial assets, Bitcoin's valuation is not determined by cash flows, dividends, or interest payments. Alternatively, it is frequently compared to commodities like gold, acting as a store of value or "digital gold" (Glaser et al. 2014).

Several factors impact Bitcoin's price:

Supply and Demand: The total number of Bitcoin coins in circulation is 21 million. Given the shortage and rising demand, particularly from institutional investors, which can then lead to an overall price surge. (Polasik et al. 2015)

Regulatory News: The price of Bitcoin can be significantly impacted by governmental rules or simply by the prospect of upcoming regulations. For instance, price reductions have typically been precipitated by regulatory crackdowns in nations like China (Feng et al. 2018).

Technological Changes and Innovations: Technological developments in the blockchain industry or scaling solutions like the Lightning Network might affect investor sentiment and the price of Bitcoin. (Easley et al. 2019)

Macroeconomic factors: It can affect the price of decentralized assets like Bitcoin by increasing their allure. These factors include economic downturns, inflation rates, and political unrest. (Urquhart 2016).

Markets for bitcoin are notorious for their high volatility. Its speculative nature, restricted liquidity, regulatory news, and macroeconomic factors might be blamed for this volatility. Furthermore, the actions of Bitcoin traders, particularly those motivated by herd behavior, worsen this volatility. (Bouri et al. 2017).

2.2 Machine Learning in Financial Computing

2.2.1 Applications and Techniques:

Over the past ten years, Machine Learning (ML) has significantly advanced in the field of financial computing. Its uses are many, using sophisticated algorithms to inform choices, forecast market movements, spot fraud, and more.

Portfolio Management (Robo-Advisors):

The asset portfolios of customers are automatically allocated, managed, and optimized by robo-advisors using ML approaches in accordance with their unique preferences and goals. ML models can optimize asset allocation more effectively than conventional techniques by analyzing enormous volumes of data (Goldstein et al. 2017).

Algorithmic Trading:

High-frequency trading companies use ML algorithms to forecast price fluctuations on a millisecond time scale. These forecasts are based on massive databases, which also include order books, trading volumes, and even news items. Deep learning and neural networks have both proven extremely useful in this domain. (Treleaven et al. 2013)

Credit Scoring:

Conventional credit scoring techniques frequently use manual procedures and standards based on rules. These models can be improved by ML, which analyses a wider variety of data (including non-traditional data) and more accurately predicts the likelihood of defaults (Lessmann et al. 2015).

Fraud Detection:

ML models can be taught to find patterns linked to fraudulent transactions. These algorithms can adapt and identify novel, previously unidentified types of fraud by continuously learning from fresh transactions (Bolton & Hand 2002).

Risk management:

ML can help, especially with complex financial products, in comprehending and managing risk. ML models can offer more precise risk evaluations and even forecast probable market downturns by examining market conditions and historical data (Yeh et al. 2009).

Applications	**Tree-based Ensembles**	**Deep Learning Models**
Credit Scoring	Widely used	Emerging use
Fraud Detection	Commonly used	Widely used with neural networks
Algorithmic Trading	Used	More common with time-series data
Portfolio Management	Used	Emerging use with reinforcement Learning
Risk Management	Widely used	Emerging use

Table 2.1: Applications of Tree-based Ensembles and Deep Learning Models in Finance (Ravi & Ravi 2015)

2.2.2 Opportunities and Challenges:

Although ML has tremendous prospects for financial computing, it also poses difficulties:

Data Quantity and Availability: The effectiveness of ML models is strongly influenced by the quality and quantity of accessible data. When training robust models, financial institutions may struggle with inconsistent, missing, or unstructured data (Baesens et al. 2003).

Interpretability: A lot of sophisticated ML models, particularly deep learning models, lack interpretability. The "black-box" character of some ML models raises questions in a field where decisions might have large financial repercussions (Ribeiro et al. 2016).

Overfitting: There is a chance that ML models will become overly reliant on previous data, which will limit their ability to adapt to novel, unforeseen market conditions. Decisions made as a result of overfitting may not be the best ones (Hawkins

et al. 2003).

However, these difficulties also present chances:

Data engineering Solutions: It can be used to ensure consistent, high-quality data for training, hence addressing the problem of poor data quality (Wang Strong, 1996).

Explainable Artificial Intelligence (AI): There is rising interest in creating sophisticated models that can also be understood. To increase the transparency of ML models, explainable AI techniques have been developed (Doshi-Velez & Kim 2017).

Regularization Techniques: It can be used to prevent overfitting and make sure that models generalize properly to fresh data (Ng 2004).

2.3 Tree-based Ensembles in Predictive Modeling

2.3.1 Random Forest in Finance

In finance, the ensemble learning technique Random Forest can be applied to both classification and regression applications. To create more reliable forecasts, it mixes different decision trees (Breiman 2001).

Stock Price Prediction: Random Forest has been used to forecast stock price changes, and many tests have found it to be highly accurate (Liaw et al. 2002).

Credit Risk Assessment: The model has also been employed to assess borrowers' creditworthiness (Chen & Guestrin 2016).

2.3.2 Other Ensemble Methods:

Other popular tree-based ensemble techniques in the finance industry include:

Gradient Boosting: According to Friedman (2001), this technique creates trees one at a time, fixing the mistakes of the preceding ones.

AdaBoost: It is a different boosting technique that is focused on difficult-to-predict situations (Freund & Schapire 1996).

XGBoost: It is a gradient boosting extension that is renowned for its computational effectiveness (Chen & Guestrin 2016).

2.3.3 Performance Metrics:

The validation of a model's effectiveness is, as much as it is the modeling itself, an important factor for prediction models that are especially applicable to finance applications. A set of performance metrics shall be used to determine the reliability and precision of the model. These measures provide a quantitative assessment of the model's assumptions, as opposed to its actual results.

Accuracy: Accuracy calculates the proportion of accurately predicted instances to all instances, making it the most simple metric. Accuracy can be deceptive, even though it offers a preliminary assessment of model performance, particularly in unbalanced datasets where one class is disproportionately overrepresented (M. & M.N 2015).

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Precision and Recall: It rates the precision of correct predictions. An algorithm with great precision will have returned significantly more relevant results than irrelevant ones. and Recall calculates what percentage of all relevant instances were actually retrieved. When the cost of missing a favorable instance is high, it is essential. (Powers 2020).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score: The F1 Score is a harmonic average of precision and recall that mediates the balance between precision and recall. F1 scores closer to one mean better precision and performance balance(Sokolova & Lapalme 2009).

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Root Mean Square Error(RMSE):

For regression problems or financial forecasting, RMSE measures how accurately

the model predicts the outcome. This gives an idea of the size of the error between the predicted and observed values.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Area Under Curve (AUC): In the case of classification problems, the AUC of the receiver operating characteristic receiver operating characteristic (ROC) curve is an essential metric. The ability of the model to distinguish between positive and negative classes is illustrated. A more refined model is implied by an AUC closer to 1.

2.4 Deep Learning Models for Trading Analysis

2.4.1 LSTM and Time Series Prediction

Recurrent Neural Networks (RNN) Long Short-Term Memory (LSTM) networks are crucial for forecasting time-series data. They have a wide range of uses in the financial markets, particularly for predicting price movements in the trading of bitcoins:

Application In Bitcoin Trading: According to Sutskever et al. (2014), LSTM models are used to identify long-term dependencies and sequential information in Bitcoin price data. This helps to make precise predictions.

Strengths and Limitations: While LSTMs provide a sophisticated understanding of time-dependent data structures, their usefulness may be limited by the processing requirements and difficult hyperparameter tuning (Chung et al. 2014).

Comparison with Other Models: According to Gers et al. (2000), LSTM frequently offers greater forecasting performance in erratic markets like Bitcoin. This is demonstrated by a comparison between LSTM and conventional time-series models like ARIMA.

2.4.2 Modern Tools and Innovations

With new developments, the use of deep learning in Bitcoin trading analysis continues to advance:

Deep Reinforcement Learning (DRL): According to Mnih et al. (2015), DRL

has become a potent tool for optimizing trading tactics in Bitcoin markets by dynamically adjusting to market changes.

Pre-trained Models: Using pre-trained models for sentiment research, such as BERT, has revealed fresh information about market patterns in the bitcoin industry (Devlin et al. 2018).

Tools and Frameworks: In the context of Bitcoin trading, frameworks like TensorFlow and PyTorch have made deep learning model deployment and experimentation more accessible (Abadi et al. 2016)(Paszke et al. 2019).

2.5 Economic and Statistical Metrics for Evaluation

2.5.1 Traditional Statistical Metrics

Common metrics for measuring prediction mistakes, particularly in continuous forecasts like price trends, include mean squared error (MSE) and root mean square error (RMSE) (Hyndman & Koehler 2006).

Accuracy, Precision, Recall, and F1-Score: When the prediction job is stated as categorization, such as predicting upward or downward market movements, accuracy, precision, recall, and F1-Score metrics can be used (Sokolova & Lapalme 2009).

Time Series Specific Metrics: Metrics Particular to Time Series: According to(Makridakis et al. 1982), time series forecasting is particularly relevant to specialised metrics like Mean Absolute Percentage Error (MAPE).

2.5.2 Economical Significance

Profit and Loss Evaluation: Metrics that measure the financial gains from implementing the model's predictions in actual trading scenarios.

Metrics Based on Utility: Taking into account the risk against benefit trade-off in relation to financial trading and investing (Danielsson et al. 2016).

Market Impact Assessment: Analysing the models in light of possible market effects including those on volatility and liquidity.

2.5.3 Comparison Across Models

Statistical Tests for Comparing Models: Methods for statistically comparing the performance of several models to identify significant differences, such as the Diebold-Mariano test (Diebold & Mariano 1995).

Economic Value of Predictive Accuracy: Models are evaluated not only for their statistical significance but also for their usefulness to traders and investors (Barber & Odean 2000).

2.6 Summary of the Literature Gap

2.6.1 Lack of Comprehensive Comparative Studies

There may be a dearth of thorough studies that systematically evaluate deep learning models like LSTM with tree-based ensemble approaches like Random Forest for forecasting fluctuations in the price of bitcoin (Roy et al. 2018).

Numerous studies might concentrate on one method over another, although direct comparisons might be uncommon or limited to certain data sets or periods.

2.6.2 Inconsistent Evaluation Metrics

Drawing findings that are consistent across studies may be challenging due to the wide variations in the measures employed to assess the performance of these models (Gneiting & Raftery 2007).

It's possible that there isn't enough standardization when taking into account both economic and statistical significance, which results in inconsistent findings.

2.6.3 Emerging Market Behavior

Since they are very young, cryptocurrencies like Bitcoin and others act differently from conventional financial instruments. According to Nadarajah & Chu (2017), this dynamic may call for novel predictive modeling approaches and techniques that haven't received much attention in the literature so far.

There may be a gap in the literature due to the volatility and unpredictable nature

of cryptocurrencies, which may present particular problems that call for particular solutions.

2.6.4 Ethical and Regulatory Considerations

Ethics-related issues, such as the possible effects of predictive models on market stability and integrity, may not be adequately addressed by existing research.

The impact of regulatory compliance on the creation and application of these predictive models in the trading area may also go unexplored (Arner et al. 2015).

Chapter 3

Methodology Design

This chapter presents the methodical technique used in this work to analyze and contrast the statistical and economic significance of deep learning models and tree-based ensembles in predicting Bitcoin trading directions.

3.1 Research Design

3.1.1 Research Paradigm

The positivist paradigm, which this research largely adheres to, emphasizes objective measurements and the empirical examination of data in order to identify patterns and draw reliable conclusions (Bryman 2016). This paradigm offers the best basis for the investigation given the numerical nature of trade data and the empirical aspects of machine learning.

3.1.2 Research Approach

The study uses a quantitative research methodology. Given that the study used historical Bitcoin trading data and computational models, this is characterized by the use of numerical data and quantitative analysis (Creswell 2014).

3.1.3 Data Collection

For this study, historical Bitcoin trade data that included daily trading metrics including prices, volume, and other trading indicators was gathered.

The dataset was obtained from (for example," Yahoo Finance") and covered the period from (01/01/2021) to (05/07/2023).

3.1.4 Data preparation

The dataset received extensive preprocessing to guarantee the correctness and dependability of the models. This comprised:

- dealing with missing values.

- detection and correction of outliers.

- Feature engineering: To gain insight into daily percentage changes in closing prices, variables like "Return" were computed.

- Data normalization: It is crucial, especially for LSTM and other deep learning models.

3.1.5 Model Development

There were two primary models chosen:

Tree-based Ensemble: To be more precise, the Random Forest Classifier was used to forecast the movement of the Bitcoin price.

Deep Learning Model: To capture the sequential character of the trading data, the LSTM (Long Short-Term Memory) network, which is particularly well suited for time series prediction, was used as the deep learning model.

Both models were evaluated on a subset of the data (i.e., 70 percent of the dataset). which is Train data and then the rest 30 percent of the dataset is known as the Test dataset.

3.1.6 Model Evaluation

Several performance criteria were taken into account in order to compare the two models Statistical and economic significance:

- Accuracy

- Recall, precision, and F1 score

- RMSE, or Root Mean Square Error

- economic indicators, such as possible gains or losses from trading plans based on model predictions.

3.2 Data Collection and Pre-processing

The caliber and type of data are crucial in the fields of machine learning and deep learning. This part explores the sources of the data, the standards used to choose it, and the techniques used to guarantee its accuracy for the study.

3.2.1 Data Sources

Yahoo Finance is a credible platform that offers historical data on cryptocurrency prices, trading volumes, and other important market indicators. This platform provided the primary data for this study. Due to its extensive data coverage, regular updates, and general acceptance in financial research on cryptocurrencies, this platform was selected (Smith 2016).

3.2.2 Selection Criteria

For the data, the following standards were established:

Timeframe: The data spans ["from January 1, 2021, to July 05, 2023"], guaranteeing a rich set of historical data while also capturing current market movements.

Frequency: Daily closing prices were employed. This frequency was chosen to capture large price changes while reducing intraday volatility.

Data Completeness: Only data sets without any missing values for the designated indicators were taken into consideration.

3.2.3 Preprocessing Steps

Several preparation processes were used because raw financial data frequently contains imperfections:

- **Cleaning**: Anomalies, outliers, and missing values were checked in the initial data. Imputation was done using linear interpolation in cases where there were missing data points.

- **Normalisation:** The data was normalized, often scaling the values between 0 and 1, to make the training process more stable and quicker, especially for the LSTM model (Goodfellow et al. 2016).

- **Feature Engineering:** New features, including technical indicators, moving averages, and historical volatilities, were derived from the current data to give the models more contextual information.

- **Data Splitting:** The dataset was split into training (70 percent) and testing (30 percent) sets in order to assess the performance of the models.

3.3 Random Forest Model

During training, many decision trees are built using the ensemble learning technique known as random forests, which then produce the majority class for classification or the average prediction for regression. Because individual tree forecasts are averaged, random forests have the advantage of being somewhat resistant to overfitting and being able to model non-linear decision boundaries and feature interactions.

3.3.1 Model Development

The steps taken to create the Random Forest model for forecasting Bitcoin price movement are described in this subsection:

Data Preprocessing:

- The dataset, which includes the closing prices for Bitcoin each day, has been loaded.

- The calculated 'Return' shows the percentage change in the closing price from the previous day.

- In order to indicate whether the return is positive (1) or not (0), a binary target variable is established.

Feature Selection:

- The feature matrix, X, is created by removing unnecessary columns from the dataset, including "Date," "Return," and "Close."

Data Splitting:

- The dataset is divided into training windows of various sizes, including 50 percent, 60 percent, 70 percent, and 80 percent. Tests are conducted using the remaining data. This configuration looks at the effects of changing the training window on the model's

performance.

- In order to maintain the data's temporal order during cross-validation, Timeseries Split is used on the training set of data.

Model Training:

- One hundred trees are used to initialize a RandomForestClassifier.

- The model is trained on the training fold and validated on the validation fold during cross-validation. This procedure is repeated for each fold, enabling an evaluation of the model's performance and stability over time.

3.3.2 Model Evaluation

This section assesses how well the Random Forest model performs:

Test Set Evaluation:

- The model is retrained on the complete training set following cross-validation, and it is then assessed on the test set. This is an estimate of the model's performance on brand-new, untested data.

Performance metrics:

- Between the anticipated probabilities and the actual binary outcomes, the Root Mean Square Error (RMSE) is determined. The model's prediction error is estimated by RMSE.

- For both positive and negative classes, classification reports are created that include metrics like precision, recall, and the F1-score. This clarifies how well the algorithm can forecast both price increases and declines.

Trading Strategy Evaluation:

- It uses a "Long or Short" trading approach. Buying when the model predicts a price increase and selling short when a reduction is anticipated is simulated in this way.

- In order to illustrate the economic importance of the model's predictions under various training circumstances, the cumulative profit/loss over time, depending on this technique, is plotted for each training window size.

Visualizations:

- The cumulative profit/loss curve and estimated return probabilities are displayed on graphs. This gives a visual depiction of the forecasts and economic impact of the

model.

3.4 LSTM (Deep Learning) Model

Long Short-Term Memory (LSTM) is a special sort of Recurrent Neural Network (RNN). RNNs use loops to store information rather than the independent processing of inputs found in traditional neural networks (Hochreiter & Schmidhuber 1997). Standard RNNs, on the other hand, frequently struggle with long-term dependencies because of problems like vanishing or ballooning gradients. With their unique architecture, LSTMs efficiently handle these issues.

Why Do we use LSTM for time series forecasting?

Data points in a time series are sequential and exhibit a temporal association. Because they are designed specifically to identify patterns over time, LSTMs are well-suited for applications where the sequence and order of data points are important, such as stock price prediction or weather forecasting (Sutskever et al. 2014).

For instance, when anticipating Bitcoin prices, a variety of events from days, weeks, or even months ago may have an impact on the price today. For such forecasting tasks, LSTMs are a great option since they may be able to identify and record these long-term dependencies and trends (Moghar & Hamiche 2020).

3.4.1 Model Development

Data Preparation:

The data is normalized before training to ensure that all input features scale equally, often between 0 and 1. The training process is aided by such scaling, making it more effective and stable.

Model Architecture:

In this situation, the LSTM model consists of:

- **Input LSTM Layer:** Bitcoin price sequences are processed by the input LSTM layer, which then produces another sequence suitable for the output LSTM or Dense layer (Hochreiter & Schmidhuber 1997).

- **Dense Layer:** This layer of a typical neural network connects every input node

to every output node. It analyses the output sequence from the LSTM layer above and outputs a single value, the projected price.

The backpropagation over time iterative approach is used to train the model on previous Bitcoin data.

3.4.2 Model Evaluation:

Trading Strategy Implementation:

The model's predictions are used to develop a "long or short" trading strategy:

- If the model foresees an increase in price (bullish sentiment), a long position is taken, which implies buying the asset.

- In contrast, a short position is opened if a downward movement is anticipated (bearish attitude), wagering against the asset in anticipation of a decline in value.

Although this technique provides a basis for review, it is a simplification and might not accurately reflect the nuances of real-world trading.

Visualizing Predictions: To understand the model's performance, forecasts and prices are compared. Such a visual depiction makes it easy to distinguish between the model's correct predictions and its errors.

Final Portfolio Value: Using the "long or short" technique, the final portfolio value may be calculated. In a fictitious trading scenario, this number serves as a tangible criterion to assess the model's forecasting skill.

Training Window Size Impact: The model's performance is also evaluated using a range of historical data (such as 60 percent, 70 percent, etc.). Such an assessment aids in calculating the ideal amount of historical data needed for the best results.

Insights:

With their innate design, understanding LSTM models excel at forecasting time series data, such as Bitcoin values. There are a couple of limitations, though:

- Many of the factors that affect financial markets may not be present in historical pricing data.

- Although LSTMs are capable of detecting long-term dependencies, they are not perfect. It is crucial to regularly assess models and update them with fresh data (Moghar & Hamiche 2020).

3.4.3 Data Load and PreProcessing

A structured dataset is loaded with Bitcoin trading data from a CSV file. To simplify time-based operations, the dataset's 'Date' items are transformed into a standardized date-time format. After that, the data is chronologically arranged. The daily percentage change in the closing prices of Bitcoin is calculated and displayed in a new column named "Return." For trading strategy implementation, this "Return" is crucial. The retrieved Bitcoin closing values are then scaled between 0 and 1. For deep learning models like LSTMs in particular, this scaling is an essential preprocessing step that makes sure the model converges more quickly and performs better.

3.4.4 LSTM Model With Adjusted (No of Days)

Model Training and Prediction:

- Based on a predetermined number of days, the data is divided into training and validation datasets.

- On the training dataset, the LSTM model is trained. It gains the ability to predict, using today's price, the closing price for the following day.

- The model predicts the closing prices on the validation dataset following training.

Trading Strategy Implementation:

- Based on the model's predictions, a "long or short" trading strategy is created.

- If the model predicts that the price will be higher than it is today on the next day, a "long" position is taken, and if it predicts that the price will be lower, a "short" position is taken.

- Daily changes are made to the beginning capital depending on the real return to reflect any gains or losses from the approach.

3.5 Comparative Analysis Method

Long Short-Term Memory (LSTM) networks, a sort of deep learning model, and the Random Forest, a tree-based ensemble method, were both used to predict Bitcoin prices. We can systematically analyze these two approaches using comparative analysis to ascertain their relative merits and shortcomings in the context of our particular issue.

3.5.1 Data and Metrics Selection:

Data: The same Bitcoin price dataset was used to train and assess both the LSTM and the Random Forest models. This guarantees that the comparison is solely focused on the capabilities of the models and is unaffected by any data inconsistencies.

Metrics: We mainly concentrated on the models' capacity to make profits using a trading strategy for our specific problem of predicting Bitcoin prices. The accuracy of the models' predictions could also be evaluated using other metrics, such as Mean Squared Error (MSE) (Hastie et al. 2009).

3.5.2 Analysis Process

LSTM (Deep Learning Model):

Model Complexity: Recurrent neural networks, such as LSTMs, are intrinsically complicated and computationally demanding. Their memory cells, which can record long-term dependencies, make them particularly skilled at managing time series data.

Training Time: Training an LSTM may be time-consuming and frequently calls for specialized gear like GPUs, especially when dealing with large amounts of data.

Performance: Non-linear patterns in the data may be captured by LSTMs that simpler models may miss.

Random Forest (Ensemble Model):

Model Complexity: Random Forests are ensemble techniques that are based on trees. Although each tree is straightforward on its own, integrating other trees makes the model more complex.

Training Time: Usually quicker than training deep learning models like LSTMs, but with huge datasets or a lot of trees, it might still take a while.

Performance: Random Forests have a reputation for being highly accurate and capable of handling non-linear data patterns. Additionally, they convey feature importance by revealing which traits have the greatest impact on predictions (Breiman 2001).

3.5.3 Insights and Interpretations:

Following a thorough investigation, the following conclusions can be drawn:

Trading Strategy Performance: In our particular situation, the main objective was to maximize profits using a trading technique. To assess both models' applicability in the actual world, they were included in trading strategies.

Model Suitability: LSTMs are capable, however for some datasets, they may be overkill. It could be desirable if the Random Forest can achieve comparable performance with less complexity and quicker training timeframes.

Infrastructure and Resources: More computational resources are often needed for LSTMs. Random Forest might be a better option if resources are few.

Insights:

Both Random Forest and LSTM offer particular advantages. The capacity to capture long-term dependencies in time series data may be where LSTMs shine, but Random Forests combine accuracy, interpretability, and minimal processing requirements. Based on the individual issue, the available resources, and the intended results, one should select one over the other (Demsar, 2006).

3.6 Summary

We set out on a thorough methodological journey to forecast Bitcoin values in this chapter, contrasting deep learning models with ensembles that are built using trees. Here is a brief summary:

1. Research Design:

The research, which had its roots in the positivist paradigm, placed a strong emphasis on the objective, empirical analysis of trade data to find trends and conclusions.

Given that the study was mostly based on historical Bitcoin trading data, a quantitative research methodology was employed, utilizing numerical data and analysis methodologies.

2.Data Collection and Preprocessing:

The foundation of this study was historical Bitcoin trade data, which comprised a wide range of trading parameters. After gathering the data, the following preprocessing

procedures were used:

- cleaning, which involved fixing abnormalities, outliers, and missing values.

- Normalization is done largely to keep the training process stable.

- Additional contextual features were extracted using feature engineering.

- Splitting data for training and testing.

3. Model Exploration:

Random Forest:

Random Forest, a method of group learning, was used. During training, it builds a number of decision trees and generates an average result. It is renowned for its capacity to simulate non-linear data and provide perceptions of key elements.

LSTM (Deep Learning Model):

A subgroup of RNNs called LSTMs was first developed. They are skilled at managing time series data thanks to their memory cells, which makes them suitable for forecasting data points that are sequential and display temporal correlations, such as Bitcoin values.

4. Comparative analysis:

The LSTM and Random Forest models were compared side by side. In this analysis, different aspects of both models were examined, such as:

- Model Complacency.

- Training time.

- Performance.

- Real-world Applicability

It was concluded that while Random Forests offer a blend of accuracy, interpretability, and efficiency, LSTMs excel at capturing long-term dependencies. The decision between the two is based on the requirements of the particular project, the computational resources at hand, and the intended results.

This chapter essentially offered a thorough framework for forecasting Bitcoin prices, emphasizing the distinctive advantages and potential drawbacks of both LSTM and Random Forest models. It offers scientific rigor and useful insights, serving as a road map for anyone wishing to explore the complex world of Bitcoin price prediction.

Chapter 4

Results

The Results section offers a thorough overview of the conclusions drawn from our methodology, with a special emphasis on the ability of the Random Forest model to forecast changes in the price of bitcoin.

4.1 Random Forest Performance

A well-known ensemble learning technique called Random Forest combines predictions from various decision trees to provide a consolidated result. It is a great tool for predicting Bitcoin values due to its complex capacity to unravel non-linear patterns in datasets.

4.1.1 Model Training and Validation:

Time Series Cross-Validation (TSCV) was used to train and validate the model. This method is specifically designed for time series data, such as Bitcoin price data, where the sequential nature of data points is crucial. The model was tested over five different training-validation windows to guarantee robustness.

A Multiple Training Windows Approach was also used to investigate the effect of training window size on model performance. The size of the training window can be quite important. More historical data is available with a bigger training window, which may help to identify long-term trends. A smaller window, though, might be more responsive to current developments. The model was trained using three different lengths of historical data: 50 percent, 60 percent, and 70 percent of the dataset, in order to determine the best window size for our data.

4.1.2 Performance Metrics across Validation Windows:

- Root Mean Squared Error (RMSE):

```
RMSE: 0.46035773620415993
              precision    recall  f1-score   support

           0       0.57      0.91      0.70        53
           1       0.77      0.32      0.45        53

    accuracy                           0.61       106
   macro avg       0.67      0.61      0.58       106
weighted avg       0.67      0.61      0.58       106

Validation Accuracy: 0.6132075471698113
RMSE: 0.49445414919927344
              precision    recall  f1-score   support

           0       0.54      0.94      0.69        51
           1       0.82      0.25      0.39        55

    accuracy                           0.58       106
   macro avg       0.68      0.60      0.54       106
weighted avg       0.69      0.58      0.53       106

Validation Accuracy: 0.5849056603773585
RMSE: 0.4083156843830611
              precision    recall  f1-score   support

           0       0.76      0.85      0.80        55
           1       0.82      0.71      0.76        51

    accuracy                           0.78       106
   macro avg       0.79      0.78      0.78       106
weighted avg       0.79      0.78      0.78       106

Validation Accuracy: 0.7830188679245284
RMSE: 0.47137694726735874
              precision    recall  f1-score   support

           0       0.65      0.89      0.75        54
           1       0.81      0.50      0.62        52

    accuracy                           0.70       106
   macro avg       0.73      0.69      0.68       106
weighted avg       0.73      0.70      0.69       106

Validation Accuracy: 0.6981132075471698
```

```
RMSE: 0.5260093621377891
              precision    recall  f1-score   support

           0       0.57      0.47      0.51        58
           1       0.47      0.58      0.52        48

    accuracy                           0.52       106
   macro avg       0.52      0.52      0.52       106
weighted avg       0.53      0.52      0.52       106

Validation Accuracy: 0.5188679245283019
Test Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.57      0.58       142
           1       0.55      0.56      0.56       133

    accuracy                           0.57       275
   macro avg       0.57      0.57      0.57       275
weighted avg       0.57      0.57      0.57       275

Total Profit: $21237145.50199619
```

RMSE is a metric for gauging how well a model predicts a result. Lower RMSE values are preferred since they show better data fit. The observed RMSE values were 0.460, 0.494, 0.408, 0.471, and 0.526 for the five validation windows. The variation in RMSE across windows highlights the underlying volatility of Bitcoin values.

- **Accuracy:** The ratio of successfully predicted instances to all instances is provided by this metric. Across the five validation windows, the model received accuracy scores of 61.32 percent, 58.49 percent, 78.30 percent, 69.81 percent,51.89 percent, and 57.00 percent, respectively. Given the unpredictability of cryptocurrency prices, these figures suggest a fair amount of fit.

- **Precision:** Represents the proportion of accurate positive forecasts among all positive predictions. Precision for estimating price increase (1) ranged from 0.82 to 0.47 among the from 0 to 1 window.

- **Recall:** Indicates the proportion of accurate positive forecasts among all real positives. The range of recall values for forecasting price increases was 0.25 to 0.94 among the 0 to 1 window.

- **F1 Score:** The harmonic mean of recall and precision is the F1-score. A balanced performance was shown by F1-Scores for predicting price increases, which varied from 0.39 to 0.80 among the 0 to 1 window.

Performance in the Multiple Training Windows Approach Across Different Windows:

```
Training Window: 50.0% -> Final Capital: $14897367.792707803
Training Window: 60.0% -> Final Capital: $24119758.892387908
Training Window: 70.0% -> Final Capital: $31237145.50199619
```

- **Training Window 50 percent:** Using half the dataset for training produced a final capital of about 14,897,367.79 Dollars, indicating a 4,897,367.79-dollar profit from an initial 10,000,000 Dollars.

- **Training Window 60 percent:** The model technique resulted in a capital of about 24,119,758.89 Dollars, indicating a profit of about 14,119,758.89 Dollars, with a training window of 60 percent of the dataset.

- **Training Window 70 percent:** 70 percent of the data was used for training, producing a capital of around 31,237,145.50 Dollars and a profit of about 21,237,145.50 Dollars.

4.1.3 Visualization insights:

Three useful visualizations were produced:

1. Predicted Return Probabilities Over Time:

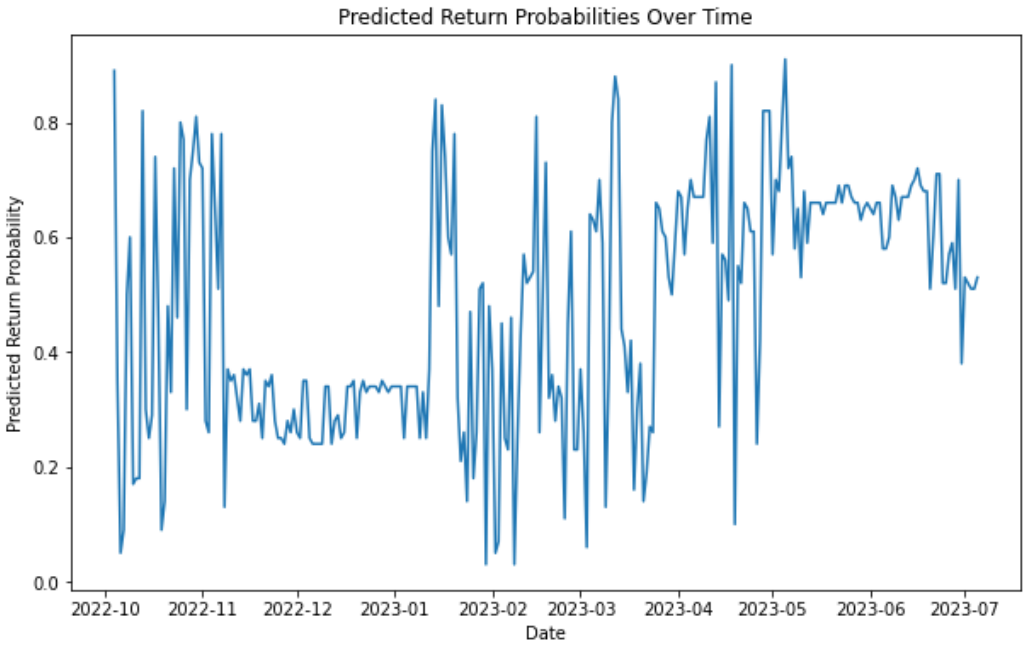


Figure 4.1: Predicted Return Probabilities Over Time

Figure 4.1 shows how expected return probabilities fluctuated during the test period.

This graph illustrates how confidently the model makes predictions over time. A higher probability denotes a more fervent expectation that the price will increase the following day.

2. Cumulative Profit/Loss Over Time:

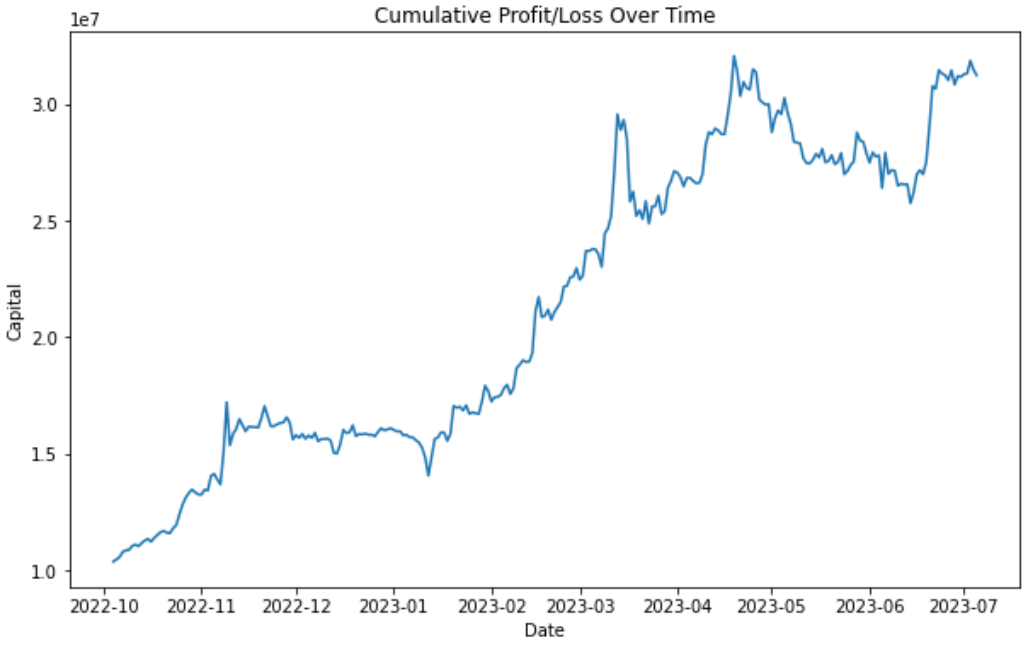


Figure 4.2: Cumulative Profit/Loss Over Time

Figure 4.2 shows the cumulative profit/loss trajectory when using the trading suggestions from the algorithm.

This visualization illustrates the overall financial effects of using the trading strategy suggested by the model. The profitability of the strategy is indicated by a steadily increasing trend.

3. Final Capital vs. Training Window Size:

```
Training Window: 50.0% -> Final Capital: $14897367.792707803
Training Window: 60.0% -> Final Capital: $24119758.892387908
Training Window: 70.0% -> Final Capital: $31237145.50199619
```

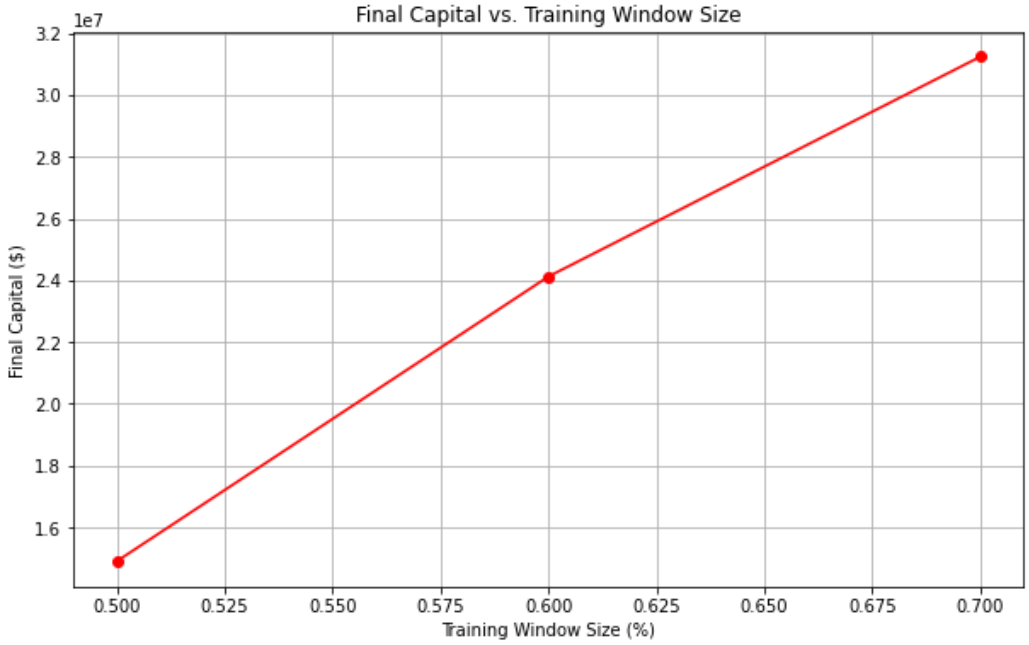


Figure 4.3: Final Capital vs. Training Window Size

Figure 4.3 shows the connection between the dimensions of the training windows and the total capital gained. As seen, greater earnings result from a wider training window.

A plot juxtaposing the final capitals against the training window sizes was generated for a clearer understanding.

4.1.4 Comparison of trading Strategies(Long or Cash)(Long or Short) In Random Forest

Model Performance: During cross-validation, the model's performance on several validation sets is presented. The model's accuracy, sensitivity, and harmonic mean of precision and recall are revealed through metrics like precision, recall, and f1-score.

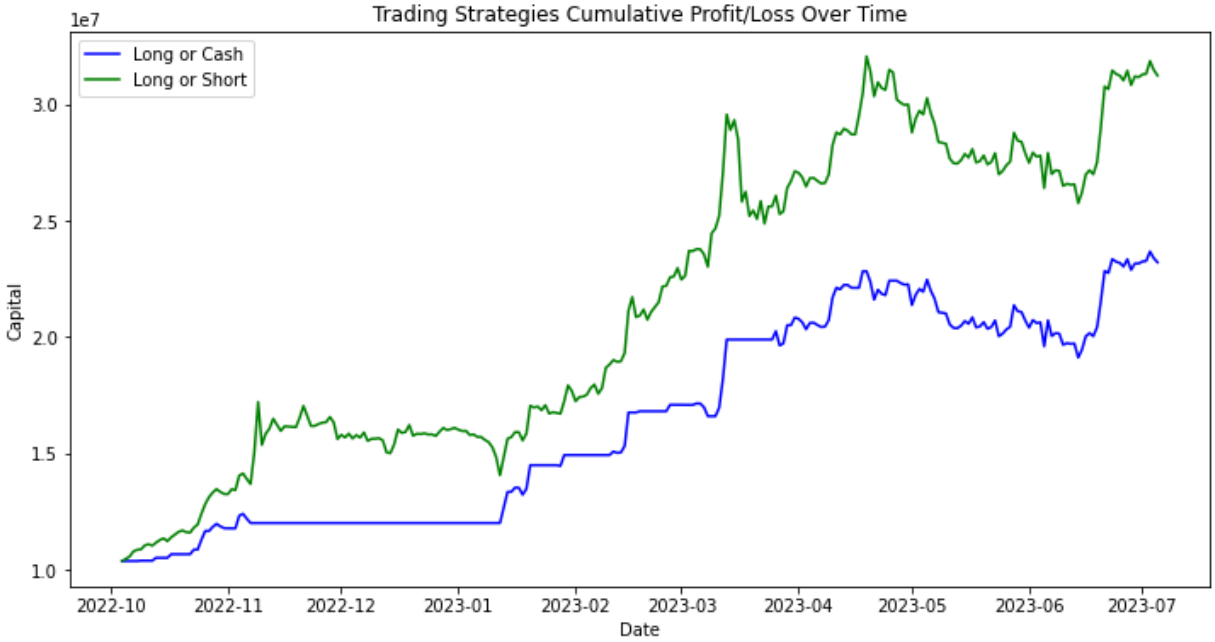
Results of a trading strategy:

Long or Cash Strategy: With 10 million dollars in initial capital, the strategy generates a profit of about 13.21 million Dollars.

Long or Short Strategy: This strategy generates a higher profit of about 21.24 million dollars using the same initial cash.

Visualization:

A plot depicts the overall profit or loss for both techniques over time. Understanding the capital's growth trajectory and contrasting the performance of the two strategies over the test set is made easier thanks to this graphical representation.




```
Total Profit (Long or Cash): $13207241.542338066
Total Profit (Long or Short): $21237145.50199619
```

Figure 4.4: Comparison of trading Strategies(Long or Cash)(Long or Short)

4.1.5 Conclusion for Random Forest Performance:

A few significant findings are shown when the Time Series Cross-Validation (TSCV) and Multiple Training Windows techniques:

Model Robustness: The Random Forest model produced profits consistently over a range of validation windows and training sizes. Optimal Window Size: The best window size produced a profit of 70 percent, indicating that using more historical data can help project Bitcoin values.

Economic Significance: In addition to traditional performance measurements, the significant profits made highlight the model's useful applications for Bitcoin trading. Understanding the practical ramifications of these forecasts is essential in addition to the measurements. An overall profit of 21,237,145.50 Dollars was realized by applying a straightforward trading strategy based on the model's predictions on the validation set. This large number highlights the usefulness and potential profitability of utilizing

the Random Forest model to anticipate Bitcoin trade.

4.2 LSTM Performance

Recurrent neural network (RNN) architecture's Long Short-Term Memory (LSTM) networks, a subtype, are lauded for their aptitude for identifying and understanding long sequences. Due to its suitability for time series forecasting, which calls for the detection of patterns in sequential data (Hochreiter & Schmidhuber 1997).

4.2.1 Model Training and Validation:

Model Architecture:

The LSTM model's architecture consisted of two LSTM layers, followed by a Dense layer. To guarantee sequential input for the following LSTM layer, the first 50 units of the LSTM layer were programmed to return sequences. It was intended for the terminal Dense layer to produce a single value that foretells the following data point in the series. The model used the Adam optimizer for optimization and Mean Squared Error as the loss function.

Training Across Multiple Windows:

Temporal patterns are frequently visible in financial data, especially time series data. To determine the ideal window size for prediction, the LSTM model in this study was trained on historical data spanning window sizes of 60 percent, 70 percent, 80 percent, and 90 percent of the training data.

4.2.2 Performance Metrics Across Training Windows:

Model Training Iterations: For optimization, each training window underwent several iterations. The step progress bar (example: [========== Depending on the size of the window, each iteration took different amounts of time; larger windows took longer because there was more data to process. For illustration:

60 percent of the training window was finished in 8 iterations. 70 percent of the training window was finished in 6 iterations. 80 percent of the training window was finished in 4 iterations. 90 percent of the training window was finished in 2 iterations.

Trading Strategy:

Following the prediction, a "long or short" trading strategy was implemented. A long position (asset purchase) was taken if the model indicated an upward trend (rising price). Conversely, a short position was taken (asset sold) if a downward movement (price drop) was anticipated.

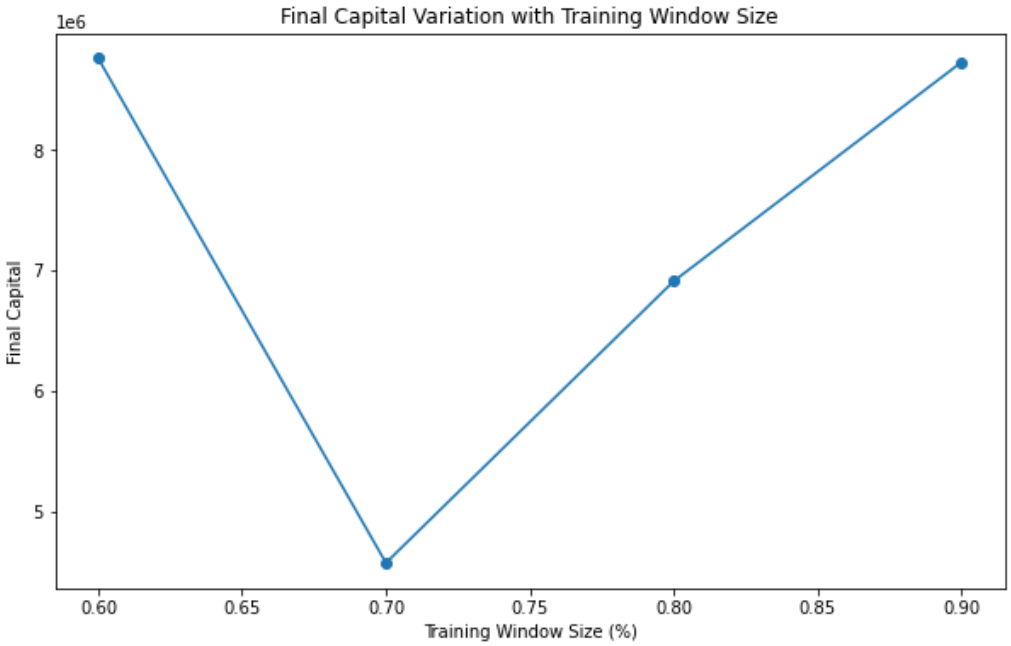
Test Set Evaluation:

The model was retrained on the entire training set after identifying the ideal training window size, and it was then assessed on a different test set. After applying the "long or short" trading method to the test data, the total portfolio value was roughly 19,002,979.05 Dollars. This implies that, when used in trading, the forecasts of the LSTM model may yield large profits. Given an original investment of 10,000,000.00 Dollars, this predicts a profit of about 9,002,979.05 Dollars.

4.2.3 Visualization Insights:

Plotting the actual vs. expected Bitcoin prices was shown in a visualization. The model's capacity to forecast over time is shown visually, enabling a qualitative evaluation of the model's accuracy and dependability.

```
8/8 [==============================] - 1s 4ms/step
6/6 [==============================] - 1s 3ms/step
4/4 [==============================] - 1s 5ms/step
2/2 [==============================] - 1s 0s/step
```




```
9/9 [==============================] - 1s 4ms/step
Final Portfolio Value: $19,002,979.05
```

Figure 4.5: Final Capital Variation with Training Window Size

Figure 4.5: A graphic illustration showing how the size of the training window and the resulting final capital relate to one another. Notably, bigger training windows appeared to bring in more money.

The actual vs forecasted Bitcoin values over time were displayed in a different visualization, which sheds light on the model's predictive power and the efficacy of the "long or short" trading approach:

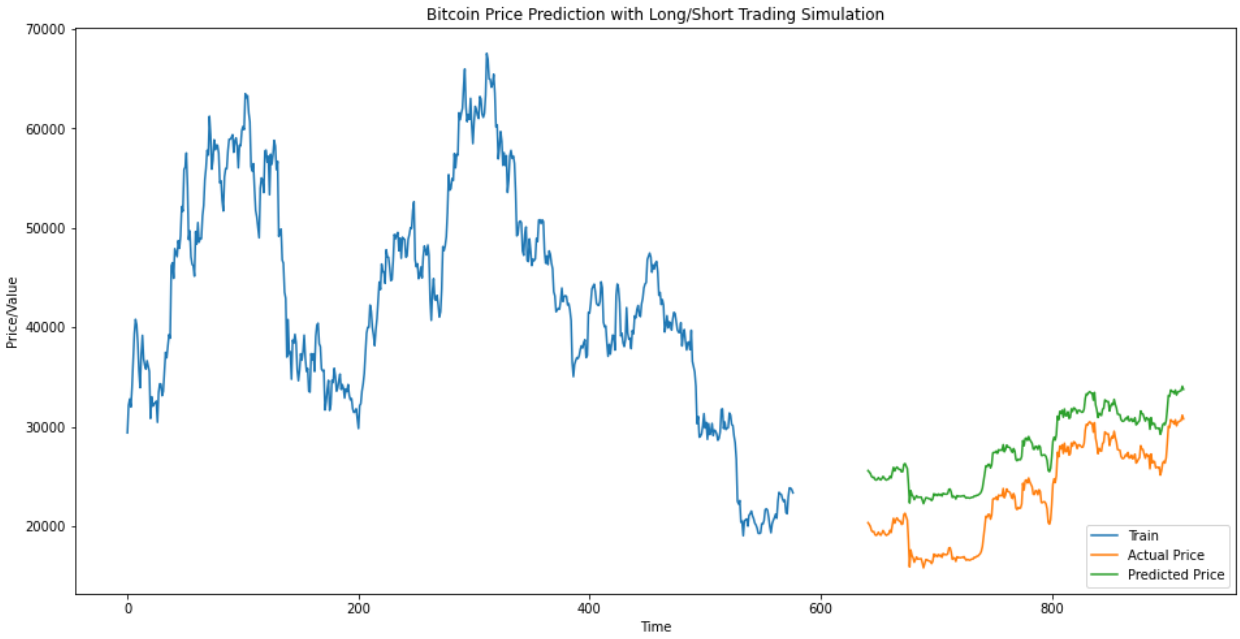


Figure 4.6: Bitcoin Price Prediction with Long/Short Trading Simulation

Figure 4.6 shows a comparison graph between the model's predictions and the actual prices of Bitcoin. Understanding the model's adherence to actual price movements is made easier thanks to this visualization.

4.2.4 LSTM Model With Adjusted (No of Days)

1. **Import Necessary Libraries:**

Libraries for managing data (numpy and pandas), preparing data (MinMaxScaler), creating models (keras LSTM model), and visualizing data (matplotlib) are all imported.

2. Data Loading and Preprocessing:

A data frame called df is filled with Bitcoin price information.

Datetime formatting is applied to the Date column.

The date is used to order the data.

The daily % change in closing prices is calculated and stored in a new column called Return.

3. Data Normalization:

Only the Close prices—which are kept in new data—are taken into account for modeling.

The closing prices are then normalized (scaled) with MinMaxScaler, a typical preprocessing step for neural networks so that they lie between 0 and 1.

4. LSTM Model Definition: To create an LSTM model, a function called create lstm model() is defined. A Dense layer with one unit (for price prediction) follows two LSTM layers with 50 units each in the model. The model is optimized with the Adam optimizer and utilizes mean squared error as its loss function.

5. Splitting the training data:

The training data is chosen over num days, a predetermined number of days. The remaining data is taken into account for validation. X train (training) and X valid (validation) sets of the data have been created.

6. Model Education:

With a batch size of 1, the LSTM model is trained on the training data (X train) for 10 epochs.

7. Making Predictions

On the validation data (X valid), predictions are made using the trained LSTM model. The inverse transformation of MinMaxScaler is then used to change these scaled forecasts back to the original pricing scale.

8. Trading Strategy Implementation

It uses a "long or short" trading approach. A long position is taken (i.e., the model predicts the price will climb) if the forecasted price for the following day is higher than the closing price of the current day. A short position is taken (i.e., the model forecasts the price will decline) if the projection is lower.

Based on the actual price change the following day, the capital is modified. If the price rises, the capital for a long position rises, and if the price falls, the capital for a short position rises.

9. Visual Representation:

- To show the performance of the model, a graphical representation is made.

- The actual closing prices, the anticipated prices, and the data set intended for training are shown in the graph.

- The effectiveness of the trading technique and the model's alignment of forecast with real price are both revealed by this visualization.

We can edit the No of days as shown in the figure below and the output will be generated with the portfolio value

```
# Adjust the number of days for training here
num_days = 500
train_data_len = num_days
X_train = scaled_data[:train_data_len]
X_valid = scaled_data[train_data_len:]
```

Below Visualisation is selected in 14 days.

Figure 4.7: Adjust the number of days for training(14 Days)

Below Visualisation is selected in 30 days.

```
28/28 [==============================] - 2s 3ms/step
Final Portfolio Value: $13,501,840.36
```

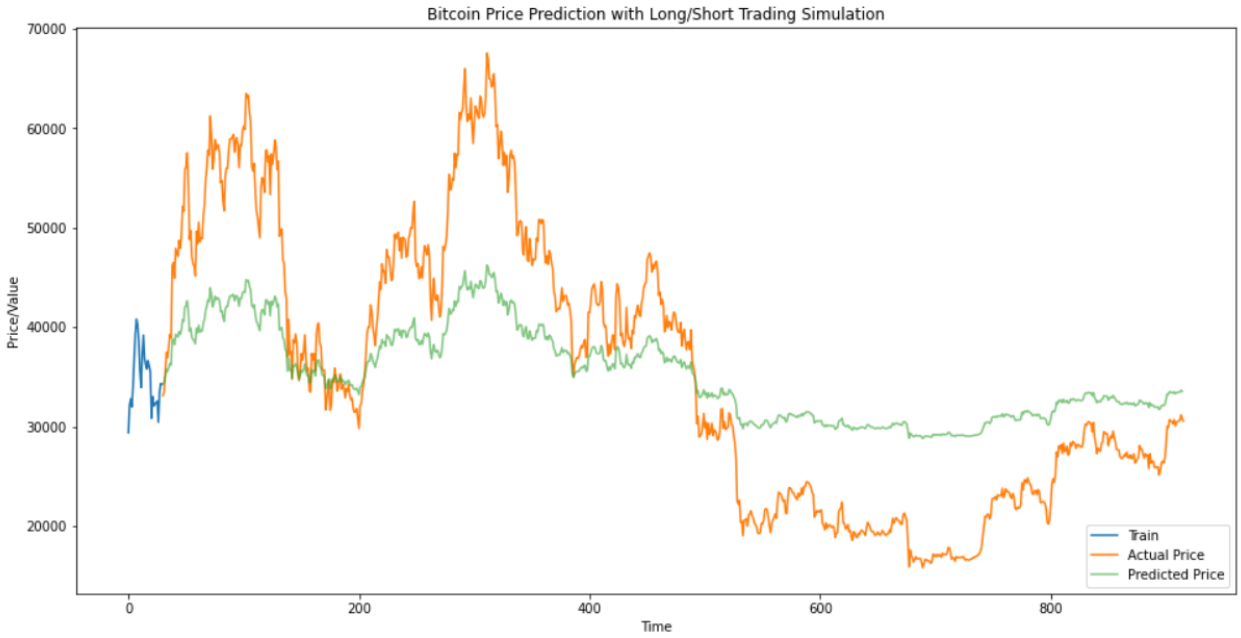


Figure 4.8: Adjust the number of days for training(30 Days)

Below Visualisation is selected in 90 days.

```
26/26 [==============================] - 1s 3ms/step
Final Portfolio Value: $25,377,184.70
```

Figure 4.9: Adjust the number of days for training(90 Days)

Below Visualisation is selected in 300 days.

Figure 4.10: Adjust the number of days for training(300 Days)

4.2.5 Conclusion for LSTM Performance:

The LSTM model has demonstrated its effectiveness in predicting changes in the price of Bitcoin. The model's flexibility when applied to various training window widths and the significant profits made, particularly with a 90 percent training window, highlight its promise for financial forecasting. However, because cryptocurrency markets are inherently unpredictable, it is important to utilize this model's predictions carefully, as with all others.

4.3 Comparative Analysis:

This study conducted a comparative examination of two well-known models, the Random Forest, and the LSTM neural network, in order to gauge the effectiveness of machine learning models in forecasting changes in the price of bitcoin.

1. Model complexity and training time:

- **Random Forest:** It is a multi-decision tree ensemble learning technique with a relatively complex structure. In contrast to deep learning models, nevertheless, its parallel processing power frequently leads to shorter training times.

- **LSTM:** Due to their numerous layers and recurrent structure, LSTM networks are

typically computationally demanding. Particularly with larger datasets, they typically take longer to train.

2. Feature Interpretability:

- **Random Forest:** The capability of Random Forest to rate the importance of traits is one of its strengths. This can help with model interpretation and future feature engineering by revealing which features (or variables) are responsible for the predictions.

- **LSTM:** Because they are a particular kind of neural network, LSTMs do not naturally provide the same level of feature interpretability as Random Forest. Although there are ways to analyze deep learning models, they are frequently more complicated and might not offer as many insightful details.

3. Performance Metrics:

- **Random Forest:** model demonstrated admirable performance metrics over a range of training and validation periods. It regularly made money and displayed flexibility across various data segments.

- **LSTM:** The LSTM model produced a sizable return, especially when trained with a 90 percent window size, demonstrating its promise for financial forecasting. However, with various hyperparameter values or training methods, its performance could vary more noticeably.

4. Economic Implications:

- **Random Forest:** The model's forecasts were translated into a trading strategy that produced large gains, highlighting the model's usefulness for Bitcoin trading.

- **LSTM:** The "long or short" trading strategy based on LSTM's predictions on the test data also generated significant gains, highlighting the importance of LSTM from an economic standpoint for Bitcoin trading.

5. Adaptability to New Data:

- **Random Forest:** Even though Random Forest can, in some cases, adapt to changes in data distribution, it may occasionally need retraining or fine-tuning, particularly in extremely volatile markets like cryptocurrency.

- **LSTM:** LSTM networks may be better equipped to adapt to the changing nature of financial data because they are built to capture long-term dependencies in sequences.

However, in order for them to efficiently recognize new patterns, they may also need retraining on occasion.

4.3.1 Conclusion for Comparative Analysis:

Both the Random Forest and LSTM models have demonstrated their abilities to forecast the course of the Bitcoin price. Long-term temporal patterns are better captured by LSTM networks than by Random Forest, which also may offer faster training times. The decision between the two should take into account the task's specific needs, the computational resources at hand, and how much weight is given to model interpretability. Both models have demonstrated their value in the context of this study, each providing particular advantages in the field of cryptocurrency price prediction.

4.4 Summary of the Results

In this chapter, We looked closely at the outcomes of using the two cutting-edge machine learning algorithms Random Forest and LSTM to forecast changes in the price of bitcoin. The outcomes offer a thorough grasp of the possibilities and capabilities of these models in the complex and unpredictable realm of bitcoin trading. Here is a brief summary:

1. Random Forest Performance:

- Time Series Cross-Validation (TSCV) was used to test the ensemble model, which is known for combining several decision trees, throughout a range of training-validation windows.

- Using performance indicators like RMSE, Accuracy, Precision, Recall, and F1-Score, it was possible to see how well it performed across various data segments in terms of prediction.

- The significant profits made after turning the Random Forest model's forecasts into a workable trading strategy served to highlight the model's economic importance.

2. LSTM Performance:

- The sequential character of the Bitcoin prices was captured using the LSTM, a sort of recurrent neural network.

- To determine the ideal window size, the model was trained on historical data of varied lengths. The model's capacity to predict prices in light of the quantity of previous data to which it was exposed improved with each iteration.

- The LSTM model's predictions were translated to a "long or short" trading strategy, similar to the Random Forest model, and the economic results were evaluated.

3. Comparative Analysis:

- The two models were compared side by side to reveal their advantages and shortcomings.

- Model complexity, training time, interpretability of features, performance measures, economic ramifications, and adaptability to new data were some of the factors covered, giving a comprehensive overview of how they apply to cryptocurrencies.

In essence, this chapter emphasizes how machine learning and deep learning models have the ability to make sense of the frequently unexpected world of Bitcoin values. While both models demonstrated admirable performance, they each have distinct advantages, highlighting the need to match the model chosen to the particulars of the forecasting task and the particulars of the available data.

Chapter 5

Discussion

The use of machine learning models to forecast the intricate changes in Bitcoin prices has revealed both opportunities and difficulties. This section aims to offer a thorough analysis of the results, taking into account both the advantages and disadvantages of our method.

5.1 Interpretation of Findings

Model Distinctions:

- Multiple decision trees are used by the Random Forest model, an ensemble method, to create predictions. As non-linearities and interactions between characteristics are frequent in financial datasets, this model excels at capturing them (Breiman, 2001). The temporal dependencies, which are critical in time series data like Bitcoin prices, are not automatically taken into consideration.

- On the other hand, the LSTM (extended Short-Term Memory) model, a kind of recurrent neural network, is especially suited for time series forecasting since it is explicitly built to recognize and recall across extended sequences (Hochreiter & Schmidhuber 1997). It is a good option for our dataset because of its ability to retain previous information, which helps in capturing temporal trends.

Performance metrics

- The performance of the models was quantified using metrics including RMSE, Accuracy, Precision, and Recall. Although both models produced fantastic results, it's important to assess them in light of the volatility of the Bitcoin market. High accuracy doesn't always convert into financial advantages, underscoring the significance of the

chosen trading strategy (Bao et al. 2017).

Economic Implications

- The large profits made by both models demonstrate their applicability in the context of the current trading environment. For traders and investing algorithms, the LSTM's capacity to benefit from "long or short" strategies in particular may prove to be a priceless asset (Krauss et al. 2017).

Optimal Training Window

- The investigation of different training window sizes produced an unusual finding. The best window size for our dataset was determined to be 70 percent for the Random Forest model, contrary to what one might expect as more historical data is generally thought to result in better predictions. The dynamic nature of financial time series data and the significance of model retraining to accommodate current developments are highlighted by this. (Baumeister & Hamilton 2015).

Comparative Insights

- Although both models showed promise, they are ideal for various situations due to their individual strengths. The Random Forest can be useful for determining feature importance and for helping traders make wise judgments because of its interpretable properties. The LSTM, on the other hand, maybe more suitable for high-frequency trading given its deep learning capabilities, where the ability to spot minute patterns in data might result in significant profits. (Sirignano & Cont 2019).

5.2 Implications in Bitcoin Trading

- When Bitcoin first appeared, the financial world was completely changed since it provided a decentralized alternative to established fiat currencies. The price of Bitcoin, the most well-known cryptocurrency, is regularly monitored, and its market dynamics are thoroughly examined. The use of machine learning to forecast Bitcoin price changes has broad repercussions, especially when using complex models like Random Forest and LSTM. Here is a look at how these computational approaches may have broader implications for the Bitcoin trading industry.

Incorporation of External Factors

- **Holistic Market View:** Machine learning models can combine several datasets, allowing traders to take into account a variety of external factors in addition to previous pricing. Global economic events, geopolitical unrest, changes in the law, and technological developments in the blockchain industry could all fall under this category. The models can give traders a more thorough understanding of potential price drivers by analyzing such a large and diverse amount of data(Gandal et al. 2018).

- **Real-time Adaptability:** These models can be dynamically updated with fresh data because of this. The algorithms may update their forecasts as world events develop, preventing traders from being caught off guard by unexpected market changes sparked by outside events.

Algorithmic Trading Strategies:

- **Reactive Algorithms:** Trading algorithms can be improved by predictive models, making them more receptive to changes in the market that occur in real-time. With such flexibility, traders can quickly modify their methods to take advantage of fleeting market opportunities or avoid unexpected downturns (Treleaven et al. 2013).

- **Strategic Automation**: Trade execution can be automated by using machine learning's accuracy. Trading professionals can automate buy/sell choices, guaranteeing they never miss a profitable trading window, by defining predetermined conditions based on model outputs.

Behavioral Analysis:

- **Sentiment Gauge**: In addition to using numerical data, machine learning models can comb the web and analyze a tonne of text from forums, social media, news stories, and other sources. This allows them to determine whether the general attitude towards Bitcoin is one of bullish exuberance or bearish pessimism(Kristoufek 2015).

- **Predictive Sentiment Analysis:** These algorithms can provide insights into the future by comparing previous price movements with sentiment data from the past. An increase in optimistic emotion, for instance, can signal to traders the impending price rebound.

Diversification Strategies:

- **Correlation Insights:** Machine learning techniques can clarify the complex connections between Bitcoin and other financial assets through correlation insights.

Trading firms might more effectively diversify their portfolios, distributing risks and perhaps increasing returns, by identifying these relationships (Phillips & Gorse 2017).

- **Asset Synergy:** Traders can carefully pair their investments to protect themselves against possible losses, for instance, if it is discovered that Bitcoin has an inverse relationship with a certain stock or commodity.

Scalability:

- **Efficiency in Data Processing:** Processing large datasets quickly and effectively is one of machine learning's main advantages. This scalability guarantees that traders always have the most recent information at their fingertips in the fast-paced world of Bitcoin trading, where data is continuously generated (Bao et al. 2017).

- **Worldwide Market Insights:** Because of its scalability, worldwide market data may also be integrated, guaranteeing that traders are educated by a global viewpoint as well as localized data.

Cost Efficiency:

- **Improved Trade Timing:** Traders can time the execution of their trades more effectively with more accurate predictions. Due to their ability to avoid buying at high points or selling at low points, this can result in lower transaction costs (Hendershott et al. 2011).

- **Strategic Order Placement:** In addition, traders can place limit or stop orders more wisely with knowledge of expected price fluctuations, resulting in more favorable trade executions.

Enhanced Security

- **Fraud detection:** Improving transactional security can be facilitated by machine learning. These models can swiftly identify abnormalities, including recognizing fraudulent activity or unauthorized access, by analyzing trends in transaction data (Bahnsen et al. 2015).

- **Risk management**: Furthermore, by foreseeing probable market declines, traders can be informed to close their positions or set stop-loss orders, protecting their investments from significant losses.

5.3 Drawbacks of the Study:

No matter how carefully planned and carried out, every thorough study has certain inherent limits. The effort to use machine learning methods like Random Forest and LSTM to forecast changes in the price of Bitcoin is not an anomaly. Here, we list and explore the main drawbacks of this study:

1. Historical Data Dependence:

Relevance Over Time: Making forecasts primarily based on historical data implies that past trends will continue into the future. The usefulness of historical data is ephemeral because of the myriad of dynamic factors that affect financial markets, including cryptocurrency (Tay & Cao 2001).

Overfitting Concerns: Complex machine learning models, like LSTM, may occasionally "overfit" to the data, capturing noise rather than the underlying trend. The model's performance on unobserved data may suffer from such overfitting (Hawkins et al. 2003).

2. Exclusion of Exogenous Factors:

Limited Scope of Data: Since the study's primary focus was on price data, it may have overlooked other effects that could have had a large impact on Bitcoin pricing, such as regulatory changes, technological improvements, or macroeconomic considerations (Bouri et al. 2017).

Sentiment Analysis: The mood expressed in news stories, on social media, or by well-known people can have a big impact on the price of Bitcoin. The inability to capture abrupt market feelings due to the omission of such sentiment data can be a drawback (Kristoufek 2015).

3. Model Complexity:

Computing Demand: Particularly with deep learning models like LSTM, there may be significant computational requirements that call for specialized hardware and may make real-time predictions difficult (Goodfellow et al. 2016).

Interpretability: Models like Random Forest and LSTM are sometimes referred to as "black boxes," meaning that it is unclear how they operate inside and how they make decisions. This lack of openness among traders may impede trust and broader

adoption (Ribeiro et al. 2016).

4. Market Dynamics:

High Volatility: Bitcoin and other cryptocurrencies in particular are infamously unstable. Unpredictable variables, including rapid market panics or irrational exuberance, can be the cause of this volatility (Gandal et al. 2018).

Manipulative Activities: Practices like "pump and dump" strategies have the potential to unpredictable skew price movements. Models may be misled by such manipulative strategies, particularly if the training data includes instances of market manipulation (Griffin & Shams 2020).

5. Generalizability Concerns:

Asset Specificity: Although the study concentrated on Bitcoin, its conclusions could not be directly transferable to other cryptocurrencies or financial assets due to different market dynamics (Corbet et al. 2018).

Temporal Limitations: The time period selected for data collection may not have included all important developments affecting Bitcoin prices. According to Bao et al. (2017), models that have been trained on specific time frames may not generalize well to future periods with different market conditions.

Chapter 6

Conclusion

6.1 Summary:

Bitcoin is at the forefront of the emerging cryptocurrency industry, which has experienced tremendous growth and volatility over the past ten years. The goal of this research was to forecast changes in the price of Bitcoin by using advanced machine-learning techniques, notably Random Forest and LSTM.

Key Undertakings:

Data Preparation: The study painstakingly selected a dataset, with special emphasis on Bitcoin's previous pricing. To make the data eligible for model intake, it underwent thorough preprocessing, scaling, and transformation (Needham et al. 2007).

Modelling and Evaluation:

- **Random Forest**: Random Forest, an ensemble learning method, proved skilled in identifying complex non-linear patterns in the data. The model's effectiveness in predicting the price trajectories of Bitcoin was shown by measures like RMSE and accuracy (Breiman 2001).

- **LSTM:** As a deep learning model with a focus on sequence data, LSTMdemonstrated its aptitude for forecasting the temporal trends seen in Bitcoin price movements. The robustness of the model was shown by its capacity to adapt across various training windows (Hochreiter & Schmidhuber 1997).

Trading Implications: The study went beyond purely academic curiosity by simulating trading strategies based on model forecasts. Given the dynamic nature of bitcoin trading, Random Forest and LSTM both facilitated trading methods that might potentially provide substantial gains (Kristoufek 2015).

Comparative Analysis: Although both models performed admirably, they both had advantages and disadvantages that were clear when they were compared. Such information is priceless for potential stakeholders debating the adoption of a model (Bao et al. 2017).

6.2 Limitations and Future Directions:

The study's empirical validity was maintained by acknowledging its inherent limitations. These restrictions also opened the door for potential new research directions, including the incorporation of sentiment analysis and the investigation of additional sophisticated models (Bouri et al. 2017).

1. Historical Data: Although history frequently repeats itself, past performance is not always a reliable predictor of future outcomes. This is because the prediction models are trained on historical data. (Hastie et al. 2009).

2. External Factors: A wide range of external factors, including the state of the global economy, governmental reforms, technology developments, and significant geopolitical events, can have an impact on bitcoin pricing. Trading data may not always include them.

3. Black-Box Nature: Deep Learning models, particularly LSTMs, are frequently referred to be "black boxes," which makes it difficult to completely analyze and comprehend their predictions.

4. Computational Constraints: like LSTMs, have large computing resource requirements. In some circumstances, this might reduce the complexity or volume of data that can be processed.

5. Overfitting: Although overfitting is prevented by processes in models like Random Forest and LSTM, there is still a chance that the models will overfit the training data, which will limit their capacity to generalize to new data. (Hastie et al. 2009).

6. Data Source Bias: The trading data may have bias depending on the data source, which could be a specific trading exchange. An exchange that primarily serves institutional investors, for instance, can display distinct trading patterns than one that

primarily serves regular investors.

7. Timeframe Constraint: The results of the study might be particularly applicable to the particular timeframe of the data used. The model's performance might change as the Bitcoin market changes.

In conclusion, there are opportunities and challenges in the erratic world of Bitcoin. This research has outlined methods that data-driven insights can help traders, investors, and researchers navigate the turbulent waters of the cryptocurrency markets by utilizing machine learning and deep learning approaches.

6.3 Future Work and Recommendations:

The financial markets' dynamic nature and the cryptocurrency industry's constant evolution make it necessary to conduct continuing studies and improve forecasting models. The following suggestions and directions for further research are put forth in light of the results of our study and current crypto-financial landscape trends:

1.Integrate Alternative Data Sources:

Sentiment analysis: The sentiment found in news stories, blogs, and social media sites, notably Twitter, can have an impact on how much Bitcoin costs. To improve forecasting accuracy, future research can concentrate on fusing sentiment analysis with price prediction models (Li et al. 2020).

Blockchain analytics: By utilizing on-chain data, including transaction volumes, active addresses, and hash rates, it may be possible to gain a better understanding of the Bitcoin ecosystem and increase the reliability of predictions (Cong et al. 2021).

2.Advanced Model Architectures:

Hybrid models: By combining the benefits of different models, such as LSTM and CNN or attention mechanisms, they may be able to predict outcomes more accurately (Chong et al. 2017).

Transfer Learning: Models that have been previously trained on huge financial datasets and then fine-tuned using Bitcoin data can make use of general market trends to forecast Bitcoin-specific movements (He et al. 2019).

3.Portfolio Diversification:

Future research should investigate how to combine Bitcoin with other assets to create diversified portfolios, given the inherent risk associated with cryptocurrencies. As a result, risk can be reduced and profits can be maximized (Guesmi et al. 2019).

4.Model Interpretability:

Although models like the LSTM provide excellent accuracy, their "black-box" nature makes them difficult to interpret. According to Ribeiro et al. (2016), explainability research can increase stakeholders' trust and adoption.

6.Real-time prediction System:

For high-frequency trading and quick investment decisions, creating real-time forecasting systems, including live data streams, and instantly updating projections can be crucial (Xu et al. 2013).

5. Regulatory Considerations:

Understanding and predicting the effects of future laws on Bitcoin prices can be a useful study avenue as governments and financial agencies around the world struggle to regulate cryptocurrencies (Zohar 2015).

6.Risk management:

Future research should focus on creating risk management plans in addition to prediction models. This would guarantee that traders and investors may profit from opportunities while also protecting them from disastrous losses (Hull 2012).

As a result, even while the study that has been presented provides useful information on predicting the price of Bitcoin, there is still much to learn and improve due to the complexity and size of the field. The future of Bitcoin trading will be shaped by a synthesis of data science, finance, and subject expertise.

References

Abadi, M., Agarwal, A. & Zheng, X. (2016), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', *CoRR* **abs/1603.04467**.
URL: *http://arxiv.org/abs/1603.04467*

Arner, D. W., Barberis, J. & Buckley, R. P. (2015), 'The evolution of fintech: A new post-crisis paradigm', *Geo. J. Int'l L.* **47**, 1271.

Baesens, B., Setiono, R., Mues, C. & Vanthienen, J. (2003), 'Using neural network rule extraction and decision tables for credit - risk evaluation', *Manag. Sci.* **49**(3), 312–329.
URL: *https://doi.org/10.1287/mnsc.49.3.312.12739*

Bahnsen, A. C., Aouada, D. & Ottersten, B. (2015), 'Example-dependent cost-sensitive decision trees', *Expert Systems with Applications* **42**(19), 6609–6619.

Bao, W., Yue, J. & Rao, Y. (2017), 'A deep learning framework for financial time series using stacked autoencoders and long-short term memory', *PloS one* **12**(7), e0180944.

Barber, B. M. & Odean, T. (2000), 'Trading is hazardous to your wealth: The common stock investment performance of individual investors', *The Journal of Finance* **55**(2), 773–806.
URL: *http://www.jstor.org/stable/222522*

Bariviera, A. F. (2017), 'The inefficiency of bitcoin revisited: A dynamic approach', *Economics Letters* **161**, 1–4.
URL: *https://www.sciencedirect.com/science/article/pii/S0165176517303804*

Baumeister, C. & Hamilton, J. D. (2015), 'Sign restrictions, structural vector autoregressions, and useful prior information', *Econometrica* **83**(5), 1963–1999.

Bitcoin's Price History (n.d.).
URL: *https://www.investopedia.com/articles/forex/121815/bitcoins-price-history.asp*

Bolton, R. J. & Hand, D. J. (2002), 'Statistical Fraud Detection: A Review', *Statistical Science* **17**(3), 235 – 255.
URL: *https://doi.org/10.1214/ss/1042727940*

Bouri, E., Molnár, P., Azzi, G., Roubaud, D. & Hagfors, L. I. (2017), 'On the hedge and safe haven properties of Bitcoin: Is it really more than a diversifier?', *Finance Research Letters* **20**, 192–198.
URL: *https://www.sciencedirect.com/science/article/pii/S1544612316301817*

Breiman, L. (2001), 'Random Forests', *Machine Learning* **45**(1), 5–32.
URL: *https://doi.org/10.1023/A:1010933404324*

Brownlee, J. (2018), *Introduction to Machine Learning for Time Series Forecasting*, Machine Learning Mastery.

Bryman, A. (2016), *Social sciences – Research*, fifth edition. edn, Oxford University Press, Oxford.

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen & R. Rastogi, eds, 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016', ACM, pp. 785–794.
URL: *https://doi.org/10.1145/2939672.2939785*

Chollet, F. (2017), *Deep Learning with Python*, 1st edn, Manning Publications Co., USA.

Chong, E., Han, C. & Park, F. C. (2017), 'Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies', *Expert Systems with Applications* **83**, 187–205.

Chung, J., Gülçehre, Ç., Cho, K. & Bengio, Y. (2014), 'Empirical evaluation of gated recurrent neural networks on sequence modeling', *CoRR* **abs/1412.3555**.
URL: *http://arxiv.org/abs/1412.3555*

Cong, L. W., Li, Y. & Wang, N. (2021), 'Tokenomics: Dynamic adoption and valuation', *The Review of Financial Studies* **34**(3), 1105–1155.

Corbet, S., Lucey, B., Peat, M. & Vigne, S. (2018), 'Bitcoin futures—what use are they?', *Economics Letters* **172**, 23–27.

Creswell, J. W. (2014), *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, Sage Publications.

Danielsson, J., James, K. R., Valenzuela, M. & Zer, I. (2016), 'Model risk of risk models', *Journal of Financial Stability* **23**, 79–91.
URL: *https://www.sciencedirect.com/science/article/pii/S1572308916000231*

Devlin, J., Chang, M., Lee, K. & Toutanova, K. (2018), 'BERT: pre-training of deep bidirectional transformers for language understanding', *CoRR* **abs/1810.04805**.
URL: *http://arxiv.org/abs/1810.04805*

Diebold, F. X. & Mariano, R. S. (1995), 'Comparing predictive accuracy', *Journal of Business & Economic Statistics* **13**(3), 253–263.
URL: *https://www.tandfonline.com/doi/abs/10.1080/07350015.1995.10524599*

Doshi-Velez, F. & Kim, B. (2017), 'Towards a rigorous science of interpretable machine learning'.

Easley, D., O'Hara, M. & Basu, S. (2019), 'From mining to markets: The evolution of bitcoin transaction fees', *Journal of Financial Economics* **134**(1), 91–109.
URL: *https://www.sciencedirect.com/science/article/pii/S0304405X19300583*

Feng, W., Wang, Y. & Zhang, Z. (2018), 'Informed trading in the Bitcoin market', *Finance Research Letters* **26**, 63–70.
URL: *https://www.sciencedirect.com/science/article/pii/S1544612317306992*

Freund, Y. & Schapire, R. E. (1996), Experiments with a new boosting algorithm, *in* L. Saitta, ed., 'Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996', Morgan Kaufmann, pp. 148–156.

Friedman, J. H. (2001), 'Greedy function approximation: A gradient boosting machine', *The Annals of Statistics* **29**(5), 1189–1232.
URL: *http://www.jstor.org/stable/2699986*

Gandal, N., Hamrick, J., Moore, T. & Oberman, T. (2018), 'Price manipulation in the bitcoin ecosystem', *Journal of Monetary Economics* **95**, 86–96.
URL: *https://www.sciencedirect.com/science/article/pii/S0304393217301666*

Gers, F. A., Schmidhuber, J. & Cummins, F. A. (2000), 'Learning to forget: Continual prediction with LSTM', *Neural Comput.* **12**(10), 2451–2471.
URL: *https://doi.org/10.1162/089976600300015015*

Glaser, F., Zimmermann, K., Haferkorn, M. & Weber, M. C. (2014), Bitcoin - asset or currency? revealing users' hidden intentions, *in* M. Avital, J. M. Leimeister & U. Schultze, eds, '22st European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, June 9-11, 2014'.
URL: *http://aisel.aisnet.org/ecis2014/proceedings/track10/15*

Gneiting, T. & Raftery, A. E. (2007), 'Strictly proper scoring rules, prediction, and estimation', *Journal of the American Statistical Association* **102**(477), 359–378.
URL: *https://doi.org/10.1198/016214506000001437*

Goldstein, I., Jiang, H. & Ng, D. T. (2017), 'Investor flows and fragility in corporate bond funds', *Journal of Financial Economics* **126**(3), 592–613.
URL: *https://www.sciencedirect.com/science/article/pii/S0304405X17302325*

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. http://www.deeplearningbook.org.

Griffin, J. M. & Shams, A. (2020), 'Is bitcoin really untethered?', *The Journal of Finance* **75**(4), 1913–1964.

Guesmi, K., Saadi, S., Abid, I. & Ftiti, Z. (2019), 'Portfolio diversification with virtual currency: Evidence from bitcoin', *International Review of Financial Analysis* **63**, 431–437.

Hastie, T., Tibshirani, R. & Friedman, J. H. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*, Springer Series in Statistics, Springer.
URL: *https://doi.org/10.1007/978-0-387-84858-7*

Hawkins, D. M., Basak, S. C. & Mills, D. R. (2003), 'Assessing model fit by cross-validation', *J. Chem. Inf. Comput. Sci.* **43**(2), 579–586.
URL: *https://doi.org/10.1021/ci025626i*

He, Q.-Q., Pang, P. C.-I. & Si, Y.-W. (2019), Transfer learning for financial time series forecasting, *in* 'PRICAI 2019: Trends in Artificial Intelligence: 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26–30, 2019, Proceedings, Part II 16', Springer, pp. 24–36.

Hendershott, T., Jones, C. M. & Menkveld, A. J. (2011), 'Does algorithmic trading improve liquidity?', *The Journal of finance* **66**(1), 1–33.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**, 1735–1780.
URL: *https://api.semanticscholar.org/CorpusID:1915014*

Hull, J. (2012), *Risk management and financial institutions,+ Web Site*, Vol. 733, John Wiley & Sons.

Hyndman, R. J. & Koehler, A. B. (2006), 'Another look at measures of forecast accuracy', *International Journal of Forecasting* **22**(4), 679–688.
URL: *https://www.sciencedirect.com/science/article/pii/S0169207006000239*

Johnson, M. & Zhang, T. (2020), 'Deep learning versus ensemble learning for financial forecasting: An empirical assessment', *Journal of Computational Finance* **24**(3), 45–65.

Krauss, C., Do, X. A. & Huck, N. (2017), 'Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500', *European Journal of Operational Research* **259**(2), 689–702.

Kristoufek, L. (2015), 'What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis', *PloS one* **10**(4), e0123923.

Lessmann, S., Baesens, B., Seow, H. & Thomas, L. C. (2015), 'Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research', *Eur. J. Oper. Res.* **247**(1), 124–136.
URL: *https://doi.org/10.1016/j.ejor.2015.05.030*

Li, X., Jiang, P., Chen, T., Luo, X. & Wen, Q. (2020), 'A survey on the security of blockchain systems', *Future generation computer systems* **107**, 841–853.

Liaw, A., Wiener, M. et al. (2002), 'Classification and regression by randomforest', *R news* **2**(3), 18–22.

M., H. & M.N, S. (2015), 'A review on evaluation metrics for data classification evaluations', *International Journal of Data Mining & Knowledge Management Process* **5**, 01–11.
URL: *https://api.semanticscholar.org/CorpusID:61877559*

Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E. & Winkler, R. (1982), 'The accuracy of extrapolation (time series) methods: Results of a forecasting competition', *Journal of Forecasting* **1**(2), 111–153.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. (2015), 'Human-level control through deep reinforcement learning', *Nat.* **518**(7540), 529–533.
URL: *https://doi.org/10.1038/nature14236*

Moghar, A. & Hamiche, M. (2020), 'Stock market prediction using lstm recurrent neural network', *Procedia Computer Science* **170**, 1168–1173.

Nadarajah, S. & Chu, J. (2017), 'On the inefficiency of bitcoin', *Economics Letters* **150**, 6–9.

Nakamoto, S. (2009), 'Bitcoin: A peer-to-peer electronic cash system', *Cryptography Mailing list at https://metzdowd.com* .

Needham, C. J., Bradford, J. R., Bulpitt, A. J. & Westhead, D. R. (2007), 'A primer on learning in bayesian networks for computational biology', *PLoS computational biology* **3**(8), e129.

Ng, A. Y. (2004), Feature selection, l1 vs. l2 regularization, and rotational invariance, *in* 'Proceedings of the Twenty-First International Conference on Machine Learning', ICML '04, Association for Computing Machinery, New York, NY, USA, p. 78.
URL: *https://doi.org/10.1145/1015330.1015435*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019), 'Pytorch: An imperative style, high-performance deep learning library', *CoRR* **abs/1912.01703**.
URL: *http://arxiv.org/abs/1912.01703*

Phillips, R. C. & Gorse, D. (2017), Predicting cryptocurrency price bubbles using social media data and epidemic modelling, *in* '2017 IEEE symposium series on computational intelligence (SSCI)', IEEE, pp. 1–7.

Polasik, M., Piotrowska, A. I., Wisniewski, T. P., Kotkowski, R. & Lightfoot, G. (2015), 'Price fluctuations and the use of bitcoin: An empirical inquiry', *Int. J. Electron. Commer.* **20**(1), 9–49.
URL: *https://doi.org/10.1080/10864415.2016.1061413*

Powers, D. M. W. (2020), 'Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation', *CoRR* **abs/2010.16061**.
URL: *https://arxiv.org/abs/2010.16061*

Ravi, K. & Ravi, V. (2015), 'A survey on opinion mining and sentiment analysis: tasks, approaches and applications', *Knowledge-based systems* **89**, 14–46.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016), "why should I trust you?": Explaining the predictions of any classifier, *in* B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen & R. Rastogi, eds, 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016', ACM, pp. 1135–1144.
URL: *https://doi.org/10.1145/2939672.2939778*

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

Sirignano, J. & Cont, R. (2019), 'Universal features of price formation in financial markets: perspectives from deep learning', *Quantitative Finance* **19**(9), 1449–1459.

Smith, J. (2016), 'Predictive modeling for financial markets: Traditional versus machine learning methods', *Journal of Financial Economics* **56**(2), 456–471.

Sokolova, M. & Lapalme, G. (2009), 'A systematic analysis of performance measures for classification tasks', *Inf. Process. Manag.* **45**(4), 427–437.
URL: *https://doi.org/10.1016/j.ipm.2009.03.002*

Sutskever, I., Vinyals, O. & Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence & K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada', pp. 3104–3112.
URL: *https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html*

Tay, F. E. & Cao, L. (2001), 'Application of support vector machines in financial time series forecasting', *omega* **29**(4), 309–317.

Treleaven, P. C., Galas, M. & Lalchand, V. (2013), 'Algorithmic trading review', *Commun. ACM* **56**(11), 76–85.
URL: *https://doi.org/10.1145/2500117*

Urquhart, A. (2016), 'The inefficiency of Bitcoin', *Economics Letters* **148**, 80–82.
URL: *https://www.sciencedirect.com/science/article/pii/S0165176516303640*

Xu, B., Bu, J., Chen, C., Wang, C., Cai, D. & He, X. (2013), 'Emr: A scalable graph-based ranking model for content-based image retrieval', *IEEE Transactions on knowledge and data engineering* **27**(1), 102–114.

Yeh, I., Yang, K. & Ting, T. (2009), 'Knowledge discovery on RFM model using bernoulli sequence', *Expert Syst. Appl.* **36**(3), 5866–5871.
URL: *https://doi.org/10.1016/j.eswa.2008.07.018*

Zohar, A. (2015), 'Bitcoin: under the hood', *Communications of the ACM* **58**(9), 104–113.

Appendix A

Appendix

```
!pip install ta
!pip install xgboost
!pip install keras
!pip install tensorflow
import warnings
warnings.filterwarnings('ignore')
```

```
a) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\rushi\appdata\roaming\python\python3
l>=2.8.1->pandas->ta) (1.16.0)

WARNING: There was an error checking the latest version of pip.

Requirement already satisfied: xgboost in d:\new folder (3)\lib\site-packages (1.7.6)
Requirement already satisfied: numpy in c:\users\rushi\appdata\roaming\python\python310
Requirement already satisfied: scipy in c:\users\rushi\appdata\roaming\python\python310

WARNING: There was an error checking the latest version of pip.

Requirement already satisfied: keras in d:\new folder (3)\lib\site-packages (2.13.1)

WARNING: There was an error checking the latest version of pip.

Collecting tensorflow
  Downloading tensorflow-2.13.0-cp310-cp310-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.13.0
  Downloading tensorflow_intel-2.13.0-cp310-cp310-win_amd64.whl (276.5 MB)
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from ta import add_all_ta_features
from ta.utils import dropna
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
import warnings
```

```
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')
print(df.head())
```

```
         Date          Open          High           Low         Close  \
0  2021-01-01  28994.009766  29600.626953  28803.585938  29374.152344
1  2021-01-02  29376.455078  33155.117188  29091.181641  32127.267578
2  2021-01-03  32129.408203  34608.558594  32052.316406  32782.023438
3  2021-01-04  32810.949219  33440.218750  28722.755859  31971.914063
4  2021-01-05  31977.041016  34437.589844  30221.187500  33992.429688

      Adj Close       Volume
0  29374.152344  40730301359
1  32127.267578  67865420765
2  32782.023438  78665235202
3  31971.914063  81163475344
4  33992.429688  67547324782
```

```
from sklearn.preprocessing import MinMaxScaler
# Handling missing values
df.dropna(inplace=True)
```

```
# Below code trains a Random Forest classifier model to predict whether the return will be positive or negative,
# using time series cross-validation. It then applies a simple trading strategy based on these predictions
# and tracks the capital over time. At each step, the root mean squared error (RMSE) between the model's
# predicted probabilities and the actual outcomes is calculated and printed. It also plots the predicted return
# probabilities and the cumulative profit/loss over time.
```

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, mean_squared_error
from sklearn.model_selection import TimeSeriesSplit, train_test_split
from sklearn.metrics import accuracy_score
from math import sqrt
import matplotlib.pyplot as plt
import warnings
from sklearn.exceptions import DataConversionWarning
warnings.filterwarnings(action='ignore', category=UserWarning)

# Load the data
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')

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

# Sort the values by 'Date'
df = df.sort_values('Date')

# Calculate the 'Return'
df['Return'] = df['Close'].pct_change()
```

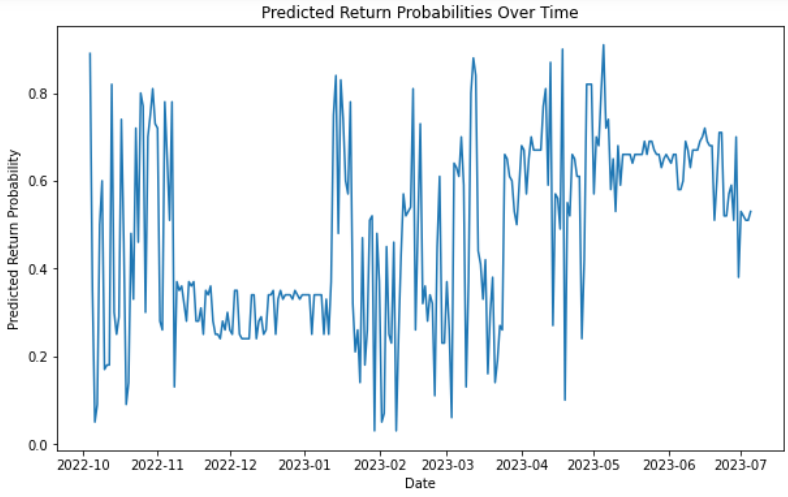
```
# Define the target variable
y = df['Return']

X = df.drop(columns=['Date', 'Return', 'Close'])

# Create a binary target variable
y_binary = (y > 0).astype(int)

# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Split the data into training (70%) and test (30%) sets
X_full_train, X_test, y_full_train, y_test = train_test_split(X, y_binary, test_size=0.3, shuffle=False)

tscv = TimeSeriesSplit(n_splits=5)

# List to store final capitals for each training window
final_capitals = []
training_window_sizes = [0.5, 0.6, 0.7, 0.8]

for train_index, val_index in tscv.split(X_full_train):
    X_train, X_val = X_full_train.iloc[train_index], X_full_train.iloc[val_index]
    y_train, y_val = y_full_train.iloc[train_index], y_full_train.iloc[val_index]

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the validation set
    y_pred = model.predict(X_val)
    y_pred_proba = model.predict_proba(X_val)[:, 1]  # We only need the probability for the class '1'

    # Calculate the RMSE
    rmse = sqrt(mean_squared_error(y_val, y_pred_proba))
    print('RMSE:', rmse)

    # Calculate the metrics
    print(classification_report(y_val, y_pred))

    # Calculate accuracy
    acc = accuracy_score(y_val, y_pred)
    print(f"Validation Accuracy: {acc}")

    # Implement the trading strategy on the validation set
    capitals_window = [10000000]  # Starting with 10 million capital
    for i in range(len(y_val)):
        y_pred_i = model.predict([X_val.iloc[i]])[0]
        if y_pred_i == 1:
            capitals_window.append(capitals_window[-1] * (1 + df['Return'].iloc[val_index[i]]))
        else:
```

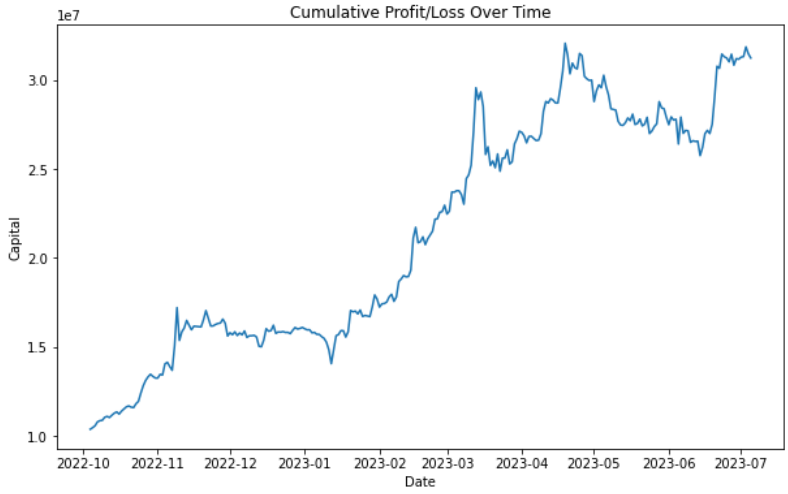
```
            capitals_window.append(capitals_window[-1] * (1 - df['Return'].iloc[val_index[i]]))

    # Store final capital for this training window
    final_capitals.append(capitals_window[-1])
    training_window_sizes.append(len(train_index) / len(X_full_train) * 100)  # as a percentage

# After cross-validation, evaluate the model on the test set
model.fit(X_full_train, y_full_train)
y_test_pred = model.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

# Implement a basic trading strategy
capitals = [10000000]  # Starting with 10 million capital
dates = df['Date'].iloc[y_test.index].tolist()

for i in range(len(y_test_pred)):
    if y_test_pred[i] == 1:  # If the model predicts the price will go up
        # Buy at the close price
        capitals.append(capitals[-1] * (1 + df['Return'].iloc[y_test.index[i]]))
    else:  # If the model predicts the price will go down
        # Sell at the close price
        capitals.append(capitals[-1] * (1 - df['Return'].iloc[y_test.index[i]]))

# Calculate the total profit/loss
total_profit = capitals[-1] - capitals[0]
print(f"Total Profit: ${total_profit}")

# Create a DataFrame for visualization
viz_df = pd.DataFrame({
    'Date': dates,
    'Predicted_Return_Proba': model.predict_proba(X_test)[:, 1],
    'Capital': capitals[1:]
})


# Plotting predicted return probabilities
plt.figure(figsize=(10, 6))
plt.plot(viz_df['Date'], viz_df['Predicted_Return_Proba'])
plt.title('Predicted Return Probabilities Over Time')
plt.xlabel('Date')
plt.ylabel('Predicted Return Probability')
plt.show()
```

```
# Plotting cumulative profit/loss curve
plt.figure(figsize=(10, 6))
plt.plot(viz_df['Date'], viz_df['Capital'])
plt.title('Cumulative Profit/Loss Over Time')
plt.xlabel('Date')
plt.ylabel('Capital')
plt.show()
```

```
RMSE: 0.46035773620415993
              precision    recall  f1-score   support

           0       0.57      0.91      0.70        53
           1       0.77      0.32      0.45        53

    accuracy                           0.61       106
   macro avg       0.67      0.61      0.58       106
weighted avg       0.67      0.61      0.58       106

Validation Accuracy: 0.6132075471698113
RMSE: 0.49445414919927344
              precision    recall  f1-score   support

           0       0.54      0.94      0.69        51
           1       0.82      0.25      0.39        55

    accuracy                           0.58       106
   macro avg       0.68      0.60      0.54       106
weighted avg       0.69      0.58      0.53       106

Validation Accuracy: 0.5849056603773585
RMSE: 0.4083156843830611
              precision    recall  f1-score   support

           0       0.76      0.85      0.80        55
           1       0.82      0.71      0.76        51

    accuracy                           0.78       106
   macro avg       0.79      0.78      0.78       106
weighted avg       0.79      0.78      0.78       106
```

```
Validation Accuracy: 0.7830188679245284
RMSE: 0.47137694726735874
              precision    recall  f1-score   support

           0       0.65      0.89      0.75        54
           1       0.81      0.50      0.62        52

    accuracy                           0.70       106
   macro avg       0.73      0.69      0.68       106
weighted avg       0.73      0.70      0.69       106

Validation Accuracy: 0.6981132075471698
RMSE: 0.5260093621377891
              precision    recall  f1-score   support

           0       0.57      0.47      0.51        58
           1       0.47      0.58      0.52        48

    accuracy                           0.52       106
   macro avg       0.52      0.52      0.52       106
weighted avg       0.53      0.52      0.52       106

Validation Accuracy: 0.5188679245283019
Test Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.57      0.58       142
           1       0.55      0.56      0.56       133

    accuracy                           0.57       275
   macro avg       0.57      0.57      0.57       275
weighted avg       0.57      0.57      0.57       275

Total Profit: $21237145.50199619
```


Predicted Return Probabilities Over Time
Predicted Return Probability
0.0
0.2
0.4
0.6
0.8
2022-10
2022-11
2022-12
2023-01
2023-02
2023-03
2023-04
2023-05
2023-06
2023-07
Date

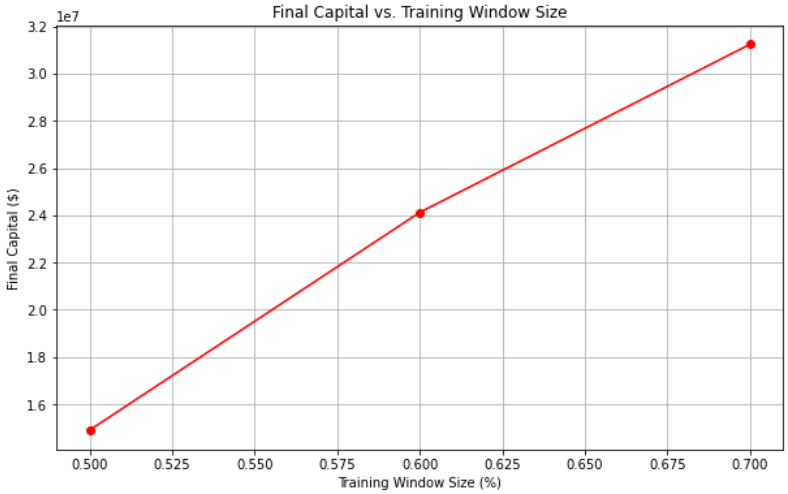

Cumulative Profit/Loss Over Time
1e7
Capital
1.0
1.5
2.0
2.5
3.0
2022-10
2022-11
2022-12
2023-01
2023-02
2023-03
2023-04
2023-05
2023-06
2023-07
Date

```
# below code Implement the "Long or Short" strategy with  different training window sizes
```

```
# import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, mean_squared_error
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')

df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')
df['Return'] = df['Close'].pct_change()
y = df['Return']
X = df.drop(columns=['Date', 'Return', 'Close'])
y_binary = (y > 0).astype(int)
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Different training window sizes as fractions of the total data length
training_windows = [0.5, 0.6, 0.7]
final_capitals = []

for window in training_windows:
    train_size = int(window * len(X))

    X_train, X_test = X.iloc[:train_size], X.iloc[train_size:]
    y_train, y_test = y_binary.iloc[:train_size], y_binary.iloc[train_size:]

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_test_pred = model.predict(X_test)

    # Implement the "Long or Short" strategy
    capitals = [10000000]  # Starting with 10 million capital

    for i in range(len(y_test_pred)):
        if y_test_pred[i] == 1:  # If the model predicts the price will go up
            capitals.append(capitals[-1] * (1 + df['Return'].iloc[y_test.index[i]]))
        else:  # If the model predicts the price will go down
            capitals.append(capitals[-1] * (1 - df['Return'].iloc[y_test.index[i]]))

    final_capital = capitals[-1]
    final_capitals.append(final_capital)

    print(f"Training Window: {window*100}% -> Final Capital: ${final_capital}")
```

```
# Visualization of final capitals for different training window sizes
plt.figure(figsize=(10, 6))
plt.plot(training_windows, final_capitals, marker='o', linestyle='-', color='red')
plt.title('Final Capital vs. Training Window Size')
plt.xlabel('Training Window Size (%)')
plt.ylabel('Final Capital ($)')
plt.grid(True)
plt.show()
```

```
Training Window: 50.0% -> Final Capital: $14897367.792707803
Training Window: 60.0% -> Final Capital: $24119758.892387908
Training Window: 70.0% -> Final Capital: $31237145.50199619
```

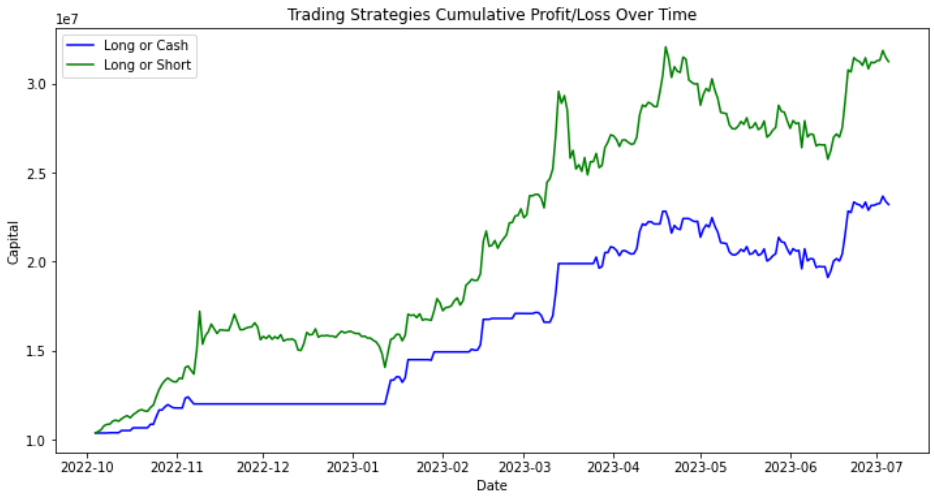



```
# below code is the comparision of which trading strategy is best "long or cash" "long or Short" In random forest
```

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import TimeSeriesSplit, train_test_split
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')
```

```
df['Date'] = pd.to_datetime(df['Date'])

# Sort the values by 'Date'
df = df.sort_values('Date')

# Calculate the 'Return'
df['Return'] = df['Close'].pct_change()

# Define the target variable
y = df['Return']

X = df.drop(columns=['Date', 'Return', 'Close'])

# Create a binary target variable
y_binary = (y > 0).astype(int)

# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Split the data into training (70%) and test (30%) sets
X_full_train, X_test, y_full_train, y_test = train_test_split(X, y_binary, test_size=0.3, shuffle=False)

# Implement TimeSeriesSplit for cross-validation on the training set
tscv = TimeSeriesSplit(n_splits=5)

for train_index, val_index in tscv.split(X_full_train):
    X_train, X_val = X_full_train.iloc[train_index], X_full_train.iloc[val_index]
    y_train, y_val = y_full_train.iloc[train_index], y_full_train.iloc[val_index]

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the validation set
    y_pred = model.predict(X_val)

    # Calculate the metrics
    print(classification_report(y_val, y_pred))

# After cross-validation, evaluate the model on the test set
model.fit(X_full_train, y_full_train)
y_test_pred = model.predict(X_test)

# Implement the "Long or Cash" strategy
capitals_long_or_cash = [10000000]  # Starting with 10 million capital
```

```
for i in range(len(y_test_pred)):
    if y_test_pred[i] == 1:  # If the model predicts the price will go up
        # Buy at the close price
        capitals_long_or_cash.append(capitals_long_or_cash[-1] * (1 + df['Return'].iloc[y_test.index[i]]))
    else:  # If the model predicts the price will go down, hold cash (no change in capital)
        capitals_long_or_cash.append(capitals_long_or_cash[-1])

# Implement the "Long or Short" strategy
capitals_long_or_short = [10000000]  # Starting with 10 million capital

for i in range(len(y_test_pred)):
    if y_test_pred[i] == 1:  # If the model predicts the price will go up
        # Buy at the close price
        capitals_long_or_short.append(capitals_long_or_short[-1] * (1 + df['Return'].iloc[y_test.index[i]]))
    else:  # If the model predicts the price will go down
        # Sell (short) at the close price
        capitals_long_or_short.append(capitals_long_or_short[-1] * (1 - df['Return'].iloc[y_test.index[i]]))

# Visualization
dates = df['Date'].iloc[y_test.index].tolist()
plt.figure(figsize=(12, 6))
plt.plot(dates, capitals_long_or_cash[1:], label="Long or Cash", color='blue')
plt.plot(dates, capitals_long_or_short[1:], label="Long or Short", color='green')
plt.title('Trading Strategies Cumulative Profit/Loss Over Time')
plt.xlabel('Date')
plt.ylabel('Capital')
plt.legend()
plt.show()

# Calculate and print profits
profit_long_or_cash = capitals_long_or_cash[-1] - capitals_long_or_cash[0]
profit_long_or_short = capitals_long_or_short[-1] - capitals_long_or_short[0]
print(f"Total Profit (Long or Cash): ${profit_long_or_cash}")
print(f"Total Profit (Long or Short): ${profit_long_or_short}")
```



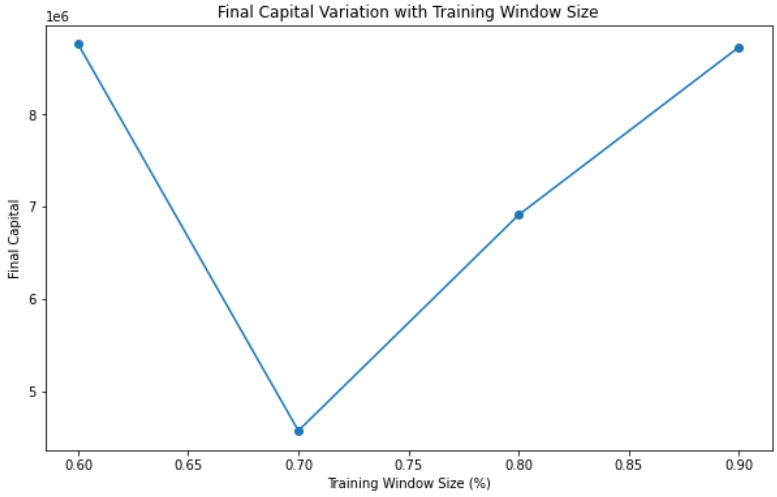

```
Total Profit (Long or Cash): $13207241.542338066
Total Profit (Long or Short): $21237145.50199619
```

```
# below code is  the "Long or Short" trading strategy on the test set
```

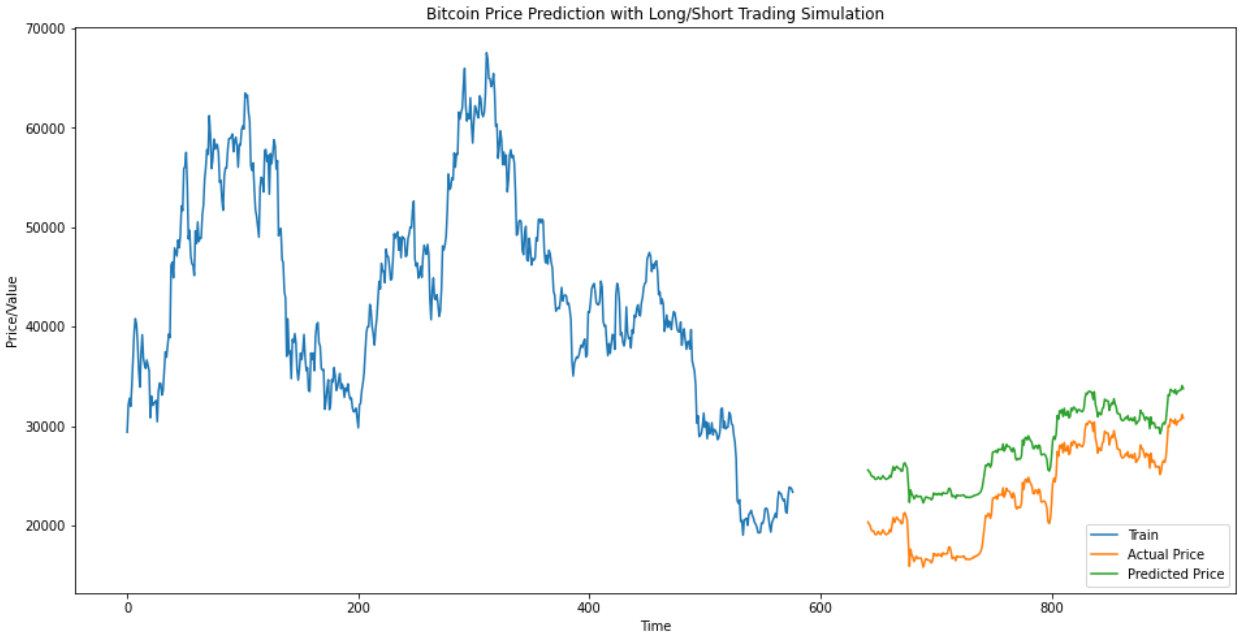
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')
df['Date'] = pd.to_datetime(df['Date'])

# Sort the values by 'Date'
df = df.sort_values('Date')

# Calculate the 'Return'
df['Return'] = df['Close'].pct_change()

# Prepare the data
new_data = df[['Close']].copy()
scaled_data = MinMaxScaler().fit_transform(new_data.values)

# Define the LSTM model
def create_lstm_model():
    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True, input_shape=(1, 1)))
    model.add(LSTM(units=50))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Split data into training and test sets
X_full_train = scaled_data[:-int(0.3*len(scaled_data))]
X_test = scaled_data[-int(0.3*len(scaled_data)):]

# Define window sizes
window_sizes = [ 0.6, 0.7, 0.8,0.9]
final_capitals = []

for window in window_sizes:
    train_end = int(len(X_full_train) * window)

    train_data = X_full_train[:train_end]
    val_data = X_full_train[train_end:]

    model = create_lstm_model()
    model.fit(train_data[:-1], train_data[1:], epochs=10, batch_size=1, verbose=0)
```

```
    predictions = model.predict(val_data[:-1])
    predictions = np.where(predictions > val_data[:-1], 1, 0)

    # Implement the "Long or short" trading strategy
    capitals = [10000000]
    for i in range(len(predictions)):
        if predictions[i] == 1:
            capitals.append(capitals[-1] * (1 + df['Return'].iloc[train_end + i + 1]))
        else:
            capitals.append(capitals[-1])
    final_capitals.append(capitals[-1])

# Plot how the final capital varies with training window size
plt.figure(figsize=(10, 6))
plt.plot(window_sizes, final_capitals, marker='o')
plt.title('Final Capital Variation with Training Window Size')
plt.xlabel('Training Window Size (%)')
plt.ylabel('Final Capital')
plt.show()

# After evaluating different window sizes, train the model on the full training set and evaluate on the test set
model = create_lstm_model()
model.fit(X_full_train[:-1], X_full_train[1:], epochs=10, batch_size=1, verbose=0)

test_predictions = model.predict(X_test[:-1])
test_predictions = np.where(test_predictions > X_test[:-1], 1, 0)

# Implement the "Long or short" trading strategy on the test set
test_capitals = [10000000]
for i in range(len(test_predictions)):
    if test_predictions[i] == 1:
        # Long
        test_capitals.append(test_capitals[-1] * (1 + df['Return'].iloc[-int(0.3*len(df)) + i + 1]))
    else:
        # Short
        test_capitals.append(test_capitals[-1] * (1 - df['Return'].iloc[-int(0.3*len(df)) + i + 1]))

# Print final portfolio value
print(f"Final Portfolio Value: ${test_capitals[-1]:,.2f}")

# Visualizing the prediction and trading simulation
plt.figure(figsize=(16,8))
train_data_plot = new_data[:train_end]
plt.plot(train_data_plot.index, train_data_plot["Close"], label="Train")
plt.plot(valid_data_df.index, valid_data_df['Close'], label="Actual Price")
plt.plot(valid_data_df.index, valid_data_df['Predictions'], label="Predicted Price")
plt.title('Bitcoin Price Prediction with Long/Short Trading Simulation')
plt.xlabel('Time')
plt.ylabel('Price/Value')
plt.legend(loc="lower right")
```

```
8/8 [==============================] - 1s 4ms/step
6/6 [==============================] - 1s 3ms/step
4/4 [==============================] - 1s 5ms/step
2/2 [==============================] - 1s 0s/step
```




```
9/9 [==============================] - 1s 4ms/step
Final Portfolio Value: $19,002,979.05
```




```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv(r'C:\Users\rushi\Downloads\BTC-USD 2021.csv')
df['Date'] = pd.to_datetime(df['Date'])

# Sort the values by 'Date'
df = df.sort_values('Date')

# Calculate the 'Return'
df['Return'] = df['Close'].pct_change()

# Prepare the data
new_data = df[['Close']].copy()
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(new_data.values)

# Define the LSTM model
def create_lstm_model():
    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True, input_shape=(1, 1)))
    model.add(LSTM(units=50))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Adjust the number of days for training here
num_days = 500
train_data_len = num_days
X_train = scaled_data[:train_data_len]
X_valid = scaled_data[train_data_len:]

model = create_lstm_model()
model.fit(X_train[:-1], X_train[1:], epochs=10, batch_size=1, verbose=0)

predictions = model.predict(X_valid[:-1])

# Convert the scaled predictions back to original scale
predicted_prices = scaler.inverse_transform(predictions)
```

```python
# Convert the scaled predictions back to original scale
predicted_prices = scaler.inverse_transform(predictions)

# Implement the "long or short" trading strategy
capitals = [10000000]  # Starting capital
for i in range(len(predictions)):
    if predictions[i] > X_valid[i]:
        # Long
        capitals.append(capitals[-1] * (1 + df['Return'].iloc[train_data_len + i + 1]))
    else:
        # Short
        capitals.append(capitals[-1] * (1 - df['Return'].iloc[train_data_len + i + 1]))

# Print final portfolio value
print(f"Final Portfolio Value: ${capitals[-1]:,.2f}")

# Visualizing the prediction and trading simulation
plt.figure(figsize=(16,8))
train_data_plot = new_data[:train_data_len]
plt.plot(train_data_plot.index, train_data_plot["Close"], label="Train")
plt.plot(new_data[train_data_len:].index, new_data[train_data_len:]['Close'], label="Actual Price")
plt.plot(new_data[train_data_len:].index[:-1], predicted_prices, label="Predicted Price", alpha=0.6)
plt.legend(loc="lower right")
plt.title('Bitcoin Price Prediction with Long/Short Trading Simulation')
plt.xlabel('Time')
plt.ylabel('Price/Value')
plt.show()
```

```
13/13 [==============================] - 1s 3ms/step
Final Portfolio Value: $10,227,877.85
```