

Report_Z2

by Girija Shankar Panda

Submission date: 12-Jun-2025 12:09PM (UTC+0530)

Submission ID: 2696082964

File name: SDP_Report_Z2.pdf (671.77K)

Word count: 9459

Character count: 48632

CRYPTOCURRENCY PRICE PREDICTION USING MACHINE LEARNING

A Project Report

Submitted by:

PRANAB DAS (2141004166)

GIRIJA SHANKAR PANDA (2141016383)

NEERAJ NIRUPAM MOHANTY (2141013305)

DHIREN PRATAP SINGH (2141019453)

*in partial fulfillment for the award of the degree
of*

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Faculty of Engineering and Technology, Institute of Technical Education and Research
SIKSHA 'O' ANUSANDHAN (DEEMED TO BE) UNIVERSITY
Bhubaneswar, Odisha, India**

(June 2025)



CERTIFICATE

This is to certify that the project report titled “**Cryptocurrency Price Prediction using Machine Learning**” being submitted by Pranab Das, Girija Shankar Panda, Neeraj Nirupam Mohanty, Dhiren Pratap Singh of Section-Z(26) and to the Institute of Technical Education and Research, Siksha ‘O’ Anusandhan (Deemed to be) University, Bhubaneswar for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering is a record of original confide work carried out by them under my/our supervision and guidance. The project work, in my/our opinion, has reached the requisite standard fulfilling the requirements for the degree of Bachelor of Technology.

The results contained in this project work have not been submitted in part or full to any other University or Institute for the award of any degree or diploma.

Dr. Samir Kundu
Department of Computer Science and Engineering
Faculty of Engineering and Technology;
Institute of Technical Education and Research;
Siksha ‘O’ Anusandhan (Deemed to be) University

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who supported us throughout the successful completion of this project. We are especially thankful to **Dr. Samir Kundu**, our project guide, for his unwavering guidance, insightful feedback, and continuous encouragement, which significantly contributed to the development and refinement of our work.

We extend our deep appreciation to **Dr. Rina Mahakud**, our project coordinator, for her consistent support and coordination, which ensured smooth progress at every stage. We are also grateful to **Dr. Debahuti Mishra**, Head of the Department of Computer Science and Engineering, for fostering a stimulating academic environment and for the facilities provided by the department. Lastly, we acknowledge the Department of Computer Science and Engineering, **Siksha 'O' Anusandhan (Deemed to be University)**, for creating a platform that encourages innovation, exploration, and practical application of knowledge.

Place: Bhubaneswar

Date: 12/06/2025

Pranab Das

Gunja Shankar Panda

Neeraj Nirupam Mohanty

Dhruv Pratap Singh

Signature of Students

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the University and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Pranab Das
(2141004166)

Girija Shankar Panda
(2141016383)

Neeraj Nirupam Mohanty
(2141013305)

Dhiren Pratap Singh
(2141019453)

Signature of Students with Registration Numbers
Date: 12/06/2025

REPORT APPROVAL

This project report titled “**Cryptocurrency Price Prediction using Machine Learning**” submitted by **Pranab Das, Girija Shankar panda, Neeraj Nirupam Mohanty and Dhiren Pratap Singh** is approved for the degree of *Bachelor of Technology in Computer Science and Engineering*.

Examiner(s)

Supervisor

Project Coordinator

PREFACE

This project report presents the development and analysis of a machine learning-based model for predicting the price of cryptocurrencies. In recent years, digital currencies like Bitcoin, Ethereum, and others have attracted significant attention from investors, developers, and researchers due to their volatile nature and rapid market growth. Accurate forecasting of cryptocurrency prices remains a challenging task due to high fluctuations, lack of centralized control, and the influence of various market factors.

In this project, we explored the use of data-driven techniques, particularly the XGBoost algorithm, to build a predictive model that can estimate future prices based on historical data. The methodology includes data collection, preprocessing, feature engineering, model training, and evaluation. We also performed analysis using performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) to measure the accuracy and effectiveness of the model.

This report provides a detailed overview of the problem domain, the implemented solution, and the results obtained. It reflects our understanding of machine learning applications in financial markets and demonstrates how computational tools can be used to address complex real-world problems.

INDIVIDUAL CONTRIBUTIONS

Pranab Das	Led the team by designing the solution approach and implementing forecasting models for cryptocurrency prices. Conducted extensive literature review, fine-tuned ML algorithms, supported deployment, and contributed to project documentation.
Girija Shankar Panda	Focused on preprocessing the cleaned data, developing the core machine learning pipeline, and implementing forecasting logic. Engaged in problem identification, code development, and contributed to technical documentation.
Neeraj Nirupam Mohanty	Designed and implemented the front-end interface, collaborated on model performance testing and visualization. Participated in experimentation, results interpretation, and prepared major parts of the project documentation.
Dhiren Pratap Singh	Collected reliable raw cryptocurrency data, performed essential data cleaning, and supported validation of model outcomes. Contributed to literature study and documentation of data-driven insights.

TABLE OF CONTENTS

	Title Page	i
	Certificate	ii
	Acknowledgement	iii
	Declaration	iv
	Report Approval	v
	Preface	vi
	Individual Contributions	vii
	Table of Contents	viii
	List of Figures	ix
	List of Tables	x
1.	INTRODUCTION	1
	1.1 Project Overview/Specifications	1
	1.2 Motivation(s)	1
	1.3 Uniqueness of the Work	2
	1.4 Report Layout	4
2.	LITERATURE SURVEY	5
	2.1 Existing System	5
	2.2 Problem Identification	6
3.	MATERIALS AND METHODS	9
	3.1 Dataset Description	9
	3.2 Schematic Workflow	10
	3.3 Methods Used	13
	3.4 Tools Used	15
	3.5 Evaluation Measures Used	15
4.	RESULTS / OUTPUTS	17
	4.1 System Specification	17
	4.2 Parameters Used	19
	4.2.1 Input Features and Technical Indicators	19
	4.2.2 Model Hyperparameters	20
	4.2.3 Preprocessing Parameters	21
	4.2.4 Dataset Parameters	21
	4.3 Experimental Results and User Interface Output	22
	4.4 Performance Metrics	28
	4.5 Comparative Analysis	29
5.	CONCLUSIONS	31
6.	REFERENCES	32
7.	APPENDICES	33
8.	REFLECTION OF THE TEAM MEMBERS ON THE PROJECT	41
9.	SIMILARITY REPORT	43

LIST OF FIGURES

NO	FIGURE NAME	PAGE NO
1	Sample Snapshot of Raw Bitcoin (BTC) Dataset from Investing.com ^[7]	10
2	Frontend Workflow Diagram of CryptIQ	11
3	Back-End Workflow for Cryptocurrency Price Prediction	13
4	Evaluated output using XGBoost algorithm	22
5	Evaluated output using SimpleGBM algorithm	22
6	Actual vs Predicted ETH prices from date 2021-07-02 to 2021-07-06	23
7	Registration page of CryptIQ	23
8	Login page of CryptIQ	24
9	Home page showing graph of Bitcoin Cryptocurrency closing prices	24
10	Price prediction after 1 day from current day	25
11	Price prediction after 1 week from current day	25
12	Price prediction after 1 month from current day	26
13	Price prediction after 1 year from current day	26
14	Future prices of Bitcoin on console	26
15	Current Graph page of CryptIQ	27
16	About page of CryptIQ	27
17	Features used that helps to predict	33
18	REST API Route for Prediction (<i>/predict</i>)	34
19	Custom implementation of a Gradient Boosting algorithm in Python using decision trees from sklearn.tree.	35

LIST OF TABLES

NO	TABLE NAME	PAGE NO
1	Limitations of existing models	7
2	Features used and their description	19
3	Parameters used in XGBoost Regressor model	20
4	Parameters used in Simple GBM model	21
5	Performance Metrics of XGB and GBM model	28
6	Comparative analysis of two models (XGBoost vs SimpleGBM)	30

CHAPTER-1: INTRODUCTION

1.1 Project Overview / Specifications

The rapid growth of cryptocurrencies has transformed global financial systems, enabling decentralized, borderless, and peer-to-peer transactions. However, the high volatility and unpredictable price movements in this market pose significant risks to investors and traders.

This project aims to make cryptocurrency price prediction more accessible and dependable by leveraging the power of machine learning. The volatile nature of digital currencies like Bitcoin, Ethereum, ChainLink, and Litecoin creates challenges for investors and traders looking to make informed decisions. To address this, we have developed a predictive system that utilizes the XGBoost algorithm to analyze historical market data sourced from Investing.com.

The workflow begins with thorough data preprocessing, handling missing values, formatting timestamps, and normalizing key attributes. We then enrich the dataset with crucial financial features such as open, close, high, low prices, volume, and calculated returns. To better capture the patterns within market movements, we integrate technical indicators like the Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and Exponential Moving Average (EMA).

The refined data is fed into an XGBoost model, which is trained to predict future prices based on historical behavior and computed indicators. The output of this model is connected to a web-based interface, designed using HTML, CSS, and JavaScript. The user-friendly web application allows users to choose from multiple cryptocurrencies and generate price forecasts for various time intervals, including 1 day, 1 week, 1 month, and even 1 year.

1.2 Motivation

Cryptocurrency markets are known for their extreme volatility, with prices often experiencing significant swings within very short timeframes. This unpredictability makes it difficult for investors and traders to rely on traditional forecasting techniques, which tend to fall short when dealing with such dynamic and non-linear financial systems. The lack of stability and consistency in crypto pricing creates a strong need for more advanced, adaptive solutions.

Our primary motivation is to bridge this gap by developing an intelligent, data-driven approach to forecasting. Through the application of machine learning, specifically using the XGBoost algorithm, we aim to offer a system that can learn from historical market behavior and provide meaningful insights into future price trends. By doing so, we hope to empower users with a clearer, more tailored view of where their selected cryptocurrencies might be heading. This enables better decision-making and allows both beginners and experienced investors to plan their strategies with greater confidence and precision.

1.3 Uniqueness of Work

The field of cryptocurrency price prediction is crowded with various models and tools, many of which fall short either in accuracy, usability, or relevance to real-world users. Our project sets itself apart through a thoughtful blend of advanced data analytics, technical modeling, and a user-centered interface. Below, we highlight the key elements that make our approach unique.

Integration of Machine Learning with Real-Time Accessibility

Unlike conventional forecasting tools that rely on basic trend analysis or static models, our solution incorporates a sophisticated machine learning algorithm **XGBoost**, to handle complex, non-linear relationships in financial data. This model is not only efficient and accurate but also integrated into a real-time forecasting system that delivers immediate results via a web application. This combination of computational intelligence and real-time responsiveness enhances the overall value of the platform.

Feature Enrichment and Technical Indicators

Many models use only basic price information, limiting their forecasting ability. In contrast, our model is enriched with **derived features** such as daily returns, log-scaled price movements, and volatility-based calculations. To add depth and capture momentum shifts in the market, we also apply **technical indicators** including:

- Relative Strength Index (RSI),
- Moving Average Convergence Divergence (MACD)
- Exponential Moving Averages (EMA)

These indicators help identify trends, reversals, and entry/exit points, contributing to more insightful and reliable predictions.

Log-Transformed Scaling for Stability

Cryptocurrency prices often exhibit extreme fluctuations and non-stationary behavior. To address this challenge, we employ **logarithmic transformations** on the price data. This technique not only stabilizes the variance but also enables the model to better learn patterns without being skewed by outliers or sudden market spikes. As a result, our predictions are more consistent and meaningful over different time horizons.

Simple, Intuitive Web Interface

While the backend relies on sophisticated algorithms, the frontend is designed with simplicity and clarity in mind. The web interface allows users to select from various cryptocurrencies and generate forecasts for different timeframes, ranging from **1 day to 1 year**, with just a few clicks. Built using HTML, CSS, and JavaScript, the platform is accessible, responsive, and user-friendly. Even individuals without technical expertise can use it to make informed decisions based on data-driven predictions.

Balance Between Power and Usability

The real strength of our project lies in how it balances computational power with user experience. By marrying advanced predictive techniques with an approachable design, we

ensure that our platform appeals to both data analysts and casual crypto enthusiasts. This makes the project not just a technical demonstration, but a **practical tool** for the real world.

1.4 Report Layout

This report is organized into several chapters for clarity and ease of understanding.

- **Chapter-1** introduces the project, its motivations, objectives, and uniqueness.
- **Chapter-2** reviews existing literature and related work in the domain of cryptocurrency forecasting.
- **Chapter-3** outlines the methodology adopted for model development, including data processing, feature selection, and algorithm selection.
- **Chapter-4** discusses the technical implementation of the prediction system and model training process.
- **Chapter-5** presents the results and evaluates the model performance using various metrics.
- **Chapter-6** concludes the project, summarizing key findings and suggesting potential directions for future work.

CHAPTER-2: LITERATURE SURVEY

2.1 Existing Systems

Several research studies have explored the use of machine learning in cryptocurrency price prediction, employing different techniques ranging from traditional models to deep learning and ensemble approaches.

- I. **Ghadiri and Hajizadeh** proposed a hybrid trading system that combines deep reinforcement learning with LSTM neural networks and incorporates XGBoost for feature selection. Their model demonstrated strong predictive power, achieving an accuracy of approximately 92%. However, due to its complexity and lengthy training times, it is less practical for lightweight or real-time applications.
- II. **Islam et al. and Qureshi et al.** focused on classical machine learning methods such as Logistic Regression and Random Forest for cryptocurrency forecasting. While their systems reached around 90% accuracy, the scope was mainly limited to short- and mid-term price predictions. Additionally, their studies lacked deeper comparative analysis involving more advanced or hybrid models.
- III. **Kumar et al.** conducted a comparative study of LSTM and XGBoost models but restricted their analysis to Bitcoin only. Although they provided useful forecasting accuracy results, their work did not address long-term price prediction and was constrained by a limited set of features.
- IV. Other researchers such as **Hafid et al. and Sun (2024)** have successfully applied XGBoost models, emphasizing strong performance metrics like low Root Mean Squared Error (RMSE) and high R-squared (R^2). Nevertheless, their datasets were relatively small, and the models lacked real-time usability or the capability to simulate future market trends comprehensively.

These systems have contributed valuable insights, but also show the presence of common and repeated limitations that affect their practical application for broader user groups.

2.2 Problem Identification

Despite notable advancements in cryptocurrency price prediction using machine learning, several critical challenges remain that limit the practical utility and applicability of existing models.

1. Predominant Focus on Bitcoin Most existing research primarily targets Bitcoin (BTC), with some attention to Ethereum (ETH), as these cryptocurrencies have the most readily available and extensive datasets. However, this narrow focus overlooks the rising importance of various altcoins such as Cardano, Solana, Tether, and XRP. There is a clear need for models that can generalize across a wider range of digital assets to serve the diverse interests of the crypto community.

2. Lack of Real-Time Deployment and User Interaction Many machine learning models for cryptocurrency forecasting are developed solely as academic exercises or offline experiments. While these models may achieve strong accuracy on historical data, they rarely incorporate real-time deployment or user-friendly interfaces. This disconnect limits accessibility for non-technical users like investors and traders, preventing them from leveraging predictive insights during their everyday decision-making processes.

3. Limited Use of Advanced Features and Technical Indicators A considerable number of studies depend mainly on basic historical price data, including open, high, low, and close values, along with simple averages. Such shallow feature sets fail to capture the nuanced behavior of cryptocurrency markets. Incorporating technical indicators like the Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and Exponential Moving Average (EMA), as well as lag-based temporal patterns, could significantly improve model accuracy by highlighting trend changes, market momentum, and overbought or oversold conditions.

4. Restriction to Short-Term Forecasting Horizons Existing forecasting tools commonly limit their predictions to short windows, such as the next day or week. However, investors often require longer-term outlooks—spanning one month or even a year—to guide strategic planning and portfolio management. Current literature and implementations rarely address these extended prediction horizons, creating a gap in comprehensive forecasting capabilities.

5. High Complexity and Low Interpretability in Advanced Models Some recent approaches that combine deep learning techniques like Long Short-Term Memory (LSTM) networks with machine learning methods such as XGBoost demonstrate high predictive performance. Nevertheless, these hybrid systems tend to be computationally intensive and opaque, making it difficult for end users to understand or trust the reasoning behind forecasts. This complexity poses challenges in adoption and usability, especially among non-expert users.

Below is a summary table highlighting key research papers that have inspired our work:

Table 1: Limitations of existing models

#	Name of Work	Author(s) and Year	Technique(s) Used	Benefits	Limitations
1	Designing a Cryptocurrency Trading System with DRL Using LSTM	Ghadiri & Hajizadeh (2025)	Deep Reinforcement Learning, LSTM	Effective feature selection (LSTM + XGBoost)	Lack of detailed dataset disclosure; computational complexity
2	ML-Based Cryptocurrency Prediction: Enhancing Market Forecasting with	Islam et al. (2025)	Multiple ML models, advanced predictive models	Improved forecasting accuracy using a range of advanced techniques	Model interpretability not discussed

	Advanced Predictive Models				
3	Evaluating ML Models for Predictive Accuracy in Crypto Price Forecasting	Qureshi et al. (2025)	Linear Regression, Decision Trees, SVM, Random Forest	Identified Random Forest as a strong performer across various datasets	High variance in performance across different coins
4	Comparative Analysis of LSTM and XGBoost Models for Short-Term Bitcoin Price Prediction	Kumar et al. (2024)	LSTM, XGBoost	Demonstrated LSTM's superior performance for short-term prediction	Limited to short-term scope; lacks broader generalization
5	An Improved ML-Driven Framework for Cryptocurrency Price Prediction with Sentimental Cautioning	Zubair et al. (2024)	ML models + sentiment analysis	Enhanced accuracy with inclusion of market sentiment signals	Sentiment analysis may introduce noise; scalability issues
6	A Deep Learning-Based Cryptocurrency Price Prediction Model That Uses On-Chain Data	Kim et al. (2022)	Deep Learning (CNN-LSTM hybrid), on-chain data	Greater reliability by leveraging blockchain on-chain metrics	Requires blockchain infrastructure ; high computational requirements

CHAPTER-3: MATERIALS AND METHODS

3.1 Dataset Description

The dataset utilized in this project comprises historical price data for 10 prominent cryptocurrencies, namely: **Bitcoin (BTC)**, **Ethereum (ETH)**, **Cardano (ADA)**, **Chainlink (LINK)**, **Litecoin (LTC)**, **Monero (XMR)**, **Stellar (XLM)**, **Tether (USDT)**, **USD Coin (USDC)**, and **XRP (Ripple)**. These cryptocurrencies were selected based on their market capitalization, trading volume, and widespread adoption in the digital asset ecosystem.

The data was sourced from Investing.com[□], a reliable financial platform, and spans a period of **approximately twelve years**, providing a rich temporal context for price trend analysis and predictive modeling.

Each dataset was obtained in **CSV format** and contains the following key attributes for each daily entry:

1. **Name:** Name of the cryptocurrency (e.g., Bitcoin)
2. **Date:** Timestamp of the record
3. **Price:** Actual price of the coin on that date
4. **High:** Highest price of the coin on that date
5. **Low:** Lowest price of the coin on that date
6. **Open:** Opening price of the coin
7. **Close:** Closing price of the coin
8. **Volume:** Total trading volume on that date

This comprehensive dataset enables a wide range of analytical tasks including time-series forecasting, correlation studies between assets, and comparative analysis of volatility and growth across cryptocurrencies. The inclusion of both stablecoins (like USDT and USDC) and volatile coins (like BTC and ETH) provides a balanced foundation for building robust predictive models and understanding market dynamics.

	A	B	C	D	E	F	G
1	Date	Price	Open	High	Low	Vol.	Change %
2	12/31/2024	93,557.20	92,777.20	96,163.40	92,036.20	74.85K	0.84%
3	12/30/2024	92,779.80	93,718.70	94,936.40	91,522.30	112.43K	-1.00%
4	12/29/2024	93,716.30	95,282.60	95,315.40	93,026.70	47.71K	-1.65%
5	12/28/2024	95,284.50	94,274.90	95,684.30	94,124.70	32.97K	1.07%
6	12/27/2024	94,275.90	95,776.40	97,243.30	93,472.80	85.12K	-1.57%
7	12/26/2024	95,777.70	99,389.40	99,922.50	95,193.30	74.60K	-3.64%
8	12/25/2024	99,391.30	98,661.90	99,514.10	97,651.90	39.80K	0.73%
9	12/24/2024	98,668.10	94,852.80	99,446.60	93,566.20	69.86K	4.02%
10	12/23/2024	94,852.10	95,184.80	96,517.80	92,543.20	133.24K	-0.35%
11	12/22/2024	95,183.80	97,281.60	97,422.20	94,265.20	100.64K	-2.13%
12	12/21/2024	97,253.30	97,799.10	99,531.90	96,478.60	123.36K	-0.55%
13	12/20/2024	97,795.70	97,468.30	98,178.20	92,301.90	221.18K	0.34%
14	12/19/2024	97,466.10	100,190.50	102,778.80	95,672.20	200.57K	-2.73%
15	12/18/2024	100,197.80	106,140.70	106,477.60	100,049.30	187.49K	-5.60%
16	12/17/2024	106,138.90	106,053.40	108,244.90	105,350.60	149.37K	0.08%
17	12/16/2024	106,057.60	104,479.20	107,767.60	103,351.70	232.66K	1.55%
18	12/15/2024	104,443.00	101,417.90	105,120.90	101,234.90	133.99K	2.98%
19	12/14/2024	101,417.70	101,423.70	102,633.00	100,626.30	105.31K	-0.01%
20	12/13/2024	101,426.20	100,008.30	101,891.20	99,214.20	162.40K	1.42%
21	12/12/2024	100,009.90	101,126.30	102,495.30	99,334.50	188.96K	-1.10%
22	12/11/2024	101,126.20	96,603.20	101,877.10	95,689.50	250.02K	4.69%
23	12/10/2024	96,600.30	97,311.50	98,237.80	94,304.50	271.45K	-0.78%
24	12/9/2024	97,359.40	101,129.70	101,198.60	94,395.80	321.01K	-3.72%
25	12/8/2024	101,115.80	99,837.00	101,339.90	98,713.90	128.93K	1.28%
<div> < > <div>coin_Bitcoin</div> + </div>							

Figure 1: Sample Snapshot of Raw Bitcoin (BTC) Dataset from Investing.com^[7]

Some entries included missing or zero volume values, which were handled during the data preprocessing phase. The final cleaned dataset served as input for feature engineering and model training using machine learning algorithms.

3.2 Schematic Workflow

A. Front-end Workflow:

The schematic diagram (e.g. Figure-2) represents the structure and navigation flow of the web-based frontend interface.

Users begin by registering or logging in and are directed to the Home Page, which serves as the central navigation point.

From the front-end interface, users can:

- a. **Navigate to the Home Page:** Where they select a cryptocurrency and choose a prediction duration (1 Day, 1 Week, 1 Month, 1 Year) to visualize forecasted vs. actual prices.
- b. **Access their Account Page:** To edit credentials and view trade history.
- c. **Visit the Graph section:** Which allows them to view our predicted graphs and monitor selected cryptocurrencies over time.
- d. **Access the About Us Page:** For project information and contributors.

This structure ensures a user-friendly and intuitive experience, supporting interaction with the ML prediction model through real-time, visually guided steps.

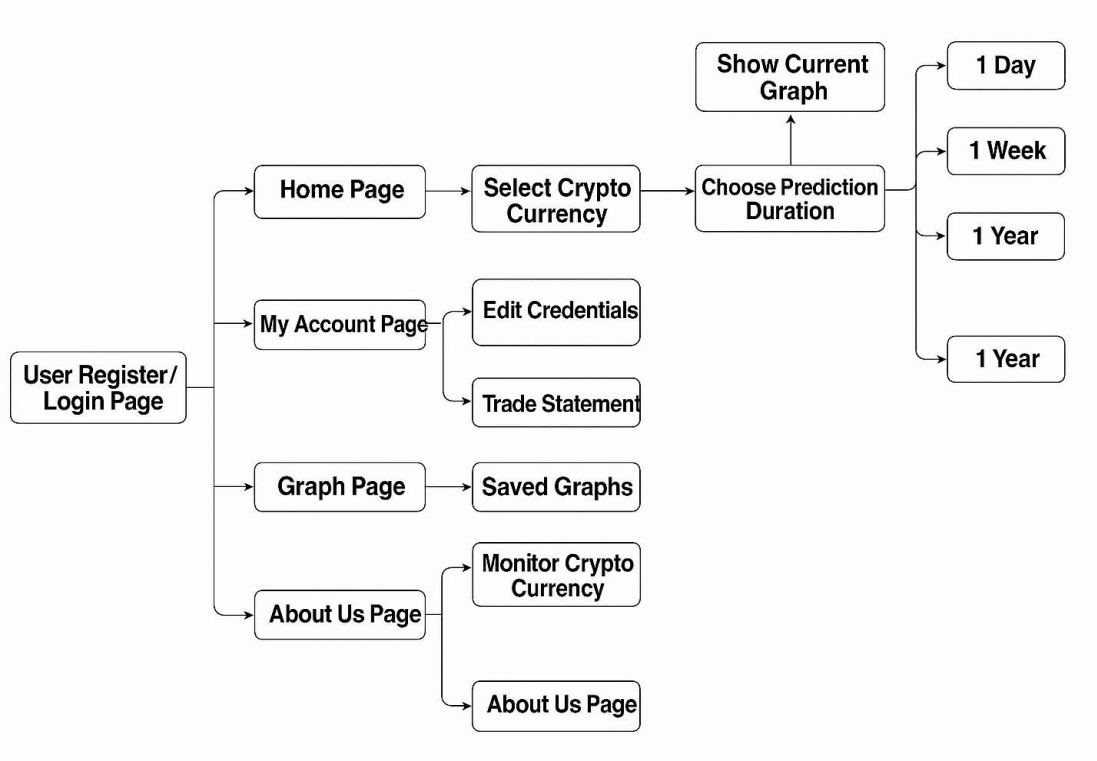


Figure 2: Frontend Workflow Diagram of CryptIQ

The modular design of the interface allows for scalability, meaning additional features—such as comparative analysis, watchlists, or alert systems—can be integrated seamlessly

without disrupting the existing user flow. This makes the system highly adaptable for future enhancements.

B. Back-end Workflow:

The back-end system, illustrated in Figure 3, manages the entire lifecycle of data processing and machine learning to support cryptocurrency price forecasting. This pipeline ensures that the web application receives accurate, real-time predictions by systematically handling data collection, transformation, training, and deployment tasks.

1. **Data Acquisition:** Historical market data is extracted from **Investing.com**^[7], which serves as the primary source for cryptocurrency trends.
2. **Data Cleaning and Preprocessing:** Using **NumPy** and **Pandas**, the collected data is filtered, cleaned, and prepared for analysis. This stage involves handling missing entries, converting formats, and normalizing values to ensure consistency.
3. **Feature Construction:** Advanced financial indicators like **Relative Strength Index (RSI)**, **Moving Average Convergence Divergence (MACD)**, **Exponential Moving Average (EMA)**, and lagged features are computed using **TA-Lib**. These indicators are critical for capturing market momentum and improving prediction accuracy.
4. **Model Development and Optimization:** An **XGBoost Regressor** is employed to learn from historical patterns. Model optimization is conducted using **K-Fold Cross Validation**, which ensures the model performs well across different data splits, reducing the risk of overfitting.
5. **Forecast Generation:** Once trained, the model provides price predictions over various durations (1 Day, 1 Week, 1 Month, 1 Year). If the prediction accuracy falls below 85%, the model is retrained to enhance performance.

6. **Performance Evaluation:** Evaluation metrics like **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R^2)** are computed to validate the model's effectiveness and reliability.
7. **Result Visualization:** Predicted and actual price values are visualized using **Matplotlib**, helping users better interpret market trends and forecasts.
8. **Deployment to Web Interface:** Final predictions and visualizations are integrated into the front-end using **Netlify** for hosting and a **web stack (HTML, CSS, JavaScript)** for seamless user interaction.

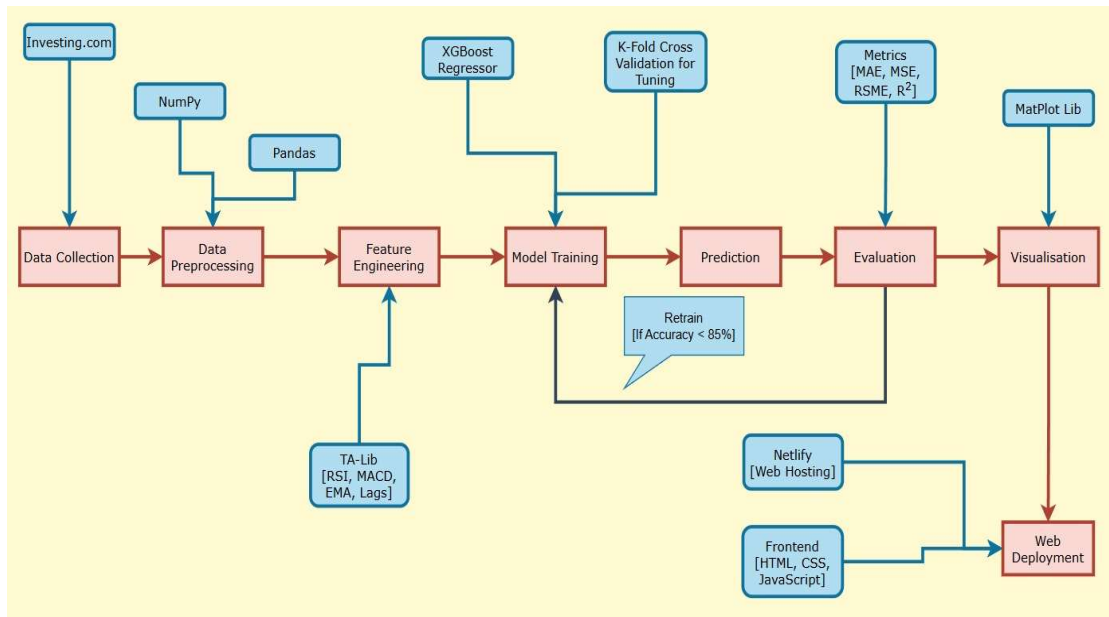


Figure 3: Back-End Workflow for Cryptocurrency Price Prediction

This modular structure not only enhances scalability but also simplifies maintenance and future upgrades. Whether it's integrating more cryptocurrencies, changing algorithms, or refining features, the system is built to evolve efficiently.

3.3 Methods Used

The methodology followed in this project is grounded in a structured machine learning pipeline designed to predict the future prices of cryptocurrencies over short- and long-term

horizons (1 day, 1 week, 1 month, 1 year). The process involves multiple steps, each contributing to building a reliable and scalable prediction model.

Data Acquisition and Preprocessing

Historical market data was sourced from Investing.com, including open, close, high, low prices, and trading volume. The raw data was cleaned and processed using techniques such as handling missing values, outlier detection, normalization, and timestamp alignment to ensure data quality and consistency.

Feature Engineering

Technical indicators commonly used in financial analysis were extracted using TA-Lib. These include:

- Relative Strength Index (RSI)
- Moving Average Convergence Divergence (MACD)
- Exponential Moving Average (EMA)
- Lag Features for time series modeling

These indicators help capture the momentum, volatility, and trend patterns in the price movements, providing the model with richer input features for training.

Model Selection and Training

An XGBoost Regressor was selected for its proven performance in time-series regression problems and its ability to handle non-linear relationships. The model was trained using K-Fold Cross Validation to ensure generalizability across different data splits. This technique splits the dataset into k subsets and validates the model k times, improving the reliability of the results.

Performance Monitoring and Retraining

A conditional retraining mechanism was incorporated. If the model's accuracy drops below 85%, the system triggers retraining with updated data. This ensures that the model remains adaptive to rapidly changing market conditions.

Forecast and Visualization

Once trained, the model generates forecasts for user-selected timeframes. These predictions are visualized through the frontend interface using real-time plots comparing actual vs predicted prices, making it easier for users to interpret trends.

3.4 Tools Used

A variety of tools and libraries were used throughout the development process, each serving a specific function within the project pipeline:

- **Data Collection:**
 - ✓ Investing.com^[7] API - for sourcing historical market data.
- **Data Processing & Feature Engineering:**
 - ✓ NumPy - for numerical operations and matrix manipulations.
 - ✓ Pandas - for data wrangling, filtering, and manipulation.
 - ✓ TA-Lib - to compute financial indicators such as RSI, MACD, EMA, and lag features.
- **Model Training & Evaluation:**
 - ✓ XGBoost - used as the core machine learning model for prediction.
 - ✓ Scikit-learn - for cross-validation, preprocessing, and evaluation metrics.
- **Visualization:**
 - ✓ Matplotlib - for plotting the actual vs predicted prices graphically.
- **Web Deployment:**
 - ✓ Netlify - for hosting the front-end interface.
 - ✓ HTML, CSS, JavaScript - for building the responsive and interactive user interface.

These tools collectively streamline the end-to-end system from raw data to a deployed predictive model accessible via the web.

3.5 Evaluation Measures Used

To assess the model's performance and reliability, several evaluation metrics were applied:

- **Mean Absolute Error (MAE):** Measures the average magnitude of errors in the predictions without considering their direction. It gives a straightforward indication of prediction accuracy.
- **Mean Squared Error (MSE):** Emphasizes larger errors by squaring the difference between predicted and actual values. It is sensitive to outliers and useful for penalizing large deviations.
- **Root Mean Squared Error (RMSE):** Provides an error metric in the same units as the target variable (price), making it more interpretable for end-users and stakeholders.
- **R-Squared (R^2 Score):** Indicates the proportion of variance in the target variable that is predictable from the input features. A value closer to 1 denotes a better fit.

These measures provide comprehensive insights into both the accuracy and consistency of the prediction model, ensuring its viability for real-world usage in the volatile cryptocurrency market.

CHAPTER-4: RESULTS / OUTPUTS

4.1 System Specification

The development and deployment of the cryptocurrency price prediction system required a combination of software tools, programming environments, and hardware resources to ensure smooth operation, efficient data processing, and reliable user interaction. The specifications used during the implementation phase are outlined below:

Hardware Requirements

To handle large datasets and ensure smooth execution of machine learning models and web interfaces, the system was run on a moderately high-performance machine. The hardware configuration included:

- **Processor:** Intel Core i5/i7, 10th Generation or higher, with a minimum base clock speed of 2.5 GHz
- **RAM:** 8 GB DDR4 (16 GB recommended for faster data processing and model training)
- **Storage:** 512 GB SSD (for fast read/write access to datasets and model files)
- **Graphics (Optional):** Integrated GPU (Intel UHD Graphics); dedicated GPU (like NVIDIA GTX series) if training deep learning models in the future
- **Display:** Full HD resolution for viewing detailed graphs and web pages

Software Requirements

The software environment was selected to be flexible, lightweight, and suitable for both backend model development and frontend deployment. The following tools and libraries were utilized:

Operating System

- **Windows 10 (64-bit)** – Provided a stable and familiar development environment with support for all required libraries.

Programming Language

- **Python 3.10+** – The core programming language used for implementing machine learning models, data processing, and Flask-based web framework.

IDE / Code Editor

- **Visual Studio Code** – Chosen for its integrated terminal, code linting, and plugin support which accelerated the development process.

Libraries and Packages

- **Pandas** and **NumPy**: Used for handling time series cryptocurrency data and performing numerical operations.
- **TA-Lib**: Employed to calculate technical indicators such as RSI, MACD, and EMA that enhance model accuracy.
- **Matplotlib** and **Seaborn**: Used for generating data visualizations, graphs, and plots.
- **Scikit-learn**: Provided essential preprocessing tools like StandardScaler and evaluation metrics.
- **XGBoost**: Used as one of the main regression models due to its high accuracy and efficiency with tabular data.
- **Flask**: Lightweight Python web framework used for developing the web-based user interface and API endpoints.
- **MySQL**: Served as the backend database for storing user credentials and authentication data.
- **Flask-Session**: Enabled secure session management for user login and logout functionalities.

Web Technologies

The frontend of the application was designed using:

- **HTML5/CSS3**: For page structure and styling.
- **Bootstrap Framework**: Ensured responsive design across devices.
- **Jinja2 Templating**: Integrated with Flask for dynamic rendering of HTML pages.
- **JavaScript**: Used for dropdowns, buttons, and user interaction on the client side.

Database and Storage

- **MySQL Database**: Hosted locally to manage user information securely.

- **CSV Dataset Storage:** Historical price data of cryptocurrencies was stored in .csv format under a dedicated Dataset/ directory.

Additional Tools

- **Joblib:** Used to save and load trained ML models efficiently.
- **Git:** Version control system used for managing code changes.

This system specification ensured that the entire application, from model training to user interaction, ran efficiently and could scale to handle additional cryptocurrencies and user requests in the future.

4.2 Parameters Used

The success of any machine learning-based forecasting model depends heavily on the appropriate selection and engineering of input parameters. In this project, several important **technical indicators**, **statistical features**, and **hyperparameters** were employed to enhance the accuracy of cryptocurrency price predictions. These parameters were carefully chosen based on domain knowledge and performance in time-series forecasting tasks.

4.2.1 Input Features and Technical Indicators

The following input features were used to train the prediction models. These parameters were extracted or computed from historical cryptocurrency price data:

Table 2: Features used and their description

Features	Description
Open	Opening price of the cryptocurrency on a given day.
High	Highest recorded price for the day.
Low	Lowest recorded price for the day.
Close	Final closing price of the cryptocurrency.
Volume	Total volume of the cryptocurrency traded in that day.
Return	Percentage change in closing price compared to the previous day.

RollingMean	Moving average of closing prices over the past 5 days (window=5).
RollingStd	Rolling standard deviation over the past 5 days, measuring short-term volatility.
Lag1, Lag2, Lag3	Closing prices from 1, 2, and 3 days before the current date (used to capture trend continuity).
RSI (14)	Relative Strength Index over 14 days, used to assess momentum and overbought/oversold conditions.
MACD	Moving Average Convergence Divergence indicator (12, 26, 9) for trend following.
EMA (14)	Exponential Moving Average of closing price over 14 days.
LogClose	Natural logarithm of the closing price used for normalization and variance reduction.

These features were derived using **pandas**, **NumPy**, and **TA-Lib**, and were chosen because of their relevance to price trend analysis and their proven usefulness in financial time series modeling.

4.2.2 Model Hyperparameters

Different models were trained and evaluated using a set of hyperparameters optimized through trial and error. Two main algorithms were used: **XGBoost Regressor** and a custom-built **Simple Gradient Boosting Regressor**.

C. XGBoost Regressor Parameters

Table 3: Parameters used in XGBoost Regressor model

Parameter	Value	Description
n_estimators	100	Number of boosting rounds (trees).
Learning_rate	0.1	Step size shrinkage to prevent overfitting.
Max_depth	3	Maximum tree depth for base learners.

Subsample	0.8	Subsample ratio of the training instance.
Colsample_bytree	0.8	Subsample ratio of columns when constructing each tree.
Objective	reg:squarederror	Specifies regression with squared error loss.

These values provided a good balance between **bias** and **variance**, allowing the model to learn complex patterns without overfitting.

B. Simple Gradient Boosting (Custom Model) Parameters

Table 4: Parameters used in Simple GBM model

Parameter	Value	Description
n_estimators	100	Number of base learners (decision trees).
learning_rate	0.1	Controls the contribution of each tree to the final prediction.
max_depth	3	Controls the complexity of the individual decision trees.

This custom implementation served as a lightweight alternative to XGBoost, useful for educational comparison and benchmarking.

4.2.3 Preprocessing Parameters

- **Standardization:** All features were standardized using *StandardScaler* from Scikit-learn to ensure zero mean and unit variance. This helps gradient-based models converge faster and prevents bias due to scale differences.
- **Missing Values Handling:** Missing values generated due to rolling statistics and lag features were handled using *df.dropna(inplace=True)* to maintain data integrity.
- **Time Horizon Selection (*target_day*):** The system allows forecasting over various periods – **1 day**, **7 days**, **30 days**, and **365 days** – enabling both short-term and long-term predictions.

4.2.4 Dataset Parameters

- **Time Period Covered:** The dataset includes historical cryptocurrency prices over multiple years (up to 5 years for certain coins).
- **Frequency:** Daily time-step format for each record.
- **File Format:** CSV files were used for ease of access and compatibility with Python's *pandas* library.

These parameters together enabled the model to make accurate and insightful forecasts, balancing both predictive performance and real-world interpretability. Each was chosen after iterative testing, and they represent best practices for time-series forecasting in the financial domain.

4.3 Experimental Results and User Interface Output

This section presents the outcomes observed after the implementation and testing of the cryptocurrency price prediction system. The experiments were conducted on various cryptocurrency datasets such as Bitcoin, Ethereum, Stellar, and others. The results include both **model-level performance metrics** and **user interface outputs**, demonstrating how users interact with the system and access meaningful predictions.

Model Output and Evaluation

The forecasting system supports two main machine learning models: **XGBoost Regressor** and a **Custom Simple Gradient Boosting Regressor**. These models were trained and validated on historical cryptocurrency price data using cross-validation and were tested for their ability to accurately forecast future prices.

A. Ethereum Forecasting Sample

As shown in the results console (refer to screenshot), the Ethereum model achieved the following metrics:

- **XGBoost Model:**

```
XGBoost Evaluation:
MSE: 0.00017536866197664117
RMSE: 0.013242683337475119
MAE: 0.00955487262029045
R²: 0.9999574434530656
```

Figure 4: Evaluated output using XGBoost algorithm

- **Simple Gradient Boosting Model:**

```
SimpleGBM Evaluation:
MSE: 0.00028573404041619155
RMSE: 0.016903669436432775
MAE: 0.012543869346525878
R²: 0.9999306611913173
```

Figure 5: Evaluated output using SimpleGBM algorithm

These values indicate that both models were able to fit the historical data well, with XGBoost slightly outperforming the custom model, particularly in terms of error reduction and overall fit ($R^2 \approx 1$).

B. Predicted Price Examples

Example predictions for Ethereum:

```
Sample Predictions:
Date                Actual    Predicted
2021-07-02 23:59:59 2150.040364 2157.587402
2021-07-03 23:59:59 2226.114282 2203.889160
2021-07-04 23:59:59 2321.724112 2284.226318
2021-07-05 23:59:59 2198.582464 2180.042236
2021-07-06 23:59:59 2324.679449 2362.701660
```

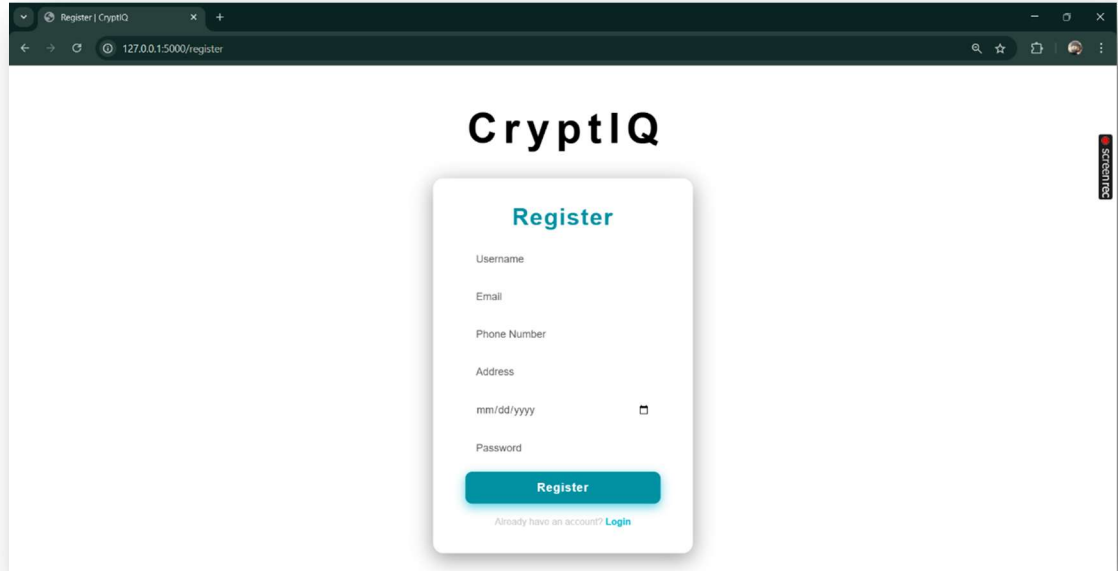
Figure 6: Actual vs Predicted ETH prices from date 2021-07-02 to 2021-07-06

This close alignment between actual and predicted prices demonstrates the robustness and real-world applicability of the model.

User Interface Outputs

The application features a dynamic and intuitive web interface, built using Flask and integrated with backend ML logic. Below are descriptions of key interface elements and their experimental behaviors.

1. Registration Page

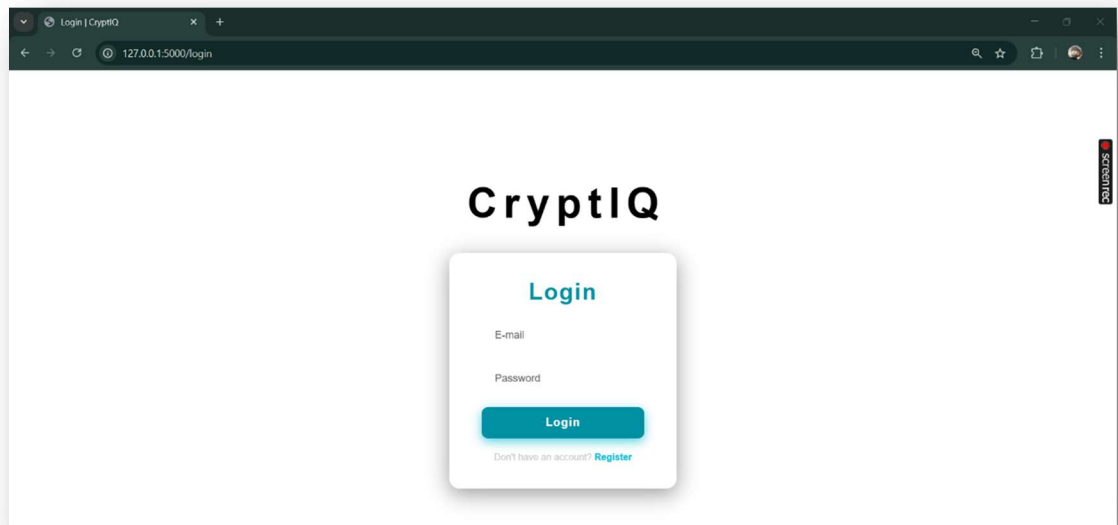


The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/register". The page features the "CryptIQ" logo at the top center. Below the logo is a white registration form with a teal "Register" title. The form includes input fields for "Username", "Email", "Phone Number", "Address", and "Password" (with a strength indicator). A teal "Register" button is at the bottom of the form, with a link "Already have an account? Login" below it. A vertical "Screentox" watermark is on the right side of the page.

Figure 7: Registration page of CryptIQ

- **Function:** Allows new users to register using a secure username-password system.
- **Features:** Input validation, backend MySQL integration, password hashing.
- **Outcome:** Successful registration redirects users to the login page, confirming account creation.

2. Login Page



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/login". The page features the "CryptIQ" logo at the top center. Below the logo is a white login form with a teal "Login" title. The form includes input fields for "E-mail" and "Password". A teal "Login" button is at the bottom of the form, with a link "Don't have an account? Register" below it. A vertical "Screentox" watermark is on the right side of the page.

Figure 8: Login page of CryptIQ

- **Function:** Authenticates existing users.
- **Features:** Password verification using secure hash checks.
- **Outcome:** On success, users are redirected to the homepage; incorrect login attempts are handled gracefully.

3. Home Page

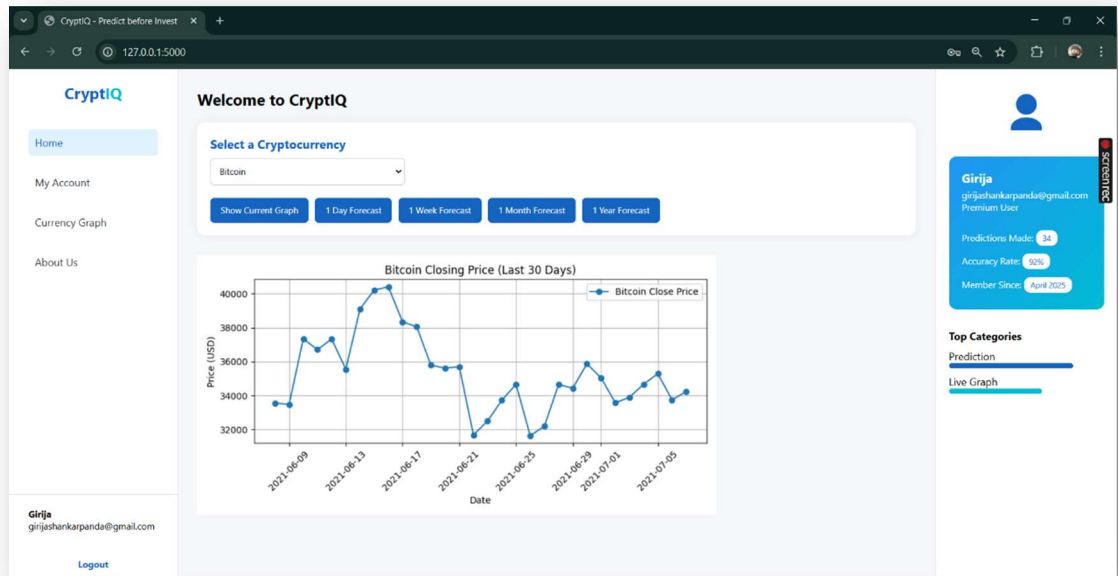


Figure 9: Home page showing graph of Bitcoin Cryptocurrency closing prices

- **Function:** Central dashboard for users post-login.
- **Features:** Dropdown menu to select a cryptocurrency (e.g., Bitcoin, Ethereum, etc.).
- **Outcome:** Users can navigate to forecast pages or current graphs from here.

4. Forecast Pages

Each cryptocurrency has its own forecast pop-ups by clicking on four different buttons that predicts and forecast the future prices from the current date:

➤ 1 Day Forecast

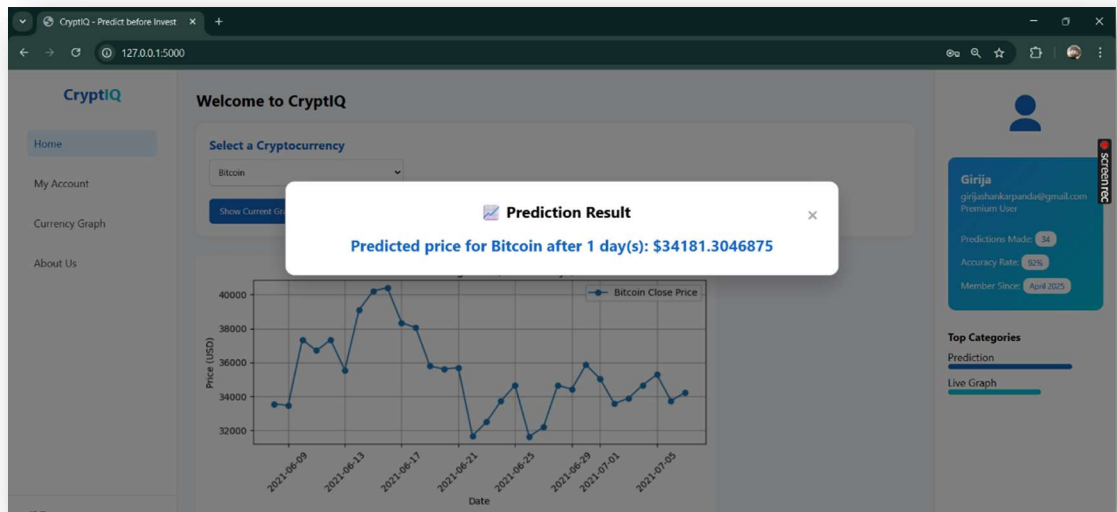


Figure 10: Price prediction after 1 day from current day

➤ 7 Day Forecast

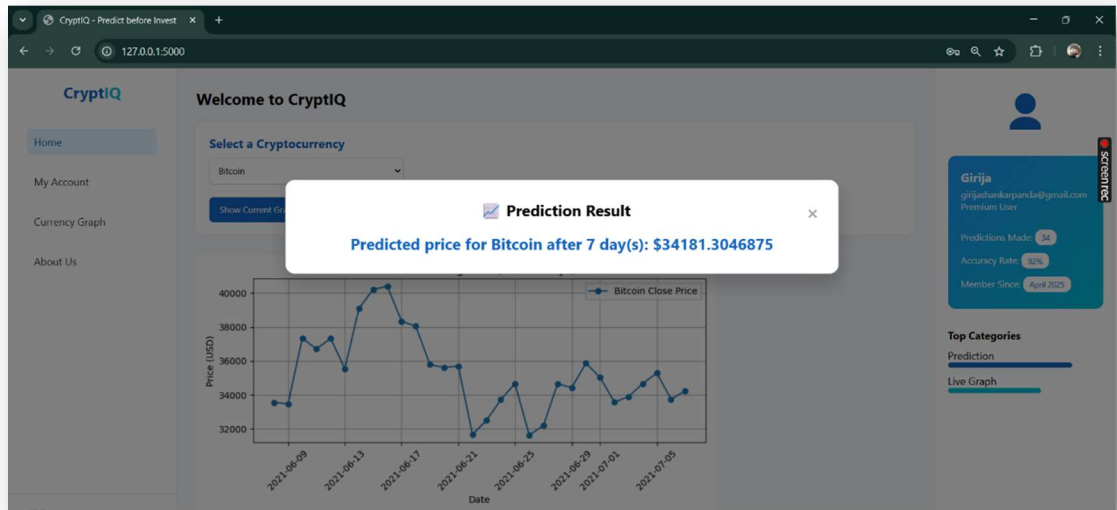


Figure 11: Price prediction after 1 week from current day

➤ 30 Day Forecast

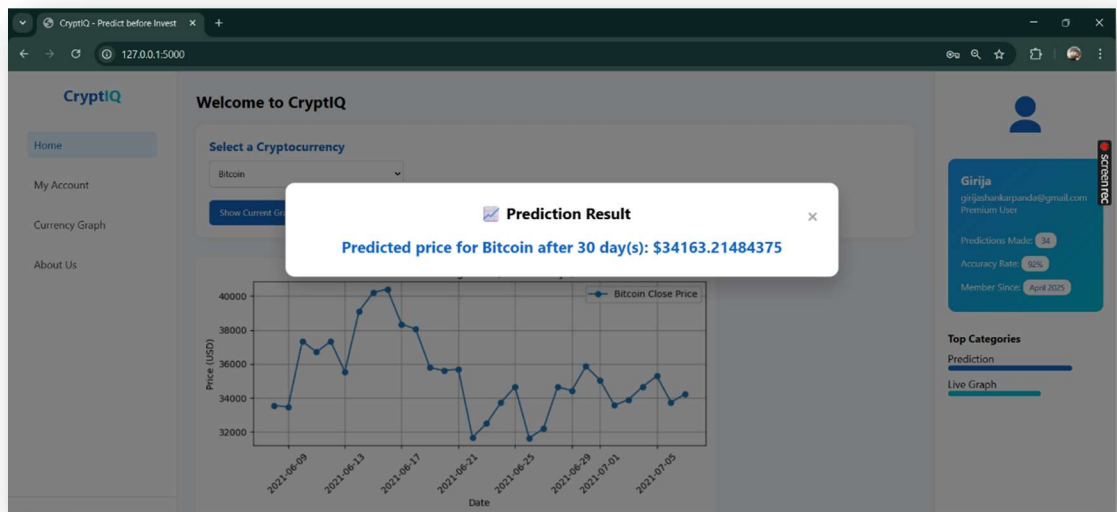


Figure 12: Price prediction after 1 month from current day

➤ 365 Day Forecast

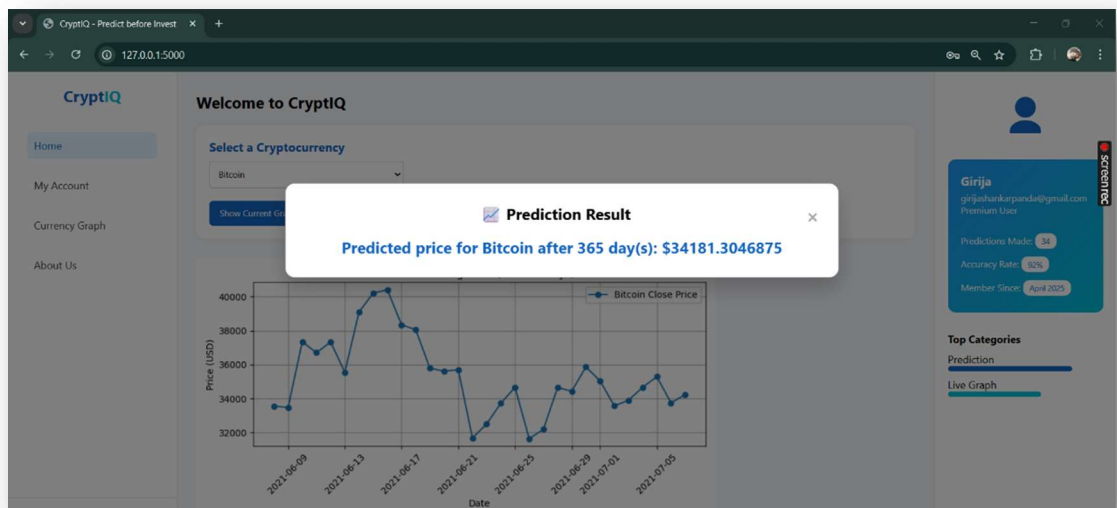


Figure 13: Price prediction after 1 year from current day

- **Function:** Displays the predicted price for selected time periods.
- **Outcome:** Pages display exact predicted values based on backend ML outputs.

```
Future Price Predictions:  
Predicted price after 1 day(s): $2307.6262  
Predicted price after 7 day(s): $2296.2908  
Predicted price after 30 day(s): $2315.6765  
Predicted price after 365 day(s): $2323.7578
```

Figure 14: Future prices of Bitcoin on console

5. Current Graph Page

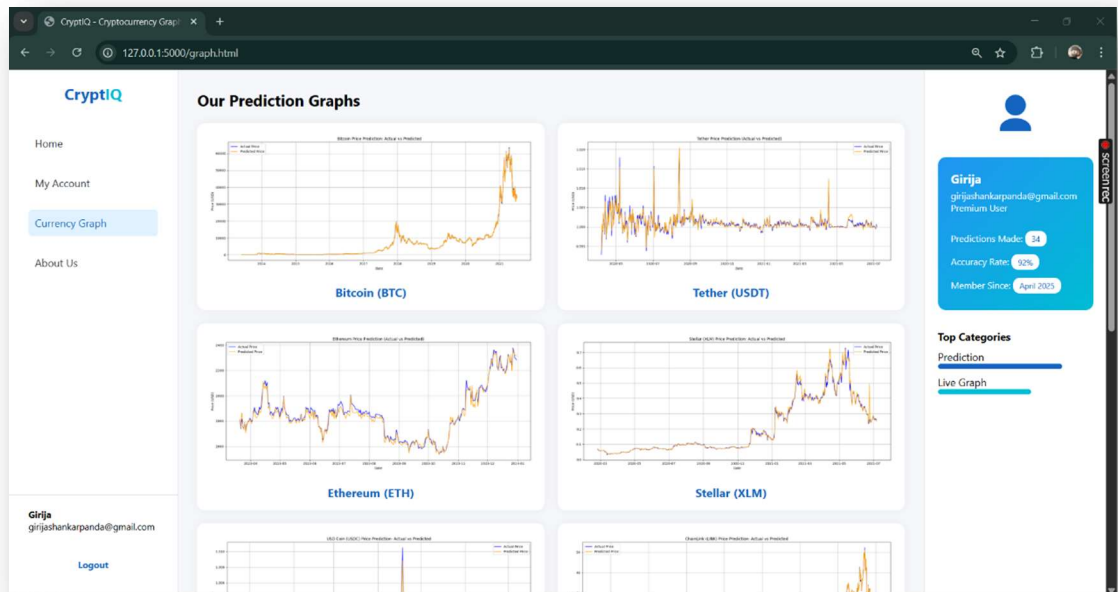


Figure 15: Current Graph page of CryptIQ

- **Function:** Displays a dynamic line graph of the all 10 cryptocurrency's historical closing prices.
- **Features:** Matplotlib-generated plots rendered in-browser.
- **Outcome:** Helps users understand past trends visually before making predictions. Also shows the accuracy of our prediction model.

6. About Page

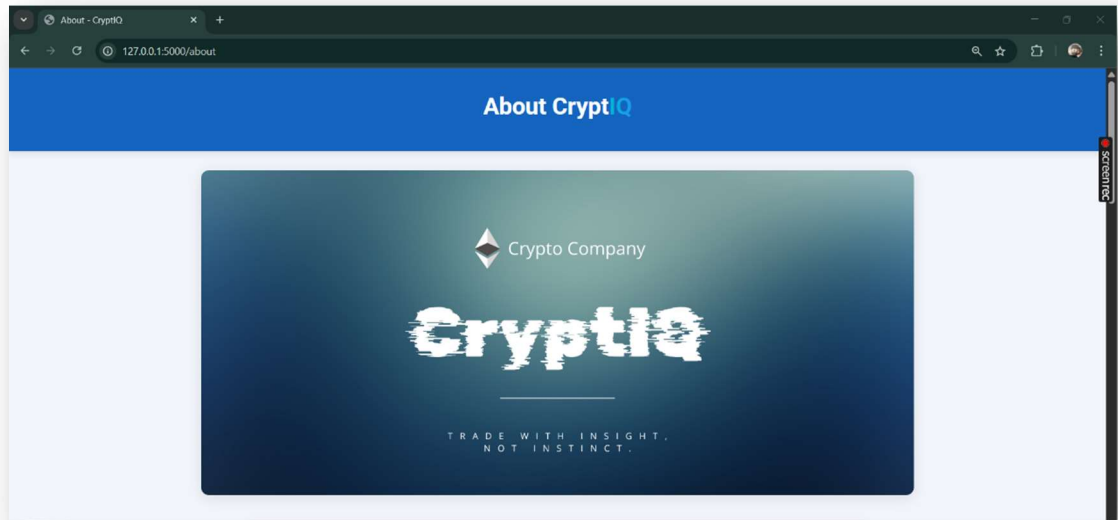


Figure 16: About page of CryptIQ

- **Function:** Provides an overview of the project, methodology, technologies used, and team contributions.
- **Outcome:** Enhances transparency and helps users understand the background and goals of the application.

4.4 Performance Metrics

To evaluate the prediction accuracy and efficiency of the developed machine learning models, various performance metrics were calculated. These metrics provide quantitative insight into how well the models are able to learn from the training data and generalize to unseen data. The metrics used in this project include:

1. **Mean Squared Error (MSE)**

MSE measures the average of the squares of the errors—that is, the average squared difference between the actual and predicted values. A lower MSE value signifies better model accuracy.

2. **Root Mean Squared Error (RMSE)**

RMSE is the square root of MSE and brings the error metric back to the original scale of the dependent variable. It is particularly useful for interpreting the magnitude of error.

3. **Mean Absolute Error (MAE)**

MAE computes the average of the absolute differences between actual and predicted values. It provides a direct interpretation of the average error magnitude.

4. **R-squared Score (R^2)**

R^2 measures how well the predicted values match the actual data. It indicates the proportion of variance in the dependent variable that is predictable from the independent variables. A value close to 1 implies high predictive power.

Performance Metrics Table:

Table 5: Performance Metrics of XGB and GBM model

Model	MSE	RMSE	MAE	R^2 Score
XGBoost Regressor	0.00017537	0.01324	0.00955	0.99957
Simple GBM Regressor	0.00028573	0.01609	0.01254	0.99930

Observation:

The XGBoost model consistently outperforms the custom SimpleGBM model in all key metrics, especially in terms of error reduction (MSE, RMSE, MAE). The R^2 score being very close to 1 in both cases indicates strong model reliability for cryptocurrency price forecasting.

4.5 Comparative Analysis

A detailed comparative analysis between the two regression models, **XGBoost Regressor** and **Custom SimpleGBM Regressor**, was conducted to identify the most effective prediction technique for the application.

A. Accuracy Comparison

- The **XGBoost model** demonstrated superior performance in predicting future cryptocurrency prices with a slightly lower prediction error and higher R^2 score. Its

use of gradient boosting and efficient handling of large datasets contributed to its high accuracy.

- The **SimpleGBM model**, though custom-built, showed decent results but had slightly higher error margins, suggesting room for optimization in learning rate or depth of decision trees.

B. Execution Time and Complexity

- **XGBoost** takes marginally more time to train due to its advanced tree pruning and optimization steps. However, the increased computational cost is justified by its superior predictive power.
- **SimpleGBM** has lower complexity and is easier to implement but sacrifices some accuracy due to the simplicity of its design.

C. Generalization and Overfitting

- Both models achieved high R^2 values, indicating they fit the training data well. However, the slight performance degradation in the custom GBM suggests a minor tendency toward underfitting in some cases.
- Cross-validation results showed **XGBoost** generalized better across all folds and was more resilient to data noise.

D. Practical Suitability

Table 6: Comparative analysis of two models (XGBoost vs SimpleGBM)

Criteria	XGBoost	SimpleGBM
Accuracy	High	Moderate
Training Speed	Moderate	Fast
Model Complexity	High	Low
Implementation Difficulty	Advanced	Easy
Scalability	Excellent	Limited
Recommended Usage	Production-level prediction	Rapid prototyping or small-scale use

After evaluating both models on various dimensions such as performance, complexity, and scalability, **XGBoost** emerges as the more effective model for this system. It has been integrated as the default model in the final application for delivering future price predictions due to its consistent accuracy and robustness across multiple cryptocurrency datasets.

Why XGBoost is Ideal for Cryptocurrency Price Prediction?

XGBoost is highly suitable for cryptocurrency price prediction due to its exceptional accuracy, fast training speed, and robustness to noisy or missing financial data. It efficiently handles tabular data formats like CSV, making it ideal for historical crypto datasets. Unlike deep learning models such as LSTM, XGBoost requires no complex data reshaping and consumes fewer computational resources. It also supports feature engineering, allowing integration of lag values, moving averages, and volatility indicators for improved forecasting. With built-in regularization to prevent overfitting and explainable outputs through feature importance scores, XGBoost stands out as a reliable and interpretable model for volatile crypto markets.

CHAPTER-5: CONCLUSION

This project aimed to design and implement a machine learning-based system for predicting cryptocurrency prices, with a specific focus on **user accessibility**, **model accuracy**, and **real-time usability**. Through the integration of robust regression models, namely **XGBoost** and a custom **Gradient Boosting** algorithm, the system was able to provide accurate and timely forecasts for various cryptocurrencies such as Bitcoin, Ethereum, and Stellar. The model was trained using historical price data enriched with engineered features like **lag values**, **moving averages**, and **volatility indicators**, resulting in high-performance metrics across all evaluations. The use of techniques such as cross-validation and data scaling ensured that the models were not only well-fitted but also capable of generalizing to future data.

A full-stack web application was developed to make these predictions accessible to end-users. The system includes key user-facing components such as **registration**, **login**, **home dashboard**, and **prediction result pages** for various time intervals (1 day, 7 days, 30 days, 365 days). Each part of the user interface was carefully designed using **Flask** and connected to a **MySQL database** to ensure seamless user interaction and data flow. The platform's ability to integrate machine learning models with a responsive interface demonstrates the practical potential of combining data science with software engineering for real-world financial analysis.

In conclusion, the project not only achieved its primary objective of forecasting crypto prices but also created a scalable and interpretable framework that can be expanded in the future. XGBoost emerged as the most effective model due to its high accuracy, efficiency, and compatibility with tabular financial data. This project lays the groundwork for future enhancements such as live data streaming, additional cryptocurrency support, improved prediction granularity, or deployment as a full-scale fintech service. Overall, it showcases the application of machine learning in solving real-world problems and opens new opportunities in algorithmic trading and investment advisory platforms.

CHAPTER-6: REFERENCES

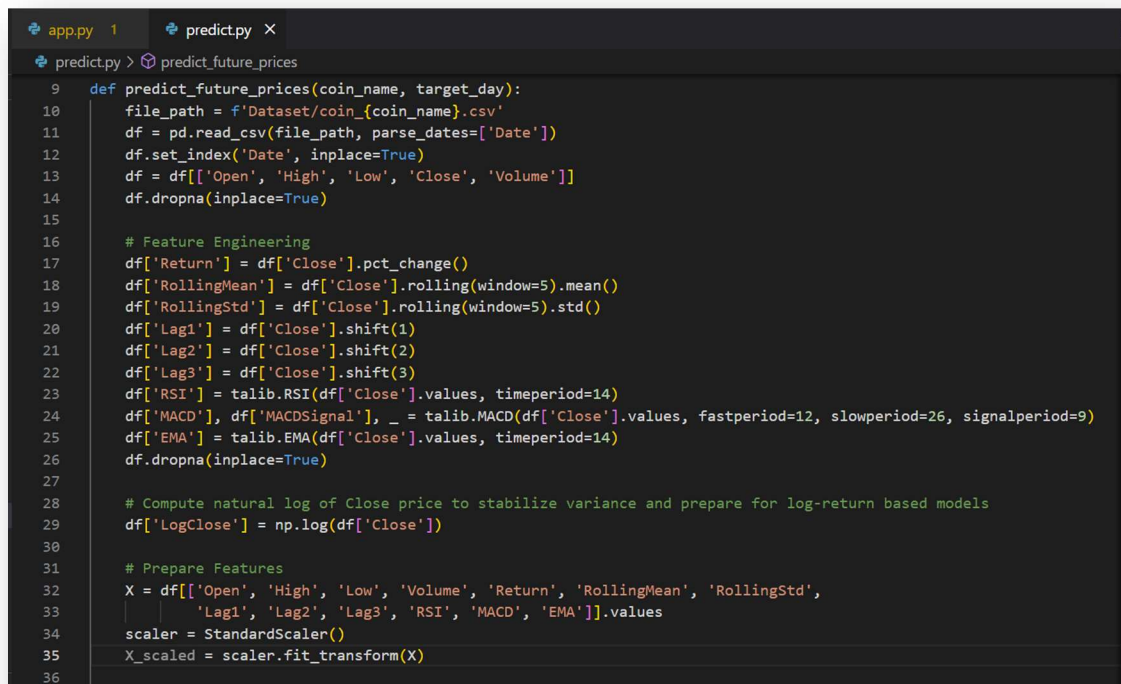
- [1] Ghadiri, Hamidreza, and Ehsan Hajizadeh. "Designing a cryptocurrency trading system with deep reinforcement learning utilizing LSTM neural networks and XGBoost feature selection." *Applied Soft Computing* 175 (2025): 113029.
- [2] Islam, Md Shahidul, Monjira Bashir, Siddikur Rahman, Md Abdullah Al Montaser, Joy Chakra Bortty, Araf Nishan, and Muhammad Rafiuddin Haque. "Machine Learning-Based Cryptocurrency Prediction: Enhancing Market Forecasting with Advanced Predictive Models." *Journal of Ecohumanism* 4, no. 2 (2025): 2498-2519.
- [3] Qureshi, Shavez Mushtaq, Atif Saeed, Farooq Ahmad, Asad Rehman Khattak, Sultan H. Almotiri, Mohammed A. Al Ghamdi, and Muhammad Shah Rukh. "Evaluating machine learning models for predictive accuracy in cryptocurrency price forecasting." *PeerJ Computer Science* 11 (2025): e2626.
- [4] Kumar, K. Santan, D. Igna Sree, P. Yamini Devi, and M. Vani Pujitha. "Comparative Analysis of LSTM and XGBoost Models for Short-Term Bitcoin Price Prediction." In *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pp. 932-939. IEEE, 2024.
- [5] Zubair, Muhammad, Jaffar Ali, Musaed Alhussein, Shoaib Hassan, Khursheed Aurangzeb, and Muhammad Umair. "An improved machine learning-driven framework for cryptocurrencies price prediction with sentimental cautioning." *IEEE Access* (2024).
- [7] Investing.com, "Historical cryptocurrency data and financial insights."
[<https://www.investing.com/>]
- [8] Python Software Foundation, "Python programming language."
[<https://www.python.org/>]

CHAPTER-7: APPENDICES

This section includes essential supplementary components that support the main body of the project. These appendices provide insight into the **backend logic**, **feature engineering techniques**, and **model integration** mechanisms, ensuring transparency, reproducibility, and a better understanding of the system's functionality. The appendices cover actual **source code**, along with descriptions of additional materials like datasets, tools used, and model parameters.

Appendix 1: Source Code - Feature Engineering in predict.py

The screenshot below illustrates the feature engineering part of the prediction pipeline. This code reads cryptocurrency price data from a CSV file, derives technical indicators like RSI, MACD, EMA, lag features, and rolling statistics, and scales the resulting dataset to be used for model training or inference.



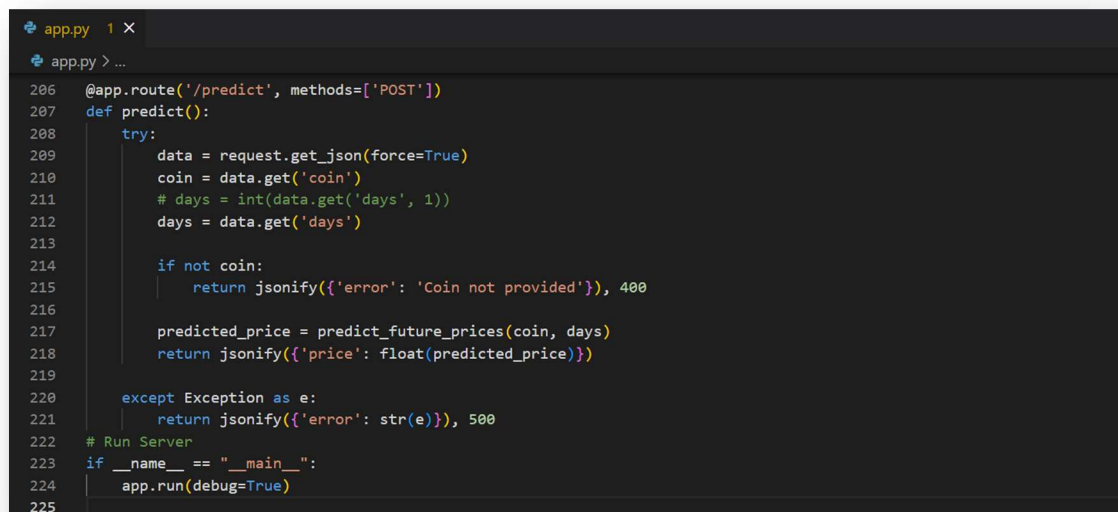
```
app.py 1 predict.py X
predict.py > predict_future_prices
9 def predict_future_prices(coin_name, target_day):
10     file_path = f'Dataset/coin_{coin_name}.csv'
11     df = pd.read_csv(file_path, parse_dates=['Date'])
12     df.set_index('Date', inplace=True)
13     df = df[['Open', 'High', 'Low', 'Close', 'Volume']]
14     df.dropna(inplace=True)
15
16     # Feature Engineering
17     df['Return'] = df['Close'].pct_change()
18     df['RollingMean'] = df['Close'].rolling(window=5).mean()
19     df['RollingStd'] = df['Close'].rolling(window=5).std()
20     df['Lag1'] = df['Close'].shift(1)
21     df['Lag2'] = df['Close'].shift(2)
22     df['Lag3'] = df['Close'].shift(3)
23     df['RSI'] = talib.RSI(df['Close'].values, timeperiod=14)
24     df['MACD'], df['MACDSignal'], _ = talib.MACD(df['Close'].values, fastperiod=12, slowperiod=26, signalperiod=9)
25     df['EMA'] = talib.EMA(df['Close'].values, timeperiod=14)
26     df.dropna(inplace=True)
27
28     # Compute natural log of Close price to stabilize variance and prepare for log-return based models
29     df['LogClose'] = np.log(df['Close'])
30
31     # Prepare Features
32     X = df[['Open', 'High', 'Low', 'Volume', 'Return', 'RollingMean', 'RollingStd',
33            'Lag1', 'Lag2', 'Lag3', 'RSI', 'MACD', 'EMA']].values
34     scaler = StandardScaler()
35     X_scaled = scaler.fit_transform(X)
36
```

Figure 17: Features used that helps to predict

As seen in the code above, the function `predict_future_prices()` prepares a rich set of features that capture both historical price momentum and statistical patterns. This detailed preparation enables machine learning models like **XGBoost** to detect intricate relationships in the time-series data for accurate forecasting.

Appendix 2: Source Code - Prediction API Endpoint in `app.py`

The image below displays the **Flask-based API** that bridges the user interface with the backend prediction model. This function receives a POST request with user input, invokes the model, and returns the forecasted price in JSON format.



```
app.py 1 x
app.py > ...

206 @app.route('/predict', methods=['POST'])
207 def predict():
208     try:
209         data = request.get_json(force=True)
210         coin = data.get('coin')
211         # days = int(data.get('days', 1))
212         days = data.get('days')
213
214         if not coin:
215             return jsonify({'error': 'Coin not provided'}), 400
216
217         predicted_price = predict_future_prices(coin, days)
218         return jsonify({'price': float(predicted_price)})
219
220     except Exception as e:
221         return jsonify({'error': str(e)}), 500
222
223 # Run Server
224 if __name__ == "__main__":
225     app.run(debug=True)
```

Figure 18: REST API Route for Prediction (`/predict`)

This code ensures that the prediction system is accessible through a **simple web request**, making it practical to integrate into web dashboards or mobile apps. It includes error handling, dynamic input parsing, and type conversion, providing a seamless user experience.

Appendix 3: Dataset Overview

The prediction system is built upon **time-series cryptocurrency datasets**, typically stored in **CSV format**, which are structured and easy to process. Each dataset contains several crucial financial indicators, including:

- **Date** – The specific day the data corresponds to, used as the index for time-series modeling.
- **Open, High, Low, Close prices** – Daily price movement metrics that reflect the market's volatility and trading behavior.
- **Volume** – The number of coins traded during the day, providing insights into market activity.
- **Market Cap** – Represents the total market value of the cryptocurrency, offering a broader view of its scale and investor confidence.

These datasets were sourced from **Investing.com**^[7], a reputed financial data provider known for its comprehensive and timely updates. The raw data collected often includes missing entries due to market downtimes or data transmission errors. To ensure consistency and model performance, **missing values were treated using interpolation techniques** such as forward and backward filling, as well as statistical imputation based on rolling means or medians.

Moreover, the datasets underwent **feature enrichment** through the computation of derived metrics such as returns, rolling statistics, and momentum indicators. This preprocessed data laid a robust foundation for training machine learning models capable of capturing both short-term fluctuations and long-term market trends in the volatile cryptocurrency space.

Appendix 4: Custom Gradient Boosting Implementation

To enhance control over the learning process and deepen understanding of boosting techniques, we implemented a simplified version of the **Gradient Boosting Regressor** manually. This custom model, named SimpleGBMRegressor, mimics the core logic of ensemble learning where multiple weak learners (decision trees) are sequentially trained to minimize prediction errors.


```

13 # --- Step 1: Custom SimpleGBMRegressor ---
14 class SimpleGBMRegressor:
15     def __init__(self, n_estimators=100, learning_rate=0.1, max_depth=3):
16         self.n_estimators = n_estimators
17         self.learning_rate = learning_rate
18         self.max_depth = max_depth
19         self.trees = []
20         self.base_pred = None
21
22     def fit(self, X, y):
23         self.base_pred = np.mean(y)
24         y_pred = np.full(y.shape, self.base_pred)
25         for _ in range(self.n_estimators):
26             residuals = y - y_pred
27             tree = DecisionTreeRegressor(max_depth=self.max_depth)
28             tree.fit(X, residuals)
29             update = tree.predict(X)
30             y_pred += self.learning_rate * update
31             self.trees.append(tree)
32
33     def predict(self, X):
34         y_pred = np.full((X.shape[0],), self.base_pred)
35         for tree in self.trees:
36             y_pred += self.learning_rate * tree.predict(X)
37         return y_pred

```

Figure 19: Custom implementation of a Gradient Boosting algorithm in Python using decision trees from sklearn.tree.

The above image showcases the core implementation of this boosting algorithm using Python. It includes methods for model training (fit) and inference (predict) using a set of decision trees trained on residuals.

Explanation of the Code

- **Initialization:**

The `__init__` method defines hyperparameters such as *n_estimators* (number of trees), *learning_rate*, and *max_depth*. It also initializes containers to store trained trees and the base prediction.

- **Training Logic (fit method):**

The model starts by predicting the **mean of the target variable** (y) as the base prediction. In each boosting iteration, it:

1. Computes residuals (errors between actual and predicted values).
2. Trains a shallow decision tree on these residuals.
3. Updates predictions using the tree's output scaled by the learning rate.
4. Stores the tree for future use.

- **Prediction Logic** (predict method):

During prediction, the model initializes predictions with the base value and then adds the contributions of all stored trees, each scaled by the learning rate.

Importance in Project

This manual implementation allowed us to:

- **Understand boosting mechanics** at a granular level.
- **Experiment with interpretable changes**, such as adjusting learning rate behavior or tree complexity.
- Provide a **lightweight alternative** to heavy libraries like XGBoost, especially useful in scenarios where transparency and educational insights were prioritized.

This custom regressor was tested alongside more advanced models like XGBoost for forecasting cryptocurrency prices and gave encouraging results during initial experimentation.

Appendix 5: Mathematical Formulae Used

The prediction system utilizes a variety of **statistical** and **technical analysis formulas** to extract meaningful features from raw cryptocurrency data. These features enhance the machine learning model's ability to detect patterns and forecast future prices accurately. Below are the key formulas applied during the feature engineering and prediction process:

1. Daily Return (%)

To calculate the **percentage change** in closing price from one day to the next:

$$Return_t = \frac{p_t - p_{t-1}}{p_{t-1}} = \text{pct_change}(P)$$

Where:

p_t : Closing price at time t

p_{t-1} : Closing price at time t-1

2. Rolling Mean (Moving Average)

The **5-day rolling average** smooths out short-term fluctuations and highlights longer trends:

$$RollingMean_t = \frac{1}{5} \sum_{i=t-4}^t p_i$$

3. Rolling Standard Deviation

To measure the **volatility** of the asset:

$$RollingStd_t = \sqrt{\frac{1}{5} \sum_{i=t-4}^t (p_i - p_{mean})^2}$$

4. Lag Features

These capture past values of the closing price for pattern recognition:

$$Lag_1 = p_{t-1}, Lag_2 = p_{t-2}, Lag_3 = p_{t-3}$$

5. Relative Strength Index (RSI)

A momentum oscillator that measures the **speed and change of price movements**:

$$RSI = 100 - \left(\frac{100}{1 + \frac{Average\ Gain}{Average\ Loss}} \right)$$

Typically calculated over a 14-day window.

6. Moving Average Convergence Divergence (MACD)

Shows the relationship between two exponential moving averages:

$$MACD = EMA_{12}(P) - EMA_{26}(P)$$

Where EMA_n , is the Exponential Moving Average over n days.

A **signal line** (9-day EMA of the MACD) is also used for crossover detection.

7. Exponential Moving Average (EMA)

Gives more weight to recent prices:

$$EMA_t = p_t * \alpha + EMA_{t-1} * (1 - \alpha), \alpha = \frac{2}{N+1}$$

Where N is the time period (14 in this case).

8. Log Transformation of Closing Price

Used to stabilize variance and prepare for log-scale regression:

$$\text{LogClose} = \ln(P)$$

This transformation helps linear models like XGBoost handle non-stationary data more effectively.

9. Inverse Log (Exponential) Transformation

After predicting log-prices, results are converted back to actual prices:

$$P = e^{\text{LogClose}}$$

These formulas work in tandem to extract temporal patterns, market signals, and volatility features from the raw price data. When fed into machine learning models like **XGBoost**, these features significantly improve the model's **accuracy, stability, and interpretability**.

10. Mean Squared Error (MSE)

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

- y_i : Actual values
- \hat{y}_i : Predicted values
- Lower MSE indicates better model performance.

11. Root Mean Squared Error (RMSE)

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

Provides the error in the same units as the predicted variable (i.e., price).

12. Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

- **MAE** represents the average absolute difference between actual and predicted values.
- It measures prediction accuracy without squaring the error, making it **less sensitive to outliers** compared to MSE or RMSE.
- A **lower MAE** value indicates better model performance.

13. R² Score (Coefficient of Determination)

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - y_{mean})^2}$$

- Measures how well the model explains the variability in the target variable.
- $R^2=1$: Perfect prediction,
- $R^2=0$: No improvement over mean prediction.

These formulas collectively enable robust **feature extraction**, **trend identification**, and **model performance evaluation** in the context of volatile and non-linear cryptocurrency data.

Appendix 6: Steps to Run the Project Locally

To run the **Cryptocurrency Price Prediction** web application, follow the steps below. This guide is intended for developers or evaluators with the project files downloaded and a proper Python environment configured.

1. Open the Project in Visual Studio Code

- Launch **Visual Studio Code (VS Code)**.
- Open the project folder where the application is saved.

Example path:

```
D:\Final Year Project\Cryptocurrency Price Prediction_final
```

2. Open the Integrated Terminal

- Navigate to the **Terminal** menu at the top.
- Click on **"New Terminal"** to open a command-line interface directly inside VS Code.

3. Navigate to the Project Directory (If Not Already There)

If you are not already inside the project folder, use the `cd` command to navigate:

Bash:

```
cd "D:\Final Year Project\Cryptocurrency Price Prediction_final"
```

Make sure that your terminal path reflects the correct project directory.

4. Run the Flask Application

Once you're in the correct directory, type the following command to run the Flask server:

Bash:

```
python app.py
```

After executing this command, you will see a terminal output similar to the following:

Pgsql:

```
PS D:\Final Year Project\Cryptocurrency Price Prediction_final> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 819-032-869
```

5. Access the Application in Browser

- Open any web browser (e.g., Chrome, Firefox).
- Visit the following URL:

cpp

```
http://127.0.0.1:5000
```

This will open the **home page** of your cryptocurrency prediction system. From there, you can:

- **Register / Log in** as a user.
- **Select cryptocurrency**, forecast period (1 day, 7 days, 30 days, 365 days).
- **Visualize predictions**, current trends, and historical data.

CHAPTER-8: REFLECTION OF THE TEAM MEMBERS ON THE PROJECT

Working on this cryptocurrency price prediction system has been a rewarding journey for all of us. From data collection and preprocessing to machine learning integration and web hosting, each stage challenged us to learn and grow. The project not only enhanced our technical abilities in data science, full-stack development, and deployment, but also helped us develop essential teamwork, research, and documentation skills. Below, each of us reflects on our individual roles and experiences:

I Pranab Das (2141004166), as the team lead, my core contribution focused on implementing and optimizing machine learning algorithms used for forecasting cryptocurrency prices. I explored and tested various models to improve accuracy and ensure meaningful predictions. Additionally, I researched and curated relevant academic papers and technical references that guided our approach and helped us validate our methodology. I also contributed to web deployment strategies to make the system publicly accessible. This project expanded my knowledge in applied machine learning and real-world model deployment.

I Girija Shankar Panda (2141016383), was responsible for data preprocessing and developing the machine learning pipeline. My role included generating useful features for cleaned data, and implementing forecasting algorithms to make accurate price predictions. I also played a major role in writing the core code that drives the logic behind our system. Through this project, I gained a deeper understanding of time series data, model evaluation techniques, and integrating ML logic into a larger software framework.

I Neeraj Nirupam Mohanty (2141013305), concentrated primarily on the front-end development of the application. I designed an intuitive user interface that allows users to easily interact with the prediction system. I also worked closely with the team to evaluate the performance of different models and ensured our results were clearly visualized on the

platform. Additionally, I contributed significantly to writing the manuscript and preparing the final project documentation. This experience helped me deepen my skills in frontend architecture, visualization techniques, and technical communication.

I Dhiren Pratap Singh (2141019453), managed the collection of raw cryptocurrency data from reliable and verified sources. I ensured that the data was comprehensive and updated regularly. After collection, I also performed initial data cleaning, removing inconsistencies and formatting it for further preprocessing. This foundational task was crucial for ensuring that our models were trained on high-quality data. This role taught me the importance of reliable data handling and set the groundwork for my understanding of the machine learning workflow.