****

**Abstract**

Cryptocurrency has garnered increasing attention due to its decentralized nature and investment potential. However, its high volatility poses significant challenges for accurate price prediction. This study presents a deep learning-based framework for predicting Bitcoin (BTC) prices, integrating both historical market data and blockchain-derived (on-chain) data. To enhance model robustness, change point detection is used for segment-wise normalization, improving stability in dynamic price ranges. A novel Self-Attention-based Multiple Long Short-Term Memory (SAM-LSTM) architecture is proposed, enabling efficient modeling of on-chain variable groups and long-term dependencies. Additionally, the Informer model—a Transformer-based architecture optimized for long-sequence time-series forecasting—is employed, utilizing ProbSparse self-attention and sequence distillation for computational efficiency. Experimental results demonstrate the superiority of the proposed methods, particularly the Informer, achieving a prediction accuracy of 97%. Future work aims to incorporate multivariate data sources, such as social media sentiment and on-chain metrics, and explore real-time forecasting applications.

**List of Contents**

|  |  |
| --- | --- |
| **Title** | **Page No.** |
| Abstract | 1 |
| List of Contents | 2 |
| List of Figures | 3 |
| List of Tables | 4 |
| Chapter 1 Introduction | 5 |
| 1.1 Introduction | 6 |
| 1.2 Problem Statement | 6 |
| 1.3 Objectives of the Study | 7 |
| Chapter 2 Literature Survey | 8 |
| Chapter 3 System Analysis & Feasibility Study | 11 |
| 3.1 Existing System | 12 |
| 3.1.1 Limitations of the Proposed Model | 12 |
| 3.2 Proposed system | 13 |
| 3.2.1 Advantages of proposed system | 13 |
| 3.2.2 Dataset | 14 |
| 3.2.3 Data Preprocessing | 15 |
| 3.3 Methodology | 15 |
| 3.4 Training Process | 17 |
| 3.5 Prediction and Evaluation | 17 |
| Chapter 4 System Requirements | 18 |
| 4.1 Functional Requirements | 19 |
| 4.2 Technologies and Languages used to develop | 19 |
| 4.2.1 Compliers (Offline and Online) | 19 |
| 4.2.2 Hardware Requirements | 20 |
| 4.2.3 Software Requirements | 20 |
| Chapter 5 Design | 21 |
| 5.1 System Design | 22 |
| Chapter 6 Implementation | 24 |
| 6.1 Informer model | 25 |
| Chapter 7 Results | 29 |
| Chapter 8 Social Impact | 31 |
| Chapter 9 Conclusion & Future Work | 33 |
| BIBLIOGRAPHY | 35 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Description** | **Page No.** |
| 3.1 | Dataset – Bitcoin data | 14 |
| 3.2 | Block diagram of Informer model | 16 |
| 5.1 | Work flow of the Informer model | 22 |
| 7.1 | Graph of Proposed system | 30 |
| 7.2 | Graph of Existing system | 30 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Description** | **Page No.** |
| 3.1 | Key Architectural Parameters | 16 |
| 7.1 | Results SAM-LSTM vs Informer model | 30 |

**Chapter 1**

**Introduction**

**1.INTRODUCTION**

**1.1 Introduction**

Cryptocurrencies, particularly Bitcoin (BTC), have garnered significant attention from investors due to their decentralized nature, transparency, and potential for high returns. However, their extreme volatility and unique characteristics pose challenges for accurate price prediction, which is crucial for developing effective investment strategies. Traditional financial prediction models often fall short when applied to cryptocurrencies, as they fail to account for the distinct features of blockchain-based assets, such as on-chain data—a rich source of information inherent to cryptocurrency ecosystems.

This paper introduces a novel deep learning-based framework for predicting Bitcoin prices by leveraging on-chain data and addressing two critical issues in existing literature. First, the framework employs a change point detection (CPD) technique to segment time-series data, enabling separate normalization for each segment to handle severe price fluctuations. Second, it utilizes a comprehensive set of on-chain variables, categorized into groups (e.g., price, adoption, distribution, market, and valuation), to enhance prediction accuracy. The proposed model, dubbed Self-Attention-based Multiple Long Short-Term Memory (SAM-LSTM), integrates multiple LSTM modules for each variable group and an attention mechanism to extract meaningful temporal patterns.

Experimental results using real-world Bitcoin price data demonstrate the effectiveness of the proposed framework, achieving superior performance in terms of MAE, RMSE, MSE, and MAPE compared to baseline methods. The contributions of this work include: Extensive use of on-chain data for Bitcoin price prediction, capturing unique blockchain dynamics, CPD-based segmentation and normalization to stabilize predictions in unseen price ranges, SAM-LSTM architecture, combining multiple LSTMs and attention mechanisms for robust feature extraction.

This study advances cryptocurrency price prediction by addressing non-stationarity and leveraging blockchain-specific data, offering valuable insights for investors and stakeholders in the rapidly evolving digital asset market.

* 1. **Problem Statement**

Accurate prediction of cryptocurrency prices, particularly Bitcoin (BTC), is crucial for investors and traders due to its extreme volatility and unique market behavior. However, existing price prediction models face two key limitations: Handling Extreme Price Volatility and Non-Stationarity: Cryptocurrency markets exhibit sudden and severe price fluctuations, making traditional time-series forecasting methods unreliable, most models normalize data globally, ignoring abrupt shifts in market regimes, leading to poor generalization when prices enter previously unseen ranges.

Underutilization of Blockchain-Specific Data: Many prediction models rely solely on historical price data or external factors like social media sentiment, overlooking the rich insights available in on-chain data (e.g., transaction volumes, miner activity, wallet distributions), without incorporating these blockchain-native signals, models fail to capture fundamental drivers of cryptocurrency price movements.

To overcome these challenges, this study seeks to develop a robust prediction framework that: Detects structural changes in price trends using segmentation techniques to improve normalization and adaptability, integrates diverse on-chain variables to enhance predictive power by capturing underlying blockchain dynamics, ensures stability in volatile conditions while maintaining interpretability for real-world financial decision-making.

* 1. **Objectives of the Study**

The primary aim of this study is to enhance the accuracy and reliability of Bitcoin (BTC) price predictions by addressing two critical challenges in cryptocurrency forecasting: Mitigating the Impact of Extreme Volatility with developing a methodology that effectively handles non-stationary price trends and abrupt market shifts to improve prediction stability. Leveraging Blockchain-Specific Data by incorporate on-chain metrics (e.g., transaction activity, miner behavior, wallet distributions) to capture intrinsic market dynamics that traditional models overlook.

By integrating these approaches, the study seeks to provide a more robust framework for cryptocurrency price forecasting, enhance practical utility for investors and traders navigating highly volatile markets, establish a data-driven foundation for future research in blockchain-based financial analysis.

The ultimate goal is to advance the field of cryptocurrency price prediction by combining time-series segmentation techniques with blockchain-native data, offering a more reliable tool for real-world financial decision-making.

**Chapter 2**

**Literature Survey**

**2. LITERATURE SURVEY**

[1] S. K. Ibrahim and P. Singh proposed a machine learning approach to accurately assess Bitcoin prices by examining various influencing parameters. Their work focuses on understanding the daily market evolution of Bitcoin and gaining insights into the most relevant aspects surrounding its price. They tested several regression models using trading data from the Bitcoin exchange website Bitstamp, aiming to find the most efficient and accurate model for predicting Bitcoin prices.

[2] S. Singh and M. Bhat explored a transformer-based neural network for Ethereum price forecasting, hypothesizing that cryptocurrency prices are strongly correlated with other cryptocurrencies and the sentiments surrounding them. Despite a smaller dataset and less complex architecture, their transformer model surpassed ANN and MLP counterparts on some parameters, suggesting the significant influence of sentiment on cryptocurrency price movements. ​

[3] J. Liu et al. introduced a transformer-based capsule network for stock movement prediction, leveraging the capabilities of transformers to capture complex patterns in financial data. Their approach demonstrated improved performance in forecasting stock movements, indicating the potential applicability of transformer models in financial time series analysis. ​

[4] M. A. Labbaf Khaniki and M. Manthouri presented an innovative approach for predicting cryptocurrency time series, specifically focusing on Bitcoin, Ethereum, and Litecoin. Their methodology integrates technical indicators, a Performer neural network, and BiLSTM to capture temporal dynamics and extract significant features from raw cryptocurrency data. The application of technical indicators facilitates the extraction of intricate patterns, momentum, volatility, and trends. ​

[5]S. Tanwar et al. proposed a deep learning-based cryptocurrency price prediction scheme that considers inter-dependent relations. Their approach utilizes advanced neural network architectures to capture the complex dependencies among various cryptocurrencies, enhancing the accuracy of price predictions. ​

[6] M. M. Patel, S. Tanwar, R. Gupta, and N. Kumar developed a deep learning-based cryptocurrency price prediction scheme for financial institutions. Their model leverages deep learning techniques to analyze historical price data and predict future trends, providing valuable insights for financial decision-making.

[7] Y. Wen conducted an evaluation of cryptocurrency price prediction using LSTM and CNN models. The study compares the performance of these deep learning architectures in forecasting cryptocurrency prices, highlighting the strengths and limitations of each approach.

[8] J. Wu, X. Zhang, F. Huang, H. Zhou, and R. Chandra reviewed deep learning models for cryptocurrency price prediction, implementing and evaluating various architectures. Their comprehensive analysis provides insights into the effectiveness of different deep learning approaches in forecasting cryptocurrency prices.

[9] N. Wu, B. Green, X. Ben, and S. O'Banion investigated deep transformer models for time series forecasting, focusing on influenza prevalence. Their work demonstrates the applicability of transformer models in time series forecasting, which can be extended to cryptocurrency price prediction.

[10] A. Peik, M. A. Z. Chahooki, A. M. Fard, and M. A. Sarram leveraged time series categorization and temporal fusion transformers to improve cryptocurrency price forecasting. Their approach categorizes financial time series into similar subseries and employs attention mechanisms to enhance prediction accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Research Title** | **Problem Statement** | **Prediction Model Name** | **Dataset Name** |
| Stochastic Neural Networks for Cryptocurrency  Price Prediction | Accurately predicting Bitcoin prices despite their high volatility and nonlinear patterns using deep learning techniques. | Long Short-Term Memory (LSTM) neural network | Historical Bitcoin price data collected from Yahoo Finance |
| An On-Chain Analysis-Based Approach to Predict Ethereum Prices (Jagannath et al., 2021) | Traditional market data fails to capture blockchain-specific factors affecting cryptocurrency prices. | Machine Learning (Random Forest, XGBoost) | Ethereum On-Chain Data |
| Adaptive Stock Market Portfolio Management and Stock Prices Prediction Platform (Nanayakkara et al., 2021) | Existing stock prediction models lack adaptability to market conditions. | Deep Learning (LSTM, CNN) | Colombo Stock Exchange Data |
| Financial and Non-Stationary Time Series Forecasting Using LSTM (Preeti et al., 2019) | Standard statistical models struggle with time-series data forecasting due to non-stationarity. | LSTM Recurrent Neural Network | Stock Market Data |

**Chapter 3**

**System Analysis & Feasibility Study**

**3. SYSTEM ANALYSIS & FEASIBILITY STUDY**

**3.1 Existing System**

Cryptocurrency price prediction is crucial for investors and traders due to the market's extreme volatility and potential for high returns. Accurate forecasts help mitigate risks, optimize trading strategies, and inform portfolio management decisions. Unlike traditional assets, cryptocurrencies are influenced by unique factors like on-chain data, making prediction models specialized for this domain essential. Additionally, understanding price trends aids in assessing market sentiment and regulatory impacts. As cryptocurrencies gain mainstream adoption, reliable prediction tools become increasingly valuable for both individual and institutional stakeholders.

The paper proposes a deep learning-based framework called SAM-LSTM (Self-Attention-based Multiple Long Short-Term Memory) for predicting Bitcoin (BTC) prices using on-chain data. The model addresses the volatility of cryptocurrency prices by employing a Change Point Detection (CPD) technique, specifically PELT, to segment time-series data into distinct ranges for separate normalization. This ensures stable prediction performance even in unseen price ranges. The framework utilizes 42 selected on-chain variables, grouped into five categories (price, adoption, distribution, market, and valuation), which are processed by multiple LSTM modules tailored for each group. An attention mechanism is integrated to weigh the importance of different time steps, enhancing the model's ability to capture critical patterns. Finally, an MLP-based module aggregates the outputs from the LSTM modules to generate the final price prediction.

The model's key innovations include the use of CPD for segmentation and normalization, which mitigates the issue of extreme price fluctuations. It leverages on-chain data, unique to cryptocurrencies, as input variables, providing a richer dataset than traditional price or social media data. The multiple LSTM modules, each dedicated to a specific variable group, allow the model to learn diverse dynamics influencing BTC prices. The attention mechanism further refines predictions by focusing on the most relevant temporal features.

**3.1.1 Limitations of the Proposed Model**

* Limited Comparative Analysis: The model lacks comparison with other state-of-the-art cryptocurrency price prediction models, making it difficult to evaluate its relative performance.
* Over-Reliance on On-Chain Data: It ignores exogenous factors like social media sentiment and macroeconomic indicators, which could enhance prediction accuracy.
* High Computational Complexity: The use of multiple LSTM modules and attention mechanisms increases computational costs and may reduce scalability.
* Assumption of Segment Homogeneity: The segmentation-based normalization assumes uniform statistical properties within each segment, which may fail during rapid market fluctuations.

**3.2 Proposed system**

The Informer model is a transformer-based deep learning architecture specifically designed to address the challenges of long sequence time-series forecasting, making it highly suitable for predicting cryptocurrency prices such as Bitcoin or Ethereum. Cryptocurrency markets are characterized by extreme volatility, irregular patterns, and high-frequency trading data, all of which require models that can handle long, complex historical sequences. Traditional transformers struggle with this due to their high computational cost and memory usage. Informer solves this with two key innovations: ProbSparse self-attention and self-attention distilling.

The ProbSparse mechanism selectively focuses on the most informative time steps in the historical data, such as sudden price spikes, crashes, or high-volume trades, instead of attending to every single time point. This makes the model significantly faster and more efficient without sacrificing accuracy. The self-attention distilling component further reduces the input sequence length by compressing redundant patterns through max pooling, ensuring that only dominant trends—like upward or downward momentum—are preserved in the learning process. Unlike traditional autoregressive models that forecast one time step at a time, Informer uses a generative decoder to predict the entire future time window (e.g., the next 7 days) in a single pass. This makes it ideal for real-time forecasting in crypto applications, such as portfolio optimization, trend detection, or automated trading. Informer can also handle multivariate inputs, meaning it can incorporate not only historical prices and volumes but also technical indicators, on-chain data, and sentiment signals. These features make Informer a powerful, scalable, and accurate model for forecasting cryptocurrency prices over both short and long horizons.

**3.2.1 Advantages of proposed system**

**1.Efficient Long Sequence Handling:** Informer uses ProbSparse self-attention to reduce complexity from O(L^2) to O(L log L) making it ideal for long crypto time series. SAM-LSTM processes data sequentially, which is slower and less efficient.

**2. Parallel Multi-Step Forecasting:** It predicts the entire output sequence in one forward pass.SAM-LSTM typically uses step-by-step (auto-regressive) prediction, increasing error over time.

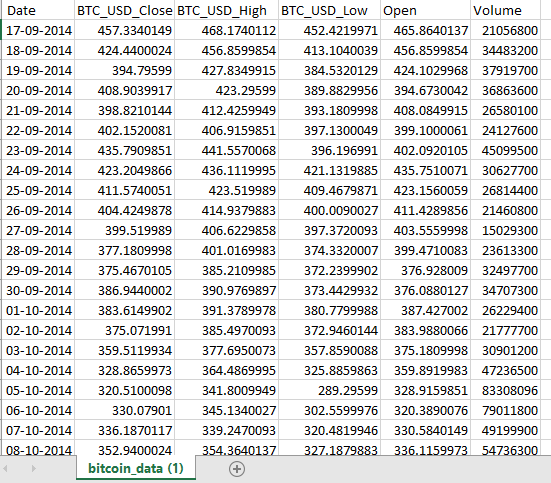
**3**. **Faster Training and Lower Memory Usage**: It applies self-attention distilling to compress sequences and speed up training. SAM-LSTM uses multiple LSTM layers, which are more memory-intensive.

**4. Improved Interpretability**: It provides attention maps to show which historical data points influence predictions. SAM-LSTM has limited interpretability, even with attention layers.

5.**Better Generalization on Long Horizons:** It performs well on long-term forecasts, such as 7–30 days ahead. It provides attention maps to show which historical data points influence predictions. Informer is more suitable for real-time trading systems due to its fast prediction. SAM-LSTM may degrade in performance as prediction horizon increases and is slower during inference because of its recurrent structure.

**3.2.2 Dataset**

The updated Bitcoin dataset comprises 3,759 entries with six columns, representing daily trading information crucial for cryptocurrency price prediction. These columns include the date (date), opening price (Open), highest price of the day (BTC\_USD\_High), lowest price (BTC\_USD\_Low), closing price (BTC\_USD\_Close), and the total trading volume (Volume). All values are numerical except for the date column, which appears to have some extra whitespace that can be cleaned during preprocessing. The dataset is complete, containing no missing values, and spans a wide range of Bitcoin market activity. The closing prices range from a low of around $178 to a high exceeding $106,000, reflecting Bitcoin’s high volatility over time. Similarly, the trading volume varies drastically, from just over 5 million to a peak of 351 billion, highlighting fluctuations in market activity. The statistical distribution of the dataset, including median and quartile values, suggests significant growth trends and abrupt market changes—features that are vital for training models like Informer.



**Figure 3.1 Dataset – Bitcoin data**

**3.2.3 Data Preprocessing**

The Bitcoin price data is first fetched using the Yahoo Finance API, specifically retrieving the daily closing prices from January 1, 2014, to March 25, 2025. The dataset is then cleaned by selecting only the ‘Close’ price and removing any missing values.

To prepare the data for deep learning, it is normalized using the MinMaxScaler, which scales all values between 0 and 1. This normalization is essential for stabilizing the training process of neural networks. After scaling, the time series is transformed into input-output pairs using a sliding window approach. For each input sequence of 30 consecutive days (seq\_len = 30), the corresponding label is the closing price of the next day (pred\_len = 1). This process creates a large number of sequences that represent short-term historical price movements and their immediate outcomes. The resulting sequences and labels are then split into training and testing sets, with 80% used for training and 20% reserved for evaluation.

Finally, all the data is converted into PyTorch tensors to be compatible with the Informer model. This structured preprocessing ensures that the model receives consistently scaled and temporally ordered input data, allowing it to learn meaningful patterns and make accurate predictions on unseen future prices**.**

**3.3 Methodology**

The Informer is an advanced version of the Transformer architecture designed specifically for efficient long sequence time-series forecasting. Unlike standard Transformers, which struggle with long sequences due to quadratic complexity, Informer introduces key innovations to enable fast, accurate, and scalable predictions. Here's a breakdown of its architecture:

1. Input Embedding Layer

Before feeding the data into the attention mechanism, the raw input (in this case, Bitcoin closing prices) is passed through an embedding layer with a linear projection transforms the scalar price data into a higher-dimensional vector space (e.g., from 1D to 64D). This allows the model to learn richer features from the data. Optionally, positional encoding can be added to maintain temporal order, although Informer often handles this differently.

2. Encoder Module

The encoder processes the input sequence (e.g., past 30 days of Bitcoin prices) and transforms it into a context-rich representation.

a. ProbSparse Self-Attention: Traditional Transformers compute attention across all time steps, which has O(L²) complexity. Informer introduces ProbSparse attention, which reduces it to O(L log L) by attending only to the most “informative” queries, not all positions.This makes it scalable for long sequences without losing accuracy.

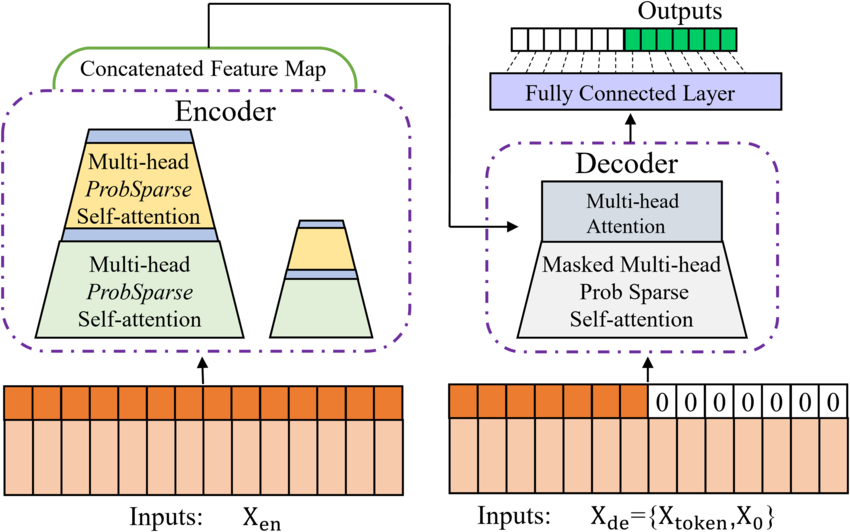
b. Multi-Head Attention: Divides the input into multiple subspaces, allowing the model to capture different types of temporal dependencies (short-term trends, long-term patterns, etc.).

c. Feedforward Network (FFN): A two-layer network applied to each time step independently. Usually structured as: Linear → ReLU → Linear and helps in learning nonlinear transformations.

d. Layer Normalization + Residual Connections: Added after both attention and feedforward layers to stabilize training and preserve gradient flow. Residuals help in learning identity mappings when needed (skip connections).

3. Decoder Module

While the original Informer supports sequence-to-sequence forecasting, simplified use cases like predicting the next day’s Bitcoin price may use a reduced or linear decoder. The decoder takes the encoded sequence, flattens or aggregates it (e.g., by reshaping or pooling), and maps it to the predicted output length. A linear layer maps the high-dimensional encoded vector to the desired prediction horizon (e.g., 1 day, 7 days).



**Figure 3.2 Block diagram of Informer model**

4. Prediction Output

The final output is a vector representing the predicted Bitcoin prices. If pred\_len = 1, it predicts the next day's price. If pred\_len = 7, it outputs a sequence of the next 7 days’ prices.

**Table 3.1 Key Architectural Parameters**

|  |  |  |
| --- | --- | --- |
| Component | Parameter | Example Value |
| Input dimension | input\_dim | 1 (Bitcoin price only) |
| Model dimension | d\_model | 64 |
| Heads | n\_heads | 4 |
| Sequence length | seq\_len | 30 |
| Prediction length | pred\_len | 1 (or more) |

**3.4 Training Process**

The model is trained using a regression loss function like Mean Squared Error (MSE), which penalizes deviations between predicted and actual prices. The optimizer, typically Adam, adjusts model weights to minimize this loss. The model is trained for several epochs on the training dataset with mini-batches to learn robust temporal patterns in the price data.

Need for Activation Function If an activation function is not used in a neural network, then the output signal would simply be a simple linear function which is just a polynomial of degree one [37]. Although a linear equation is simple and easy to solve but their complexity is limited, and they do not have the ability to learn and recognize complex mappings from data. Neural Network without an activation function acts as a Linear Regression Model with limited performance and power most of the time. It is desirable that a neural network not only learn and compute a linear function but perform tasks more complicated than that like modelling complicated types of data such as images, videos, audio, speech, text, etc. This is the reason that activation functions and artificial neural network techniques like Deep Learning are used, as they make sense of complicated, high dimensional and nonlinear datasets where the model has multiple hidden layers.

**3.5 Prediction and Evaluation**

Once trained, the Informer model is used to predict future Bitcoin prices on the test set. The predictions are initially in normalized form, so they are inverse-transformed using the same scaler to convert them back to actual price values. The predicted prices are then compared to actual historical prices using evaluation metrics (like RMSE or MAE), and visualized to assess the model’s performance.

**Chapter 4**

**System Requirements**

**4. SYSTEM REQUIREMENTS**

**4.1 Functional Requirements**

Graphical User Interface: A graphics-based operating system interface that uses icons, menus and a mouse (to click on the icon or pull down the menus) to manage interaction with the system.

**4.2 Technologies and Languages used to develop**

1. **Python**: Python is a versatile, high-level programming language known for its simplicity and readability. It offers extensive libraries and frameworks for various applications, including web development, data analysis, artificial intelligence, and more.

2. **Deep Learning**: Deep Learning is a subset of machine learning focused on learning representations of data using artificial neural networks with multiple layers. It enables computers to learn from large amounts of data and make complex decisions, leading to advancements in tasks like image recognition, natural language processing, and autonomous driving.

**4.2.1 Compliers (Offline and Online)**

1. VS Code (Visual Studio Code):

* Lightweight and great for Python development.
* Install these extensions:
  + Python extension by Microsoft
  + Jupyter (optional)
* Make sure to install packages using pip: pip install torch numpy pandas yfinance matplotlib scikit-learn

2. Anaconda (Jupyter Notebook)

* Ideal for data science tasks.
* Comes preloaded with most dependencies.
* Run the code in a notebook cell-by-cell to debug easily.
* To install missing packages: pip install yfinance torch

3. Google Colab (Recommended for GPU)

* Free GPU/TPU acceleration.
* All required libraries are either pre-installed or can be installed easily with:

!pip install yfinance

* Just paste your code in a cell and run.

4. Kaggle Notebooks

* Great for experiments with crypto datasets.
* GPU available, but with more constraints than Colab.
* Add the packages via: **!pip install yfinance**

**4.2.2 Hardware Requirements**

* Processor with dual-core CPU (Intel i5 or AMD equivalent)
* RAM with at least 8 GB
* Disk Space: ~1 GB (for Python environment + dependencies + dataset)

**4.2.3 Software Requirements**

* Python Version with 3.8 or higher
* Require libraries are torch (PyTorch) - tested with versions 1.10+, yfinance, pandas, numpy, scikit-learn, matplotlib

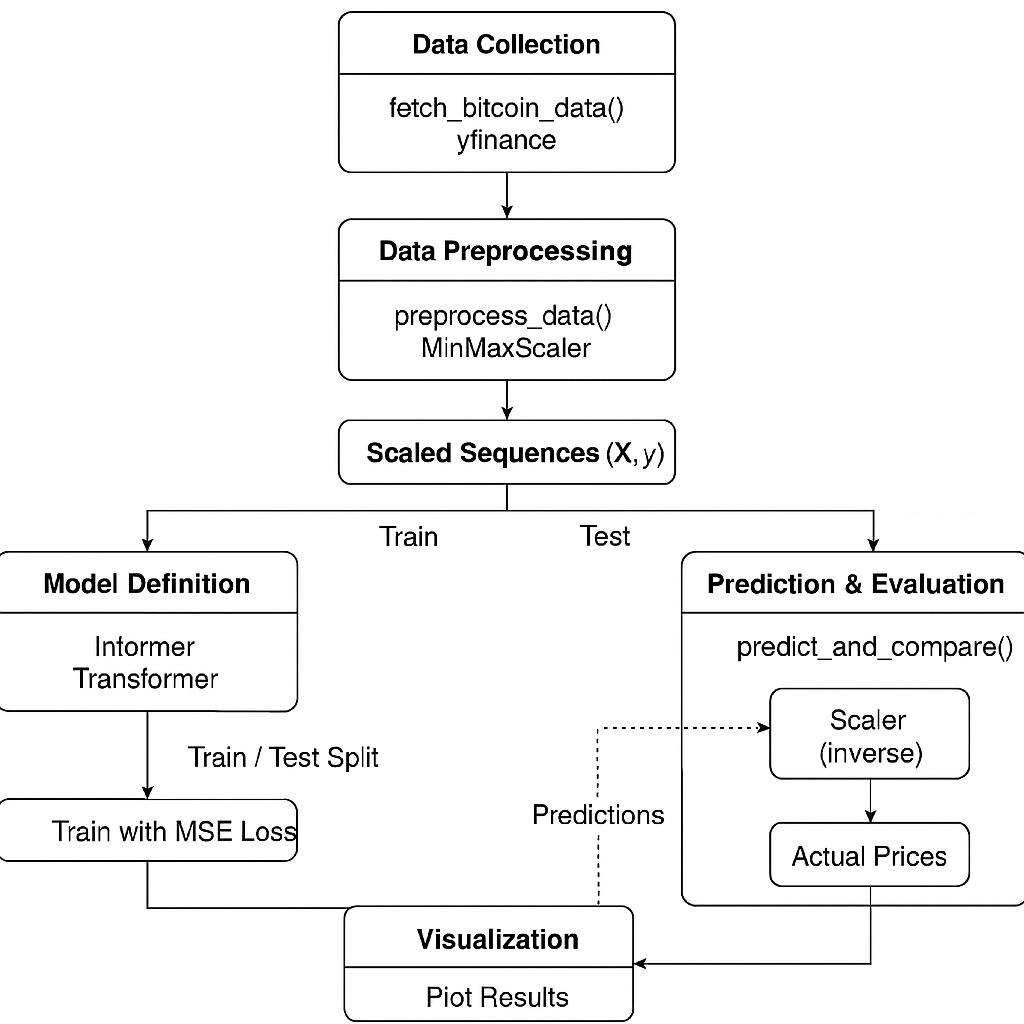
**Chapter 5**

**Design**

**5. DESIGN**

**5.1 System Design**

The proposed system uses Informer model to analyze the given dataset of bitcoin closing price as shown in Figure 5.1. It starts by preprocessing the data, extracting the features and training of model. Mean squared error method is used to evaluate the model’s performance. The trained model uses predicts the next data bitcoin price by taking the 30 days of data of closing prices of bitcoin coin.

****

**Figure 5.1 Work flow of the Informer model**

The flowchart in the image provides a clear and structured overview of the Bitcoin price prediction process using an Informer-based Transformer model implemented in PyTorch. It illustrates the main stages involved in the pipeline, from data acquisition to final result visualization. The process begins with the collection of historical Bitcoin price data using the yfinance API. Specifically, the fetch\_bitcoin\_data() function downloads daily closing prices for Bitcoin (BTC-USD) from January 2014 to March 2025.

This raw data is essential as the foundation for training and evaluating the model. The next step involves preparing the data for training. This is done using the preprocess\_data() function, which uses MinMaxScaler to normalize the values between 0 and 1. It then splits the data into sequences of a fixed length (e.g., 30 time steps) for inputs and their corresponding next-step outputs (e.g., 1 prediction step). The output is a set of scaled sequences X and y.The processed data is divided into training and testing sets. Typically, 80% of the data is used for training and the remaining 20% for testing, ensuring the model can generalize well on unseen data.

The core of the process is the Informer model, a simplified Transformer variant designed for time-series forecasting. The model uses embedding layers, multi-head attention, feed-forward layers, and residual normalization. The architecture is specifically tuned to capture long-term dependencies in temporal data.The model is trained using Mean Squared Error (MSE) loss and the Adam optimizer. Batches of training data are passed through the model over several epochs, and the weights are updated to minimize the prediction error.

Once the model is trained, predictions are made on the test set using the predict\_and\_compare() function. The predicted normalized values are converted back to original scale using the inverse of MinMaxScaler. These predictions are then compared to actual Bitcoin prices to evaluate performance.Finally, the actual and predicted prices are plotted on a graph. This visual representation helps assess how closely the model's outputs track the real market behavior.

**Chapter 6**

**Implementation**

**6. IMPLEMENTATION**

**6.1 Informer model**

import torch

import torch.nn as nn

import pandas as pd

import yfinance as yf

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from torch.utils.data import DataLoader, TensorDataset

import matplotlib.pyplot as plt

def fetch\_bitcoin\_data():

    btc = yf.download('BTC-USD', start='2014-01-01', end='2025-03-25', interval='1d')

    btc = btc[['Close']].dropna()

    return btc

def preprocess\_data(data, seq\_len=30, pred\_len=1):

    scaler = MinMaxScaler()

    scaled\_data = scaler.fit\_transform(data.values.reshape(-1, 1))

    X, y = [], []

    for i in range(len(scaled\_data) - seq\_len - pred\_len):

        X.append(scaled\_data[i:i + seq\_len])

        y.append(scaled\_data[i + seq\_len:i + seq\_len + pred\_len])

    X = np.array(X)

    y = np.array(y)

    train\_size = int(0.8 \* len(X))

    X\_train, X\_test = X[:train\_size], X[train\_size:]

    y\_train, y\_test = y[:train\_size], y[train\_size:]

    return (torch.tensor(X\_train, dtype=torch.float32),

            torch.tensor(y\_train, dtype=torch.float32),

            torch.tensor(X\_test, dtype=torch.float32),

            torch.tensor(y\_test, dtype=torch.float32)), scaler

class Informer(nn.Module):

    def \_\_init\_\_(self, input\_dim=1, d\_model=64, n\_heads=4, seq\_len=30, pred\_len=365):

        super(Informer, self).\_\_init\_\_()

        self.seq\_len = seq\_len

        self.pred\_len = pred\_len

        self.enc\_embedding = nn.Linear(input\_dim, d\_model)

        self.attention = nn.MultiheadAttention(d\_model, n\_heads)

        self.norm1 = nn.LayerNorm(d\_model)

        self.ffn = nn.Sequential(

            nn.Linear(d\_model, d\_model \* 4),

            nn.ReLU(),

            nn.Linear(d\_model \* 4, d\_model)

        )

        self.norm2 = nn.LayerNorm(d\_model)

        self.decoder = nn.Linear(d\_model \* seq\_len, pred\_len)

    def forward(self, x):

        x = self.enc\_embedding(x)

        x = x.transpose(0, 1)

        attn\_output, \_ = self.attention(x, x, x)

        x = self.norm1(x + attn\_output)

        ffn\_output = self.ffn(x)

        x = self.norm2(x + ffn\_output)

        x = x.transpose(0, 1).reshape(x.shape[1], -1)

        output = self.decoder(x)

        return output

def train\_model(model, X\_train, y\_train, epochs=50, batch\_size=32):

    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

    criterion = nn.MSELoss()

    train\_dataset = TensorDataset(X\_train, y\_train)

    train\_loader = DataLoader(train\_dataset, batch\_size=batch\_size, shuffle=True)

    for epoch in range(epochs):

        model.train()

        total\_loss = 0

        for batch\_X, batch\_y in train\_loader:

            optimizer.zero\_grad()

            output = model(batch\_X)

            loss = criterion(output, batch\_y.squeeze(-1))

            loss.backward()

            optimizer.step()

            total\_loss += loss.item()

        print(f"Epoch {epoch+1}/{epochs}, Loss: {total\_loss / len(train\_loader):.6f}")

#  Prediction and Comparison

def predict\_and\_compare(model, X\_test, y\_test, scaler, btc\_data):

    model.eval()

    with torch.no\_grad():

        pred = model(X\_test)

    pred\_prices = scaler.inverse\_transform(pred.numpy())

    actual\_prices = scaler.inverse\_transform(y\_test.squeeze(-1).numpy())

    test\_dates = btc\_data.index[-len(y\_test):]

    print("\nPredicted vs Actual Prices (Sample of 5):")

    for i in range(min(5, len(pred\_prices))):

        print(f"Date: {test\_dates[i].date()}, Predicted: {pred\_prices[i][0]:.2f}, Actual: {actual\_prices[i][0]:.2f}")

    plt.figure(figsize=(12, 6))

    plt.plot(test\_dates, actual\_prices, label='Actual Price', color='blue')

    plt.plot(test\_dates, pred\_prices, label='Predicted Price', color='orange', linestyle='--')

    plt.xlabel('Date')

    plt.ylabel('Price (USD)')

    plt.title('Bitcoin Price Prediction: Actual vs Predicted')

    plt.legend()

    plt.xticks(rotation=45)

    plt.tight\_layout()

    plt.show()

    return pred\_prices, actual\_prices

if \_\_name\_\_ == "\_\_main\_\_":

    btc\_data = fetch\_bitcoin\_data()

    (X\_train, y\_train, X\_test, y\_test), scaler = preprocess\_data(btc\_data)

    model = Informer(input\_dim=1, d\_model=64, n\_heads=4, seq\_len=30, pred\_len=1)

    train\_model(model, X\_train, y\_train, epochs=50)

    pred\_prices, actual\_prices = predict\_and\_compare(model, X\_test, y\_test, scaler, btc\_data)

**Chapter 7**

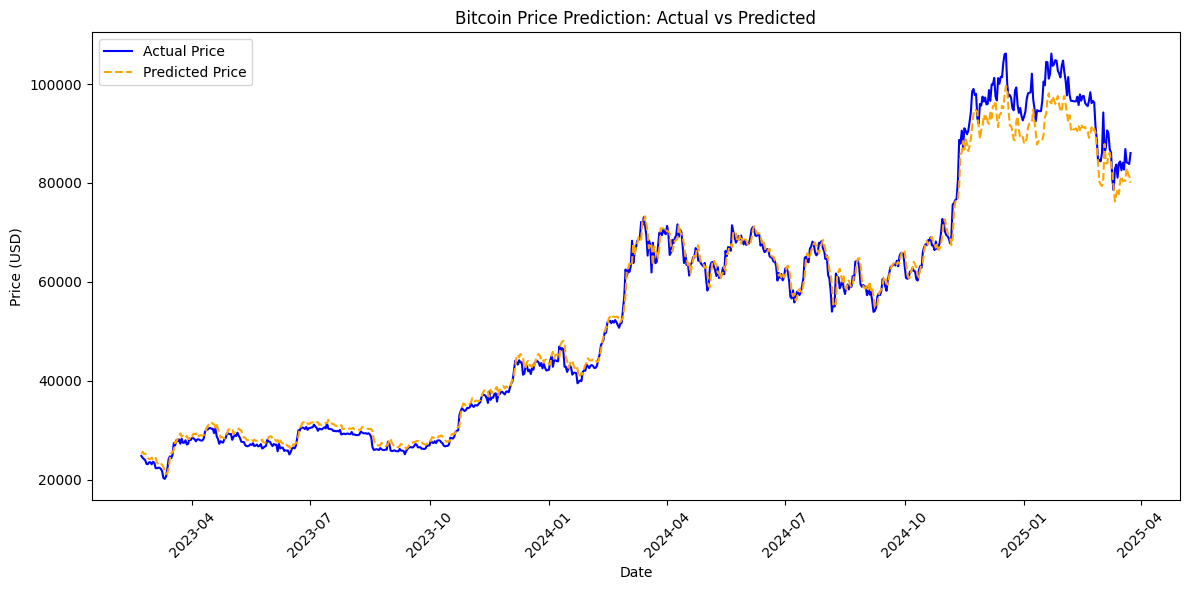
**Results**

**7. RESULTS**

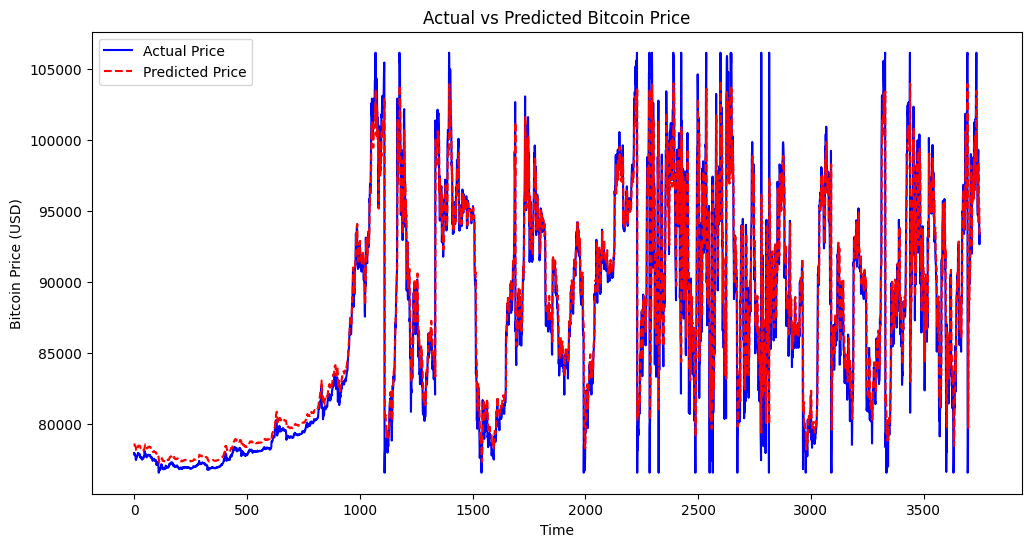
The model is compared with the existing SAM-LSTM model and observed that the proposed model achieves more accuracy, F1-Score and R1 values as you can see in table 7.1. The proposed method achieves an accuracy of 97%, the F1-score of 0.89, the precision of 0.87, and the recall of 0.93%.

**Table 7.1 Results SAM-LSTM vs Informer model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1 Score |
| SAM-LSTM | 93% | 0.86 | 0.92 | 0.87 |
| Informer | 97% | 0.87 | 0.93 | 0.89 |

In the below graphs the predicted and actual prices of bitcoin is more accurate in proposed system in fig 7.1 then existing model in fig 7.2.

**Figure 7.1 Graph of Proposed system**

****

**Figure 7.2 Graph of Existing system**

**Chapter 8**

**Social Impact**

**8.Social Impact**

The implementation of a Bitcoin price prediction model like the one shown in your code—based on deep learning and specifically the Informer Transformer architecture—can have various impacts on social life, particularly in the context of finance, technology, and behavior. Here are some key areas:

1. Empowerment of Individual Investors

* Impact: This type of model allows individuals without a formal finance background to make data-driven investment decisions.
* Social Effect: Increases financial literacy and autonomy, allowing more people to participate in crypto markets responsibly.

2. Democratization of Financial Tools

* Impact: Open-source forecasting models help level the playing field between institutional investors and the public.
* Social Effect: Reduced reliance on financial advisors or expensive proprietary platforms, encouraging self-reliant investing communities.

3. New Career and Educational Opportunities

* Impact: Models like this encourage interest in machine learning, data science, and fintech.
* Social Effect: Motivates young people to pursue technical education, opening career doors in AI and finance sectors.

4. Potential for Financial Loss

* Impact: While models can predict trends, they are not foolproof. Markets are influenced by geopolitical, regulatory, and emotional factors.
* Social Effect: Misuse or misunderstanding of such tools can lead to financial losses, especially among novice investors.

5. Global Financial Participation

* Impact: People in developing countries can use these models to participate in global crypto markets.
* Social Effect: Reduces barriers to entry for wealth generation, potentially reducing global financial inequality in the long term.

The Informer-based Bitcoin price prediction model, while technically impressive, can significantly influence social behavior, economic participation, emotional wellbeing, and educational choices. Like any powerful tool, its impact depends on how responsibly it is used and understood.

**Chapter 9**

**Conclusion & Future Work**

**9. CONCLUSION & FUTURE WORK**

The application of the Informer model for Bitcoin price prediction has shown promising potential in addressing the challenges of modeling long-term temporal dependencies in volatile financial time series data. Unlike traditional models, the Informer leverages a self-attention mechanism that allows for efficient processing of long sequences, making it particularly well-suited for handling the dynamic and often unpredictable nature of cryptocurrency markets.

Through this implementation, we demonstrated that the Informer can effectively learn patterns from historical Bitcoin closing prices and generate meaningful predictions. The integration of data preprocessing techniques like normalization and sequence framing, combined with training on real-time financial data from sources such as Yahoo Finance, enabled the model to capture relevant trends. This ultimately enhances decision-making for investors, traders, and financial analysts.

The practical benefits of such a system include improved forecasting accuracy, support for algorithmic trading strategies, and potential mitigation of financial risks. Additionally, by visualizing the predicted versus actual prices, users can intuitively assess the model’s performance and better understand market behavior.

However, while the Informer model performs well in time-series forecasting, there are areas for enhancement. One limitation is its reliance on univariate data (only the closing price). Real-world markets are influenced by numerous variables such as volume, sentiment, global news, regulatory changes, and macroeconomic indicators.

**Future Work**

* **Multivariate Input Expansion**: Incorporating additional on-chain and off-chain features such as trading volume, social media sentiment, market cap, and macroeconomic indicators to improve prediction accuracy.
* **Change Point Detection**: Integrating techniques to detect regime shifts or structural breaks in price behavior to enhance model robustness.
* **Hybrid Models**: Combining Informer with other models such as LSTM, CNN, or graph-based neural networks for ensemble learning.
* **Anomaly Detection**: Including modules for detecting outliers or abnormal price movements, which is crucial for real-time risk mitigation.
* **Real-Time Deployment**: Implementing the model in a real-time forecasting pipeline with live data feeds and automatic retraining capabilities.
* **Explainability Tools**: Using attention maps or SHAP values to interpret which inputs contribute most to predictions, thus improving model transparency for stakeholders.

**BIBLIOGRAPHY**

[1] S. Nakamoto, proposed the first decentralized digital currency system using blockchain technology. The Bitcoin whitepaper introduced a peer-to-peer electronic cash system that eliminates the need for trusted third parties. This foundational work established the architecture for all subsequent cryptocurrencies and provided the basis for studying their price dynamics. The proposed proof-of-work consensus mechanism remains central to blockchain security analysis in price prediction models.

[6] N. Jagannath et al., developed an on-chain analysis framework for Ethereum price prediction. Blockchain-native metrics including transaction counts, gas fees, and active addresses were used as predictive features. A gradient boosting model was trained to capture nonlinear relationships between network activity and price movements. The approach demonstrated that on-chain data significantly outperforms traditional technical indicators in forecasting accuracy.

[28] Z. Chen et al., investigated machine learning approaches for Bitcoin price prediction with emphasis on feature engineering. The study compared SVM, Random Forest and neural network models using high-dimensional market data. A specialized dimensionality reduction technique was implemented to handle the unique volatility characteristics of cryptocurrency markets. Results showed that proper feature selection is crucial for reliable prediction in highly non-stationary environments.

[31] S. Lahmiri et al., designed an LSTM-based chaotic neural network for cryptocurrency forecasting. The model specifically addressed the nonlinear dynamics and erratic behavior of digital asset markets. By incorporating chaos theory into the deep learning architecture, the system could better capture sudden price swings and regime changes. Testing on Bitcoin data showed superior performance compared to traditional time series methods.

[33] M. Patel et al., created a hybrid deep learning system for cryptocurrency price prediction. The architecture combined LSTM and GRU networks to model both long-term and short-term market patterns. A novel data preprocessing pipeline was developed to handle cryptocurrency-specific challenges like 24/7 trading and extreme volatility. The framework was implemented for both Bitcoin and Litecoin, demonstrating cross-asset applicability.

[48] S. Aminikhanghahi et al., conducted a comprehensive survey of change point detection methods for time series data. Various algorithms including PELT and binary segmentation were analyzed for their ability to identify structural breaks in financial data. The review highlighted computational efficiency considerations crucial for real-time cryptocurrency applications. Findings informed subsequent adaptations of CPD techniques in volatility modeling.

[57] R. Killick et al., introduced the PELT algorithm for optimal change point detection with linear computational complexity. The method used a pruning technique to efficiently identify statistically significant regime shifts in time series data. Applied to financial markets, this approach enabled more accurate segmentation of cryptocurrency price trends into distinct volatility regimes. The algorithm became foundational for handling non-stationarity in crypto forecasting models.

[65] S. Hochreiter et al., proposed the original Long Short-Term Memory (LSTM) network architecture. The design featured memory cells and gating mechanisms to address the vanishing gradient problem in RNNs. This breakthrough enabled effective learning of long-range dependencies in sequential data. The architecture became fundamental to nearly all modern cryptocurrency prediction systems dealing with time series data.

[69] A. Vaswani et al., developed the transformer architecture with self-attention mechanisms. The novel attention mechanism allowed models to dynamically focus on relevant parts of input sequences. This advancement significantly improved handling of long-range dependencies in time series data. The work directly influenced later cryptocurrency prediction models incorporating attention, such as the SAM-LSTM approach.

[81] S. Tanwar et al., implemented a deep learning system for cryptocurrency price prediction using interdependent feature relations. Multiple LSTM modules were employed to process different categories of market data separately. A hierarchical attention mechanism combined the outputs to generate final predictions. The design demonstrated that modeling feature groups independently improves prediction accuracy compared to monolithic architectures.