# Crypto Currency Price Prediction with Sentiment Analysis

Isuri Upeksha.

Information Technology Service Center, Faculty of Science

University of Colombo.

Supervisor

Dr. Prabath Liyanage.

In partial fulfillment of the requirements for the degree of

Bachelor of Science Honors in IT and Management.

March 31, 2025.

# Acknowledgements

This study explores how sentiment analysis and deep learning can work together to predict cryptocurrency prices, aiming to improve accuracy and provide valuable insights for investors.

I am deeply grateful to those who have supported me throughout this journey. First, I sincerely thank Dr. Prabath Liyanage, my supervisor, Senior Lecturer, Department of Mathematics, Faculty of Science, University of Colombo, for his invaluable guidance and encouragement. His support motivated me to complete my work successfully.

I am equally grateful to my friends and colleagues for their thoughtful discussions and feedback. Special thanks to my family for their unwavering love and support.

Lastly, I appreciate the Information Technology Unit 2 of the Faculty of Science for their assistance.

# Abstract

In this research, news sentiment and deep learning were used to improve cryptocurrency price prediction. Cryptocurrency prices change very quickly and are difficult to predict, so normal methods do not work well. One reason is that the effect of news and public opinion on prices is often not considered.

So, in this research Six popular cryptocurrencies were studied: Bitcoin (BTC), Ethereum (ETH), Solana (SOL), Ripple (XRP), Litecoin (LTC), and Cardano (ADA). News headlines and descriptions were collected. Natural language processing (NLP) was used to clean and understand the text. Sentiment scores were calculated using VADER and improved with a finance dictionary called the Loughran-McDonald dictionary.

These sentiment scores were combined with past price data. Then, two deep learning models LSTM and GRU were used to make predictions. Many settings were tested using a tool called Keras Tuner to find the best model.

The results showed that GRU gave more accurate predictions than LSTM. Also, news headlines gave slightly better results than full news descriptions of most cryptocurrencies.

# Contents

# List Of Figures

# List Of Tables

# Chapter 1

# 1. Introduction

## 1.1. Summary

This section shows an overview of the project, highlighting the significance of cryptocurrency price prediction, the role of sentiment analysis, and the integration of deep learning techniques. It also describes the main and specific objectives of the research.

## 1.2. Background of study

Cryptocurrency markets have become very popular due to the substantial price swings and the possibility of huge profits. Unlike physical money, cryptocurrencies are decentralized. Which means, it is not issued by the government or other financial institutions. So, they are highly volatile and difficult to predict. As digital currencies are being accepted globally, more attention is being given by traders, investors, and researchers to create accurate forecasting methods. These methods are used to understand how the market behaves and to lower financial risks.

In traditional finance, experts use different methods to predict prices, but cryptocurrencies are unpredictable, so we need better ways to make predictions. One useful method is sentiment analysis, which looks at public opinions and news to help forecast price movements.

Many studies have attempted to predict cryptocurrency prices using machine learning and statistical techniques, but most do not fully consider the influence of news sentiment. News is spread quickly. Negative news or new rules can cause prices to go down, while positive news can make prices go up. To improve price prediction, historical price data should be combined with the news sentiment from those days.

In this research, Bitcoin (BTC), Ethereum (ETH), Solana (SOL), Ripple (XRP), Litecoin (LTC), and Cardano (ADA) are studied to improve cryptocurrency price predictions using deep learning and sentiment analysis. Natural Language Processing (NLP) techniques, like VADER and the Loughran-McDonald Finance Sentiment Dictionary, are used to extract sentiment scores from news headlines and descriptions. These sentiment scores are combined with historical price data. Then, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models are trained. The goal of this study is to build a complete forecasting system that uses both past price trends and news sentiment.

## 1.3. Problem Statement

Cryptocurrency prices are changed quickly and in unpredictable ways, so their future values are hard to predict. Traditional financial models are often not successful because the effects of news and public opinion on the market are not considered. Financial news and social media are used by many traders and investors to make decisions, but this information is not used well in many existing price prediction models.

One of the main problems with current forecasting methods is that they do not properly integrate historical price trends with market sentiment (public opinion). Another challenge is that traditional models struggle to capture long-term patterns and the sentiment behind financial news, which can have a strong impact on price movements.

This research aims to improve cryptocurrency price predictions by combining historical price data with sentiment analysis. By combining price trends and sentiment analysis, this study seeks to develop a more accurate forecasting model.

## 1.4. Main Objective

- The main objective of this research is to develop an effective cryptocurrency price prediction model by integrating sentiment analysis with deep learning techniques.

## 1.5. Specific Objectives

- Extract sentiment from cryptocurrency related news using VADER and the Loughran-McDonald Finance Sentiment Dictionary.
- Develop and compare deep learning models to predict price movements.
- Select the best-performing model for cryptocurrency price prediction.

# Chapter 2

# 2. Literature Review

## 2.1. Summary

The literature review shows the existing studies on cryptocurrency price forecasting, sentiment analysis, and deep learning models. It discusses various forecasting methods, their limitations, and the potential of sentiment-based approaches in improving forecast accuracy.

## 2.2. Related Works

(Bhatt, 2023) focuses on predicting Bitcoin prices using machine learning and social media sentiment analysis. The study uses historical Bitcoin market data, blockchain transaction data, and Twitter sentiment data to improve price predictions. Twitter-Roberta and Vader were used for sentiment analysis. Several machine learning models were tested, including K-Nearest Neighbors (KNN), Logistic Regression, Gaussian Naive Base, Support Vector Machine (SVM), Extreme Sequence Boosting (XGBoost), and a multi-model fusion model. The **multi-model fusion model with Twitter-Roberta produced the best results with 90% accuracy** and was shown to improve social media sentiment predictions [1].

(Malhotra, Chandwani, Agarwala, & Mann, 2022) focuses on predicting the final price of Bitcoin using various machine learning (ML) and deep learning (DL) models and selecting the most effective approach to predict Bitcoin price using the mean absolute error (MAE). Bitcoin price data was collected over a four-month period from the CryptoCompare API. Several machine learning and deep learning models were tested, including regression-based models such as linear regression, ridge regression, and LASSO regression, deep learning models such as long short-term memory (LSTM) and gated recurrent unit (GRU), and a statistical approach using the Autoregressive Integrated Moving Average (ARIMA) model. The main findings of the study show that **ARIMA performed better than any other model, with a minimum mean absolute error (MAE) of 0.02294**. Although they performed better than the regression models, LSTM and GRU were slightly less successful than ARIMA. The least successful model for predicting the price of Bitcoin was LASSO regression [2].

(JM Low, 2023) focuses on predicting cryptocurrency prices (Bitcoin, Ethereum, Litecoin) using deep learning and sentiment analysis. Social media (Reddit) data has been used for sentiment analysis. This study proposes a **Deep Learning Cryptocurrency Forecasting considering Sentiment (DLCFS) system** that considers market features, trading volume, and sentiment data from Reddit to improve price prediction accuracy. Cryptocurrency price data for Bitcoin, Ethereum, and Litecoin was collected from the Binance API, while sentiment data was gathered from Reddit submissions using the Pushshift API. The data set covers the period from January 1, 2020, to November 20, 2022. VADER (Valence Aware Dictionary and Sentiment Reasoner), Flair (Pre-trained NLP model), and TextBlob (Lexicon-based analysis) have been used for sentiment analysis. Several machine learning and deep learning models were tested, including regression-based models such as Support Vector Regression (SVR), Bayesian Ridge Regression, Thiel-Sen Regression, and Long Short-Term Memory (LSTM), a deep learning model designed to capture long-term dependencies in time series data. The **DLCFS LSTM model showed the highest accuracy**, outperforming all other models, achieving 99.18% for Bitcoin, 96.82% for Litecoin, and 99.05% for Ethereum. In contrast, traditional regression models, including linear regression and Theil-Sen regression models, showed the lowest accuracy [3].

(Franco Valencia, 2019) focuses on predictability of daily price movements (direction only) in four major cryptocurrencies Bitcoin (BTC), Ethereum (ETH), Ripple (XRP), and Litecoin (LTC) using machine learning (ML) and sentiment analysis. A binary classification approach was employed, predicting whether prices would increase or decrease. They use 60 days of market data from CryptoCompare API and Over 20 million tweets, analyzed with VADER sentiment analysis. Several ML models are tested, including Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Random Forest (RF). This study found that **Multi-Layer**

**Perceptron (MLP) were the best at predicting daily price changes for most cryptocurrencies.** For Bitcoin, the best results came from using both Twitter and market data. Litecoin was the easiest to predict, with the SVM model giving the highest accuracy. Ethereum was the hardest to predict, and no model did much better than random guessing. Twitter data alone helped a bit for Ripple and Litecoin but wasn't as useful as market data. Using both Twitter and market data sometimes improved results, but in some cases, it actually made predictions worse. [4].

(Tarif) focuses on predict the price direction of Bitcoin (BTC) in using a combination of machine learning techniques and public sentiment analysis from social media (Twitter & Reddit), comparing ARIMA and LSTM models. So found that the **LSTM model predicted Bitcoin prices more accurately than the ARIMA model**. The ARIMA model had an RMSE of 209.263, while the LSTM model had lower errors 198.448 for single feature and 197.515 for multi feature, which was the best result. Although LSTM takes longer to run because it is more complex, it gives better predictions. This is because LSTM can remember patterns over time, making it more effective for predicting Bitcoin prices, which change quickly and often. [5].

(C Lamon, 2017) focuses on predict the next-day prices of three cryptocurrencies: Bitcoin, Ethereum, and Litecoin. Instead of guessing how people feel (positive or negative) from news or tweets, the team used actual price changes to train their model. **For Bitcoin (BTC), the best performing model was Logistic Regression**. It correctly predicted 44% of price increases and 62% of price drops. Even though the accuracy wasn't perfect, the model was better at catching days with larger price changes, which is important for trading. **For Ethereum (ETH), the best model was Bernoulli Naive Bayes**. It was very good at predicting price increases, getting 76% of them right, but it struggled with predicting drops, with only 16% accuracy. Despite that, the model was still able to detect some of the biggest daily changes in price. **For Litecoin (LTC), Logistic Regression gave the best results**, but overall performance was poor. The model mostly guessed that prices would drop, and it failed to predict a major price increase during the test period. This likely happened because the training data didn't reflect the sharp rise in Litecoin's price seen in the test set [6].

# Chapter 03

## 3. Methodology

This section focused on the theories and methodologies that were used in the developing the cryptocurrency price prediction model with sentiment analysis.

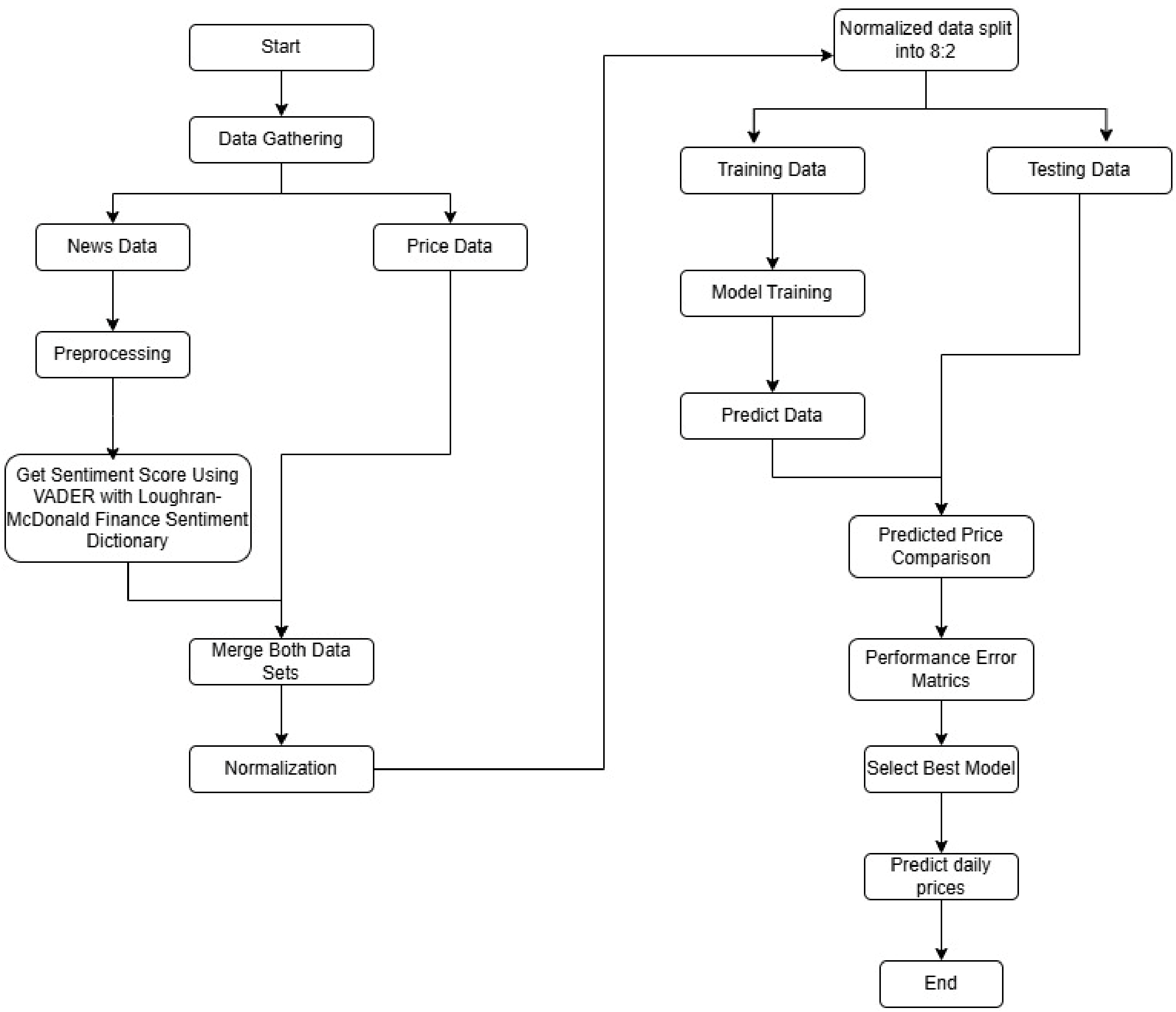



*Figure 1 : Model Architecture*

Figure 1 shows the architectural process stages of the cryptocurrency price prediction system model. The framework consists of several main stages, as outlined below:

## 3.1. Data Gathering

During the data collection phase, the data types required for the model were historical cryptocurrency price data for Bitcoin, Ethereum, Solana, XRP, Litecoin, and ADA, as well as cryptocurrency related news for sentiment analysis.

**News Data:** Cryptocurrency related news were scraped from the Crypto news website (Crypto News: Latest Cryptocurrency News and Analysis) and stored in a MySQL database.

**Price Data:** Historical cryptocurrency price data was retrieved from Yahoo Finance.

### 3.1.1. Crypto Selection Criteria

*Table 1: Selection Criteria for Cryptocurrency*

| | |
|---|---|
| **Market Relevance** | Top cryptocurrencies by market cap and trading volume to ensure a significant impact on the market. |
| **Technological Significance** | Covers multiple blockchain types (smart contracts, DeFi, payments, etc.). |
| **Data Availability** | Consistent historical price and news data for sentiment analysis and predictive modeling. |
| **Usage** | cryptocurrencies that are widely used and have a robust developer and community base. |

**Selected Cryptocurrencies:**

*Table 2: Reasons for selection, Selected Crypto*

| Cryptocurrency | Market Cap (as of 2024) | Reasons for select |
|---|---|---|
| Bitcoin (BTC) | $X Trillion | • Market leader and overall crypto health indicator.<br>• first and most recognized cryptocurrency |
| Ethereum (ETH) | $Y Billion | • Ethereum is the second-largest cryptocurrency and is widely used in decentralized applications and smart contracts.<br>• It plays a major role in the growing decentralized finance (DeFi) ecosystem. |
| Solana (SOL) | $Z Billion | • Rapid growth with low transaction costs.<br>• Strong Developer Community<br>• Attractive for decentralized applications and DeFi |
| Cardano (ADA) | $A Billion | • Strong research focus and unique proof-of-stake approach.<br>• Strong Developer Community<br>• Cardano prioritizes security in its blockchain design, using formal methods and rigorous testing to ensure its protocols are safe and robust, making it a reliable platform for smart contracts and decentralized applications (dApps). |
| Ripple (XRP) | $B Billion | • High liquidity and partnerships with major financial institutions. |

| | | |
|---|---|---|
| | | • focused on enabling fast and cost-effective cross-border payments. |
| Litecoin (LTC) | $C Billion | • Established altcoin benchmark.<br>• faster transaction times compared to Bitcoin. |

**Excluded Cryptocurrencies:**

*Table 3: Reasons for exclude, Excluded Crypto*

| Exclusion Factor | Examples | Justification |
|---|---|---|
| Low Market Relevance | Small-cap tokens, niche coins | • Insufficient trading data and negligible market impact. |
| Limited Data Availability | New cryptocurrencies (PEPE, WLD) | • Inconsistent historical price and sentiment data for modeling. |
| Lack of Adoption | Meme coins (e.g. Dogecoin) | • High volatility driven by speculation, not underlying technology or utility. |
| Stablecoins | USDT, USDC | • Predictive modeling is not necessary when prices are stable. |

### 3.1.2. Web Scrapping

News articles related to cryptocurrencies were collected from the CryptoNews website using a web scraping method built in Python. The tools Selenium, BeautifulSoup, and undetected chromedriver were used for this task.

The website pages were opened automatically using a browser controlled by Selenium. To avoid being blocked by the website, a special version of Chrome called undetected chromedriver was used. This made the browser look like it was being used by a real person.

Several pages from different news sections such as Bitcoin news, Ethereum news, Altcoin news, and Blockchain news were visited one by one. On each page, the browser was made to scroll down, so more news items would load. Then the content of the page was read and processed using BeautifulSoup.

For each news article, the date, headline, and short description were collected. These details were taken from the page using the class names of the HTML elements.

The collected articles were then checked for keywords like “Bitcoin”, “Ethereum”, “Ripple”, “Litecoin”, “Solana” or “Cardano”. If a keyword was found, the article was saved in the correct table in a MySQL database. For example, articles about Bitcoin were saved in the btc_news table.

To keep the news data up to date, a daily update script was created. This script runs the same scraping process, but only for the latest articles from the first 2 pages of each news section.

## 3.2. Data Preprocessing

The data preprocessing phase was essential to ensure that the collected news data was clean, structured, and suitable for sentiment analysis. This process involved multiple steps, including text cleaning, sentiment extraction, and sentiment score adjustment using a financial lexicon.

### 3.2.1. Text Preprocessing

To extract meaningful insights from news articles, the raw text from headlines and descriptions is passed through a preprocessing pipeline using Natural Language Processing (NLP) techniques. The preprocessing steps are included:

- Lowercasing: Converts all text to lowercase for uniformity.
- Removing URLs and special characters: Eliminates hyperlinks and non-alphabetic characters to remove irrelevant content.
- Whitespace normalization: Reduces extra spaces to maintain text consistency.
- Stop word removal: Eliminates frequently occurring words (e.g., “the,” “is,” “and”) that do not contribute to sentiment analysis.
- Lemmatization: Converts words to their base forms (e.g., “running” → “run”) using the spaCy NLP model, ensuring better text representation.

### 3.2.2. Sentiment Analysis

Once the text was preprocessed, sentiment analysis was performed using VADER (Valence Aware Dictionary and sEntiment Reasoner).

This provided an initial measure of how positive or negative each news headline and description was.

Since general-purpose sentiment analysis models like VADER did not fully capture financial and cryptocurrency-specific sentiment, an additional lexicon-based sentiment adjustment was applied. This adjustment was based on the Loughran-McDonald Finance Sentiment Dictionary, with modifications to include cryptocurrency-specific keywords.

- Positive words: Increased sentiment scores by +0.2 per occurrence.
- Negative words: Decreased sentiment scores by -0.2 per occurrence.
- The final sentiment score was adjusted and normalized within the range [-1,1].

The final score is recalculated as:

$$\textbf{Adjusted Sentiment} = \textbf{VADER Score} + \sum \textbf{(LM Adjustments)}$$

Positive Words:

- gain
- growth
- bullish
- opportunity
- profit
- Layer 2 scaling
- stability
- strong
- whale accumulation
- etf approval
- surge
- expansion
- adoption
- approval
- partnership
- Metaverse expansion
- all-time-high
- liquidity
- stake
- innovation
- mainstream
- rally
- institutional
- breakthrough
- Positive outlook
- halving
- scarcity
- integration

Negative Words:

- crash
- debt
- ban
- ponzi
- rug pull
- delisting
- loss
- bankruptcy
- fud
- liquidation
- dumping
- exchange insolvency
- bearish
- downturn
- lawsuit
- fraud
- sell-off
- inflation

- depeg
- decline
- regulatory crackdown
- crypto winter
- hacked
- scam
- negative outlook
- volatility
- recession

| | date | Final_Headline_Sentiment | Final_Description_Sentiment |
|---|---|---|---|
| **0** | 2020-07-27 | 0.0000 | 0.20000 |
| **1** | 2021-03-25 | 0.4019 | 0.72670 |
| **2** | 2021-06-09 | 0.2000 | 0.78180 |
| **3** | 2021-07-12 | 0.0000 | -0.19740 |
| **4** | 2021-08-16 | 0.0000 | 0.42150 |

*Figure 2: Example dataset after getting final sentiment scores*

## 3.3. Data Merging and Normalization

The sentiment scores were combined with historical price data using the date column. This ensured that each price record had a matching sentiment score for the same date. If a sentiment score was missing, use forward fill method. The combined dataset was normalized to standardize the feature values. This process ensured that all features had a similar scale, making the data suitable for deep learning models and improving model performance.

## 3.4. Model development

The normalized dataset was divided into training and testing sets in an 8:2 ratio. Both LSTM and GRU deep learning models were trained using historical price and sentiment data. Once trained, these models generated predicted price values for the testing dataset.

The predicted prices from both models were compared with the actual values. To evaluate their accuracy, performance metrics such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Accuracy score were used. The model that achieved the best performance was selected for the final predictions.

### 3.4.1. Hyperparameter Tuning

To ensure optimal model performance, a process called hyperparameter tuning was employed. In this process, certain settings of the model were adjusted to identify the best combination for achieving the highest performance. Keras Tuner was used to automate the exploration of different settings and to determine the most effective ones. The settings that were tuned included:

- **Number of units (neurons):** The GRU and LSTM layers were assigned units or neurons, which enable the model to learn from the data. Values between 32 and 128 units for the first layer and between 16 and 64 units for the second layer were tested.
- **Dropout rate:** Dropout is used as a technique to prevent the model from overfitting, which involves learning the noise in the data too much. Dropout rates between 0.1 and 0.5 were tested.
- **Optimizer:** The optimizer is used to help the model learn more efficiently. Two types Adam and RMSprop were tested.

The hyperparameter tuning was performed using Keras Tuner's Hyperband algorithm, which efficiently searched for the best hyperparameter configurations. Hyperband automatically focuses more on the best-performing model settings and stops testing the less successful ones early, which helps save time and resources. The tuner was set to improve the model by minimizing the validation loss (val_loss), and the best settings were chosen for the final model training.

### 3.4.2. Model Evaluation

After both models were trained and their settings were tuned, their performance in predicting prices was evaluated. The test dataset (the 20% that had been reserved earlier) was used to assess the predictions.

Three evaluation metrics were used to measure the accuracy of the models:

- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Accuracy Score

After comparing these metrics, the best model was selected as the one with the lowest errors.

## 3.5. Theories

### 3.5.1. Vader

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool. It uses a combination of pre-labelled lexical features (key words which are labeled as having a positive or negative sentiment) to classify new words into having either positive or negative sentiment. The compound VADAR score is a normalized, weighted composite score that is calculated based on the sum of all the lexicon ratings which have been standardized to range between -1 (most extreme negative) and +1 (most extreme positive).

- positive sentiment: compound score >= 0.5
- neutral sentiment: (compound score > -0.5) and (compound score < 0.5)
- negative sentiment: compound score <= -0.5

### 3.5.2. Loughran-McDonald Finance Sentiment Dictionary

The Loughran-McDonald (LM) Finance Sentiment Dictionary is recognized as a specialized lexicon designed for sentiment analysis within financial contexts. Unlike general-purpose sentiment dictionaries (e.g., VADER, SentiWordNet, or AFINN), the LM dictionary is tailored to the language found in financial documents, earnings reports, and market analyses.

In this project, the LM Finance Sentiment Dictionary was applied to enhance the sentiment analysis of cryptocurrency news data. By refining the raw sentiment scores produced by VADER, financial-specific sentiment terms were more accurately interpreted, contributing to improved cryptocurrency market analysis and price forecasting. To implement the LM Dictionary, two key word sets based on positive and negative sentiment categories commonly used in financial markets were defined.

### 3.5.3. LSTM Model Architecture

LSTM (Long Short-Term Memory) is a specialized recurrent neural network (RNN) architecture designed to handle sequential data efficiently, particularly for tasks requiring long-term dependencies, such as time series forecasting, speech recognition, and natural language processing (NLP). LSTM architecture involves the memory cell which is controlled by three gates as illustrate in Figure 3. LSTM Architecture the input gate, the forget gate and the output gate. These gates decide what information to add to, remove from and output from the memory cell.

- input gate: Controls what information is added to the memory cell.
- Forget gate: Determines what information is removed from the memory cell.

- Output gate: Controls what information is output from the memory cell.

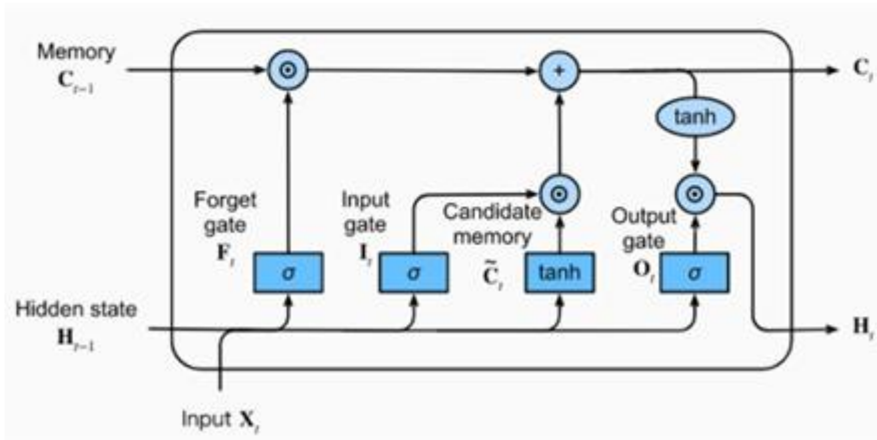



*Figure 3: LSTM Architecture*

### 3.5.4. GRU Model Architecture

GRU stands for Gated Recurrent Unit, which is a type of recurrent neural network (RNN) architecture that is similar to LSTM (Long Short-Term Memory).

Like LSTM, GRU is designed to model sequential data by allowing information to be selectively remembered or forgotten over time. However, GRU has a simpler architecture than LSTM, with fewer parameters, which can make it easier to train and more computationally efficient.

The main difference between GRU and LSTM is the way they handle the memory cell state. In LSTM, the memory cell state is maintained separately from the hidden state and is updated using three gates: the input gate, output gate, and forget gate. In GRU, the memory cell state is replaced with a "candidate activation vector," which is updated using two gates, the reset gate and update gate.

The reset gate determines how much of the previous hidden state to forget, while the update gate determines how much of the candidate activation vector to incorporate into the new hidden state.

Overall, GRU is a popular alternative to LSTM for modeling sequential data, especially in cases where computational resources are limited or where simpler architecture is desired.

The GRU architecture consists of the following components,

- Input Layer - Receives sequential data and feeds it into the GRU.
- Hidden Layer - Maintains a memory state, updating at each time step based on the current input and previous state.
- Reset Gate - Controls how much past information to forget, allowing the model to focus on recent data.
- Update Gate - Decides how much of the candidate activation should be incorporated into the new hidden state.
- Candidate Activation - A modified version of the previous state, adjusted using the reset gate and the current input.
- Output Layer - Processes the final hidden state to generate predictions.

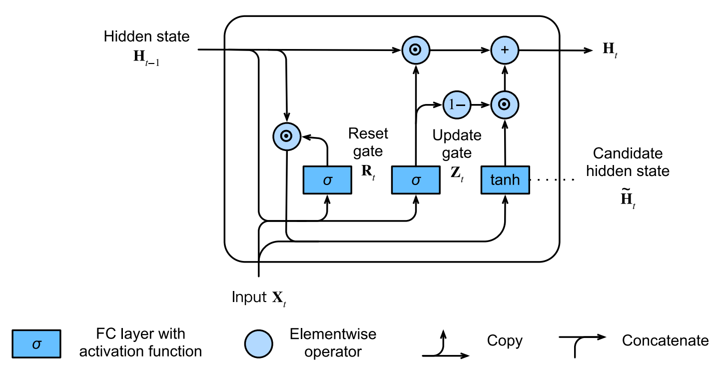



*Figure 4: GRU Architecture*

### 3.5.5. Mean Absolute Error (MAE)

Mean Absolute Error is defined as the average of absolute difference between forecasted values and true values.

$$\boldsymbol{MA}\mathbf{E} == \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{t=1}^{t=n} |\boldsymbol{y}' - \boldsymbol{y}|$$

Where $y'$ is forecasted value and $y$ is the actual value. n is the total number of values in the test set. MAE tells how big an error can expect from the forecast on average. The error values are in the original units of forecasted values and MAE = 0 indicates that there is no error in the forecasted values.

The lower the MAE value, the better the model is a value of zero means there is no error in the forecast. In other words, when comparing multiple models, the model with the lowest MAE is considered better.

### 3.5.6. Mean Absolute Percentage Error (MAPE)

MAPE is defined as the percentage of the average of absolute difference between forecasted values and true values, divided by true value.

$$\boldsymbol{MA}\mathbf{PE} == \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{t=1}^{t=n} \frac{|\boldsymbol{y}' - \boldsymbol{y}|}{\boldsymbol{y}} * \mathbf{100}\%$$

Where $y'$ is forecasted value and $y$ is the actual value. n is the total number of values in the test set. Because of the in denominator, it works best with data without zeros and extreme values. If this value is extremely small or large, the MAPE value also takes an extreme value.

### 3.5.7. Accuracy Score

Accuracy score obtains using the following formula for the evaluate the final model performance.

$$\boldsymbol{Weigted\ mean\ absolute\ deviation\ (WMAD)}\ =\ \frac{\sum_{t=1}^{t=n}|y'-y|}{\sum_{t=1}^{t=n}y}$$

**Accuracy = (1 - $\boldsymbol{WMAD}$ )**

Where $y'$ is forecasted value and $y$ is the actual value. n is the total number of values in the test set.

### 3.5.8. Keras Tuner's Hyperband algorithm

Keras Tuner is an automated hyperparameter tuning framework for deep learning models. It helps find the best model configurations (e.g., number of LSTM units, dropout rates, learning rates) to improve performance.

**Hyperband**

Hyperband is an advanced hyperparameter optimization algorithm that efficiently selects the best-performing models while reducing computation time.

**How Hyperband Works,**

1. Randomly selects multiple hyperparameter combinations (trials).
2. Trains models for a few epochs and evaluates performance.

3. Eliminates underperforming models and allocates more resources (epochs) to the better performing ones.
4. Repeats the process until the best model configuration is found.

# Chapter 04

# 4. Experimental Results

## 4.1. Summary

This section presents experimental results obtained from applying GRU and LSTM models for cryptocurrency price prediction using both news descriptions and headlines.

## 4.2. Hyperparameter Tuning

Part 3.4.1 shows a summary of the parameters chosen for each model. For all models, all possible combinations of the hyperparameters were investigated during the hyperparameter tuning process and the combinations presented in table 4 produced the best results.

*Table 4: Results of hyperparameter tuning*

| **Crypto** | **Sentiment Type** | **Model** | **Layer 1** | | **Layer 2** | | **Optimizer** |
|---|---|---|---|---|---|---|---|
| | | | **Units** | **Dropout Rate** | **Units** | **Dropout Rate** | |
| **ADA** | Description | GRU | 128 | 0.1 | 64 | 0.1 | Adam |
| | | LSTM | 96 | 0.3 | 64 | 0.1 | Adam |
| | Headline | GRU | 128 | 0.3 | 64 | 0.1 | Adam |
| | | LSTM | 128 | 0.1 | 64 | 0.1 | Adam |
| | Description | GRU | 32 | 0.2 | 32 | 0.2 | Adam |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **BTC** | | LSTM | 128 | 0.3 | 48 | 0.1 | Adam |
| | Headline | GRU | 64 | 0.5 | 64 | 0.3 | Adam |
| | | LSTM | 96 | 0.1 | 48 | 0.3 | Adam |
| **ETH** | Description | GRU | 96 | 0.1 | 64 | 0.1 | Adam |
| | | LSTM | 128 | 0.2 | 64 | 0.2 | Adam |
| | Headline | GRU | 128 | 0.3 | 48 | 0.2 | Adam |
| | | LSTM | 128 | 0.4 | 48 | 0.3 | Adam |
| **LTC** | Description | GRU | 128 | 0.2 | 64 | 0.1 | Adam |
| | | LSTM | 128 | 0.1 | 32 | 0.2 | Adam |
| | Headline | GRU | 128 | 0.4 | 48 | 0.1 | Adam |
| | | LSTM | 128 | 0.1 | 16 | 0.3 | Adam |
| **XRP** | Description | GRU | 128 | 0.2 | 32 | 0.3 | Adam |
| | | LSTM | 96 | 0.1 | 64 | 0.1 | Adam |
| | Headline | GRU | 128 | 0.4 | 16 | 0.4 | Adam |
| | | LSTM | 128 | 0.2 | 32 | 0.1 | Adam |
| **SOL** | Description | GRU | 64 | 0.2 | 64 | 0.2 | Adam |
| | | LSTM | 128 | 0.1 | 32 | 0.2 | Adam |
| | Headline | GRU | 64 | 0.3 | 48 | 0.3 | Adam |
| | | LSTM | 128 | 0.5 | 32 | 0.2 | Adam |

## 4.3. Error Analysis

In this section, we evaluate the performance of our price prediction models by analyzing the discrepancies between actual and predicted values.

### 4.3.1. Cardano (ADA)

#### 4.3.1.1. Actual vs Predicted Price Visualization

This section shows a visual comparison of the model's performance by plotting the actual vs. predicted prices over time for ADA.

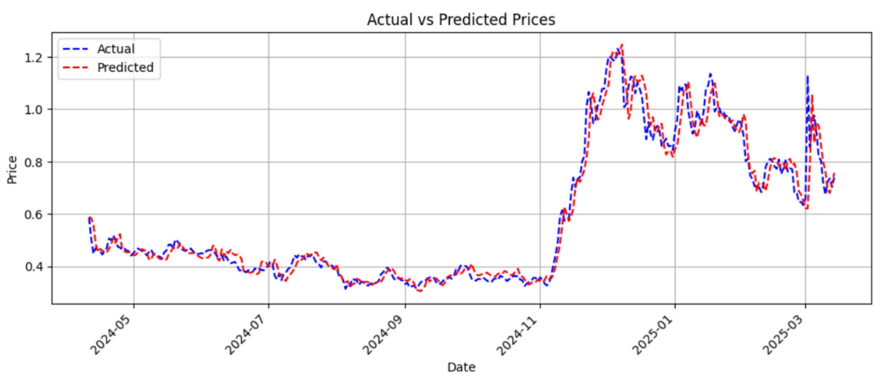



*Figure 5: ADA Price Chart of Actual vs Predicted Price (with news Description) - LSTM Model*

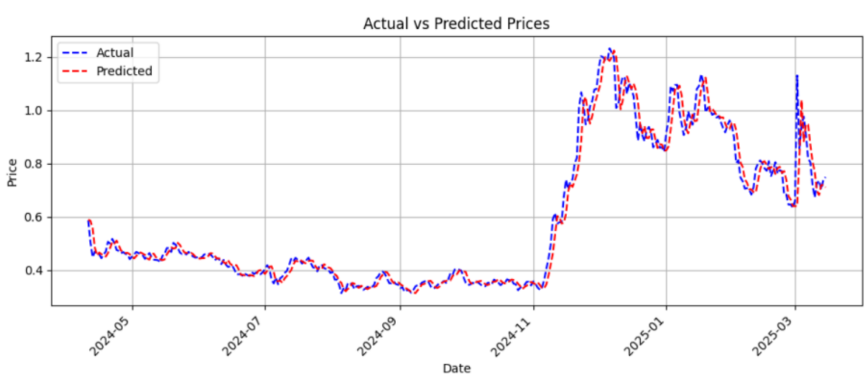



*Figure 6: ADA  Price Chart of Actual vs Predicted Price (with news Headline) - LSTM Model*

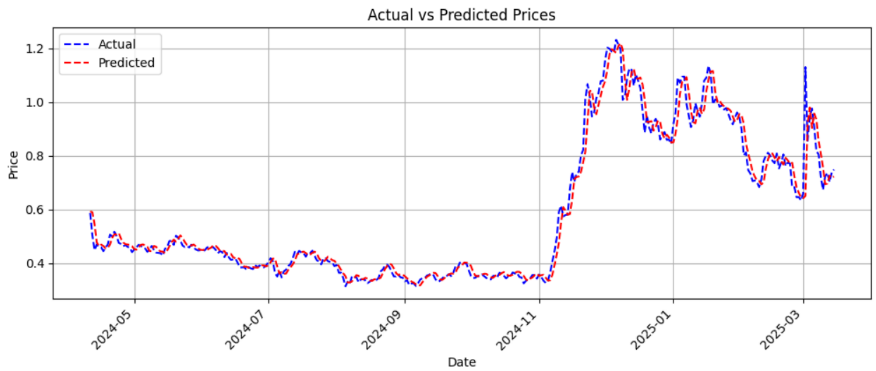



*Figure 7: ADA Price Chart of Actual vs Predicted Price (with news Headline) - GRU Model*


*Figure 8: ADA Price Chart of Actual vs Predicted Price (with news Description) - GRU Model*

### 4.3.1.2. Error Metrics

*Table 5: Results of ADA model performance matrices*

| | **With Description** | | **With Headline** | |
|---|---|---|---|---|
| | **GRU** | **LSTM** | **GRU** | **LSTM** |
| MAE | 0.0269 | 0.0326 | 0.0256 | 0.0281 |
| MAPE % | 4.111 | 5.159 | 3.936 | 4.275 |
| Accuracy | 0.954 | 0.944 | 0.956 | 0.952 |

Table 5 shows the accuracy, MAE and MAPE values for each model with news description data and with news headline data for ADA. From this table, we can see that the news headline models have better results than the news descriptions of both GRU and LSTM Models, because they have lower MAE and MAPE values and higher accuracy. And also, we can see GRU model is better than LSTM model because it also has lower MAE and MAPE values and has higher accuracy.

Accordingly, we can say that the best combination for price prediction for ADA is news headlines with GRU.

#### 4.3.2. Bitcoin (BTC)

##### 4.3.2.1. Actual vs Predicted Price Visualization

This section shows a visual comparison of the model's performance by plotting the actual vs. predicted prices over time for BTC.

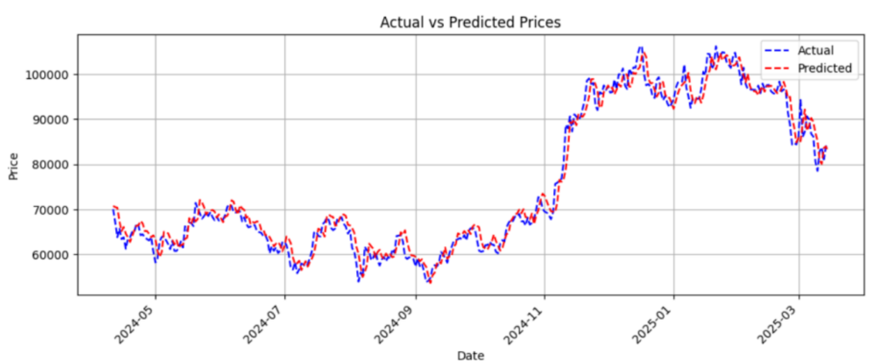



*Figure 9: BTC Price Chart of Actual vs Predicted Price (with news Description) - LSTM Model*

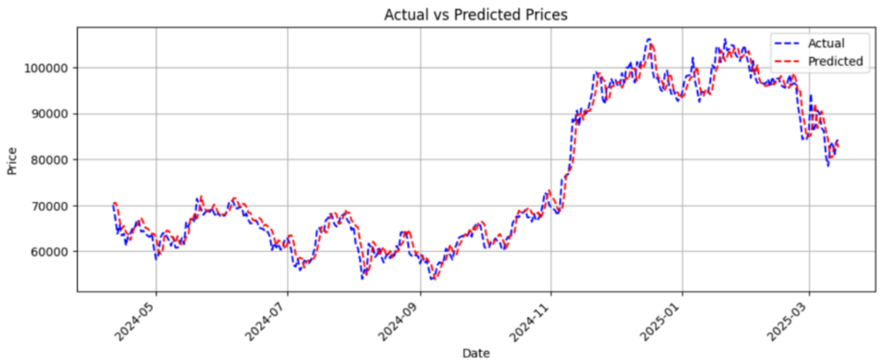



*Figure 10: BTC Price Chart of Actual vs Predicted Price (with news Headline) - LSTM Model*

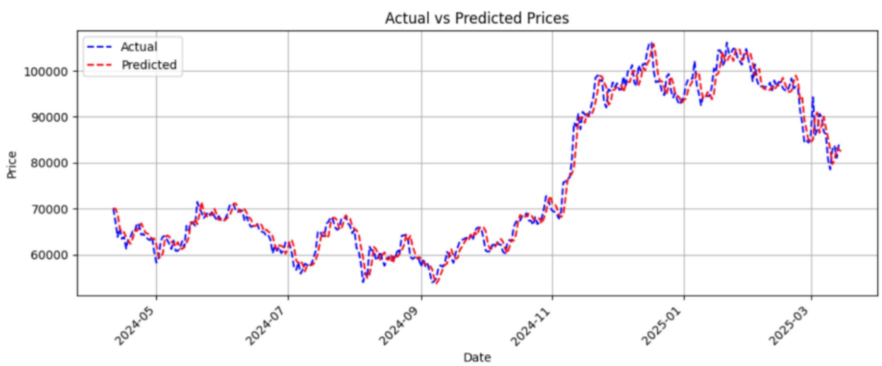



*Figure 11: BTC Price Chart of Actual vs Predicted Price (with news Headline) - GRU Model*

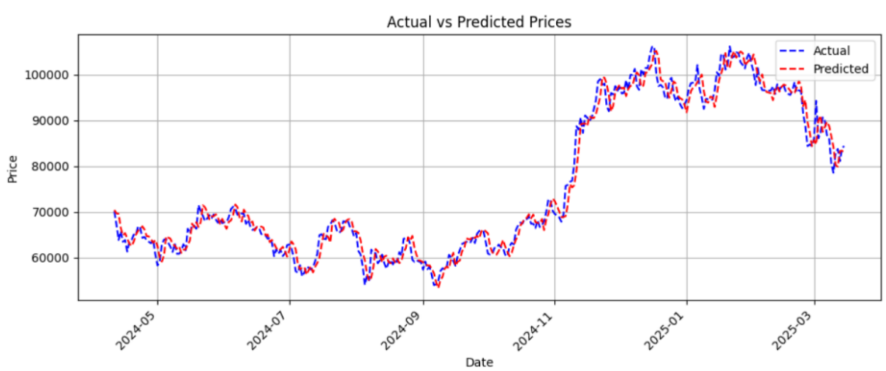



*Figure 12: BTC Price Chart of Actual vs Predicted Price (with news Description) - GRU Model*

#### 4.3.2.2. Error Metrics

*Table 6: Results of BTC model performance matrices*

| | With Description | | With Headline | |
|---|---|---|---|---|
| | **GRU** | **LSTM** | **GRU** | **LSTM** |
| MAE | 1792.94 | 1947.62 | 1754.60 | 1883.73 |
| MAPE % | 2.375 | 2.628 | 2.324 | 2.529 |
| Accuracy | 0.9763 | 0.9742 | 0.9768 | 0.9750 |

Table 6 shows the accuracy, MAE and MAPE values for each model with news description data and with news headline data for BTC. From this table, we can see that the news headline models have better results than the news descriptions of both GRU and LSTM Models, because they have lower MAE and MAPE values and higher accuracy. And also, we can see GRU model is better than LSTM model because it also has lower MAE and MAPE values and has higher accuracy.

Accordingly, we can say that the best combination for price prediction for BTC is news headlines with GRU.

### 4.3.3. Ethereum (ETH)

#### 4.3.3.1. Actual vs Predicted Price Visualization

This section shows a visual comparison of the model's performance by plotting the actual vs. predicted prices over time for ETH.

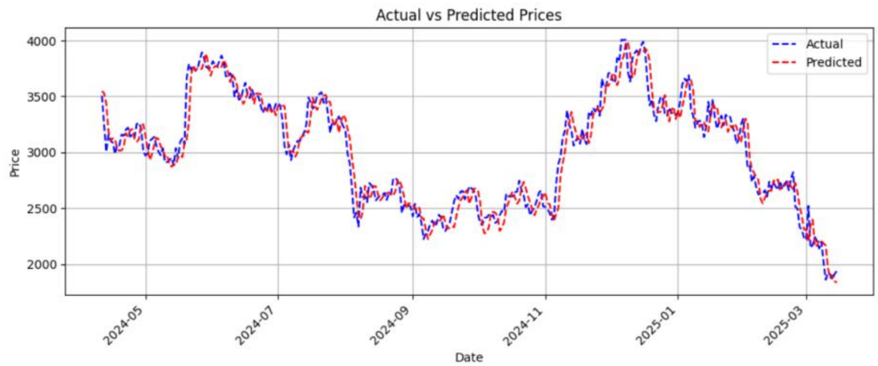



*Figure 13: ETH Price Chart of Actual vs Predicted Price (with news Description) - LSTM Model*

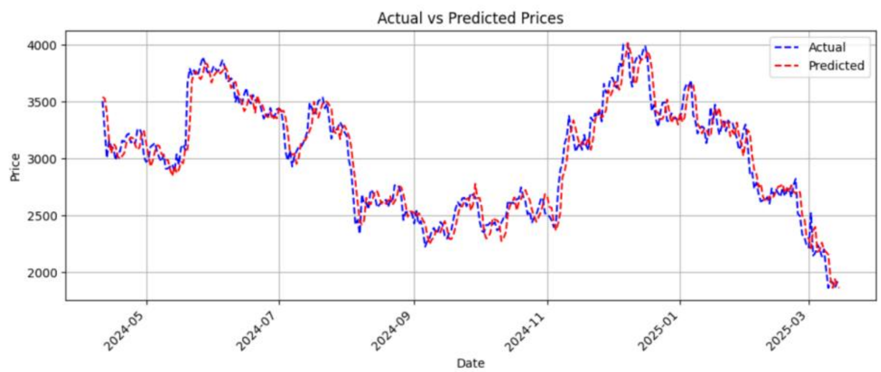



*Figure 14: ETH Price Chart of Actual vs Predicted Price (with news Headline) - LSTM Model*

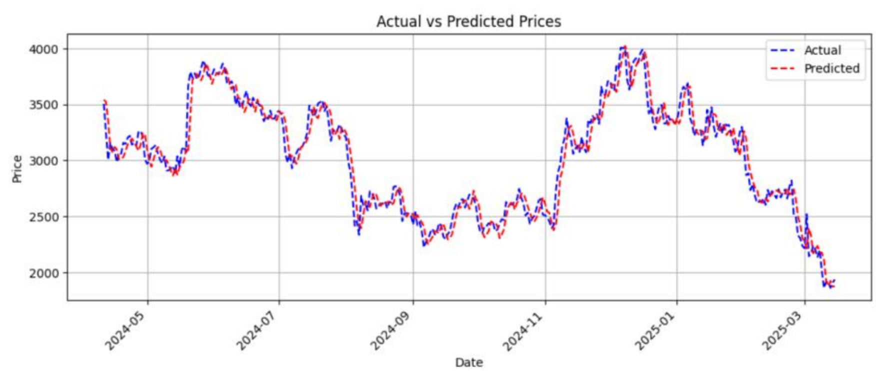



*Figure 15: ETH Price Chart of Actual vs Predicted Price (with news Headline) - GRU Model*

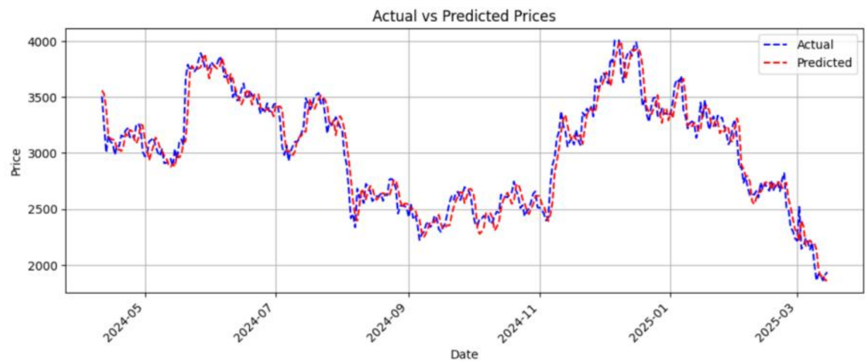



*Figure 16: ETH Price Chart of Actual vs Predicted Price (with news Description) - GRU Model*

### 4.3.3.2. Error Metrics

*Table 7: Results of ETH model performance matrices*

| | With Description | | With Headline | |
|---|---|---|---|---|
| | **GRU** | **LSTM** | **GRU** | **LSTM** |
| MAE | 89.56 | 97.84 | 88.04 | 98.49 |
| MAPE % | 3.064 | 3.324 | 2.981 | 3.348 |
| Accuracy | 0.9700 | 0.9675 | 0.9708 | 0.9673 |

Table 7 shows the accuracy, MAE and MAPE values for each model with news description data and with news headline data for ETH. From this table, we can see that the news headline models have better results than the news descriptions in GRU Model, because they have lower MAE and MAPE values and higher accuracy. But in LSTM model news description model has best results than news headlines model. And also, we can see GRU model is better than LSTM model because it also has lower MAE and MAPE values and has higher accuracy.

Accordingly, we can say that the best combination for price prediction for ETH is news headlines with GRU.

### 4.3.4. Litecoin (LTC)

#### 4.3.4.1. Actual vs Predicted Price Visualization

This section shows a visual comparison of the model's performance by plotting the actual vs. predicted prices over time for LTC.

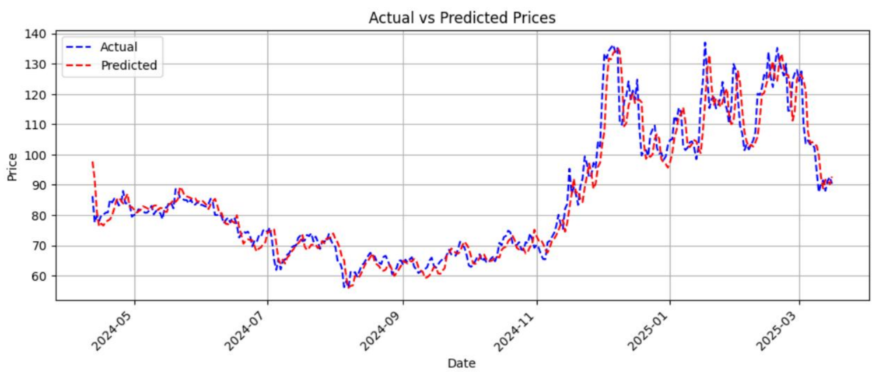



*Figure 17: LTC Price Chart of Actual vs Predicted Price (with news Description) - LSTM Model*


*Figure 18: LTC Price Chart of Actual vs Predicted Price (with news Headline) - LSTM Model*


*Figure 19: LTC Price Chart of Actual vs Predicted Price (with news Headline) - GRU Model*

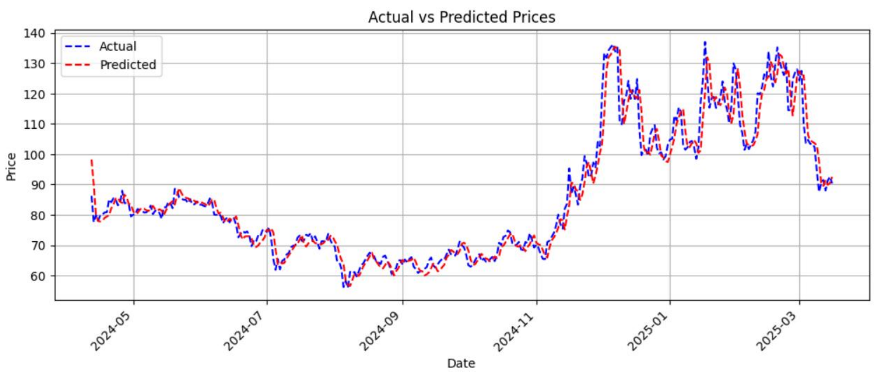



*Figure 20: LTC Price Chart of Actual vs Predicted Price (with news Description) - GRU Model*

#### 4.3.4.2. Error Metrics

*Table 8: Results of LTC model performance matrices*

| | **With Description** | | **With Headline** | |
|---|---|---|---|---|
| | **GRU** | **LSTM** | **GRU** | **LSTM** |
| MAE | 3.05 | 3.66 | 3.14 | 3.32 |
| MAPE % | 3.328 | 4.023 | 3.429 | 3.602 |
| Accuracy | 0.9645 | 0.9574 | 0.9636 | 0.9615 |

Table 8 shows the accuracy, MAE and MAPE values for each model with news description data and with news headline data for LTC. From this table, we can see that the news descriptions models have better results than the news headline in GRU Model, because they have lower MAE and MAPE values and higher accuracy. But in LSTM model news headlines model has best results than news description model. And also, we can see GRU model is better than LSTM model because it also has lower MAE and MAPE values and has higher accuracy.

Accordingly, we can say that the best combination for price prediction for LTC is news descriptions with GRU.

### 4.3.5. Ripple (XRP)

#### 4.3.5.1. Actual vs Predicted Price Visualization

This section shows a visual comparison of the model's performance by plotting the actual vs. predicted prices over time for XRP.

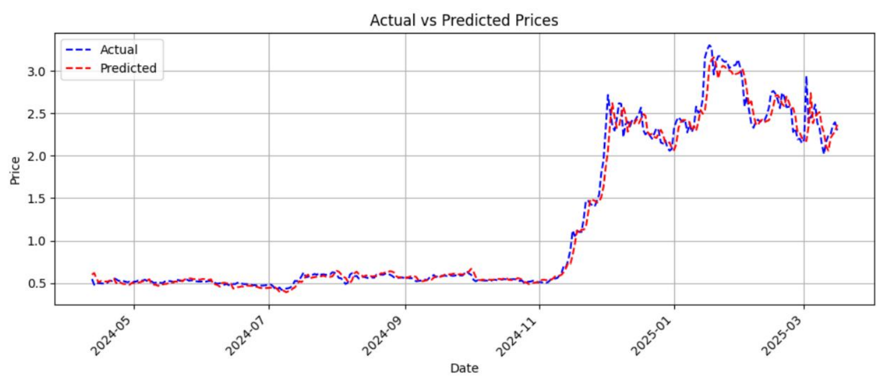



*Figure 21: XRP Price Chart of Actual vs Predicted Price (with news Description) - LSTM Model*

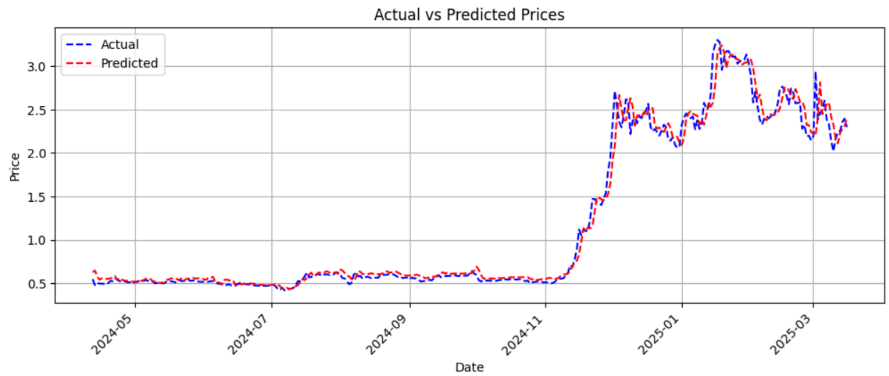



*Figure 22: XRP Price Chart of Actual vs Predicted Price (with news Headline) - LSTM Model*

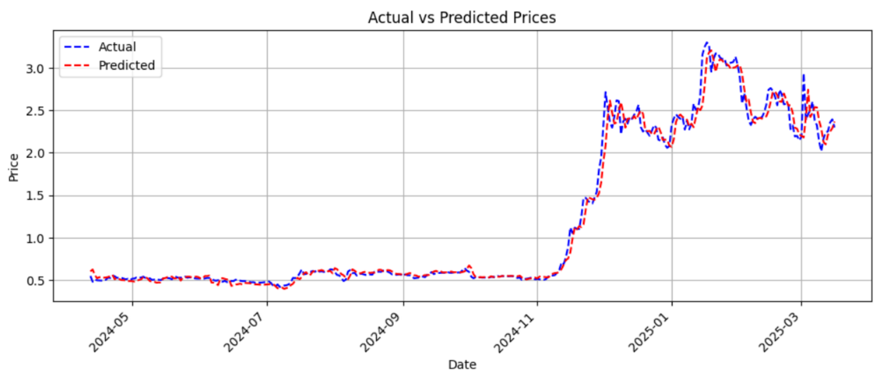



*Figure 23: XRP Price Chart of Actual vs Predicted Price (with news Headline) - GRU Model*


*Figure 24: XRP Price Chart of Actual vs Predicted Price (with news Description) - GRU Model*

#### 4.3.5.2. Error Metrics

*Table 9: Results of XRP model performance matrices*

| | With Description | | With Headline | |
|---|---|---|---|---|
| | **GRU** | **LSTM** | **GRU** | **LSTM** |
| MAE | 0.0619 | 0.0639 | 0.0601 | 0.0668 |
| MAPE % | 5.0037 | 5.0014 | 4.7489 | 5.6542 |
| Accuracy | 0.9482 | 0.9465 | 0.9497 | 0.9441 |

Table 9 shows the accuracy, MAE and MAPE values for each model with news description data and with news headline data for XRP. From this table, we can see that the news headline models have better results than the news descriptions in GRU Model, because they have lower MAE and MAPE values and higher accuracy. But in LSTM model news description model has best results than news headlines model. And also, we can see GRU model is better than LSTM model because it also has lower MAE and MAPE values and has higher accuracy.

Accordingly, we can say that the best combination for price prediction for XRP is news headlines with GRU.

#### 4.3.6. Solana (SOL)

##### 4.3.6.1. Actual vs Predicted Price Visualization

This section shows a visual comparison of the model's performance by plotting the actual vs. predicted prices over time for Solana.

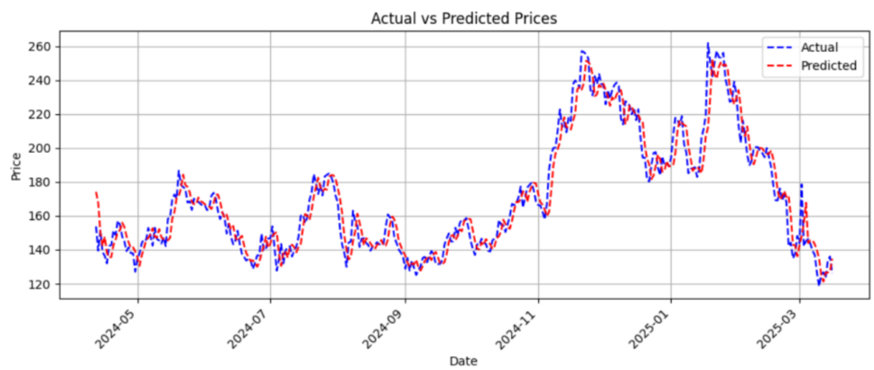



*Figure 25: SOL Price Chart of Actual vs Predicted Price (with news Description) - LSTM Model*

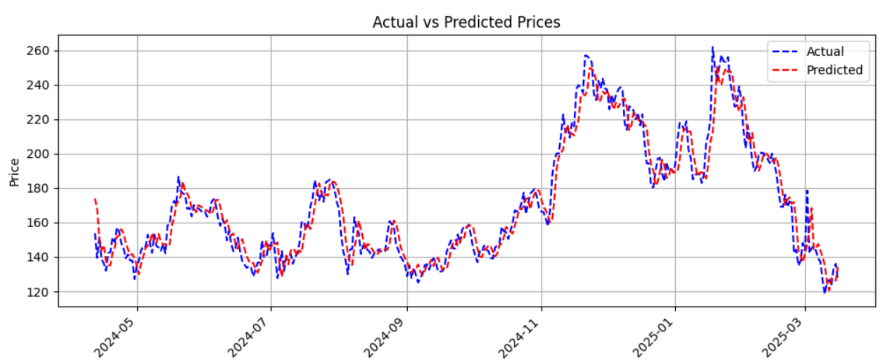



*Figure 26: SOL Price Chart of Actual vs Predicted Price (with news Headline) - LSTM Model*

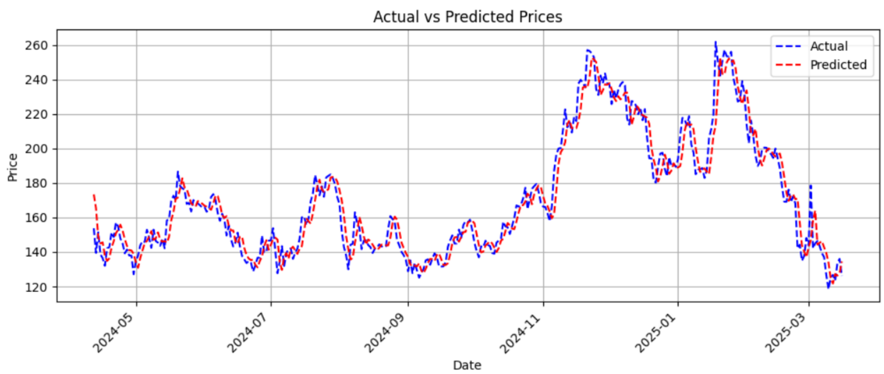



*Figure 27: SOL Price Chart of Actual vs Predicted Price (with news Headline) - GRU Model*

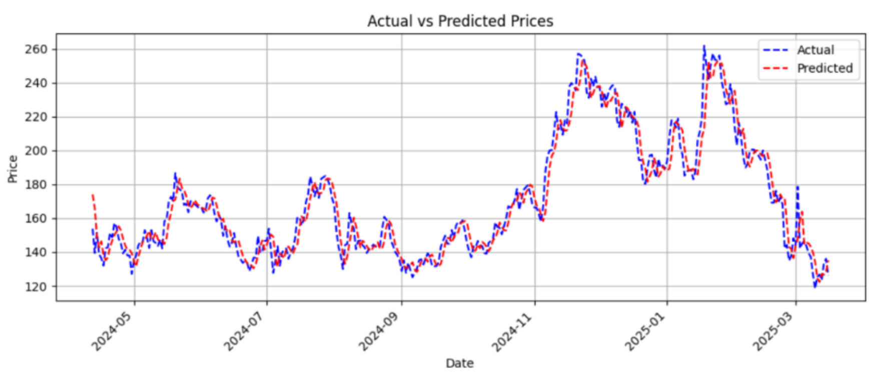



*Figure 28: SOL Price Chart of Actual vs Predicted Price (with news Description) - GRU Model*

#### 4.3.6.2. Error Metrics

*Table 10: Results of SOL model performance matrices*

| | **With Description** | | **With Headline** | |
|---|---|---|---|---|
| | **GRU** | **LSTM** | **GRU** | **LSTM** |
| MAE | 6.736 | 7.057 | 6.760 | 7.204 |
| MAPE % | 3.9639 | 4.1502 | 3.9835 | 4.2435 |
| Accuracy | 0.9604 | 0.9586 | 0.9603 | 0.9577 |

Table 10 shows the accuracy, MAE and MAPE values for each model with news description data and with news headline data for SOL. From this table, we can see that the news descriptions models have better results than the news headline in both LSTM and GRU

Models, because they have lower MAE and MAPE values and higher accuracy. And also, we can see GRU model is better than LSTM model because it also has lower MAE and MAPE values and has higher accuracy.

Accordingly, we can say that the best combination for price prediction for SOL is news descriptions with GRU.

*Table 11: Results of model performance matrices for all models*

| | | With Description | | With Headline | |
|---|---|---|---|---|---|
| | | **GRU** | **LSTM** | **GRU** | **LSTM** |
| **ADA** | MAE | 0.0269 | 0.0326 | 0.0256 | 0.0281 |
| | MAPE % | 4.111 | 5.159 | 3.936 | 4.275 |
| | Accuracy | 0.954 | 0.944 | 0.956 | 0.952 |
| **BTC** | MAE | 1792.94 | 1947.62 | 1754.60 | 1883.73 |
| | MAPE % | 2.375 | 2.628 | 2.324 | 2.529 |
| | Accuracy | 0.9763 | 0.9742 | 0.9768 | 0.9750 |
| **ETH** | MAE | 89.56 | 97.84 | 88.04 | 98.49 |
| | MAPE % | 3.064 | 3.324 | 2.981 | 3.348 |
| | Accuracy | 0.9700 | 0.9675 | 0.9708 | 0.9673 |
| **LTC** | MAE | 3.05 | 3.66 | 3.14 | 3.32 |
| | MAPE % | 3.328 | 4.023 | 3.429 | 3.602 |
| | Accuracy | 0.9645 | 0.9574 | 0.9636 | 0.9615 |
| **XRP** | MAE | 0.0619 | 0.0639 | 0.0601 | 0.0668 |
| | MAPE % | 5.0037 | 5.0014 | 4.7489 | 5.6542 |
| | Accuracy | 0.9482 | 0.9465 | 0.9497 | 0.9441 |
| **SOL** | MAE | 6.736 | 7.057 | 6.760 | 7.204 |
| | MAPE % | 3.9639 | 4.1502 | 3.9835 | 4.2435 |
| | Accuracy | 0.9604 | 0.9586 | 0.9603 | 0.9577 |

Table 11 shows Results of model performance matrices for all models. The values highlighted in bold are the best performance accuracy in terms of performance error metrics achieved as the comparison between the models.

# Chapter 05

## 5. Discussion

In this research, the combination of sentiment analysis with deep learning models was explored to improve cryptocurrency price predictions. Cryptocurrencies are known for their price volatility, and traditional financial models often struggle to predict their movements accurately. One major reason is that traditional models usually focus only on numerical price data and ignore how public opinion or news affects the market.

To address this, sentiment analysis was added to the prediction process. News headlines and descriptions related to six major cryptocurrencies Bitcoin (BTC), Ethereum (ETH), Cardano (ADA), Solana (SOL), Ripple (XRP), and Litecoin (LTC) were collected. These texts were cleaned and processed using Natural Language Processing (NLP) techniques. Then, VADER, a popular sentiment analysis tool, along with the Loughran-McDonald Finance Sentiment Dictionary, was used to adjust sentiment scores based on financial keywords. This approach allowed the emotional tone (positive or negative) of the news to be captured more accurately.

Two types of deep learning models LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) were used to build the prediction models. Both models were trained using two different input types, news descriptions and news headlines. Historical price data was also included in the models. From the results, we found that the **GRU model performed better than the LSTM model in most cases**. GRU gave more accurate predictions. Also, news headlines gave slightly better results than full news descriptions for most cases, possibly because headlines are short and focus on the most important information.

However, for some cryptocurrencies, such as Solana (SOL) and Litecoin (LTC), the models performed slightly better with news descriptions. This suggests that the effectiveness of headline vs. description may vary based on the nature of the coin and the type of news it typically receives.

Overall, the models were able to predict prices with good accuracy. This shows that combining market data with sentiment from news can help improve price forecasting for cryptocurrencies.

# 6. Conclusion

This project successfully showed that deep learning models can predict cryptocurrency prices more accurately when combined with sentiment analysis from news articles.

Both price data and news sentiment were used to train the LSTM and GRU models. After the models were tested and compared, it was found that

- GRU models gave better results than LSTM models.
- News headlines helped more than full news descriptions in most cases.

It was also found that the **GRU model with news headlines gave the best results for most cryptocurrencies**. GRU was shown to be more accurate and faster to train than LSTM. High accuracy and low error rates were observed in the results.

The best accuracy values observed were 95.6% for ADA (GRU with headlines), 97.68% for BTC (GRU with headlines), 97.08% for ETH (GRU with headlines), 96.45% for LTC (GRU with descriptions), 94.97% for XRP (GRU with headlines), and 96.04% for SOL (GRU with descriptions).

# 7. Future Works

This study successfully combined sentiment analysis with deep learning models to improve cryptocurrency price forecasting. However, there are several areas where future research could improve and expand on this work:

1. Use More Data Sources
    - Add data from social media (Twitter, Reddit) and financial forums to improve sentiment analysis.
    - Use more financial news sources, blogs, and market analysis reports.
    - Try advanced AI models (like BERT, FinBERT, and GPT) for better sentiment detection.

2. Fine-Tune Sentiment Adjustment
    - The current model used a fixed 0.2 adjustment value for financial sentiment scoring.
    - Future experiments will test a range of values (e.g., 0.1 to 0.3) to find the optimal adjustment level that leads to the lowest prediction error.

3. Improve Model Performance
    - Use better deep learning models like Transformers, Attention-based LSTMs, or TCNs (Temporal Convolutional Networks).
    - Combine different models to get better predictions.
    - Use advanced tuning methods (like Bayesian Optimization) to find the best model settings.

4. Create a Real-Time Prediction System
    - Develop a system that updates predictions in real-time as new news articles and price data appear.
    - Create a web or mobile app where traders can instantly check price predictions.

# 8. References


1. Bhatt, S., Ghazanfar, M. and Amirhosseini, M., 2023. Sentiment-Driven Cryptocurrency Price Prediction: A Machine Learning Approach Utilizing Historical Data and Social Media Sentiment Analysis. *Machine Learning and Applications: An International Journal (MLAIJ)*, *10*(2/3), pp.1-15. [Sentiment Driven – Cryptocurrency price prediction:... - Google Scholar]

2. Malhotra, B., Chandwani, C., Agarwala, P. and Mann, S., 2022, October. Bitcoin price prediction using machine learning and deep learning algorithms. In *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 1-6). IEEE. [Bitcoin Price Prediction Using Machine Learning and... - Google Scholar]

3. Low, J.M., Tan, Z.J., Tang, T.Y. and Salleh, N.M., 2023, October. Deep learning and sentiment analysis-based cryptocurrency price prediction. In *International Visual Informatics Conference* (pp. 40-51). Singapore: Springer Nature Singapore. [Deep learning and sentiment analysis-based cryptocurrency... - Google Scholar]

4. Valencia, F., Gómez-Espinosa, A. and Valdés-Aguirre, B., 2019. Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. *entropy*, *21*(6), p.589. [Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning]

5. Raju, S.M. and Tarif, A.M., 2020. Real-time prediction of BITCOIN price using machine learning techniques and public sentiment analysis. *arXiv preprint arXiv:2006.14473*. [[2006.14473] Real-Time Prediction of BITCOIN Price using Machine Learning Techniques and Public Sentiment Analysis]

6. Lamon, C., Nielsen, E. and Redondo, E., 2017. Cryptocurrency price prediction using news and social media sentiment. *SMU Data Sci. Rev*, *1*(3), pp.1-22. [5237280.pdf]

7. Passalis, N., Avramelou, L., Seficha, S., Tsantekidis, A., Doropoulos, S., Makris, G. and Tefas, A., 2022. Multisource financial sentiment analysis for detecting Bitcoin price change indications using deep learning. *Neural Computing and Applications*, *34*(22), pp.19441-19452. [Multisource financial sentiment analysis for detecting Bitcoin price change indications using deep learning | Neural Computing and Applications]

8. Bhatt, S., Ghazanfar, M. and Amirhosseini, M., 2023. Machine learning based cryptocurrency price prediction using historical data and social media sentiment. *Computer Science & Information Technology (CS & IT)*, *13*(10), pp.1-11. [Machine Learning based Cryptocurrency Price Prediction using historical data and Social Media Sentiment: UEL Research Repository]

9. Yao, W., Xu, K. and Li, Q., 2019, June. Exploring the influence of news articles on bitcoin price with machine learning. In *2019 IEEE Symposium on computers and communications (ISCC)* (pp. 1-6). IEEE. [Exploring the Influence of News Articles on Bitcoin Price with Machine Learning | IEEE Conference Publication | IEEE Xplore]

10. Aslam, N., Rustam, F., Lee, E., Washington, P.B. and Ashraf, I., 2022. Sentiment analysis and emotion detection on cryptocurrency related tweets using ensemble LSTM-GRU model. *Ieee Access*, *10*, pp.39313-39324. [Sentiment Analysis and Emotion Detection on Cryptocurrency Related Tweets Using Ensemble LSTM-GRU Model | IEEE Journals & Magazine | IEEE Xplore]

# 9. Appendix

## 9.1. Web Scrapping

```
import time
import random
import mysql.connector
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
from webdriver_manager.chrome import ChromeDriverManager
from datetime import datetime, timedelta
import undetected_chromedriver as uc
```

```
# MySQL connection setup
db_config = {
    'host': 'localhost',  # MySQL host
    'user': 'root',       # MySQL username
    'password': [redacted]   # MySQL password
    'database': 'crypto_news'  # Database name
}

# Establish MySQL connection
try:
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()
    print("Database connection established.")
except mysql.connector.Error as e:
    print(f"Error connecting to MySQL: {e}")
    exit(1)

# Keywords for filtering cryptocurrency categories
crypto_keywords = {
    "BTC_news": ["bitcoin", "BTC", "Solona"],
    "Ethereum_news": ["ethereum", "ETH", "Ethereum"],
    "XRP_news": ["xrp", "ripple", "Ripple"],
    "Solana_news": ["solana", "SOL-USD", "Solona"],
    "ADA_news": ["cardano", "ADA", "Cardano"],
    "LTC_news": ["litecoin", "LTC", "Litecoin"]
}

# Create a table for each cryptocurrency category
for table_name in crypto_keywords.keys():
    cursor.execute(f'''
    CREATE TABLE IF NOT EXISTS {table_name} (
        id INT AUTO_INCREMENT PRIMARY KEY,
        date TEXT,
        headline TEXT,
        description TEXT
    )
    ''')
```

```python
# News categories and corresponding URLs
urls = [
    "https://cryptonews.com/news/bitcoin-news/",
    "https://cryptonews.com/news/altcoin-news/",
    "https://cryptonews.com/news/ethereum-news/",
    "https://cryptonews.com/news/defi-news/",
    "https://cryptonews.com/news/blockchain-news/"
]

# Initialize Chrome options
options = Options()
options.add_argument("--disable-gpu")
options.add_argument("--start-maximized")
options.add_argument("--disable-infobars")  # Prevent Chrome from showing automation info

# Find Chrome version installed
from selenium import webdriver
chrome_version = webdriver.Chrome().capabilities['browserVersion']  # Get installed Chrome version
print(f"Installed Chrome Version: {chrome_version}")

# Manually specify the ChromeDriver version using the ChromeDriverManager
driver_path = ChromeDriverManager().install()  # This will automatically find the correct driver version
driver = uc.Chrome(service=Service(driver_path), options=options)

# Function to save scraped data into the respective table
def save_to_db(date_text, headline_text, description_text, table_name):
    try:
        cursor.execute(f'''
        INSERT INTO {table_name} (date, headline, description)
        VALUES (%s, %s, %s)
        ''', (date_text, headline_text, description_text))
        conn.commit()
        print(f"Saved news to {table_name}: {headline_text}")
    except mysql.connector.Error as e:
        print(f"Error saving to {table_name}: {e}")
```

```python
# Function to convert relative date to absolute date
def convert_relative_date(relative_date):
    from datetime import datetime, timedelta
    today = datetime.today()
    if "day" in relative_date:
        days = int(relative_date.split()[0])
        return (today - timedelta(days=days)).strftime("%b %d, %Y")
    elif "hour" in relative_date:
        return today.strftime("%b %d, %Y")
    elif "week" in relative_date:
        weeks = int(relative_date.split()[0])
        return (today - timedelta(weeks=weeks)).strftime("%b %d, %Y")
    elif "month" in relative_date:
        months = int(relative_date.split()[0])
        return (today - timedelta(days=months * 30)).strftime("%b %d, %Y")
    elif "year" in relative_date:
        years = int(relative_date.split()[0])
        return (today - timedelta(days=years * 365)).strftime("%b %d, %Y")
    else:
        return relative_date
```

```python
# Function to scrape data from a given URL
def scrape_url(url):
    for page in range(1, 443):
        target_url = url if page == 1 else f"{url}page/{page}/"
        try:
            driver.get(target_url)
            time.sleep(random.randint(2, 5))
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
            time.sleep(random.randint(2, 4))

            soup = BeautifulSoup(driver.page_source, 'html.parser')
            news_items = soup.select('a.archive-template-latest-news--mini')

            for item in news_items:
                # Extract the date
                date = item.select_one('.archive-template-latest-news__time')
                date_text = date.get_text(strip=True) if date else 'N/A'

                # Extract the headline
                headline = item.select_one('.archive-template-latest-news__title')
                headline_text = headline.get_text(strip=True) if headline else ''

                # Extract the description
                news_description = item.select_one('.archive-template-latest-news__description')
                description_text = news_description.get_text(strip=True) if news_description else ''

                full_text = f"{headline_text} {description_text}".lower()

                # Check for crypto keywords and assign the correct table
                for table_name, keywords in crypto_keywords.items():
                    if any(kw.lower() in full_text for kw in keywords):
                        save_to_db(date_text, headline_text, description_text, table_name)

        except Exception as e:
            print(f"Failed to scrape {target_url}: {e}")
            continue
```

```python
# Scrape data from each URL
for url in urls:
    scrape_url(url)

# Close the browser and database connection
driver.quit()
cursor.close()
conn.close()

print("Scraping and saving to separate tables is complete.")
```

## Database Structure:

| | Tables_in_crypto_news |
|---|---|
| ▶ | ada_news |
| | btc_news |
| | ethereum_news |
| | ltc_news |
| | solana_news |
| | xrp_news |

| | Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|---|
| ▶ | id | int | NO | PRI | NULL | auto_increment |
| | date | text | YES | | NULL | |
| | headline | text | YES | | NULL | |
| | description | text | YES | | NULL | |

| id | date | headline | description |
|---|---|---|---|
| 1 | Jan 01, 2025 | Hive Digital Relocates HQ to Texas, Citing Pro-Bitcoin Stance of President… | Bitcoin mining firm Hive Digital Technologies has announced plans to move its … |
| 2 | Dec 16, 2024 | Bitcoin Soars Past $106,000 Amid Hopes of U.S. Strategic Bitcoin Reserve | 6,000 on Monday, fueled by spec… |
| 3 | Dec 13, 2024 | Japanese Lawmaker Asks Government to Create National Bitcoin Reserve | A Japanese lawmaker has asked the government to launch a national Bitcoin … |
| 4 | Nov 28, 2024 | ETH Co-Founder Charles Hoskinson Predicts Bitcoin Could Hit $500,000 W… | Ethereum blockchain and Cardano founder Charles Hoskinson says that Bitcoi… |
| 5 | Nov 11, 2024 | Bitcoin Hits New High Over $82k on Hopes of Trump's Pro-Crypto Policies | Bitcoin reached a record high above $82,000 on Monday, fueled by Donald Tr… |
| 6 | Aug 07, 2024 | Canada Sets Strict CIRO Compliance Deadline for Crypto Trading Platforms | Canada-based crypto trading platforms (CTPs) are facing tight regulatory de… |
| 7 | Jul 28, 2024 | Bitcoin Price Prediction as Trump Backs 'Strategic Bitcoin Stockpile' in Bold … | Bitcoin (BTC), the world's leading cryptocurrency, has experienced volatility. … |

## Database Daily Update Code:

```python
import random
import time
from datetime import datetime, timedelta
from bs4 import BeautifulSoup
import mysql.connector
import undetected_chromedriver as uc
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

# MySQL connection setup
db_config = {
    'host': 'localhost',  # MySQL host
    'user': 'root',       # MySQL username
    'password': '[redacted]',  # MySQL password
    'database': 'crypto_news'  # Database name
}

# Establish MySQL connection
try:
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()
    print("Database connection established.")
except mysql.connector.Error as e:
    print(f"Error connecting to MySQL: {e}")
    exit(1)

# Keywords for filtering cryptocurrency categories
crypto_keywords = {
    "BTC_news": ["bitcoin", "BTC", "Bitcoin"],
    "Ethereum_news": ["ethereum", "ETH", "Ethereum"],
    "XRP_news": ["xrp", "ripple", "Ripple"],
    "Solana_news": ["solana", "SOL-USD", "Solona"],
    "ADA_news": ["cardano", "ADA", "Cardano"],
    "LTC_news": ["litecoin", "LTC", "Litecoin"]
}
```

```python
# Create tables for each cryptocurrency category if not exists
for table_name in crypto_keywords.keys():
    cursor.execute(f'''
    CREATE TABLE IF NOT EXISTS {table_name} (
        id INT AUTO_INCREMENT PRIMARY KEY,
        date TEXT,
        headline TEXT,
        description TEXT
    )
    ''')

# Function to check if a headline already exists in the table
def is_news_exists(headline, table_name):
    cursor.execute(f"SELECT COUNT(*) FROM {table_name} WHERE headline = %s", (headline,))
    return cursor.fetchone()[0] > 0


def convert_relative_date(relative_date):
    try:
        if "day" in relative_date:
            days_ago = int(relative_date.split()[0])
            date = datetime.now() - timedelta(days=days_ago)
        elif "hour" in relative_date:
            hours_ago = int(relative_date.split()[0])
            date = datetime.now() - timedelta(hours=hours_ago)
        elif "minute" in relative_date:
            minutes_ago = int(relative_date.split()[0])
            date = datetime.now() - timedelta(minutes=minutes_ago)
        else:
            # Assuming the date format is like 'Jan 19, 2025'
            date = datetime.strptime(relative_date, '%b %d, %Y')
        return date.strftime('%Y-%m-%d')
    except ValueError as e:
        print(f"Error parsing date '{relative_date}': {e}")
        return 'N/A'
```

```python
# Function to save scraped data into the database
def save_to_db(date_text, headline_text, description_text, table_name):
    if not is_news_exists(headline_text, table_name):
        try:
            cursor.execute(f'''
            INSERT INTO {table_name} (date, headline, description)
            VALUES (%s, %s, %s)
            ''', (date_text, headline_text, description_text))
            conn.commit()
            print(f"Saved news to {table_name}: {headline_text}")
        except mysql.connector.Error as e:
            print(f"Error saving to {table_name}: {e}")
```

```python
# Function to scrape data from the URL
def scrape_url(url, pages=2):
    try:
        for page in range(1, pages + 1):
            target_url = url if page == 1 else f"{url}page/{page}/"
            driver.get(target_url)

            # Randomized scrolling to simulate human behavior
            for _ in range(3):  # Simulate 3 scrolls down the page
                driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
                time.sleep(random.randint(3, 5))  # Wait for content to load

            # Parse page content with BeautifulSoup
            soup = BeautifulSoup(driver.page_source, 'html.parser')

            # Find news items on the page
            news_items = soup.select('a.archive-template-latest-news--mini')

            for item in news_items:
                # Extract the date
                date = item.select_one('.archive-template-latest-news__time')
                relative_date = date.get_text(strip=True) if date else 'N/A'
                date_text = convert_relative_date(relative_date) if relative_date != 'N/A' else 'N/A'

                # Extract the headline
                headline = item.select_one('.archive-template-latest-news__title')
                headline_text = headline.get_text(strip=True) if headline else ''

                # Extract the description
                news_description = item.select_one('.archive-template-latest-news__description')
                description_text = news_description.get_text(strip=True) if news_description else ''

                full_text = f"{headline_text} {description_text}".lower()

                # Check for keywords and assign the correct table
                for table_name, keywords in crypto_keywords.items():
                    if any(kw.lower() in full_text for kw in keywords):
                        save_to_db(date_text, headline_text, description_text, table_name)

    except Exception as e:
        print(f"Error while scraping {url}: {e}")
```

```python
# Initialize Chrome options for undetected_chromedriver
options = Options()
options.add_argument("--disable-gpu")
options.add_argument("--start-maximized")
options.add_argument("--disable-infobars")
options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36")
options.add_argument("--disable-blink-features=AutomationControlled")

# Initialize undetected_chromedriver
driver_path = ChromeDriverManager().install()  # Automatically finds the correct driver
driver = uc.Chrome(service=Service(driver_path), options=options)  # Using undetected_chromedriver

# Example URLs to scrape
urls = [
    "https://cryptonews.com/news/bitcoin-news/",
    "https://cryptonews.com/news/altcoin-news/",
    "https://cryptonews.com/news/ethereum-news/",
    "https://cryptonews.com/news/defi-news/",
    "https://cryptonews.com/news/blockchain-news/"
]

# Scrape data from each URL
for url in urls:
    scrape_url(url)

# Close the browser and database connection
driver.quit()
cursor.close()
conn.close()

print("Scraping completed.")
```

## 9.2. Prediction Models

Git Hub link: https://github.com/IsuriUp/Final_Project.git

## 9.3. Prediction Web Application

```
PREDICTION APP > utils.py > predict_next_price
import re
import joblib
import numpy as np
import pandas as pd
import spacy
import yfinance as yf
import mysql.connector
import tensorflow as tf
from datetime import datetime, timedelta
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tensorflow.keras.models import load_model
```

```
# Load NLP Model & Sentiment Analyzer
nlp = spacy.load("en_core_web_sm", disable=["parser", "ner"])
analyzer = SentimentIntensityAnalyzer()
```

```
# Define Paths for Models & Scalers
MODEL_PATHS = {
    "ADA": r"C:\Users\acer\Desktop\Research_New\Research_N\ADA\best_model_H_gru.h5",
    "BTC": r"C:\Users\acer\Desktop\Research_New\Research_N\BTC\best_model_H_gru.h5",
    "ETH": r"C:\Users\acer\Desktop\Research_New\Research_N\ETH\best_model_H_gru.h5",
    "SOL": r"C:\Users\acer\Desktop\Research_New\Research_N\SOL\best_model_D_gru.h5",
    "XRP": r"C:\Users\acer\Desktop\Research_New\Research_N\XRP\best_model_H_gru.h5",
    "LTC": r"C:\Users\acer\Desktop\Research_New\Research_N\LTC\best_model_D_gru.h5",
}

SCALER_PATHS = {
    "ADA": r"C:\Users\acer\Desktop\Research_New\Research_N\ADA\scaler_GH.pkl",
    "BTC": r"C:\Users\acer\Desktop\Research_New\Research_N\BTC\scaler_GH.pkl",
    "ETH": r"C:\Users\acer\Desktop\Research_New\Research_N\ETH\scaler_GH.pkl",
    "SOL": r"C:\Users\acer\Desktop\Research_New\Research_N\SOL\scaler_GD.pkl",
    "XRP": r"C:\Users\acer\Desktop\Research_New\Research_N\XRP\scaler_GH.pkl",
    "LTC": r"C:\Users\acer\Desktop\Research_New\Research_N\LTC\scaler_GD.pkl",
}
```

```python
# Load Financial Sentiment Dictionary
lm_positive = {    "gain", "growth", "bullish", "opportunity", "profit", "surge", "expansion",
    "adoption", "approval", "partnership", "innovation", "mainstream", "rally",
    "institutional", "breakthrough", "stability", "strong", "all-time-high",
    "liquidity", "halving", "scarcity", "whale accumulation", "stake", "integration",
    "etf approval", "layer 2 scaling", "metaverse expansion", "positive outlook"}

lm_negative = {"crash", "loss", "bearish", "decline", "debt", "bankruptcy", "downturn",
    "regulatory crackdown", "ban", "fud", "lawsuit", "hacked", "ponzi",
    "liquidation", "fraud", "scam", "rug pull", "dumping", "sell-off",
    "volatility", "delisting", "depeg", "inflation", "recession", "exchange insolvency",
    "crypto winter", "negative outlook"}

# Function to dynamically load model and scaler
def load_crypto_model(crypto):
    """Loads the trained model and scaler for the selected cryptocurrency."""
    model_path = MODEL_PATHS.get(crypto)
    scaler_path = SCALER_PATHS.get(crypto)

    if model_path and scaler_path:
        model = load_model(model_path)
        scaler = joblib.load(scaler_path)
        return model, scaler
    else:
        raise ValueError(f"Model or Scaler not found for {crypto}")
```

```python
# Text Preprocessing for Sentiment Analysis
def preprocess_text(text):
    """Clean, lemmatize, and remove stopwords."""
    if pd.isnull(text):
        return ""
    text = text.lower()
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    doc = nlp(text)
    return " ".join([token.lemma_ for token in doc if not token.is_stop])

# Function to adjust sentiment based on financial lexicon
def financial_sentiment_adjustment(text, vader_score):
    """Adjust sentiment score using Loughran-McDonald financial lexicon."""
    words = text.split()
    adjustment = sum(0.2 if word in lm_positive else -0.2 if word in lm_negative else 0 for word in words)
    return max(min(vader_score + adjustment, 1), -1)
```

```python
# Fetch Latest News Data
def fetch_latest_news(db_table):
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="Ariyarathne99@",
            database="crypto_news"
        )
        query = f"SELECT date, headline, description FROM {db_table} WHERE date >= CURDATE() - INTERVAL 2 DAY"
        news_data = pd.read_sql_query(query, conn)
        conn.close()
        return news_data
    except mysql.connector.Error as err:
        print(f"Error fetching news: {err}")
        return pd.DataFrame(columns=["date", "headline", "description"])

# Fetch Historical Price Data
def fetch_price_data(symbol, window_size=2):
    """Fetches the last `window_size` days of price data."""
    end_date = datetime.today().strftime('%Y-%m-%d')
    start_date = (datetime.today() - timedelta(days=window_size + 2)).strftime('%Y-%m-%d')

    price_data = yf.download(symbol, start=start_date, end=end_date, interval="1d")
    if price_data.empty:
        return [0] * window_size  # Handle missing data
    return price_data["Close"].tail(window_size).values
```

```python
# Predict Next Price
def predict_next_price(crypto, symbol, db_table):
    """Predicts today's closing price for the given cryptocurrency."""
    model, scaler = load_crypto_model(crypto)

def predict_next_price(crypto, symbol, db_table):
    """Predicts today's closing price for the given cryptocurrency."""
    model, scaler = load_crypto_model(crypto)

    # Sentiment analyzer
    analyzer = SentimentIntensityAnalyzer()

    # Fetch and preprocess news
    news_df = fetch_latest_news(db_table)
    if not news_df.empty:
        news_df["cleaned_text"] = news_df["headline"].astype(str).apply(preprocess_text)
        news_df["sentiment"] = news_df["cleaned_text"].apply(lambda x: financial_sentiment_adjustment(x, analyzer.polarity_scores(x)["com
        latest_sentiment = news_df["sentiment"].iloc[-1]
    else:
        latest_sentiment = 0

    # Fetch price data (last 2 closing prices)
    latest_prices = fetch_price_data(symbol)  # shape (2,)

    # Combine sentiment and price for scaling
    input_data = np.column_stack((np.array([[latest_sentiment]] * 2), latest_prices))  # shape (2, 2)
    scaled_input = scaler.transform(input_data)

    # Build final model input shape (1, 3, 1)
    X_sequence = scaled_input[:, 1].reshape(1, 2, 1)  # past Close prices
    X_sentiment = np.array([[latest_sentiment]]).reshape(1, 1, 1)  # today's sentiment
    X_input = np.concatenate((X_sequence, X_sentiment), axis=1)
```

```python
    # Predict scaled value and inverse scale
    predicted_scaled_price = model.predict(X_input)
    predicted_price = scaler.inverse_transform([[0, predicted_scaled_price[0][0]]])[0][1]

    predicted_date = datetime.today().strftime('%Y-%m-%d')
    return predicted_price, predicted_date
```

(variable) predicted_scaled_price: Any

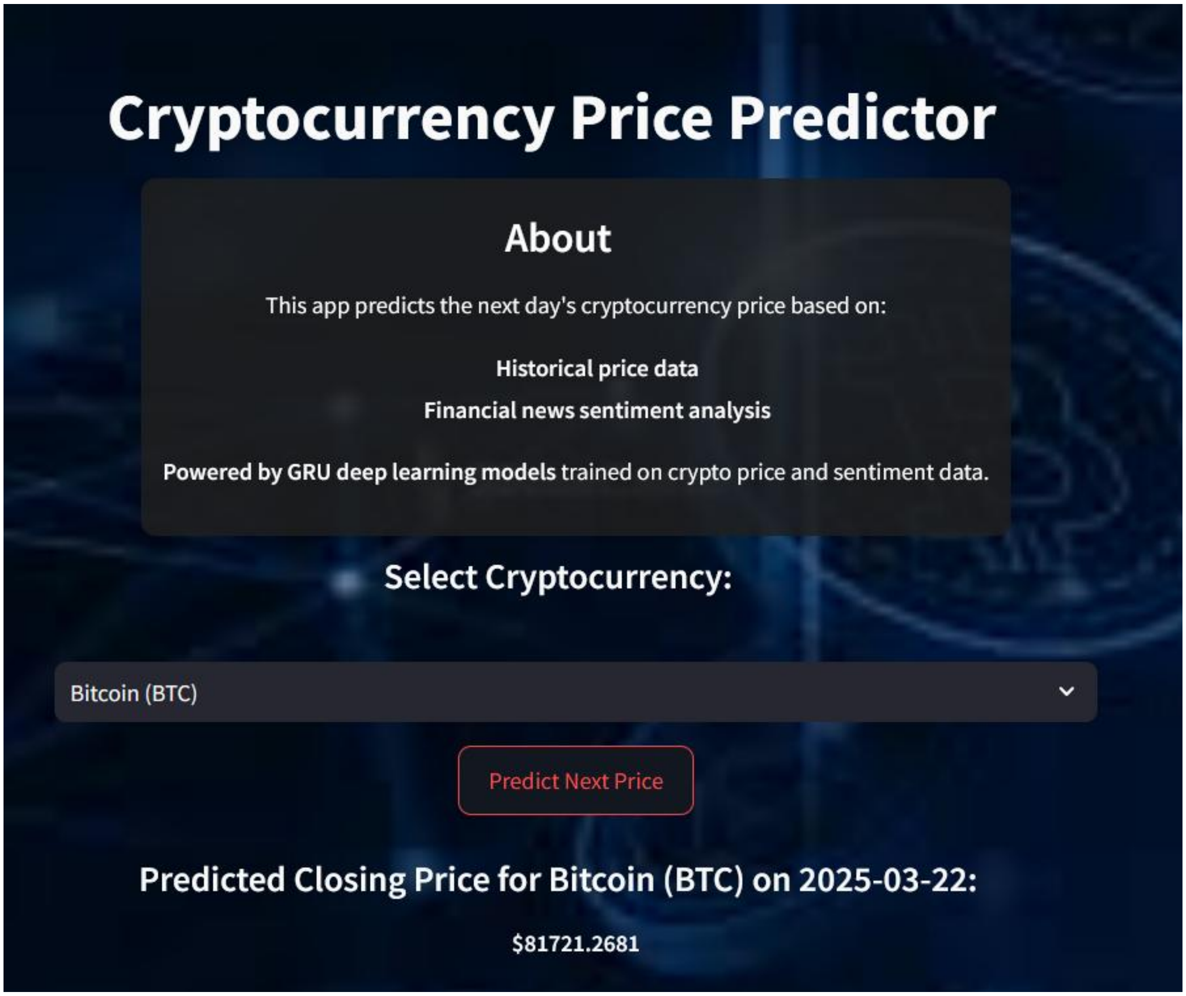
PREDICTION APP > app.py > ...

```python
import streamlit as st
import base64
from utils import predict_next_price

# Helper: Convert local image to base64
def get_base64_bg_image(image_path):
    with open(image_path, "rb") as img_file:
        encoded = base64.b64encode(img_file.read()).decode()
    return f"data:image/jpg;base64,{encoded}"

# Load your background image from local path
bg_image = get_base64_bg_image(r"C:\Users\acer\Downloads\images.jpg")

# Inject CSS for styling and background
st.markdown(f"""
    <style>
    .stApp {{
        background-image: url("{bg_image}");
        background-size: cover;
        background-position: center;
        background-attachment: fixed;
    }}

    .centered {{
        text-align: center;
    }}

    @media (prefers-color-scheme: light) {{
        .highlight-box {{
            background-color: rgba(255, 255, 255, 0.85);
            color: #000000;
        }}
    }}
```

```
    @media (prefers-color-scheme: dark) {{
        .highlight-box {{
            background-color: rgba(30, 30, 30, 0.85);
            color: #ffffff;
        }}
    }}

    .highlight-box {{
        padding: 15px;
        border-radius: 10px;
        box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
        display: inline-block;
        text-align: center;
        max-width: 700px;
        margin: auto;
    }}

    .stButton > button {{
        display: block;
        margin: auto;
        font-size: 18px;
        font-weight: bold;
        padding: 10px 20px;
    }}
    </style>
""", unsafe_allow_html=True)

# Title
st.markdown("<h1 class='centered'> Cryptocurrency Price Predictor </h1>", unsafe_allow_html=True)
```

```
# About Section
st.markdown("""
    <div class="centered">
        <div class="highlight-box">
            <h3>About</h3>
            <p>This app predicts the next day's cryptocurrency price based on:</p>
            <ul style="list-style-type: none; padding: 0;">
                <li><b>Historical price data</b></li>
                <li><b>Financial news sentiment analysis</b></li>
            </ul>
            <p><b>Powered by GRU deep learning models</b> trained on crypto price and sentiment data.</p>
        </div>
    </div>
""", unsafe_allow_html=True)

# Cryptocurrency Selection
st.markdown("<h4 class='centered'> Select Cryptocurrency:</h4>", unsafe_allow_html=True)

crypto_options = {
    "Cardano (ADA)": ("ADA", "ADA-USD", "ada_news"),
    "Bitcoin (BTC)": ("BTC", "BTC-USD", "btc_news"),
    "Ethereum (ETH)": ("ETH", "ETH-USD", "ethereum_news"),
    "Solana (SOL)": ("SOL", "SOL-USD", "solana_news"),
    "Ripple(XRP)": ("XRP", "XRP-USD", "xrp_news"),
    "Litecoin (LTC)": ("LTC", "LTC-USD", "ltc_news"),
}

crypto_choice = st.selectbox("", list(crypto_options.keys()), index=0)
crypto_key, symbol, db_table = crypto_options[crypto_choice]
```

```
# Predict Button
if st.button("Predict Next Price"):
    with st.spinner("Analyzing news and price data..."):
        predicted_price, predicted_date = predict_next_price(crypto_key, symbol, db_table)
        st.markdown(f"""
            <div class='centered'>
                <h4>Predicted Closing Price for {crypto_choice} on {predicted_date}:</h4>
                <p><b>${predicted_price:.4f}</b></p>
            </div>
        """, unsafe_allow_html=True)

```


Cryptocurrency Price Predictor
About
This app predicts the next day's cryptocurrency price based on:
Historical price data
Financial news sentiment analysis
Powered by GRU deep learning models trained on crypto price and sentiment data.
Select Cryptocurrency:
Bitcoin (BTC)
Predict Next Price
Predicted Closing Price for Bitcoin (BTC) on 2025-03-22:
$81721.2681