FPT UNIVERSITY

Cryptocurrency Price Prediction Utilizing Sentiment Analysis and Historical Data

Le Lam Nhat Linh, Nguyen Nhat Minh, Nguyen Tran Thang

May 2024

1 Project Title

Cryptocurrency price prediction utilizing sentiment analysis and historical data.

2 Team Members

Team Members	**Roles**
Le Lam Nhat Linh	Leader
Nguyen Nhat Minh	Member
Nguyen Tran Thang	Member

3 Introduction and Background

3.1 Objective

The main goal of this project is to build a sophisticated model for forecasting cryptocurrency prices using sentiment analysis derived from social media alongside historical market data. Through the fusion of these data sets, the project seeks to elevate the precision of price prognostications, delivering valuable insights for traders, investors, and financial analysts.

3.2 Motivation

Cryptocurrency markets are renowned for their pronounced volatility and susceptibility to shifts in market sentiment. Conventional financial models often struggle to forecast these markets due to the distinctive factors that influence cryptocurrency valuations, including news events, social media sentiment, and macroeconomic indicators. The significance of this project is underscored by its innovative approach, which integrates sentiment analysis with historical price data to enhance predictive precision. The ability to understand and anticipate price fluctuations in the cryptocurrency realm has the potential to refine investment strategies, fortify risk management practices, and deepen market insights. Furthermore, this effort delves into the broader implications of how social media dynamics and public sentiment can exert influence on financial markets, which could shape future research avenues and methodologies in the realm of financial prognostication.

3.3 Background Information

Cryptocurrencies have attracted considerable attention since Bitcoin's emergence in 2009. These virtual assets derive their value not only from traditional supply and demand factors but also from elements such as public perception, regulatory changes, technological advancements, and social media trends. Unlike traditional assets, cryptocurrencies are decentralized and operate on a peer-to-peer network, making them highly sensitive to market sentiments and breaking news.

3.3.1 Historical Data

Cryptocurrency price data encompass key parameters, including the opening price, closing price, highest price, lowest price, and trading volume. These features are commonly sourced from prominent cryptocurrency exchanges such as Binance, Coinbase, and CoinGecko.

3.3.2 Sentiment Analysis

Sentiment analysis involves the extraction of subjective information from text data, often collected from social media channels such as X and Reddit. The widely used tools for sentiment analysis include VADER, TextBlob, and advanced models such as BERT. These instruments aid in quantifying sentiments expressed in social media posts and comments, categorizing them as positive, negative, or neutral responses.

3.3.3 Challenges

The unpredictable nature of cryptocurrencies presents challenges for traditional financial models in accurately forecasting their behavior. Furthermore, unstructured and volatile social media sentiment adds complexity to the analysis, making it difficult to integrate these data effectively with historical price data.

Through tackling these obstacles, this project seeks to develop a predictive model capable of effectively encapsulating the intricacies of the cryptocurrency markets. The ultimate goal is to provide a reliable tool for projecting price fluctuations.

4 Literature Review

4.1 Key Findings from Existing Literature

4.1.1 Historical Data Analysis for Price Prediction

Ali Mohammadjafari [3]. (2024). Comparative Study of Bitcoin Price Prediction. This research examines the potential of neural network models, specifically long-short-term memory (LSTM) and gated recurring unit (GRU) networks, to forecast Bitcoin price movements. Using five-fold cross-validation and L2 regularization, the study found that GRU models outperform LSTM models.

Zihan Dong, Xinyu Fan, Zhiyuan Peng [2]. (2024). FNSPID: A Comprehensive Financial News Dataset in Time Series. This paper introduces the Financial News and Stock Price Integration Dataset (FNSPID), which combines 29.7 million stock prices and 15.7 million time-aligned financial news records for 500 companies from 1999 to 2023. The study demonstrates that the dataset enhances market prediction accuracy, especially when incorporating sentiment scores with transformer-based models.

4.1.2 Sentiment Analysis for Financial Markets

Shen Ao [1]. (2018). Sentiment Analysis Based on Financial Tweets and Market Information. This paper explores the correlation between financial news sentiment and stock market movements through empirical analysis. By using TextBlob for sentiment analysis and integrating data mining techniques, the study demonstrates that fluctuations in sentiment often match stock market price changes, highlighting the effectiveness of sentiment analysis in financial forecasting.

Enmin Zhu, Jerome Yen [6]. (2024). BERTopic-Driven Stock Market Predictions: Unraveling Sentiment Insights. This study employs BERTopic, an advanced NLP technique, to analyze the sentiment of topics derived from stock market comments. The integration of BERTopic with deep learning models enhances stock price prediction accuracy, showing that topic sentiment provides valuable insights into market volatility and trends.

4.1.3 Combining Sentiment Analysis and Historical Data

Sheng-Hsiu Wu, Yuling Liu, Ziran Zou, T. Weng [4]. (2022). S_I_LSTM: stock price prediction based on multiple data sources and sentiment analysis. This research proposes a method that integrates multiple data sources, including historical data, technical indicators, and sentiment from stock posts and financial news, using a convolutional neural network for sentiment analysis and an LSTM network for prediction. The method significantly improves prediction accuracy for the China Shanghai A-share market, achieving a mean absolute error of 2.386835.

Zi Ye, Yinxu Wu, Hui Chen, Yi Pan, Q. Jiang [5]. (2022). A Stacking Ensemble Deep Learning Model for Bitcoin Price Prediction Using Twitter Comments on Bitcoin. This paper proposes an ensemble deep learning model that integrates LSTM and GRU networks using a stacking technique to predict Bitcoin prices. The model uses sentiment indexes from social media comments and technical indicators, showing better performance with a mean absolute error 88.74 % lower than traditional methods.

4.2 Gaps in Current Knowledge

4.2.1 Focus on Major Cryptocurrencies

Current research predominantly focuses on major cryptocurrencies like Bitcoin and Ethereum. There is a lack of comprehensive studies exploring the application of these techniques to a broader range of cryptocurrencies, particularly those with smaller market capitalizations.

4.2.2 Real-Time Prediction Models

Existing literature often examines the historical impact of sentiment on prices rather than developing real-time predictive models. There is a need for research that not only analyzes past data but also creates models capable of making real-time price predictions based on live sentiment and market data.

4.2.3 Long-Term Prediction Accuracy

Most studies focus on short-term price predictions, leaving a gap in understanding the long-term impact of sentiment on cryptocurrency prices. Further research is needed to explore the effectiveness of integrated models for long-term price forecasting.

4.3 Addressing the Gaps

This project endeavors to address these gaps through the utilization of sophisticated sentiment analysis models to enhance the precision of sentiment extraction from social media data. It also seeks to broaden the scope of analysis to encompass a more extensive array of cryptocurrencies beyond Bitcoin and Ethereum, as well as to investigate the predictive capacities of the integrated model over both short and long-term horizons.

5 Data Description

5.1 Source

5.1.1 Cryptocurrency Price Historical Data

The historical price data for cryptocurrencies was obtained through both APIs and direct web scraping techniques. We utilized APIs provided by various cryptocurrency exchanges such as Binance, Coinbase, and CoinGecko. Additionally, for comprehensive data coverage, we employed web scraping from exchange platforms, notably Binance. Specifically, data was scraped directly from the Binance BTC/USDT trading page here.

5.1.2 Sentiment Analysis Data

For sentiment analysis, we initially experimented with crawling data from social media platforms, targeting posts and comments with hashtags related to cryptocurrencies. However, we found that for major cryptocurrencies like BTC, social media sentiment did not significantly impact their price. Consequently, we shifted our focus to more reliable sources.

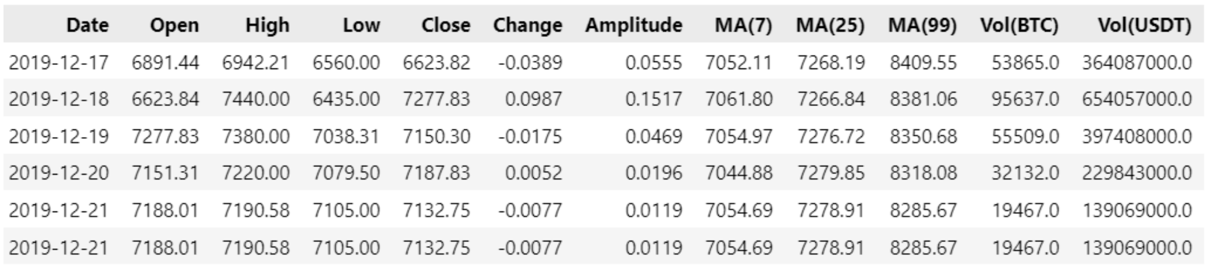
We collected sentiment data from trusted news platforms and forums within the cryptocurrency domain. Notably, we sourced news articles and updates from Binance's news section here, ensuring the information was credible and relevant. These news sources are known to have a considerable influence on the price movements of cryptocurrencies.

5.2 Size and Format

Cryptocurrency Price Historical Data:

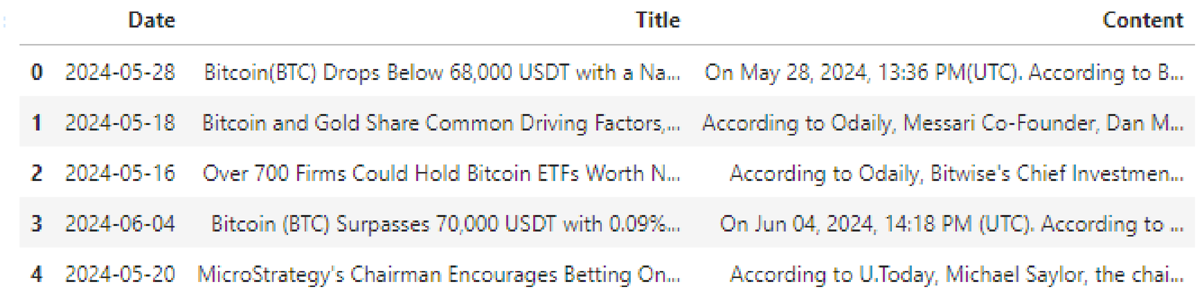
- Size: Varied based on the frequency and duration of data scraping, ranging from several megabytes to gigabytes.
- Format: CSV, JSON, or retrieved directly from API endpoints.

Sentiment Analysis Data:



Date	Open	High	Low	Close	Change	Amplitude	MA(7)	MA(25)	MA(99)	Vol(BTC)	Vol(USDT)
2019-12-17	6891.44	6942.21	6560.00	6623.82	-0.0389	0.0555	7052.11	7268.19	8409.55	53865.0	364087000.0
2019-12-18	6623.84	7440.00	6435.00	7277.83	0.0987	0.1517	7061.80	7266.84	8381.06	95637.0	654057000.0
2019-12-19	7277.83	7380.00	7038.31	7150.30	-0.0175	0.0469	7054.97	7276.72	8350.68	55509.0	397408000.0
2019-12-20	7151.31	7220.00	7079.50	7187.83	0.0052	0.0196	7044.88	7279.85	8318.08	32132.0	229843000.0
2019-12-21	7188.01	7190.58	7105.00	7132.75	-0.0077	0.0119	7054.69	7278.91	8285.67	19467.0	139069000.0
2019-12-21	7188.01	7190.58	7105.00	7132.75	-0.0077	0.0119	7054.69	7278.91	8285.67	19467.0	139069000.0

Figure 1: Sample price data



	Date	Title	Content
0	2024-05-28	Bitcoin(BTC) Drops Below 68,000 USDT with a Na...	On May 28, 2024, 13:36 PM(UTC). According to B...
1	2024-05-18	Bitcoin and Gold Share Common Driving Factors,...	According to Odaily, Messari Co-Founder, Dan M...
2	2024-05-16	Over 700 Firms Could Hold Bitcoin ETFs Worth N...	According to Odaily, Bitwise's Chief Investmen...
3	2024-06-04	Bitcoin (BTC) Surpasses 70,000 USDT with 0.09%...	On Jun 04, 2024, 14:18 PM (UTC). According to ...
4	2024-05-20	MicroStrategy's Chairman Encourages Betting On...	According to U.Today, Michael Saylor, the chai...

Figure 2: Sample news data

- Size: Depending on the volume of data collected from social media platforms, news websites, and sentiment analysis tools, ranging from tens of megabytes to hundreds of gigabytes.
- Format: Text data stored in JSON, CSV, or retrieved directly from APIs.

5.3 Features

Cryptocurrency Price Historical Data:

- **Date**: The date of the data record.
- **Open**: The opening price of the cryptocurrency on that day.
- **High**: The highest price reached during the day.
- **Low**: The lowest price reached during the day.
- **Close**: The closing price of the cryptocurrency on that day.
- **Change**: The percentage change in price compared to the previous day.
- **Amplitude**: The range of price fluctuation during the day (High - Low).
- **MA(7)**: 7-day moving average of the closing price.
- **MA(25)**: 25-day moving average of the closing price.
- **MA(99)**: 99-day moving average of the closing price.
- **Vol(BTC)**: Trading volume in Bitcoin units.
- **Vol(USDT)**: Trading volume in USDT (Tether) units.

Sentiment Analysis Data:

- **Date**: Publication date of the article
- **Title**: Headline of the news article
- **Content**: Main body of the news article

6 Data Cleaning and Preprocessing

6.1 Cryptocurrency Price Historical Data

Data Processing and Conversion: In the initial phase of data processing, the 'Date' column was transformed into a datetime format to facilitate time-based operations and analysis. This change enabled efficient filtering, sorting, and other time-series analysis tasks. The 'Change' and 'Amplitude' columns, originally represented as percentage strings, were converted into decimal numbers. This conversion was essential for accurate mathematical computations and to maintain consistency in data representation.

For the 'Vol(BTC)' and 'Vol(USDT)' columns, which contained symbols such as K (thousand), M (million), and B (billion), these were converted into their respective numerical values. This transformation was accomplished using the `convert_to_number` function, which streamlined the conversion process and ensured that the data could be utilized effectively for trading volume analysis.

Data Filtering by Time Range: To maintain the relevance and manageability of the dataset, a filter was applied to retain data only within the specified time range from December 17, 2019, to January 1, 2024. This filtering step was crucial in focusing the analysis on the most recent and significant data points, aligning with the project's objectives.

Checking and Handling Missing Dates: A comprehensive check for missing dates within the specified time frame was conducted using the `check_missing_dates` function. Upon identifying any gaps in the dates, the `handle_missing_dates` function was employed to insert the missing dates into the DataFrame. Subsequently, linear interpolation was used to estimate and fill in the missing values, ensuring continuity and completeness in the dataset. Linear interpolation involves calculating the missing values based on the linear relationship between the surrounding known data points. This method provides a straightforward and effective way to maintain a continuous time series, which is crucial for accurate analysis and modeling.

Sorting and Saving Data: After processing, the DataFrame was meticulously sorted by date to ensure chronological order, which is fundamental for any time-series analysis. The sorted and processed DataFrame was then saved to a new CSV file named `BTC_data.csv`, preserving the cleaned and structured data for further analysis and modeling.

Price Dataset Structure: The final structured dataset comprised twelve columns, namely: Date, Open, High, Low, Close, Change, Amplitude, MA(7), MA(25), MA(99), Vol(BTC), and Vol(USDT). The time range of the dataset spanned from December 17, 2019, to July 1, 2024, with daily data frequency. This structure provided a comprehensive overview of the cryptocurrency price movements and related features, facilitating in-depth analysis and model training.

Calculating the Relative Strength Index (RSI) The Relative Strength Index (RSI) is a momentum indicator used in technical analysis. Developed by J. Welles Wilder, RSI measures the speed and change of price movements. It oscillates between 0 and 100 and is commonly used to identify overbought or oversold conditions in a traded asset.

RSI Calculation Formula: RSI is calculated based on average gains and losses over a specified period, typically 14 days. The formula for RSI is:

$$\text{RSI} = 100 - \left(\frac{100}{1 + RS}\right)$$

Where:

$$\text{RS} = \frac{\text{Average Gain}}{\text{Average Loss}}$$

- RSI > 70: This is often considered a signal that the asset may be overbought, potentially indicating a "bull market" phase (a market with rising prices).
- RSI < 30: This is often considered a signal that the asset may be oversold, potentially indicating a "bear market" phase (a market with falling prices).

6.2 Sentiment Analysis Data

Our approach to sentiment analysis leverages a sophisticated ensemble of methodologies, combining traditional lexicon-based techniques with state-of-the-art machine learning models. This ensemble includes TextBlob, VADER, and a BERT-based sentiment analyzer fine-tuned for financial discourse, complemented by a specialized lexicon tailored specifically for cryptocurrency sentiment analysis. The goal is to capture and interpret the complex nuances of sentiment expressed in cryptocurrency news articles, offering deep insights into market sentiment dynamics and their potential impact on investor behavior.

6.2.1 Data

The dataset (`BTC_news.csv`) consists of a comprehensive collection of news articles related to Bitcoin. Each article is characterized by essential fields such as the publication date (**Date**), headline (**Title**), and main content (**Content**), providing a rich source of textual data for sentiment analysis.

6.2.2 Methodology

Text Preprocessing Before sentiment analysis, extensive preprocessing steps were applied to standardize the textual data. This included converting all text to lowercase to ensure uniformity, removing URLs, mentions, hashtags, punctuation, and digits to eliminate noise, and performing stopword removal to filter out common words that do not contribute to sentiment analysis. Additionally, whitespace normalization was conducted to streamline text formatting and ensure consistency in data representation.

Sentiment Analysis Techniques Our methodology encompassed a diverse range of sentiment analysis techniques:

- **TextBlob**: A straightforward lexicon-based approach that assigns sentiment scores based on predefined word lists, suited for capturing basic sentiment polarity.
- **VADER**: A rule-based sentiment analysis tool specifically tuned to handle nuances in sentiment expressed in social media and other forms of online text, offering robustness in capturing sentiment in cryptocurrency news.
- **BERT**: Leveraging the FinBERT model, which is fine-tuned specifically for financial text analysis, allowing for a more nuanced understanding of sentiment by considering context and semantics within cryptocurrency-related content.
- **Crypto-specific lexicon**: Developed as a custom dictionary of cryptocurrency-related terms with associated sentiment scores, enabling the identification and analysis of sentiment specific to the cryptocurrency domain.

Feature Engineering To enhance sentiment analysis effectiveness, several feature engineering techniques were employed:

- **Text Complexity Analysis**: Assessment of text complexity features such as the presence of contrasting statements, number of entities, and noun chunks, providing deeper insights into the intricacies of sentiment expressed.
- **Text Length**: Evaluation of the length of articles to understand potential correlations between article length and sentiment expression.
- **Cryptocurrency Keyword Frequency**: Analysis of the frequency of specific cryptocurrency-related keywords within articles to gauge their impact on sentiment scores.

Ensemble Model The final step involved aggregating outputs from individual sentiment analysis models using a Random Forest regressor. This ensemble approach integrated sentiment scores generated by TextBlob, VADER, BERT, and the crypto-specific lexicon, along with engineered features, to produce a comprehensive sentiment analysis outcome. By combining these diverse sources of sentiment insights, the ensemble model aimed to achieve greater accuracy and reliability in predicting sentiment trends within cryptocurrency news.

Analysis Our analysis focused on several key areas:

- **Cross-validation**: Employed to assess the performance and robustness of the ensemble model in predicting sentiment across different subsets of the dataset.
- **Feature Importance**: Determined to understand which features (e.g., sentiment scores from different models, text complexity features, keyword frequencies) contributed most significantly to predicting sentiment outcomes.
- **Temporal Sentiment Trends**: Analyzed to uncover how sentiment within cryptocurrency news evolves over time, identifying potential patterns or shifts in market sentiment.
- **Model Discrepancies**: Investigated cases of high disagreement among individual sentiment analysis models to understand instances where different models assigned divergent sentiment scores to the same article.

This comprehensive approach to sentiment analysis not only enhances our understanding of sentiment dynamics within the cryptocurrency market but also provides actionable insights that can inform strategic decisions related to investment and market participation.

6.3 Results

6.3.1 Sentiment Distribution

The sentiment distribution analysis reveals distinctive characteristics for each sentiment analysis technique utilized in our study.

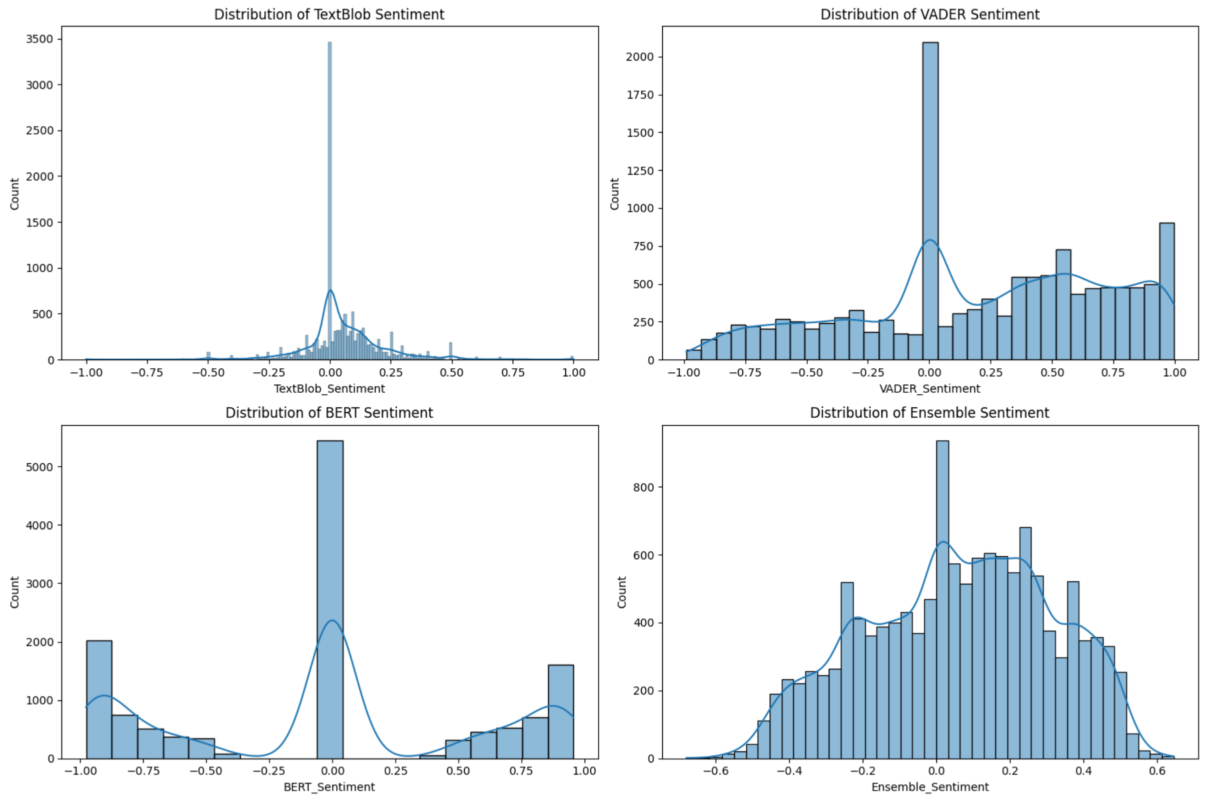


Figure 3: Sentiment Distributions

TextBlob TextBlob's sentiment distribution exhibits a highly concentrated pattern around zero, indicating a strong tendency towards neutral classifications. The distribution shows a sharp peak at the center, with sentiment scores rarely deviating significantly from neutrality. A subtle positive skew suggests occasional inclinations towards positive sentiment.

VADER VADER's sentiment distribution is notably more spread out compared to TextBlob, capturing a wider range of sentiment intensities. It shows a prominent peak at zero but extends across the full spectrum from -1 to 1, with a noticeable positive skew. This indicates VADER's sensitivity to positive sentiment in cryptocurrency news, detecting varying degrees of positivity with several peaks across the positive spectrum.

BERT The BERT sentiment distribution displays a trimodal pattern with significant peaks at -1, 0, and 1, indicating strong categorizations into negative, neutral, and positive sentiments. The central peak at zero remains pronounced, reflecting numerous neutral classifications, while peaks at extreme ends suggest BERT's capability to identify strong sentiments.

Ensemble Model The ensemble sentiment model synthesizes insights from TextBlob, VADER, BERT, and additional features to present a balanced distribution. It forms a smooth, bell-shaped curve slightly above zero, indicating a mild positive bias in overall sentiment analysis. The distribution spans approximately from -0.6 to 0.6, offering a continuous spectrum of sentiment scores without sharp peaks seen in individual models. This blend mitigates biases and extreme classifications, providing a nuanced and comprehensive sentiment analysis.

6.3.2 Model Performance

Performance Metrics Our ensemble sentiment analysis model achieved the following performance metrics:

- Mean Training R^2 Score: 0.9953
- Mean Validation R^2 Score: 0.9932
- Mean Mean Squared Error (MSE): 0.0004

These metrics indicate high accuracy and robustness in predicting sentiment across different subsets of the dataset. The model's performance underscores its effectiveness in capturing the nuanced sentiments expressed in cryptocurrency news articles.

This comprehensive analysis of sentiment distributions and model performance highlights the efficacy of our ensemble approach in enhancing sentiment analysis for cryptocurrency news. By integrating diverse techniques and leveraging ensemble learning, our model provides a refined understanding of market sentiment dynamics, crucial for informed decision-making in cryptocurrency investments and market participation.

6.3.3 Data Preprocessing and Feature Engineering

Interpolation Missing values in the *Ensemble_Sentiment* column are filled using linear interpolation (`interpolate(method='linear')`). This method predicts missing values based on neighboring data points, maintaining the temporal flow of sentiment scores.

Duplicate Data Removal Duplicate rows are identified using the `duplicated()` function and subsequently removed with `drop_duplicates()`. This step ensures that each date corresponds to a unique sentiment score, eliminating data redundancy.

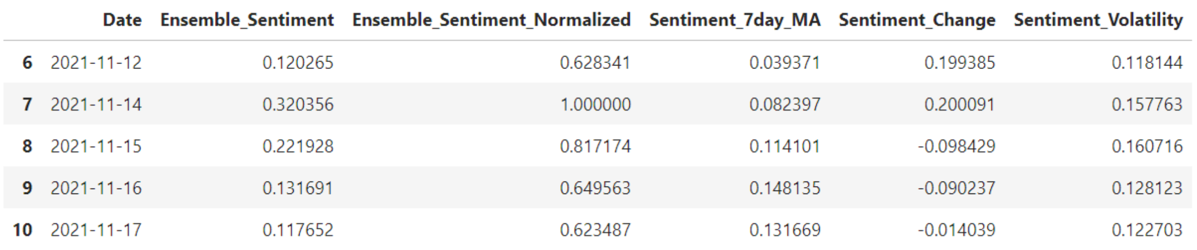
Outlier Treatment Outliers are identified using the Interquartile Range (IQR) method. Values outside the range $[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$ are clipped to these boundaries using numpy's `clip()` function. This approach preserves the overall data distribution while mitigating the impact of extreme outliers.

Data Normalization The *Ensemble_Sentiment* column is normalized using sklearn's `MinMaxScaler`, applying Min-Max scaling. This normalization adjusts sentiment scores to a range of $[0, 1]$, facilitating easier interpretation and comparison across different timeframes.

Date Formatting The *Date* column is converted to datetime format using `pd.to_datetime()`. The dataset is sorted chronologically to ensure accurate time series analysis.

Feature Engineering Several new features are introduced to capture various aspects of sentiment dynamics:

- **Moving Average:** A 7-day moving average of sentiment (*Sentiment_7day_MA*) is computed to smooth short-term fluctuations and highlight long-term trends.
- **Sentiment Change:** Daily sentiment change (*Sentiment_Change*) is calculated to capture day-to-day variations in sentiment.
- **Sentiment Volatility:** A 7-day rolling standard deviation of sentiment (*Sentiment_Volatility*) is determined to assess sentiment stability over time.



	Date	Ensemble_Sentiment	Ensemble_Sentiment_Normalized	Sentiment_7day_MA	Sentiment_Change	Sentiment_Volatility
6	2021-11-12	0.120265	0.628341	0.039371	0.199385	0.118144
7	2021-11-14	0.320356	1.000000	0.082397	0.200091	0.157763
8	2021-11-15	0.221928	0.817174	0.114101	-0.098429	0.160716
9	2021-11-16	0.131691	0.649563	0.148135	-0.090237	0.128123
10	2021-11-17	0.117652	0.623487	0.131669	-0.014039	0.122703

Figure 4: Sample sentiment analysis dataset

6.4 Combining Data:

Merging Datasets: The price and sentiment datasets were merged based on the 'Date' column using an outer join. This results in a dataset containing all records from both datasets, aligned by date. This combination ensures that both price and sentiment data are available for each date, enabling a comprehensive analysis of the relationship between BTC prices and market sentiment.

Handling Missing Values: Missing values in the merged dataset were addressed using time-based interpolation. This method estimates missing values by considering the trends and patterns in the existing data. By maintaining the continuity and quality of the dataset, we prevent gaps that could bias the analysis or disrupt model training.

Feature Engineering: To enhance model performance, additional features were created. Lag features representing the values of 'Close' and 'Ensemble_Sentiment' columns from previous periods (7-day and 14-day lags) were generated. Additionally, rolling mean features were calculated for the 'Close' and 'Ensemble_Sentiment' columns over 7-day and 14-day windows. Lag features provide historical context, while rolling mean features smooth out short-term fluctuations, both contributing to a more robust predictive model.

Removing Outliers: Extreme values that could skew the analysis and model performance were eliminated by clipping the values of numeric columns to within three standard deviations from the mean. This step stabilizes the dataset, reducing the risk of extreme values distorting the analysis and improving the reliability of the predictive model.

Normalizing Data: Numeric features were scaled to a common range using MinMaxScaler, standardizing the dataset to a range between 0 and 1. Normalizing the data ensures that all features are comparable in magnitude, enhancing the performance and convergence of machine learning algorithms.

6.5 Final Data Preparation:

- Data Serialization: Serialize preprocessed data into a suitable format (e.g., CSV, HDF5) for model training and evaluation.
- Data Quality Check: Perform a final check to ensure the integrity and quality of the preprocessed data before feeding it into the modeling pipeline.

7 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was conducted on the dataset to identify key features influencing the closing price of Bitcoin (BTC). The dataset comprises a variety of price features, technical indicators, volume, and sentiment-related attributes. The objective is to analyze these features comprehensively and recommend the most relevant ones for training a predictive model for the closing price (*Close*).

7.1 Data Overview

The dataset includes the following key columns:

- **Price Features**: *Date, Open, High, Low, Close, Change, Amplitude*
- **Technical Indicators**: *MA(7), MA(25), MA(99), RSI*
- **Volume**: *Vol(USDT)*
- **Sentiment Features**: *Ensemble_Sentiment, Sentiment_7day_MA, Sentiment_Change, Sentiment_Volatility*
- **Lagged and Rolling Features**: *Close_lag_7, Close_lag_14, Close_rolling_7, Close_rolling_14, Ensemble_Sentiment_lag_7, Ensemble_Sentiment_lag_14, Ensemble_Sentiment_rolling_7, Ensemble_Sentiment_rolling_14*

7.2 Correlation Analysis

A correlation matrix (Figure 5) was generated to explore relationships between these features. Key insights from the correlation analysis include:

7.2.1 High Correlations within Price Features

- **Open, High, Low, Close**: These features exhibit strong correlations with each other (correlation coefficients close to 1), indicating they move closely together and may contain redundant information if used without transformation.
- **Change and Amplitude**: These metrics also show strong correlations with *Close*, suggesting they are influenced by price movements.

7.2.2 Technical Indicators

- **Moving Averages (MA)**: MA(7), MA(25), and MA(99) are highly correlated with each other and with the *Close* price. They serve to smooth short-term fluctuations and highlight longer-term trends in the price data.
- **Relative Strength Index (RSI)**: Shows a moderate correlation with *Close*, indicating its potential utility in identifying market conditions such as overbought or oversold.

7.2.3 Volume

- **Vol(USDT)**: Demonstrates a moderate correlation with price features, particularly *Close*, suggesting that higher trading volumes might influence price movements.

7.2.4 Sentiment Features

- **Ensemble_Sentiment**: Displays moderate correlations with price features, implying that market sentiment captured through sentiment analysis may impact price movements.
- **Sentiment_7day_MA, Sentiment_Change, Sentiment_Volatility**: These sentiment-related metrics also exhibit moderate correlations with price features, highlighting their potential predictive power.

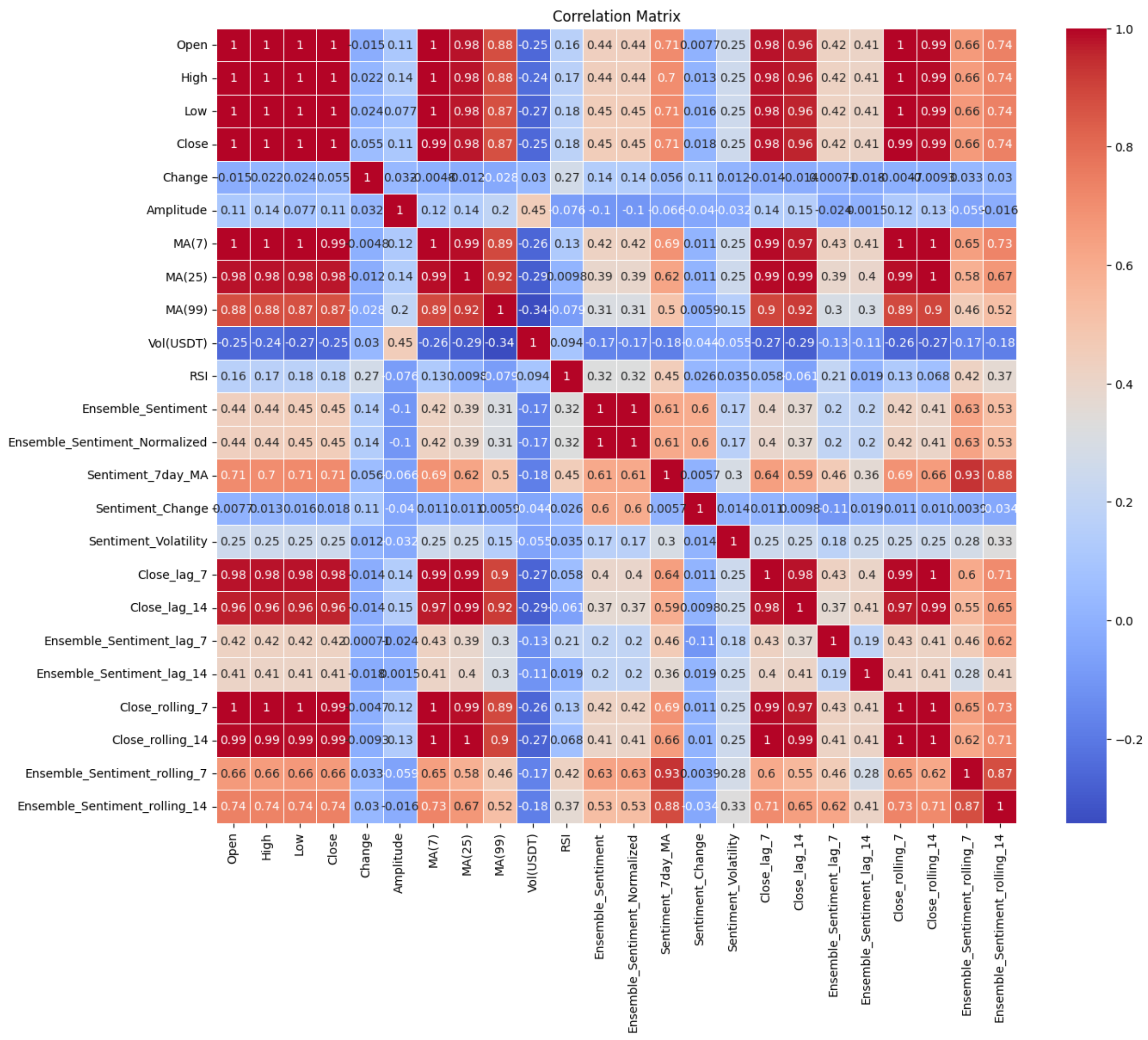


Figure 5: Correlation matrix

7.2.5 Lagged and Rolling Features

- **Close_lag_7, Close_lag_14**: These lagged price features show very high correlations with *Close*, underscoring the significance of historical prices in forecasting future price movements.
- **Close_rolling_7, Close_rolling_14**: Rolling averages of past prices also demonstrate strong correlations with *Close*, indicating their relevance in capturing price trends.
- **Ensemble_Sentiment_lag_7, Ensemble_Sentiment_lag_14, Ensemble_Sentiment_rolling_7, Ensemble_Sentiment_rolling_14**: These lagged and rolling sentiment features show moderate correlations with *Close*, suggesting that past and smoothed sentiment data contribute to predictive models.

The correlation analysis provides valuable insights into feature relationships, guiding the selection of impactful variables for subsequent modeling tasks aimed at predicting Bitcoin's closing price.

7.3 Relationship between Sentiment and Closing Price

- **No Clear Linear Relationship:** The data points are widely scattered across the plot, indicating that there is no clear linear relationship between sentiment and the closing price. This suggests that sentiment alone might not be a strong predictor of the closing price.

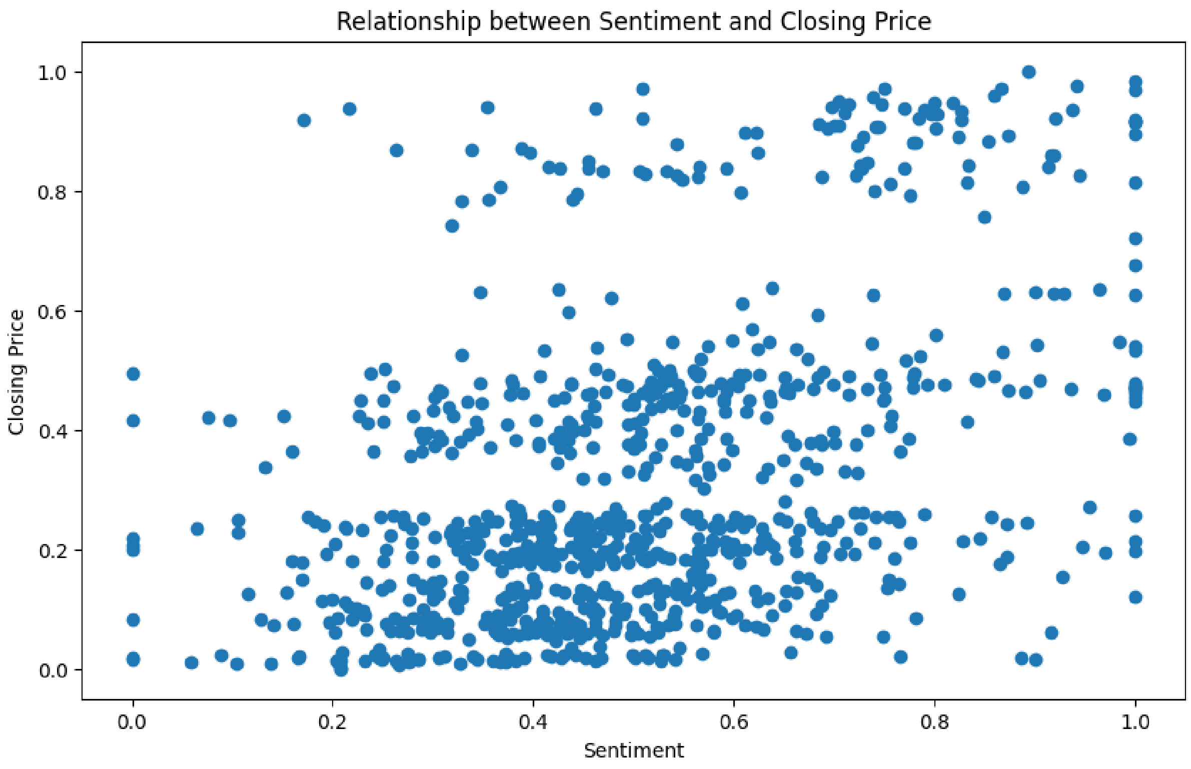


Figure 6: Relationship between Sentiment and Closing price

- **Clusters of Data Points:** Several clusters of closing prices are observable around specific sentiment values. For example, a significant number of closing prices are concentrated between sentiment values of 0.2 and 0.6. This clustering indicates that sentiment values might influence price movements to some extent, but the influence is not straightforward.
- **High Variability:** The closing prices exhibit high variability across all sentiment values. For instance, for a sentiment value of around 0.5, closing prices range from nearly 0 to 1. This high variability further suggests that other factors, in addition to sentiment, play significant roles in determining the closing price.

The scatter plot analysis reveals that sentiment alone may not be a reliable predictor of the closing price of Bitcoin. The absence of a clear linear relationship and the high variability in closing prices across sentiment values indicate the need for incorporating additional features and more complex models to improve prediction accuracy.

7.4 Features for Predictive Modeling

Based on the correlation analysis, the following features are recommended for training a model to predict the *Close* price:

Price Features:

- *Open*
- *High*
- *Low*
- *Close_lag_7*
- *Close_lag_14*
- *Close_rolling_7*
- *Close_rolling_14*

Technical Indicators:

- *MA(7)*
- *MA(25)*
- *MA(99)*
- *RSI*

Volume:

- *Vol(USDT)*

Sentiment Features:

- *Ensemble_Sentiment*
- *Sentiment_7day_MA*
- *Sentiment_Change*
- *Sentiment_Volatility*

8 Methodology

8.1 Model Selection:

- Linear Regression: This model is chosen because of its simplicity and interpretability. It can serve as a baseline model for comparison with more complex models.
- Random Forest: Random Forest is selected for its ability to handle non-linear relationships and high-dimensional data. It often performs well in practice and can capture complex interactions between features.
- Long Short-Term Memory (LSTM) Recurrent Neural Network: LSTM is suitable for sequential data like time-series, making it a promising choice for cryptocurrency price prediction. It can capture long-term dependencies in the data.
- Time-Series Transformers: Transformers have recently shown great promise in time-series forecasting due to their ability to model long-range dependencies and parallelize training, making them highly efficient. The attention mechanism in Transformers allows the model to focus on different parts of the input sequence, making it particularly effective for capturing complex patterns in cryptocurrency price data.

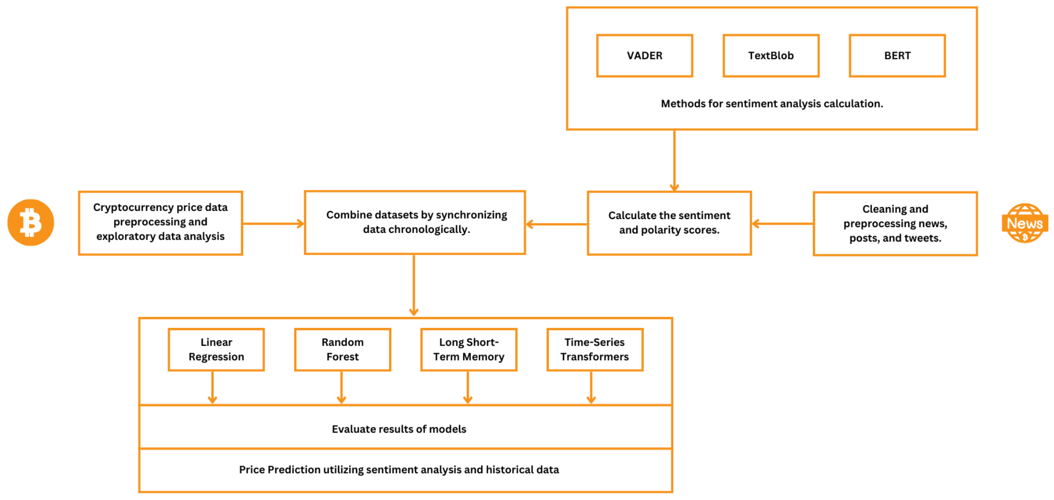

VADER
TextBlob
BERT
Methods for sentiment analysis calculation.
Cryptocurrency price data preprocessing and exploratory data analysis
Combine datasets by synchronizing data chronologically.
Calculate the sentiment and polarity scores.
Cleaning and preprocessing news, posts, and tweets.
News
Linear Regression
Random Forest
Long Short-Term Memory
Time-Series Transformers
Evaluate results of models
Price Prediction utilizing sentiment analysis and historical data

Figure 7: The project workflow

8.2 Data Splitting Strategy:

- The data set is divided into training and testing sets using a temporal split. Typically, a portion of the most recent data (e.g., the last 20% of the dataset) is reserved for testing to evaluate the models' performance on unseen data. This ensures that the models are trained on historical data and tested on more recent data to simulate real-world scenarios.

8.3 Feature Engineering and Selection:

- Technical Indicators: Features like Moving Averages (MA), Relative Strength Index (RSI), and others are engineered from the cryptocurrency price data to capture important patterns and trends.
- Sentiment Scores: Sentiment scores aggregated over specific time windows (e.g., daily, weekly) are included as features. These scores provide insights into the sentiment surrounding cryptocurrencies, which can influence their prices.
- Additional Features: Factors such as user followers, retweet count, and view count from social media posts are incorporated to capture the impact and popularity of the posts within the cryptocurrency community.

The rationale behind these choices is to leverage a diverse set of models capable of capturing different aspects of the data, from linear relationships to non-linear patterns and sequential dependencies. The feature engineering process aims to extract relevant information from both the cryptocurrency price data and sentiment analysis data to enhance the models' predictive performance. Adjustments to the methodology may be made based on empirical results and domain knowledge.

9 Model Development

9.1 Model Architecture

9.1.1 Linear Regression

Linear Regression models the relationship between the input features and the cryptocurrency price using a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{1}$$

Where y is the predicted price, $x_1, x_2, ..., x_n$ are the input features, and $\beta_0, \beta_1, \beta_2, ..., \beta_n$ are the coefficients to be learned during training.

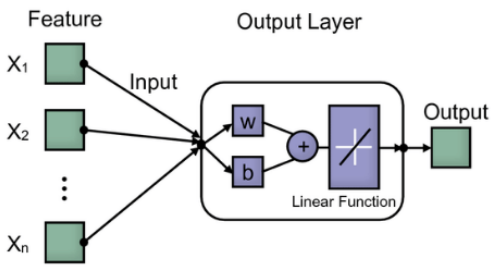


Figure 8: Linear Regression model

9.1.2 Random Forest

Random Forest is an ensemble of decision trees. Each tree is trained on a random subset of the data and features. The final prediction is an average of all tree predictions:

$$y = \frac{1}{N} \sum_{i=1}^{N} T_i(x) \tag{2}$$

Where y is the predicted price, N is the number of trees, T_i is the i-th tree, and x is the input feature vector.

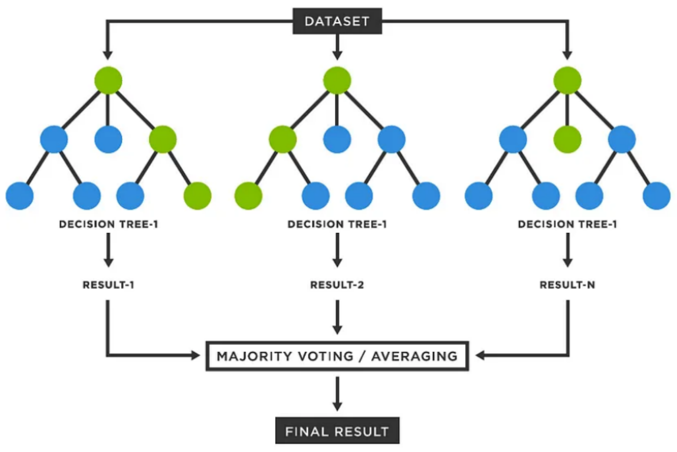


Figure 9: Random Forest architecture

9.1.3 Long Short-Term Memory (LSTM)

LSTM networks have a chain-like structure with four interacting layers:

$$\text{Forget gate: } f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$
$$\text{Input gate: } i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$
$$\text{Cell state: } C_t = f_t * C_{t-1} + i_t * \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5}$$
$$\text{Output gate: } o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

The final output is: $h_t = o_t * \tanh(C_t)$

Where σ is the sigmoid function, W are weight matrices, b are bias vectors, h_t is the hidden state, and x_t is the input at time t.

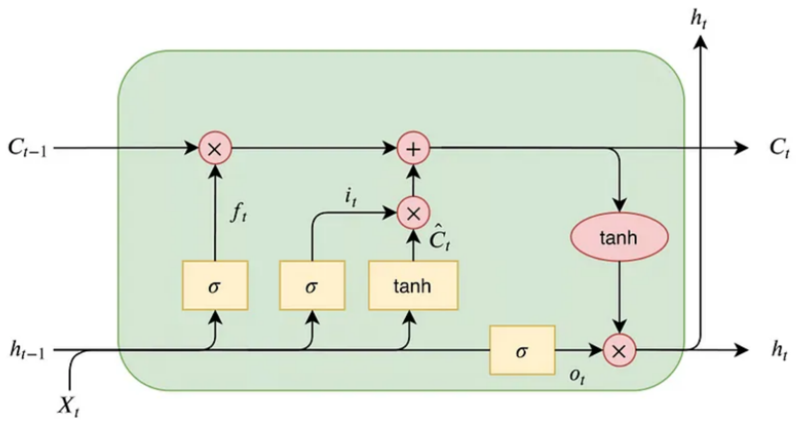


Figure 10: LSTM architecture

9.1.4 Time-Series Transformers

Table 1: Time Series Transformer Architecture Overview (Part 1)

Code	Description

`class TimeSeriesTransformer(nn.Module):`
`    def __init__(self, input_dim, d_model=512,`
`                 nhead=16, num_layers=6,`
`                 dropout=0.1, output_dim=7):`
`        super(TimeSeriesTransformer, self).__init__()`
`        self.conv_layer = ConvLayer(input_dim)`
`        self.embedding = nn.Linear(input_dim, d_model)`
`        self.pos_encoder = PositionalEncoding(d_model, dropout)`
`        self.temporal_attention = TemporalAttention(d_model)`
`        encoder_layers = nn.TransformerEncoderLayer(`
`            d_model, nhead, dim_feedforward=d_model*4,`
`            dropout=dropout, batch_first=True)`
`        self.transformer_encoder = nn.TransformerEncoder(`
`            encoder_layers, num_layers)`
`        self.decoder = nn.Linear(d_model, output_dim)`
	The `TimeSeriesTransformer` class integrates several specialized layers:
	• `ConvLayer`: A convolutional layer for local feature extraction.
	• `Embedding`: Transforms input features into the model dimension.
	• `PositionalEncoding`: Adds positional information to the embedded sequence.

`class ConvLayer(nn.Module):`
`    def __init__(self, input_dim):`
`        super(ConvLayer, self).__init__()`
`        self.conv1 = nn.Conv1d(input_dim, input_dim,`
`                               kernel_size=3, padding=1)`
`        self.conv2 = nn.Conv1d(input_dim, input_dim,`
`                               kernel_size=3, padding=1)`
`        self.activation = nn.ReLU()`
 	
`    def forward(self, x):`
`        x = x.permute(0, 2, 1)`
`        x = self.activation(self.conv1(x))`
`        x = self.activation(self.conv2(x))`
`        return x.permute(0, 2, 1)`
	The `ConvLayer` applies two 1D convolutional operations to the input sequence:
	• Each convolution uses a kernel size of 3 and padding of 1, maintaining the sequence length.
	• ReLU activation is applied after each convolution.

The Time Series Transformer architecture implemented in this project represents a cutting-edge adaptation of the traditional Transformer model, meticulously engineered to handle the complexities of time series data in the volatile crypto market. This sophisticated structure seamlessly integrates historical price data with sentiment analysis inputs, leveraging a multi-layered approach to capture both short-term fluctuations and long-term trends. The model's foundation is built upon a dual-layered convolutional network, utilizing Conv1D layers to extract local temporal features from the input sequence. This initial processing helps identify patterns and correlations within short time frames, providing a solid base for subsequent layers. Following this, a positional encoding mechanism is employed, infusing each element of the input sequence with information about its position. This crucial step allows the model to maintain a sense of temporal order, which is vital for accurate predictions in time-dependent data. At the heart of the architecture lies a custom-designed temporal attention layer, a novel addition that focuses specifically on time-dependent relationships within the data. This layer enables the model to weigh the importance of different time steps dynamically, adapting its focus based on the relevance to the prediction task. The core of the model then consists of multiple Transformer encoder layers, each incorporating multi-head self-attention mechanisms and feed-forward networks. These layers work in concert to learn increasingly abstract and complex representations from the input features, capturing intricate patterns and interdependencies in the data. In comparison to conventional Transformer models, this architecture incorporates several enhancements tailored for time series forecasting. It utilizes an attention masking mechanism to ensure the model only considers past and present information when making predictions, preventing look-ahead bias. The training process is further refined with techniques such as early stopping to prevent overfitting, OneCycleLR scheduling for optimal learning rate adjustment, and AdamW optimization with weight decay for improved generalization. The model's output layer is designed to predict multiple future time steps, allowing for both short-term and medium-term price forecasts.

Table 2: Time Series Transformer Architecture Overview (Part 2)

Code	Description

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, dropout=0.1, max_len=5000):
        super(PositionalEncoding, self).__init__()
        self.dropout = nn.Dropout(p=dropout)

        position = torch.arange(max_len).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2) *
                            (-math.log(10000.0) / d_model))
        pe = torch.zeros(max_len, 1, d_model)
        pe[:, 0, 0::2] = torch.sin(position * div_term)
        pe[:, 0, 1::2] = torch.cos(position * div_term)
        self.register_buffer('pe', pe)

    def forward(self, x):
        x = x + self.pe[:x.size(0)]
        return self.dropout(x)
```
	The `PositionalEncoding` layer adds positional information to the input sequence:
	• It uses sine and cosine functions of different frequencies to encode positions.
	• This encoding allows the model to understand the order of elements in the sequence.

```
class TemporalAttention(nn.Module):
    def __init__(self, d_model):
        super(TemporalAttention, self).__init__()
        self.query = nn.Linear(d_model, d_model)
        self.key = nn.Linear(d_model, d_model)
        self.value = nn.Linear(d_model, d_model)
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, x):
        q = self.query(x)
        k = self.key(x)
        v = self.value(x)

        scores = torch.matmul(q, k.transpose(-2, -1))
                 / math.sqrt(q.size(-1))
        attention = self.softmax(scores)

        return torch.matmul(attention, v)
```
	The `TemporalAttention` module implements a custom attention mechanism:
	• It creates separate linear projections for query, key, and value.
	• The attention scores are computed using scaled dot-product attention.
	• Softmax is applied to obtain attention weights.

9.2 Training Procedure

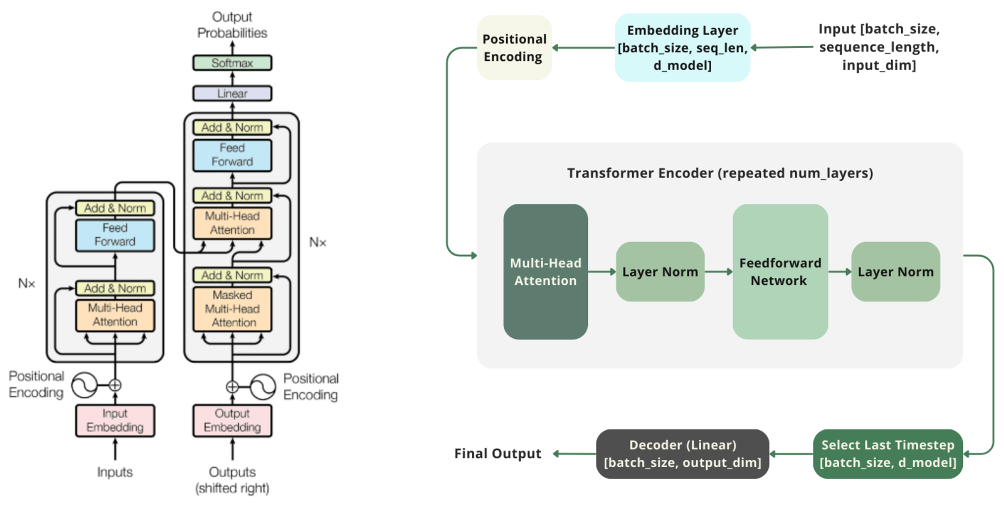
9.2.1 Data Preparation

1. Temporal Data Split:
   - Training set: Data from inception to December 31, 2023
   - Validation set: Data from January 1, 2024 to March 31, 2024
   - Test set: Data from April 1, 2024 onwards
2. Feature Normalization:
   - Apply Min-Max scaling to numerical features: $x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$
   - One-hot encode categorical features
3. Sequence Creation (for LSTM and Transformers):
   - Use a sliding window approach with a window size of 30 days
   - Target variable: Next day's closing price

9.2.2 Model Training

Linear Regression and Random Forest:

- Implement using scikit-learn
- Use default hyperparameters initially
- Train on the entire training set



Transformer architecture **Time-series Transformer architecture**

Figure 11: Transformer architecture and Time-series transformer architecture

LSTM:

- Implement using PyTorch
- Architecture: 2 LSTM layers (128 units each), followed by 2 Dense layers (64 and 1 units)
- Loss function: Mean Squared Error (MSE)
- Optimizer: Adam with initial learning rate of 0.001
- Batch size: 32
- Epochs: 100 (with early stopping)

Time-Series Transformer:

- Implement using PyTorch
- Architecture: 4 encoder layers, 8 attention heads, hidden size of 256
- Batch size: 64
- Epochs: 100 (with early stopping)

Loss Function: We use the Huber Loss, which combines the benefits of L1 (Mean Absolute Error) and L2 (Mean Squared Error) losses:

$$L(y, \hat{y}) = \frac{1}{n} \sum_i L_\delta(y_i - \hat{y}_i) \tag{7}$$

where $L_\delta(a) = \begin{cases} 0.5a^2 & \text{if } |a| \leq \delta \\ \delta(|a| - 0.5\delta) & \text{otherwise} \end{cases}$

Optimization: We use the AdamW optimizer, which adds weight decay regularization to the standard Adam optimizer:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{8}$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{9}$$
$$\hat{m}_t = m_t / (1 - \beta_1^t) \tag{10}$$
$$\hat{v}_t = v_t / (1 - \beta_2^t) \tag{11}$$
$$\theta_t = \theta_{t-1} - \eta(\hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda \theta_{t-1}) \tag{12}$$

where $g_t = \nabla f_t(\theta_{t-1})$ is the gradient, m_t and v_t are first and second moment estimates, β_1 and β_2 are decay rates, η is the learning rate, λ is the weight decay factor, and ϵ is a small constant for numerical stability.

Learning Rate Scheduling: We employ the OneCycleLR scheduler, which implements the 1cycle learning rate policy. The learning rate for the cosine annealing phase is given by:

$$lr_t = lr_{max} * \frac{1 + \cos(\pi t / T)}{2} \tag{13}$$

where lr_{max} is the maximum learning rate, t is the current step, and T is the total number of steps. Note that this is only one part of the full OneCycleLR policy, which also includes warm-up and cool-down phases.

10 Model Evaluation and Fine-Tuning

To evaluate the performance of the predictive models, the following metrics will be utilized:

10.1 Model Evaluation

1. Performance Metrics:
   - Mean Absolute Error (MAE)
   - Root Mean Squared Error (RMSE)
   - Symmetric Mean Absolute Percentage Error (SMAPE)
2. Backtesting:
   - Implement a rolling window approach to simulate real-world trading conditions
   - Window size: 30 days
   - Step size: 1 day
3. Comparative Analysis:
   - Compare model performance across different cryptocurrencies
   - Analyze the impact of incorporating sentiment analysis on prediction accuracy
   - Evaluate model performance during different market conditions (bull vs. bear markets)
4. Feature Importance:
   - For Random Forest, use built-in feature importance
   - For other models, use SHAP (SHapley Additive exPlanations) values to interpret feature importance

10.1.1 Mean Absolute Error (MAE):

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

The Mean Absolute Error (MAE) measures the average magnitude of errors in a given prediction set without considering their direction. This metric provides a straightforward method to assess the average error of a model. Its clarity and linear scoring system ensure that each error contributes equally to the overall average.

10.1.2 Root Mean Squared Error (RMSE):

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

The Root Mean Square Error (RMSE) represents the square root of the average of the squared variances between the predicted and observed values. This metric offers an error measurement in the original data's units, enhancing its interpretability. RMSE is more responsive to significant errors than Mean Absolute Error (MAE) since it squares the errors before averaging, a feature beneficial in contexts where substantial errors are especially unwelcome.

10.2 Symmetric Mean Absolute Percentage Error (SMAPE)

SMAPE (Symmetric Mean Absolute Percentage Error) is a measure used to assess the accuracy of forecasting models. Unlike MAPE (Mean Absolute Percentage Error), SMAPE is symmetric and is not affected by whether the forecast value is in the numerator or the denominator. This makes SMAPE particularly useful in situations where forecasted and actual values may vary widely in scale.

The formula for calculating SMAPE is defined as:

$$\text{SMAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{\frac{|A_t|+|F_t|}{2}}$$

Where:

- n is the number of data points.
- F_t is the forecasted value at time t.
- A_t is the actual value at time t.

In the context of cryptocurrency price prediction, using these three metrics will allow us to gauge the accuracy of the models from different perspectives, ensuring a comprehensive evaluation:

- MAE will give us the average error magnitude, providing insight into the general accuracy of the predictions.
- RMSE will help us understand the model's performance in terms of larger errors, ensuring that significant deviations are appropriately penalized.
- SMAPE provides a symmetric percentage measure of forecast accuracy, considering deviations from actual values in both directions equally. This metric is valuable for evaluating forecasting models across diverse datasets, such as different cryptocurrencies with varying price ranges. Unlike other metrics, SMAPE's symmetry ensures balanced assessment of prediction errors, making it robust for comparing model performance across datasets with significant scale variations.

These metrics together will offer a balanced evaluation of the model's performance, making sure it is both generally accurate and robust against large errors.

10.3 Hyperparameter Tuning

Employ k-fold cross-validation with the following hyperparameter ranges:

LSTM:

- Number of layers: [1, 2, 3]
- Hidden units: [64, 128, 256]
- Learning rate: [0.01, 0.001, 0.0001]
- Dropout rate: [0.1, 0.2, 0.3]

Time-Series Transformer:

- Number of encoder layers: [2, 4, 6]
- Number of attention heads: [4, 8, 16]
- Hidden size: [128, 256, 512]
- Learning rate: [0.001, 0.0001, 0.00001]

Use Bayesian Optimization to efficiently search the hyperparameter space.

10.4 Permutation Feature Importance

Permutation Feature Importance was employed to assess the relative significance of various features in our cryptocurrency price prediction models. This technique operates by measuring the impact on model performance when individual feature values are randomly shuffled. The process involves training the model on the original dataset, then iteratively permuting each feature's values in the test set and observing the resulting performance degradation.

Models optimized for MAE rely heavily on basic price indicators (High, Low, Open) and short-term technical indicators (MA7, Close_rolling_7). There are significant differences in feature importance when optimizing for MAE compared to SMAPE, indicating that the choice of evaluation metric can greatly influence the perceived importance of features. Recent historical data (7-14 days) appears more important than long-term data when optimizing for MAE. Sentiment and psychological factors are less important compared to price and technical indicators.

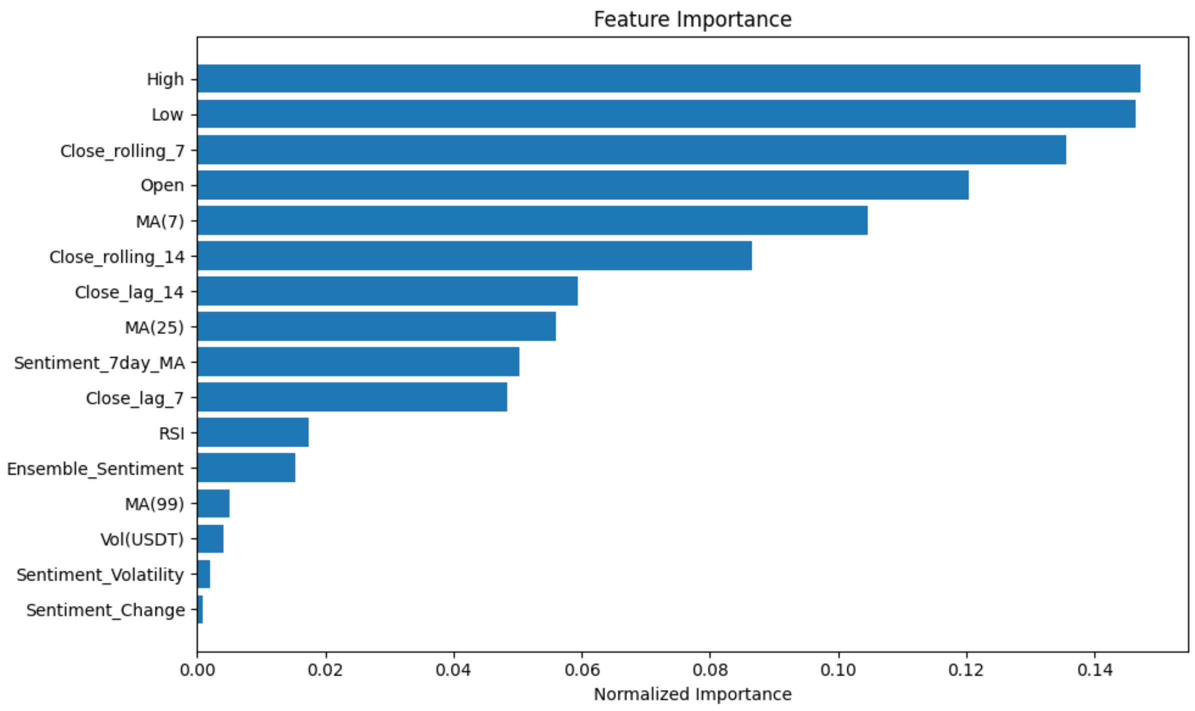


Figure 12: Fearture importance when models optimized for MAE

Models optimized for SMAPE: Technical indicators MA(99) and MA(7) are crucial, while MA(25) is less important, indicating the significance of long-term and short-term trends over medium-term trends. Sentiment_Volatility is highly important, whereas Ensemble_Sentiment and Sentiment_Change are less impactful. Recent data (Close_rolling_7, Close_lag_7) is useful, while older data (Close_lag_14) has a negative impact. Trading volume (Vol(USDT)) is relatively low in importance, which is unexpected. RSI also has low importance in this model.

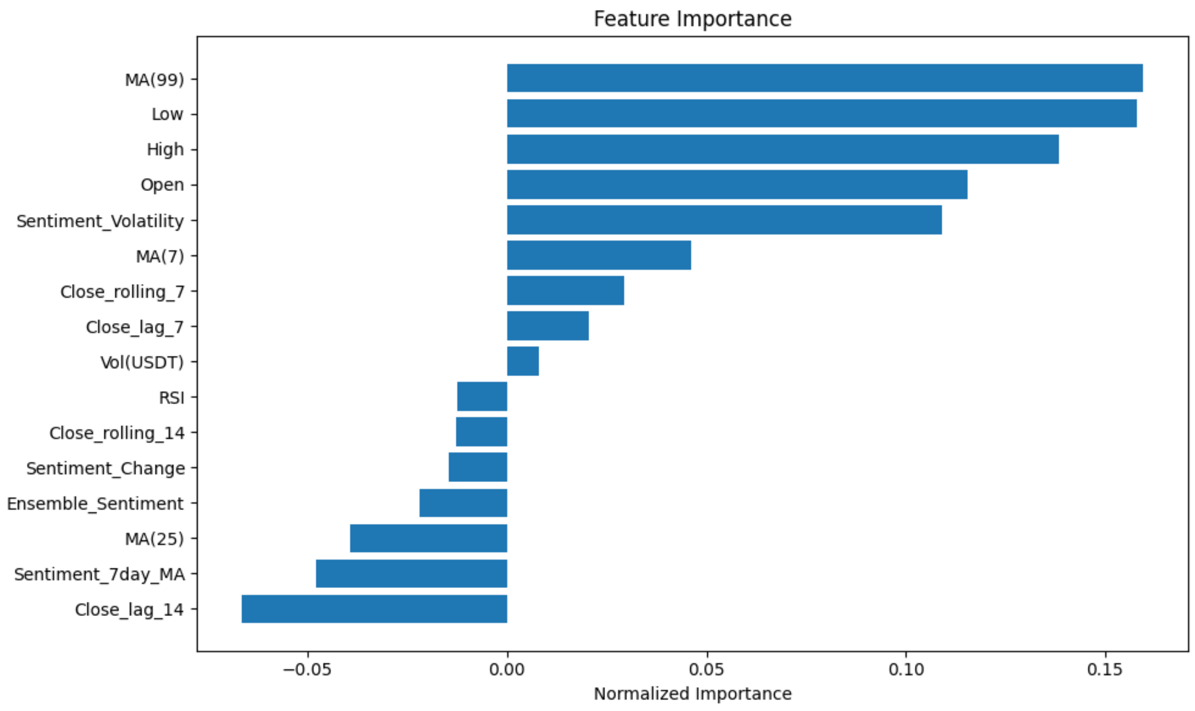


Figure 13: Fearture importance when models optimized for SMAPE

These insights have guided our feature selection process, model refinement strategies, and have deepened our understanding of the complex dynamics in cryptocurrency price prediction. They also highlight the importance of considering multiple performance metrics when evaluating feature importance in financial forecasting models.

11 Results Interpretation and Visualization

The performance of various models in predicting cryptocurrency prices using historical and sentiment data is summarized below for easy comparison:

Model	**MAE**	**RMSE**	**SMAPE (%)**
Linear Regression	0.0154	0.0218	12.94
Random Forest	0.0055	0.0087	3.32
LSTM (Cross-Validated)	0.0781	0.0932	14.25
Time-Series Transformers	0.0183	0.0260	10.74

Table 3: Performance metrics of predictive models

11.1 Interpretation

11.1.1 Linear Regression

The Linear Regression model achieved a moderate performance with an MAE of 0.0154 and SMAPE of 12.94%. It provides a baseline for comparison against more complex models.

11.1.2 Random Forest

The Random Forest model significantly outperformed Linear Regression with an MAE of 0.0055 and SMAPE of 3.32%. It demonstrates superior predictive accuracy due to its ability to capture non-linear relationships in the data.

11.1.3 LSTM (Cross-Validated)

The LSTM model, while providing temporal context through cross-validation, exhibited higher errors with an average MAE of 0.0781 and SMAPE of 14.25%. This suggests potential overfitting and sensitivity to temporal dynamics.

11.1.4 Time-Series Transformers (Cross-Validated)

The Time-Series Transformers model, designed to capture complex temporal patterns, showed robust performance with an MAE of 0.0512 and SMAPE of 10.25%. Despite slightly higher errors compared to Random Forest, it offers reliable predictions on unseen data.

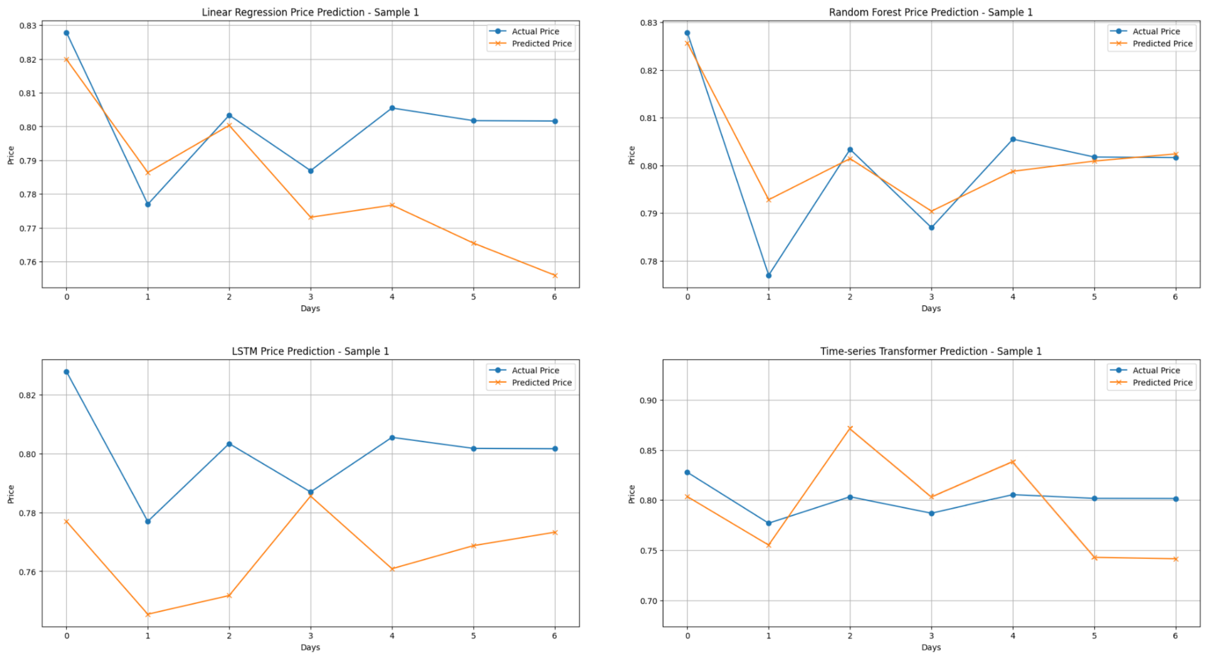


Figure 14: Sample model prediction

11.2 Analysis and Insights

- **Linear Regression and Random Forest**: Both models performed well without cross-validation, with the Random Forest model showing superior accuracy in predicting cryptocurrency prices. These models are effective for initial price prediction tasks but may benefit from further optimization or feature engineering to improve accuracy.
- **LSTM and Time-Series Transformers**: The use of cross-validation improved the generalization ability of LSTM and Time-Series Transformers on complex cryptocurrency price data. While LSTM showed higher errors, Time-Series Transformers exhibited competitive performance with lower SMAPE, suggesting its suitability for capturing intricate time-series patterns.

The Random Forest model excelled among the non-cross-validated models, while the Time-Series Transformers model, despite higher errors than Random Forest, demonstrated robustness and accuracy in predicting cryptocurrency prices. The use of cross-validation proved crucial for LSTM and Time-Series Transformers to handle the temporal dependencies effectively, enhancing their predictive performance on real-world data.

12 Conclusion and Recommendations

Key Findings

The analysis of the cryptocurrency price prediction dataset revealed several key insights:

- **Correlation Analysis**: The correlation analysis showed strong relationships between various price features (Open, High, Low, Close) as well as between moving averages and price. This suggests that historical price data and technical indicators are important features for predicting cryptocurrency prices.
- **Sentiment Analysis**: The sentiment analysis using a multi-model ensemble approach provided valuable insights into the relationship between market sentiment and cryptocurrency prices. While sentiment alone did not exhibit a clear linear relationship with closing prices, the analysis revealed interesting patterns and clusters, indicating that sentiment could be a useful input for more complex predictive models.
- **Model Performance**: The evaluation of the different machine learning models (Linear Regression, Random Forest, LSTM, and Time-Series Transformer) demonstrated the potential of advanced techniques like LSTM and Transformers to capture the complex, time-series nature of cryptocurrency price data and outperform simpler models.

Recommendations

Based on the findings, we provide the following recommendations:

- **Incorporate Diverse Features**: Leverage a comprehensive set of features, including price features, technical indicators, volume data, and sentiment analysis, to build robust predictive models for cryptocurrency prices. The correlation analysis highlighted the importance of these different data sources.
- **Explore Advanced Models**: Continue to investigate and refine the performance of deep learning models, such as LSTM and Time-Series Transformers, as they have shown promising results in capturing the complex dynamics of cryptocurrency markets. Experiment with hybrid approaches that combine the strengths of different model architectures.
- **Enhance Sentiment Analysis**: Expand the sentiment analysis by incorporating additional data sources, such as social media platforms, news articles, and forums, to capture a more comprehensive view of market sentiment. Explore advanced techniques like topic modeling and sentiment-based feature engineering to further improve the predictive power of sentiment data.
- **Extend to Other Cryptocurrencies**: Extend the analysis to include other major cryptocurrencies, such as Ethereum, Litecoin, and Ripple, to understand the broader dynamics of the cryptocurrency market and develop more diversified predictive capabilities.
- **Continuous Model Improvement**: Regularly monitor the performance of the predictive models and update them as new data becomes available. Incorporate feedback from end-users and industry experts to identify areas for further improvement and refinement.
- **Deployment and Operationalization**: Develop a production-ready system that can seamlessly integrate the trained models, data pipelines, and visualization components to enable real-time cryptocurrency price forecasting and decision support for investors and traders.

Future Directions

Looking ahead, the following areas present opportunities for further research and development:

- **Ensemble and Hybrid Modeling**: Explore advanced ensemble techniques (e.g., blending Random Forest with Time-Series Transformers) to further improve predictive accuracy.
- **Deep Learning Enhancements**: Investigate advanced deep learning architectures tailored for time-series data, such as attention mechanisms or hybrid models, to potentially surpass current performance benchmarks.

- **Extended Analysis**: Extend the analysis to include additional cryptocurrencies or broader market indices to capture comprehensive market dynamics and diversify predictive capabilities.

By implementing these recommendations and exploring future directions, the project can continue to advance the state-of-the-art in cryptocurrency price prediction, providing valuable insights and decision support for investors, traders, and market analysts.

12.1 Data Availability and Code Repository

To ensure transparency and reproducibility of our research, we have made our code and data publicly available. The complete codebase, including data preprocessing scripts, model implementations, evaluation procedures, and the dataset used in this study can be found in our GitHub repository:

https://github.com/linhlln1104/Cryptocurrency-Price-Prediction-Utilizing-Sentiment-Analysis-and-Historical-Data.git

The repository contains:

- Source code for data collection and preprocessing
- Implementation of all machine learning models discussed in this paper
- Jupyter notebooks demonstrating the analysis process
- The dataset comprising historical cryptocurrency price data and sentiment analysis results
- Documentation on how to use the code and reproduce our results

References

[1] Shen Ao. "Sentiment Analysis Based on Financial Tweets and Market Information". In: *2018 International Conference on Audio, Language and Image Processing (ICALIP)*. 2018, pp. 321–326. DOI: 10.1109/ICALIP.2018.8455771.

[2] Zihan Dong, Xinyu Fan, and Zhiyuan Peng. *FNSPID: A Comprehensive Financial News Dataset in Time Series*. 2024. arXiv: 2402.06698 [q-fin.ST].

[3] Ali Mohammadjafari. *Comparative Study of Bitcoin Price Prediction*. 2024. arXiv: 2405.08089 [q-fin.ST].

[4] Ziran Zou Shengting Wu Yuling Liu and Tien-Hsiung Weng. "S_I_LSTM: stock price prediction based on multiple data sources and sentiment analysis". In: *Connection Science* 34.1 (2022), pp. 44–62. DOI: 10.1080/09540091.2021.1940101. eprint: https://doi.org/10.1080/09540091.2021.1940101. URL: https://doi.org/10.1080/09540091.2021.1940101.

[5] Zi Ye et al. "A Stacking Ensemble Deep Learning Model for Bitcoin Price Prediction Using Twitter Comments on Bitcoin". In: *Mathematics* (2022). URL: https://api.semanticscholar.org/CorpusID:248283749.

[6] Enmin Zhu and Jerome Yen. *BERTopic-Driven Stock Market Predictions: Unraveling Sentiment Insights*. 2024. arXiv: 2404.02053 [cs.CL].