Time Series Analysis with Crypto currency

By: Mohitha Bandi

Email: mohitha12026@gmail.com

1. Introduction

As part of my internship, I undertook a project focused on analyzing and forecasting cryptocurrency prices, specifically Bitcoin (BTC), using time series techniques. The project involved collecting data from public APIs, cleaning and processing it, performing in-depth analysis, building predictive models, and creating an interactive dashboard to visualize results. This report details my work, highlighting the technical skills I developed, the challenges I faced, and the insights I gained. The project aimed to provide actionable insights into cryptocurrency market trends and demonstrate the application of data science in financial analysis.

2. Project Objectives

The main goals of this project were:

- Data Collection: Gather historical Bitcoin price data from reliable sources like Binance and CoinGecko APIs.
- Data Preprocessing : Clean, normalize, and enrich the data with features to make it suitable for analysis.
- Time Series Analysis : Explore price trends, volatility patterns, and feature correlations to understand market dynamics.
- Price Forecasting: Develop and compare various forecasting models to predict future Bitcoin prices.
- Interactive Visualization: Build a user-friendly dashboard to display price trends, technical indicators, forecasts, and risk assessments.
- Learning and Skill Development: Enhance my proficiency in Python, data analysis, time series modeling, and dashboard development.

3. Methodology

I used Python as the primary programming language, leveraging libraries such as pandas, numpy, matplotlib, seaborn, scikit-learn, TensorFlow, statsmodels, Prophet, PyTorch, and Dash. The project was divided into several phases, each building on the previous one.

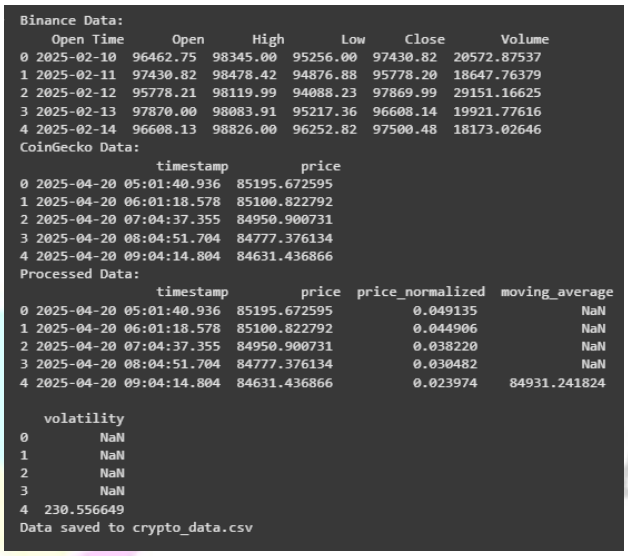
3.1 Data Acquisition

To analyze Bitcoin prices, I collected data from two sources:

Binance API: I used the Binance API to fetch daily OHLCV (Open, High, Low, Close, Volume) data for the BTCUSDT trading pair. The `fetch_binance_data` function retrieved 100 days of data, including columns like Open Time, Open, High, Low, Close, and Volume. Timestamps were converted to datetime format, and numerical columns were cast to float for analysis.

CoinGecko API: I obtained hourly Bitcoin price data in USD over 30 days (720 data points) using the `fetch_coingecko_data` function. The data included timestamps and prices, which were also converted to datetime format.

The data was stored in pandas DataFrames for further processing, ensuring consistency in format and structure.



```
Binance Data:
    Open Time      Open      High       Low     Close       Volume
0 2025-02-10  96462.75  98345.00  95256.00  97430.82  20572.87537
1 2025-02-11  97430.82  98478.42  94876.88  95778.20  18647.76379
2 2025-02-12  95778.21  98119.99  94088.23  97869.99  29151.16625
3 2025-02-13  97870.00  98083.91  95217.36  96608.14  19921.77616
4 2025-02-14  96608.13  98826.00  96252.82  97500.48  18173.02646
CoinGecko Data:
                 timestamp         price
0 2025-04-20 05:01:40.936  85195.672595
1 2025-04-20 06:01:18.578  85100.822792
2 2025-04-20 07:04:37.355  84950.900731
3 2025-04-20 08:04:51.704  84777.376134
4 2025-04-20 09:04:14.804  84631.436866
Processed Data:
                 timestamp         price  price_normalized  moving_average
0 2025-04-20 05:01:40.936  85195.672595          0.049135             NaN
1 2025-04-20 06:01:18.578  85100.822792          0.044906             NaN
2 2025-04-20 07:04:37.355  84950.900731          0.038220             NaN
3 2025-04-20 08:04:51.704  84777.376134          0.030482             NaN
4 2025-04-20 09:04:14.804  84631.436866          0.023974    84931.241824

   volatility
0         NaN
1         NaN
2         NaN
3         NaN
4  230.556649
Data saved to crypto_data.csv
```

3.2 Data Preprocessing

To prepare the data for analysis, I performed several preprocessing steps:

Handling Missing Values: I checked for null values and used forward and backward filling to impute missing data in price-related columns (e.g., price, moving average, volatility).

Normalization: I normalized the price data using MinMaxScaler to scale values between 0 and 1, creating a `price_normalized` column to standardize the data for modeling.

- Feature Engineering:

- Computed a 5-period moving average (`moving_average`) to smooth price trends.

- Calculated volatility as the 5-period rolling standard deviation of prices.

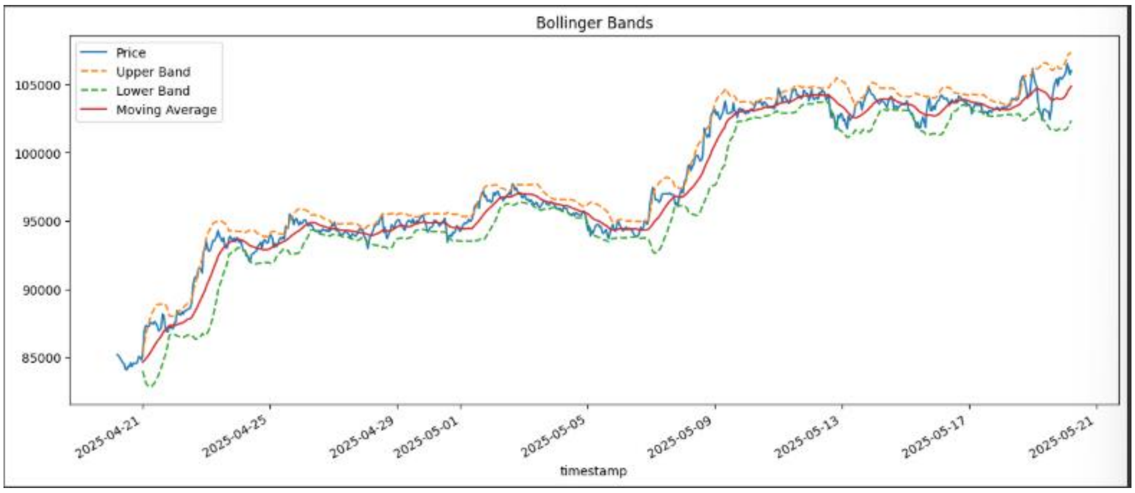
- Added daily returns (`daily_return`) as percentage price changes and price differences (`price_change`).

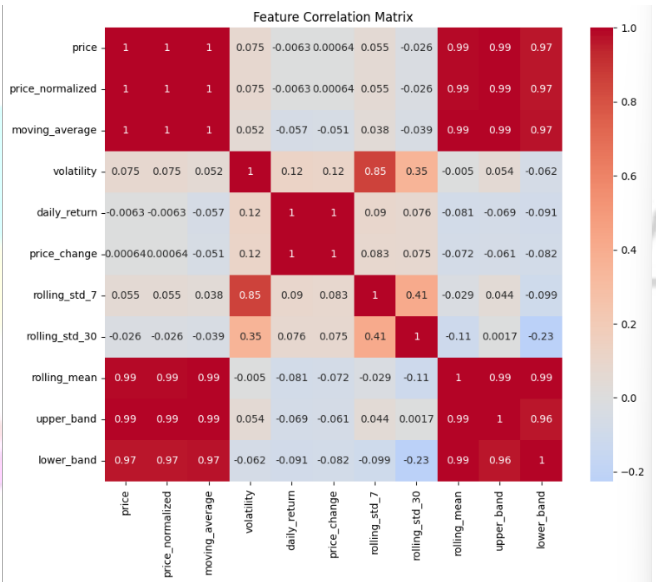
- Included 7-day and 30-day rolling standard deviations (`rolling_std_7`, `rolling_std_30`) to capture short- and long-term volatility.

- Created Bollinger Bands using a 20-period moving average and ±2 standard deviations (`rolling_mean`, `upper_band`, `lower_band`).

- Extracted temporal features like hour, day of week, and month from timestamps to analyze time-based patterns.

- Data Storage: The processed data was saved to `crypto_data.csv` and `processed_crypto_data.csv` for further use.





	price	price_normalized	moving_average	volatility	daily_return	price_change	rolling_std_7	rolling_std_30	rolling_mean	upper_band	lower_band
price	1	1	1	0.075	-0.0063	0.00064	0.055	-0.026	0.99	0.99	0.97
price_normalized	1	1	1	0.075	-0.0063	0.00064	0.055	-0.026	0.99	0.99	0.97
moving_average	1	1	1	0.052	-0.057	-0.051	0.038	-0.039	0.99	0.99	0.97
volatility	0.075	0.075	0.052	1	0.12	0.12	0.85	0.35	-0.005	0.054	-0.062
daily_return	-0.0063	-0.0063	-0.057	0.12	1	1	0.09	0.076	-0.081	-0.069	-0.091
price_change	0.00064	0.00064	-0.051	0.12	1	1	0.083	0.075	-0.072	-0.061	-0.082
rolling_std_7	0.055	0.055	0.038	0.85	0.09	0.083	1	0.41	-0.029	0.044	-0.099
rolling_std_30	-0.026	-0.026	-0.039	0.35	0.076	0.075	0.41	1	-0.11	0.0017	-0.23
rolling_mean	0.99	0.99	0.99	-0.005	-0.081	-0.072	-0.029	-0.11	1	0.99	0.99
upper_band	0.99	0.99	0.99	0.054	-0.069	-0.061	0.044	0.0017	0.99	1	0.96
lower_band	0.97	0.97	0.97	-0.062	-0.091	-0.082	-0.099	-0.23	0.99	0.96	1

3.3 Time Series Analysis

I conducted an exploratory analysis to understand Bitcoin price behavior:

- Visualization:

  - Plotted price trends over time to observe overall market movements.

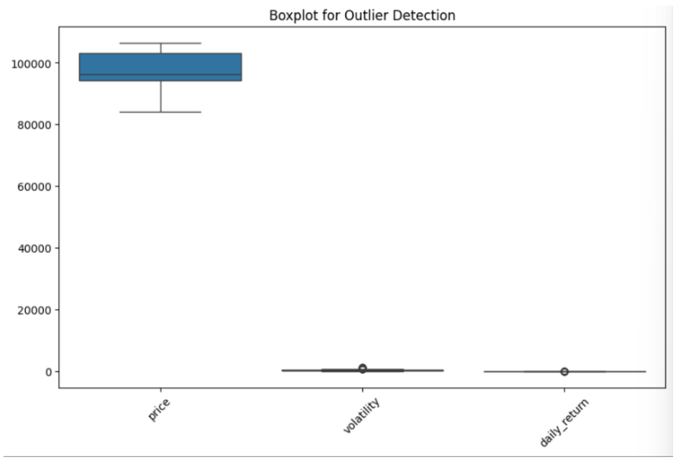
  - Compared prices with the 5-period moving average to identify trends.

  - Visualized Bollinger Bands to highlight price boundaries and potential trading signals.

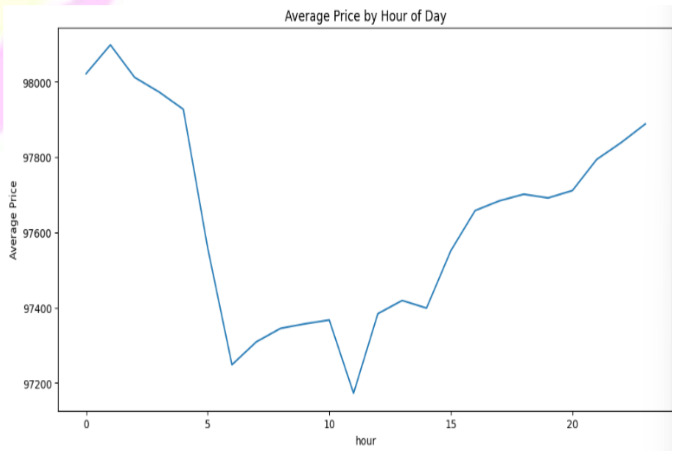
  - Plotted volatility over time and marked high-volatility periods (above the 90th percentile) to identify risky market conditions.

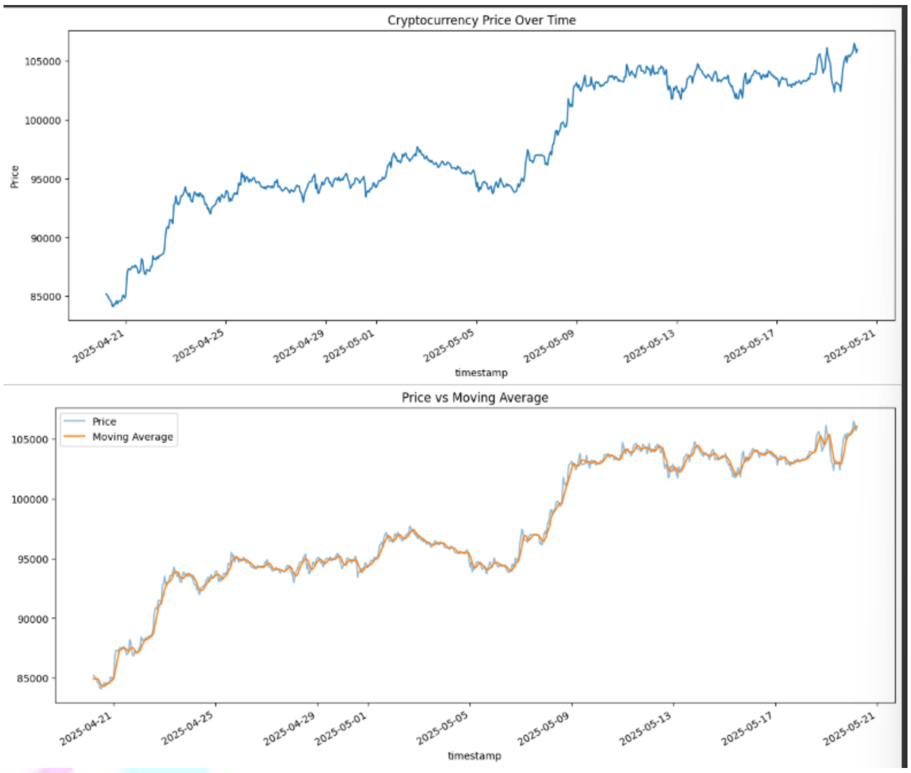
- Correlation Analysis: I generated a correlation matrix and visualized it as a heatmap using seaborn. Key findings included strong correlations (0.99) between price, normalized price, moving average, and Bollinger Bands, indicating these features move closely together.

- Outlier Detection: I used boxplots to identify outliers in price, volatility, and daily returns. Outliers were capped using the Interquartile Range (IQR) method, where values beyond 1.5 * IQR from the 25th or 75th percentiles were adjusted to the nearest bound.



-Temporal Analysis: I grouped prices by hour to calculate average prices, revealing intraday patterns useful for trading strategies.



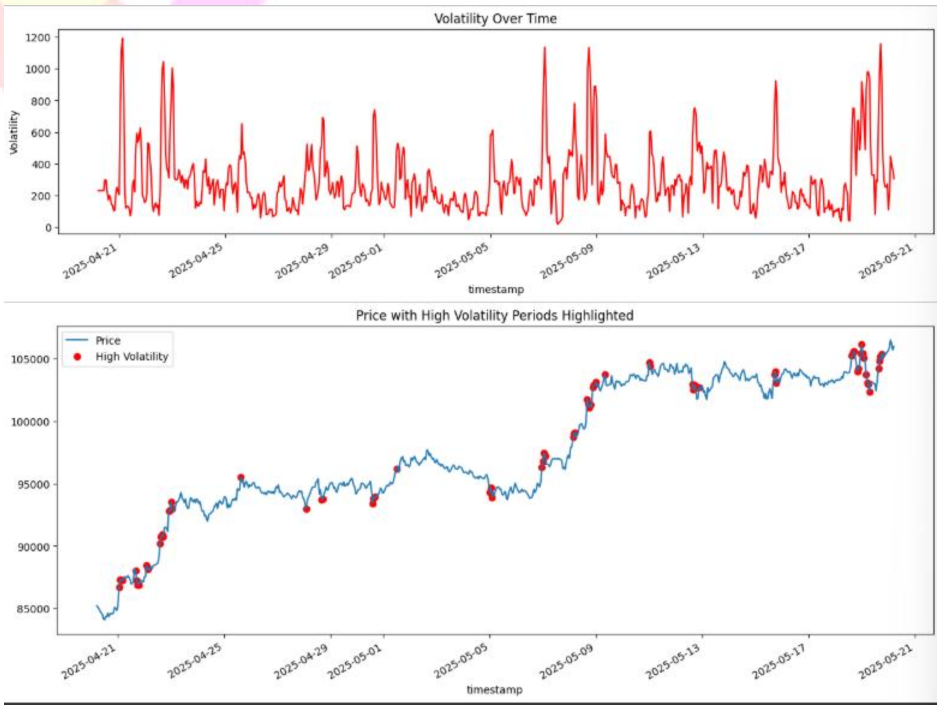


Key Statistics (from April 20, 2025, to May 20, 2025):

- Dataset size: 720 hourly observations.

- Price range: $84,093.83 (min) to $106,518.36 (max), mean: $97,630.36, standard deviation: $5,366.58.

- Volatility: Mean of 290.20, with peaks indicating significant market fluctuations.



3.4 Time Series Forecasting

I implemented and compared multiple forecasting models to predict Bitcoin prices:

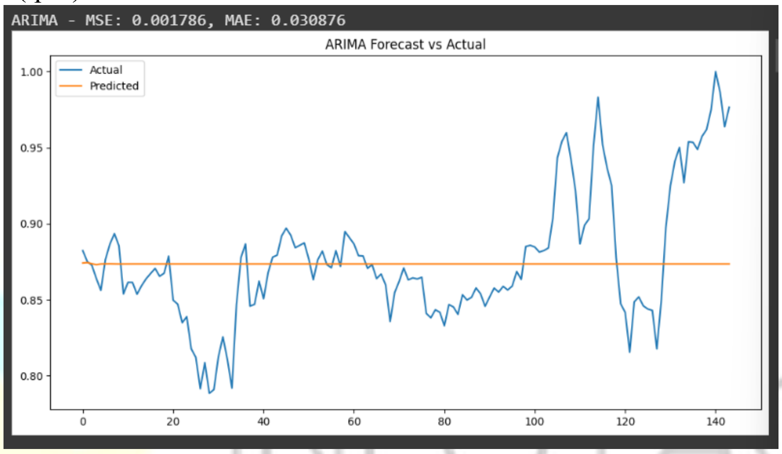
- Data Preparation:

  - Used `price_normalized` as the target variable, with an 80-20 train-test split (576 training, 144 testing data points).

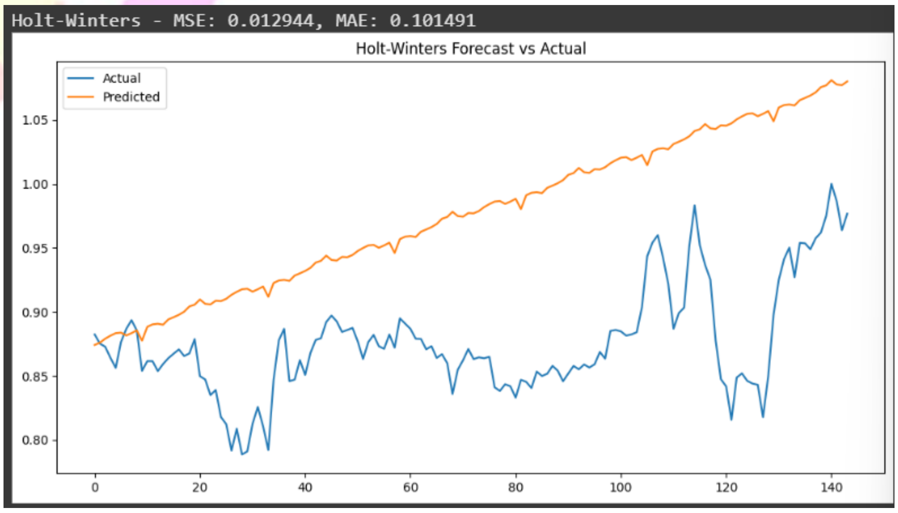
  - For machine learning models, I created features using a 24-hour window to capture historical context.

Statistical Models:

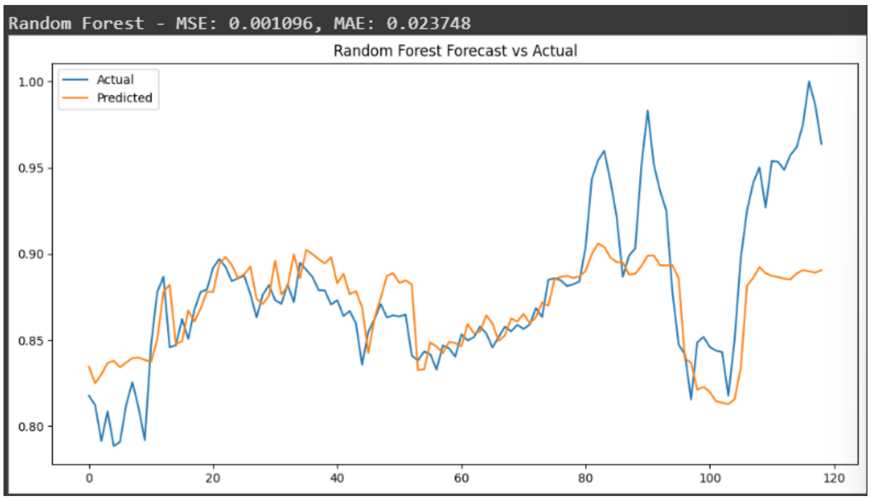
1. **ARIMA :** A statistical model with parameters for autoregression (p=5), differencing (d=1), and moving average (q=0). It achieved an MSE of 0.001786 and MAE of 0.030876.



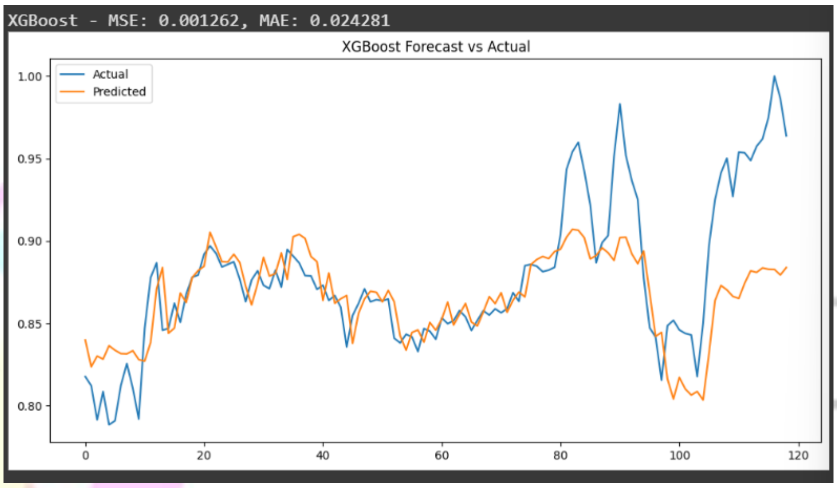
2. **Holt-Winters (Exponential Smoothing):** Incorporated additive trend and 24-hour seasonality. It had a higher error (MSE: 0.012944, MAE: 0.101491), suggesting it was less effective for this dataset.



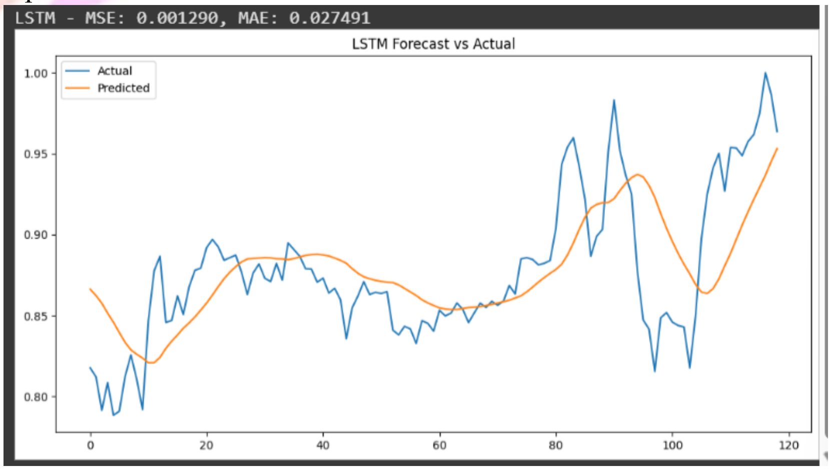
3. Random Forest: A machine learning model with 100 estimators, achieving the best performance with MSE: 0.001096 and MAE: 0.023748.


Random Forest - MSE: 0.001096, MAE: 0.023748

4. **XGBoost:** A gradient boosting model with 100 estimators and a learning rate of 0.1, yielding MSE: 0.001262 and MAE: 0.024281.

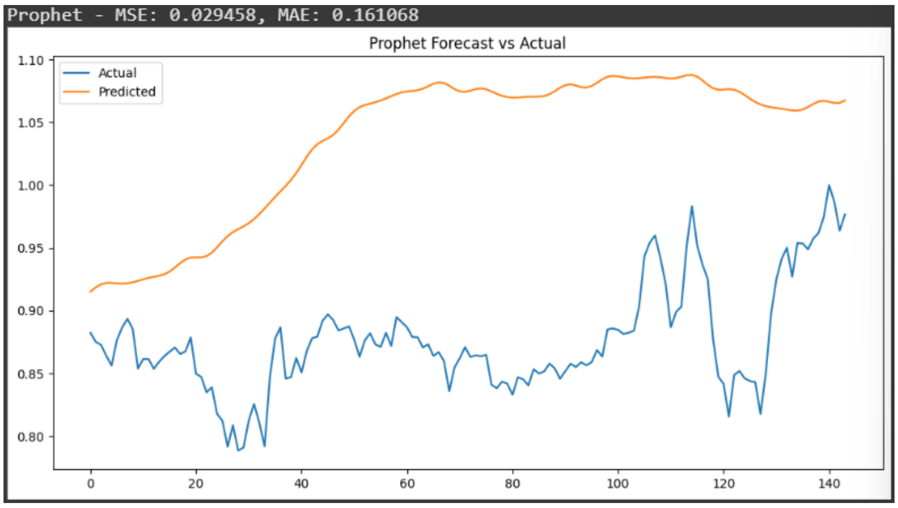

XGBoost - MSE: 0.001262, MAE: 0.024281

5. **LSTM:** A deep learning model with two LSTM layers (50 units each) and a dense output layer, trained for 20 epochs. It achieved MSE: 0.001290 and MAE: 0.027491.

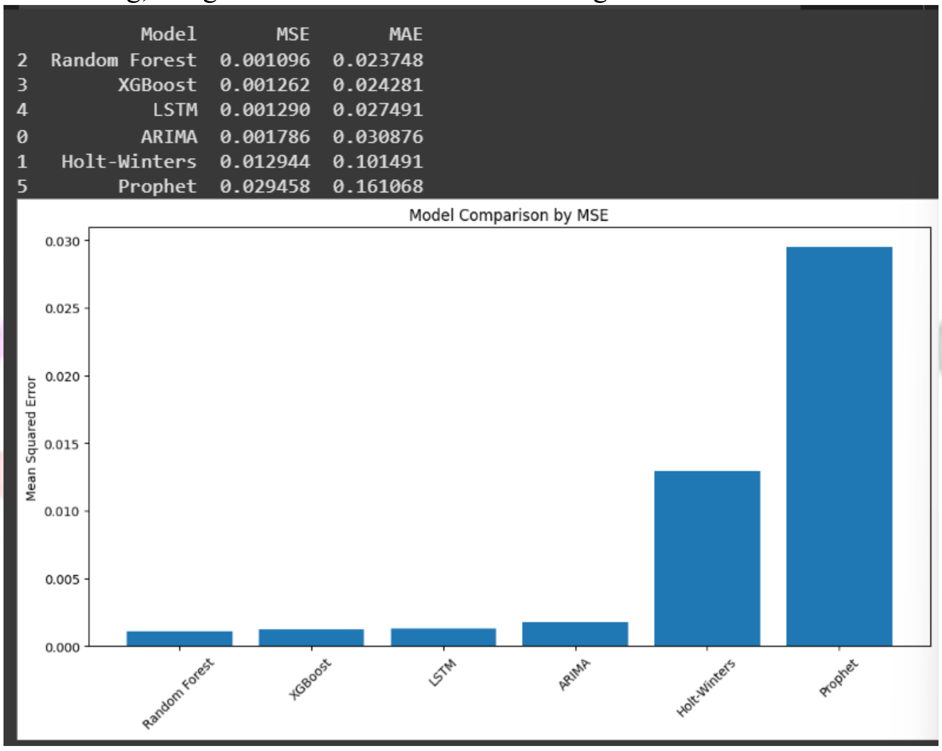

LSTM - MSE: 0.001290, MAE: 0.027491

6. **Prophet:** Facebook’s forecasting model with daily seasonality, but it performed poorly with MSE: 0.029458 and MAE: 0.161068, likely due to the high-frequency nature of the data.

```
Prophet - MSE: 0.029458, MAE: 0.161068
```



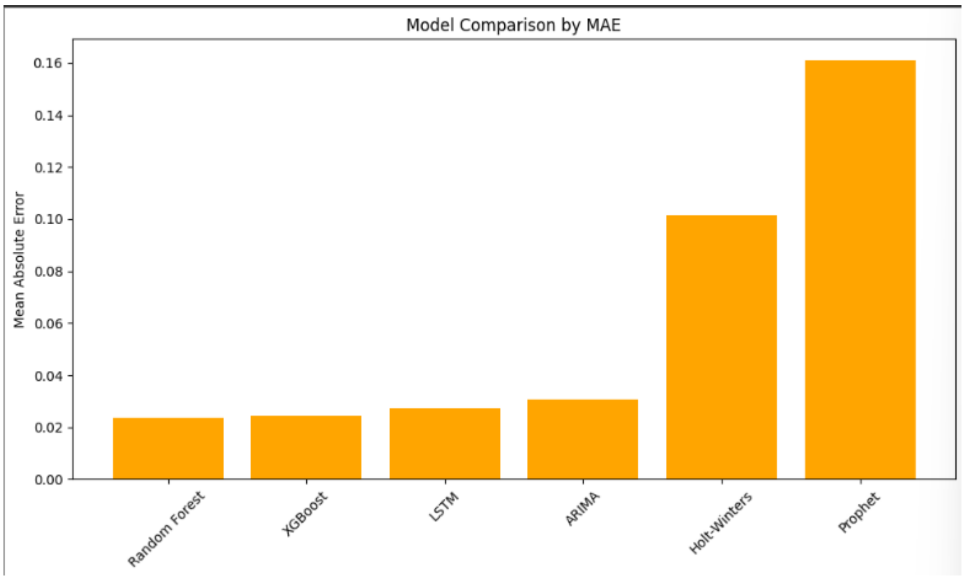
7. **Temporal Fusion Transformer (TFT):** An advanced deep learning model using PyTorch Forecasting, designed for multi-horizon forecasting with attention mechanisms.



```
            Model       MSE       MAE
2   Random Forest  0.001096  0.023748
3         XGBoost  0.001262  0.024281
4            LSTM  0.001290  0.027491
0           ARIMA  0.001786  0.030876
1    Holt-Winters  0.012944  0.101491
5         Prophet  0.029458  0.161068
```

<u>Evaluation:</u>

- I evaluated models using Mean Squared Error (MSE) and Mean Absolute Error (MAE).
- Random Forest outperformed others, likely due to its ability to capture non-linear patterns in the volatile cryptocurrency market.
- Visualizations of actual vs. predicted values helped assess model fit.



Market Sentiment Analysis:

1. Data Collection Sources: Twitter API: Fetched tweets containing keywords like "bitcoin," "crypto," or "cryptocurrency".

NewsAPI: Collected news articles related to cryptocurrency over the past 7 days.

Data Storage: Raw data saved in crypto_sentiment_raw.csv.

2. Text Preprocessing Steps Applied: Lowercasing: Converted all text to lowercase.

URL & Mention Removal: Removed hyperlinks and Twitter handles (@user, #hashtags).

Punctuation Removal: Eliminated special characters.

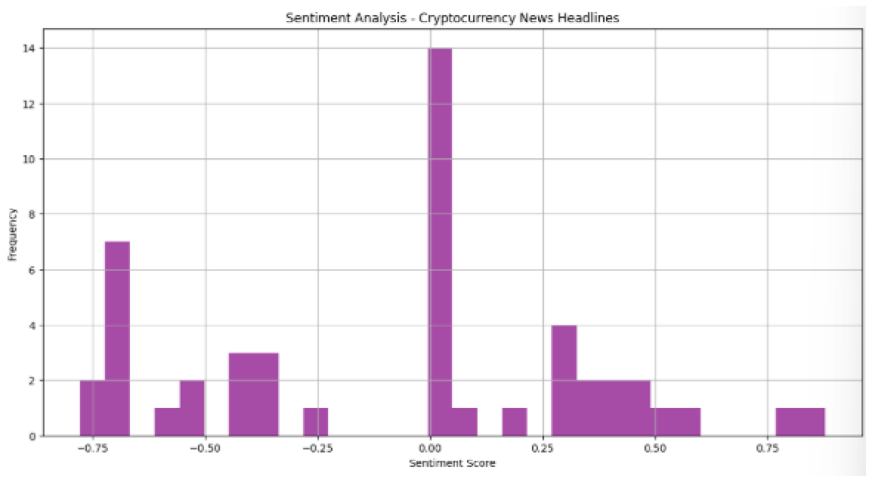
Tokenization: Split text into words.

Stopword Removal: Filtered out common words (e.g., "the," "and").

Lemmatization: Reduced words to their base form (e.g., "running" → "run").

Libraries Used: nltk (Punkt Tokenizer, Stopwords, WordNetLemmatizer)

re (Regex for cleaning)



3. Sentiment Analysis Methods Applied: VADER (Valence Aware Dictionary and sEntiment Reasoner)

Rule-based sentiment analyzer tuned for social media.

Output: Compound score between [-1 (Negative), +1 (Positive)].

TextBlob

Uses a pre-trained Naive Bayes classifier.

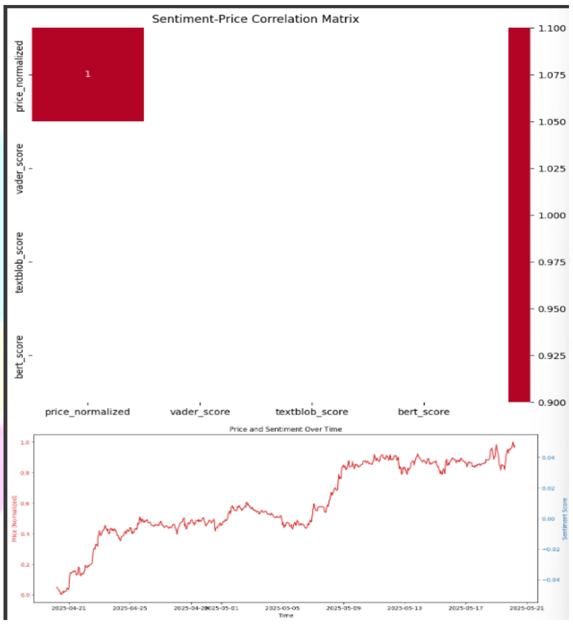
Output: Polarity score between [-1 (Negative), +1 (Positive)].

BERT (Transformers-based Model):

Fine-tuned bertweet-base-sentiment-analysis for financial text.

Output: POSITIVE=1, NEUTRAL=0, NEGATIVE=-1 (with confidence score).

Results: Each tweet/news article was assigned three sentiment scores.



3.5 Interactive Dashboard:

I developed an interactive dashboard using Dash and Plotly to present the analysis and forecasts:

Components:

Price Chart: Displayed price trends, moving averages, and Bollinger Bands with selectable timeframes (24 hours, 1 week, 1 month, 3 months).

Indicators Chart: Showed volatility and daily returns, with color-coded bars for positive/negative returns.

Sentiment Volume Chart: Visualized hourly sentiment data volume from `crypto_sentiment_raw.csv` (though sentiment analysis details were limited in the document).

Predictive Analytics: Allowed users to select ARIMA, LSTM, or Prophet models to view 7-day price forecasts with confidence intervals.

Risk Assessment: Displayed metrics like volatility, max drawdown, and Sharpe ratio, along with investment recommendations based on risk levels (High, Medium, Low) and technical indicators (e.g., price/MA ratio, Bollinger Band position).

Features:

- Real-time updates every 5 minutes using `dcc.Interval`.

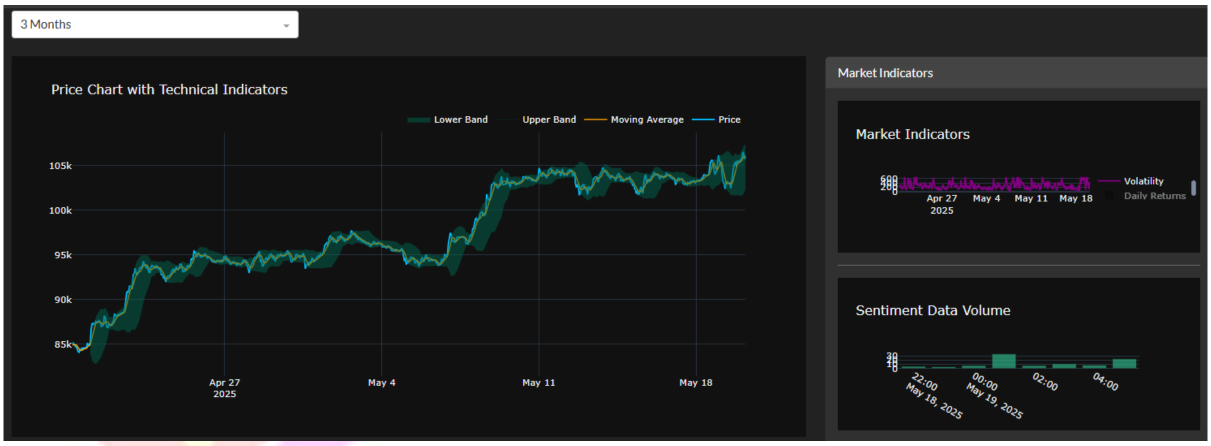
- Dark theme (`plotly_dark`) for better readability.

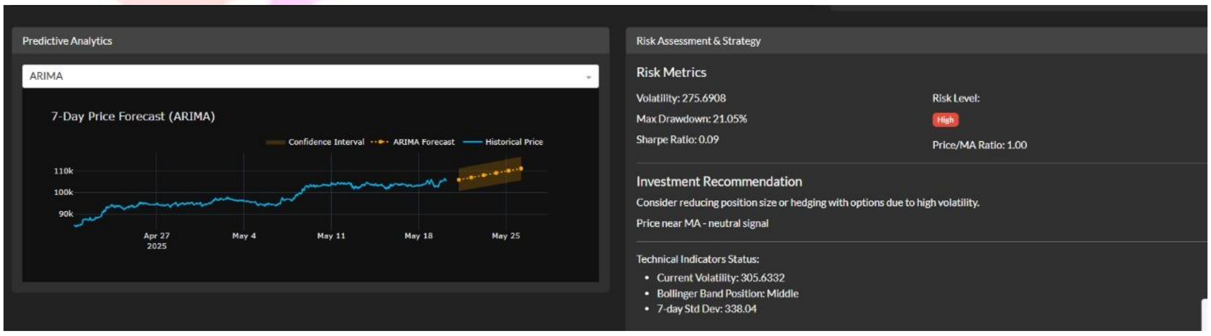
- Responsive layout with `dash-bootstrap-components` for professional styling.

- Risk Assessment Logic:

- Classified risk based on volatility: High (>0.05), Medium (0.02–0.05), Low (<0.02).

- Provided recommendations, such as reducing position size during high volatility or dollar-cost averaging in moderate conditions.





Risk Assessment & Strategy

Risk Metrics

Volatility: 275.6908

Max Drawdown: 21.05%

Sharpe Ratio: 0.09

Risk Level:

High

Price/MA Ratio: 1.00

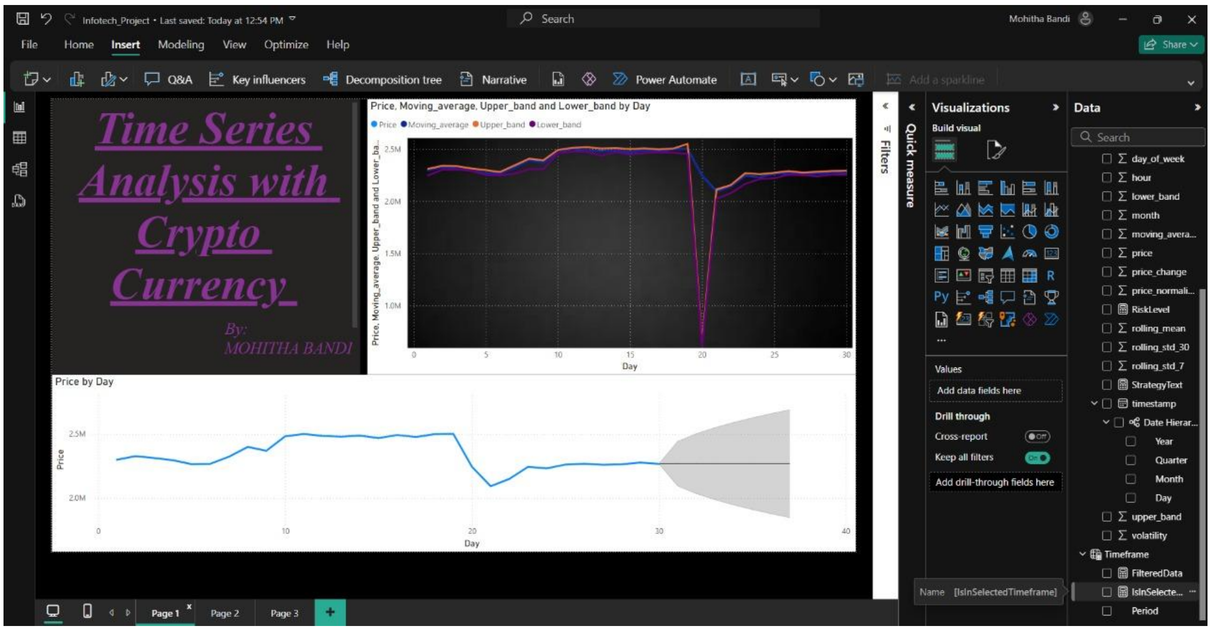
Investment Recommendation

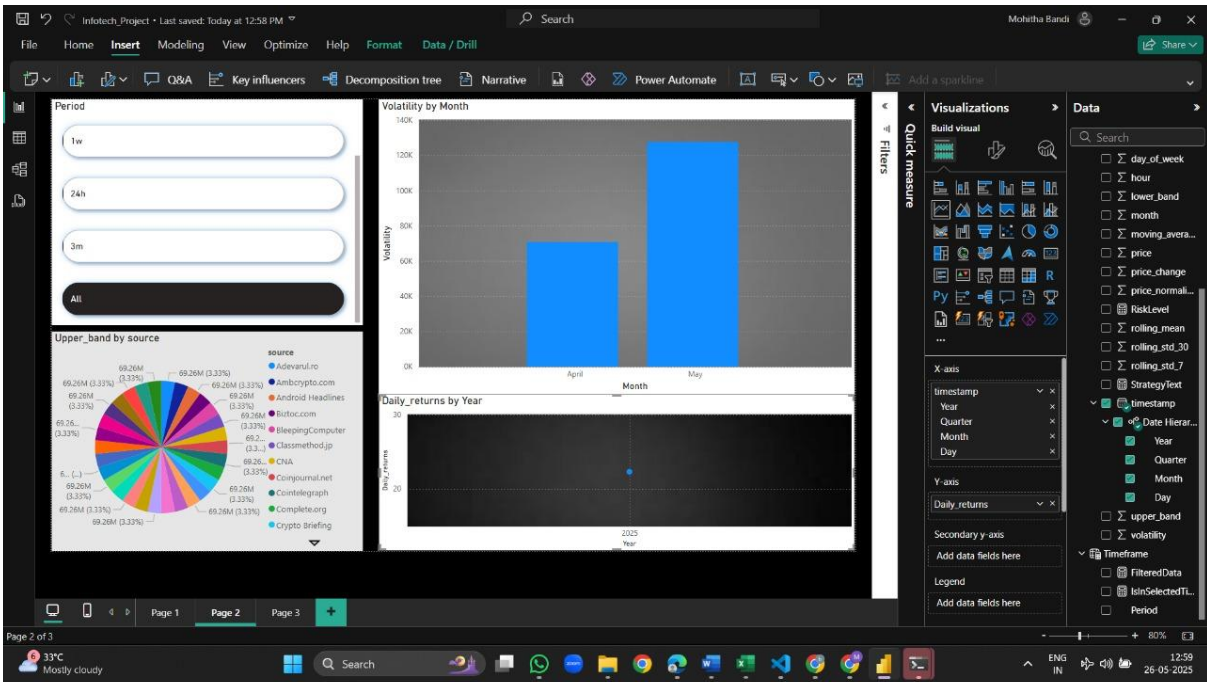
Consider reducing position size or hedging with options due to high volatility.

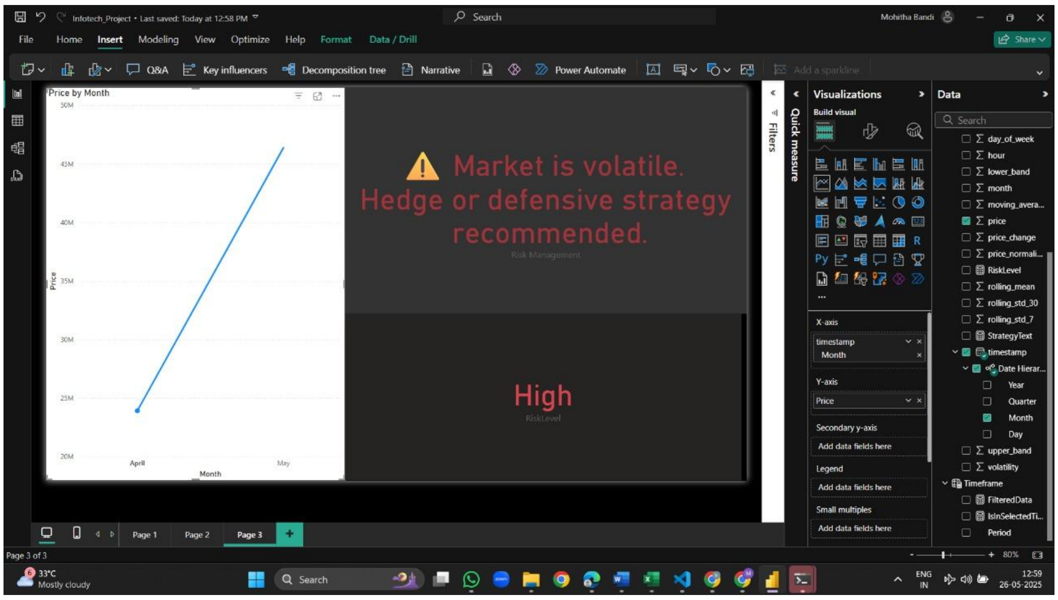
Price near MA - neutral signal

Technical Indicators Status:

- Current Volatility: 305.6332
- Bollinger Band Position: Middle
- 7-day Std Dev: 338.04


Infotech_Project • Last saved: Today at 12:54 PM
Time Series Analysis with Crypto Currency
By: MOHITHA BANDI
Price, Moving_average, Upper_band and Lower_band by Day
Price
Moving_average
Upper_band
Lower_band
Day
Price by Day
Price
Visualizations
Data
Page 1
Page 2
Page 3


Infotech_Project • Last saved: Today at 12:58 PM
Period
1w
24h
3m
All
Volatility by Month
Volatility
April
May
Month
Upper_band by source
source
Adevarul.ro
Ambcrypto.com
Android Headlines
Biztoc.com
BleepingComputer
Classmethod.jp
CNA
Coinjournal.net
Cointelegraph
Complete.org
Crypto Briefing
Daily_returns by Year
2025
Year
Visualizations
Data
Page 1
Page 2
Page 3


Market is volatile.
Hedge or defensive strategy recommended.

High

3.6 Technical Implementation

Libraries Used: pandas, numpy, matplotlib, seaborn, scikit-learn, TensorFlow, statsmodels, prophet, pytorch-lightning, pytorch-forecasting, dash, dash-bootstrap-components, and others for sentiment analysis (tweepy, nltk, vaderSentiment, transformers).

Challenges:

- Managing API rate limits and ensuring data consistency across Binance and CoinGecko.

- Handling high-frequency data with potential noise and outliers.

- Tuning model hyperparameters to balance accuracy and overfitting.

4. Results and Key Findings

➢ Data Insights:

- Bitcoin prices fluctuated significantly, with a standard deviation of $5,366.58 over 30 days.

- High-volatility periods were identified, aiding in risk-aware decision-making.

- Hourly price patterns revealed intraday trends, useful for short-term trading.

➢ Forecasting Performance:

- Random Forest was the most accurate model, with the lowest MSE (0.001096) and MAE (0.023748), highlighting its suitability for cryptocurrency data.

- LSTM and XGBoost also performed well, while Prophet struggled with high-frequency data.

Dashboard Utility

- The dashboard provided an intuitive interface for visualizing price trends, technical indicators, and forecasts.

- Risk assessment tools offered practical insights, such as hedging strategies during volatile periods.

Learning Outcomes:

- Gained hands-on experience with Python libraries for data analysis and visualization.

- Developed expertise in time series forecasting and model evaluation.

- Learned to build interactive web applications using Dash.

5. Challenges Faced:

- Data Quality: Ensuring consistency between Binance and CoinGecko data required careful preprocessing.
- Model Tuning: Selecting appropriate parameters for models like ARIMA and LSTM was time-consuming and required experimentation.
- Dashboard Development: Integrating real-time updates and handling large datasets in Dash posed technical challenges.
- Time Constraints: Balancing comprehensive analysis with internship deadlines was demanding but taught me efficient time management.

6. Conclusion:

This internship project was a valuable opportunity to apply data science techniques to real-world financial data. By collecting and processing Bitcoin price data, conducting time series analysis, building forecasting models, and developing an interactive dashboard, I created a robust tool for market analysis and decision-making. The project enhanced my technical skills in Python programming, data preprocessing, time series modelling, and web development, while deepening my understanding of cryptocurrency markets. The successful implementation of the dashboard and forecasting models demonstrated the practical utility of data science in finance.

7. Future Enhancements:

- AI-powered portfolio management system.
- Integration with DeFi (Decentralized Finance) protocols.
- Mobile app integration for on-the-go trading insights

9. Acknowledgments

I sincerely thank my internship mentors and team for their support and feedback throughout this project. Their guidance helped me navigate complex technical challenges and deliver a successful outcome. This experience has been instrumental in shaping my career aspirations in data science.

BY:

Mohitha Bandi

E-mail: mohitha12026@gmail.com