Assignment 3

ML Data Product

3 November 2025

Group 13

Student Last Name	Student First Name	Student ID	Group Allocation	Github Username	API Repo URL	Custom Package Repo URL
Leong	Dylan	25400603	Student A	dylanjjleong-uts	https://github.com/dylanjjleong-uts/AMLA_AT3_MLDataProduct_API	https://github.com/dylanjjleong-uts/AMLA_AT1_Group8_KaggleNBADraft_CollabRepo/tree/main/my_krml_25400603
Wongwatcharapaiboon	Kittituch	25544646	Student B	KittituchW	https://github.com/KittituchW/AMLA_at3_25544646_FastAPI	https://github.com/KittituchW/adv_mla_at1_25544646
Ratanawarocha	Ratticha	24996427	Student C	ratticha	https://github.com/ratticha/AML_AT3_FastAPI	https://github.com/ratticha/amla_at1_24996427
Saito	Shawya	25552249	Student D	Shawynot33	https://github.com/Shawynot33/at3_api	https://github.com/Shawynot33/my_krml_25552249

Github	Project Repo: https://github.com/KittituchW/AMLA_AT3_Group13 Streamlit App Repo: https://github.com/KittituchW/AMLA_Group13_Streamlit

36120 - Advanced Machine Learning Application
Master of Data Science and Innovation
University of Technology of Sydney

Table of Contents

1. Executive Summary

This project focused on developing an end-to-end data product for cryptocurrency investors, providing both historical market insights and predictive analytics for four major cryptocurrencies: Bitcoin, Ethereum, XRP, and Solana. The primary objective was to create an interactive Streamlit application that enables users to explore historical price and volume data while accessing next-day high price predictions generated by machine learning models. This project addresses the difficulty of forecasting in highly volatile cryptocurrency markets, aiming to provide short-term insights that can assist investors in decision making.

To achieve this, individual models were trained for each token: CatBoost for Bitcoin, Linear Regression for Ethereum, LightGBM for XRP, and XGBoost for Solana. While these models offered modest improvements, or in some cases no improvement over a naive baseline that predicts tomorrow's high as yesterday's high, the project successfully demonstrated a full pipeline. This included model development, deployment via FastAPI on Render using Docker, and a cloud-deployed Streamlit interface, resulting in a functional platform that integrates predictive modelling with interactive visualization. The project highlights the practical application of data science in the context of cryptocurrency investment and provides a foundation for further refinement of forecasting models.

2. Business Understanding

a. Business Use Cases

This project focuses on developing predictive machine-learning models to forecast the next-day high prices of four major cryptocurrencies — Bitcoin (BTC), Ethereum (ETH), Solana (SOL), and Ripple (XRP). In the rapidly evolving digital-asset market, investors face persistent challenges arising from extreme volatility, limited regulatory clarity, and a 24/7 trading cycle. These conditions make manual, sentiment-driven decisions both time-consuming and error-prone.

The business use case centres on empowering retail and institutional investors with data-driven forecasts that support short-term trading and risk-management strategies. By integrating predictive analytics into an interactive Streamlit dashboard connected through FastAPI, users can visualise historical data, monitor market behaviour, and access next-day forecasts in real time.

This initiative responds to two main challenges and opportunities:

1. **Market Unpredictability:** Cryptocurrency prices are shaped by complex, non-linear factors such as investor sentiment, global macroeconomic conditions, and liquidity. Traditional linear or rule-based methods struggle to capture these dynamic interactions.
2. **Data Opportunity:** The availability of large-scale, high-frequency data from reliable public sources (Kraken, CoinDesk, CoinGecko) enables ML models to uncover hidden temporal and behavioural patterns that human traders may overlook.

Machine-learning algorithms are particularly suited to this environment because they can model non-linear dependencies, adapt to shifting market conditions, and learn intricate relationships between variables such as price trends, volume, and volatility. Each algorithm contributes distinct advantages in time-series forecasting and interpretability:

- **CatBoost (BTC)**: robust to multicollinearity and effective with noisy financial data.
- **Linear Regression (ETH):** interpretable and efficient for capturing linear relationships.
- **LightGBM (XRP):** powerful gradient boosting with strong generalisation and speed.
- **XGBoost (SOL):** optimised for speed and accuracy in tabular financial datasets.

Together, these models create a diversified predictive framework that mirrors professional multi-model trading systems, improving robustness and confidence in forecast reliability. The project also contributes to broader Fintech goals of enhancing transparency, enabling responsibility AI adoption, and democratising access to advanced market analytics for both institutional and emerging retail investors.

b. Key Objectives

The primary objective of this project is to develop and evaluate predictive machine-learning models capable of forecasting the next-day high prices of leading cryptocurrencies, thereby supporting transparent, data-driven investment decisions. The project aligns with the FinTech sector's broader goals of improving risk management, enabling algorithmic trading, and fostering responsible AI adoption in financial analytics.

No.	Objective	Stakeholders / Requirements	ML Approach
1	Develop and evaluate predictive models for BTC, ETH, XRP, and SOL to forecast next-day high prices.	Investors, traders, and analyst require accurate short-term price forecasts to guide entry and exit decisions.	Regression-based and gradient-boosting algorithms are applied to capture both linear and non-linear relationships, enhancing predictive precision.
2	Identify key market features driving short-term price movement (volatility, trading volume, market capitalisation, trend indicators).	Data analysts and FinTech platforms seek interpretable insights and transparent model behaviour.	Feature-importance and correlation analyses reveal the strongest predictive variables for each cryptocurrency, improving explainability.
3	Benchmark and compare model performance across consistent evaluation metrics (MAE, RMSE, MAPE, R^2).	Stakeholders expect transparent, reproducible, and comparable model results.	Standardised metrics enable quantitative assessment of accuracy and stability, informing model selection and business recommendations.
4	Integrate all models into a unified Streamlit dashboard and FastAPI framework for deployment.	End users and partners need accessible, real-time forecasting tools that support continuous updates.	Automated pipelines and API-based deployment ensure scalability, maintainability, and cross-platform usability.

3. Data Understanding

3.1 Dataset Overview and Collection

The project focuses on four major cryptocurrencies—Bitcoin (BTC), Ethereum (ETH), Ripple (XRP), and Solana (SOL)—each paired against the US Dollar (USD). The datasets capture daily trading activity through the OHLCV structure (Open, High, Low, Close, Volume), which represents standard market indicators for analysing asset performance and investor behaviour.

Data collection was conducted in two stages:

1) **Development Stage** for model training and experimentation using pre-downloaded datasets
2) **Production Stage** for real-time data retrieval through public APIs (Kraken, CoinDesk, and CoinGecko).

3.1.1 Development Stage

Pre-downloaded historical datasets were used to ensure consistency and reproducibility during the development phase. The data contain daily OHLCV and market-capitalisation values covering multiple years, enabling models to learn both short- and long-term market dynamics.

Cryptocurrency	Trading Pair	Period Covered	Frequency
Bitcoin (BTC)	BTC/USD	22 Oct 2024 – 28 May 2025	Daily
Ethereum (ETH)	ETH/USD	1 Jan 2015 – 31 Dec 2024	Daily
Ripple (XRP)	XRP/USD	1 Jan 2015 – 31 Dec 2024	Daily
Solana (SOL)	SOL/USD	11 Apr 2020 - 31 Dec 2024	Daily

Limitations:

- Missing records and small gaps may exist due to historical exchange outages or low trading volume days.
- Extreme price fluctuations were retained to preserve realistic volatility patterns essential for model learning.

3.1.2 Production Stage

In the production phase, a FastAPI service retrieves live OHLC data to produce next-day high-price forecasts. Multiple data sources are integrated to ensure redundancy and reliability—Kraken for Ethereum, Ripple, and Solana; CoinDesk and CoinGecko for Bitcoin.

Cryptocurrency	API Provider	Endpoint	Frequency
Bitcoin (BTC)	CoinDesk / CoinGecko	https://data-api.coindesk.com/index/cc/v1/historical/days https://api.coingecko.com/api/v3/coins/bitcoin/market_chart	Daily
Ethereum (ETH)	Kraken	https://api.kraken.com/0/public/OHLC?pair=XETHZUSD	Daily
Ripple (XRP)	Kraken	https://api.kraken.com/0/public/OHLC?pair=XXRPZUSD	Daily
Solana (SOL)	Kraken	https://api.kraken.com/0/public/OHLC?pair=SOLUSD	Daily

Limitations:

- API rate limits (approximately one request per second) restrict continuous calls.
- Bitcoin's CoinDesk/CoinGecko APIs may differ slightly in data granularity and naming conventions; preprocessing harmonises formats before model input.
- The API finalizes daily data after the market closes (UTC), so same-day predictions rely on the previous day's close.
- Temporary downtime or network delays may interrupt data retrieval during deployment.

3.2 Key Variables and Feature Definitions

The core dataset comprises twelve key variables derived from the OHLCV structure and time metadata. These variables represent essential market indicators used to describe price movement, trading volume, and capitalization trends.

Attributes	Type	Description	Significance
Open	Float	opening price of the asset for the day.	Indicates early market sentiment and initial price level.
High	Float	Highest price recorded during the day.	Reflects maximum investor optimism and market resistance.
Low	Float	lowest price recorded during the day.	Captures downside movement and intra-day volatility.

Close	Float	final price at the end of the day.	Represents daily market consensus and closing sentiment.
Volume	Float	Total trading volume within the day.	Measures liquidity and overall trading activity.
MarketCap	Float	Total market capitalization of the asset.	Represents overall value and investor confidence in the asset.
timeopen, time_close, time_high, time_low	Object	Timestamps marking key trading events.	Enable temporal mapping of price movements.
timestamp	Object	Raw Unix timestamp for reference.	Used for data indexing or joining operations.

These foundational variables enable a structured representation of market behaviour and form the basis for further transformations such as rolling averages, volatility measures, and return calculations introduced later in the feature engineering process.

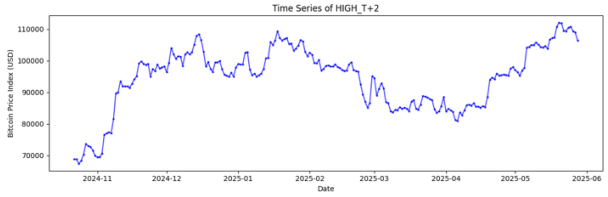
3.3 Exploratory Data Analysis (EDA)

3.3.1 Bitcoin (BTC)

Graph	Key Insights

1. Summary of Data
`DatetimeIndex: 365 entries, 2024-10-22 to 2025-10-21`
`Data columns (total 20 columns):`
`#  Column  Non-Null Count  Dtype`
`0  UNIT  365 non-null  object`
`1  TYPE  365 non-null  object`
`2  MARKET  365 non-null  object`
`3  INSTRUMENT  365 non-null  object`
`4  OPEN  365 non-null  float64`
`5  HIGH  365 non-null  float64`
`6  LOW  365 non-null  float64`
`7  CLOSE  365 non-null  float64`
`8  FIRST_MESSAGE_TIMESTAMP  365 non-null  datetime64[ns]`
`9  LAST_MESSAGE_TIMESTAMP  365 non-null  datetime64[ns]`
`10  FIRST_MESSAGE_VALUE  365 non-null  float64`
`11  HIGH_MESSAGE_VALUE  365 non-null  float64`
`12  HIGH_MESSAGE_TIMESTAMP  365 non-null  datetime64[ns]`
`13  LOW_MESSAGE_VALUE  365 non-null  float64`
`14  LOW_MESSAGE_TIMESTAMP  365 non-null  datetime64[ns]`
`15  LAST_MESSAGE_VALUE  365 non-null  float64`
`16  TOTAL_INDEX_UPDATES  365 non-null  int64`
`17  VOLUME  365 non-null  float64`
`18  QUOTE_VOLUME  365 non-null  float64`
`19  market_cap  365 non-null  float64`
`dtypes: datetime64[ns](4), float64(11), int64(1), object(4)`
`memory usage: 59.9+ KB`
	The Bitcoin (BTC) dataset comprises 365 daily observations spanning from 22 October 2024 to 21 October 2025, capturing the most recent market dynamics. Indexed by UTC dates, it contains 20 features representing currency metadata (unit, type, market, instrument), prices (OHLC), volume measures (volume, quote volume), temporal descriptors (message timestamps and values, and market-level characteristics (total index updates, market capitalisation).

2. Bitcoin HIGH (day + 2) Price Trends

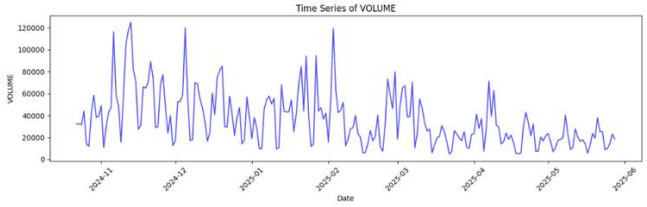


From this point forward, all exploratory data analysis (EDA) is performed exclusively on the training set to prevent data leakage. Plotting the HIGH (day + 2) target variable across time from 22 October 2024 to 28 May 2025, we observe relatively smooth volatility overall, with no extended periods of extreme fluctuations. However, there are distinct phases where prices shift to new, higher levels that persist over time. For instance, the series spikes +16.53% over two days—from 7 November 2024 (HIGH (day + 2) = USD 76,929.29) to 9 November 2024 (HIGH (day + 2) = USD 89,644.70). Thereafter, it stabilizes around that range, remaining above USD 90,000 until a major drop between 24–27 February 2025, which briefly reverts before falling again on 6 March 2025, establishing a new level around USD 85,000. Similar upward shifts occur on 19–20 April 2025, 6 May 2025, and 26 May 2025. While this pattern suggests a pattern of structural breaks that persist across time, modelling the raw series directly may not effectively capture short-term, day-to-day dynamics within each regime.

3. Bitcoin Price Detrended (2024—2025)

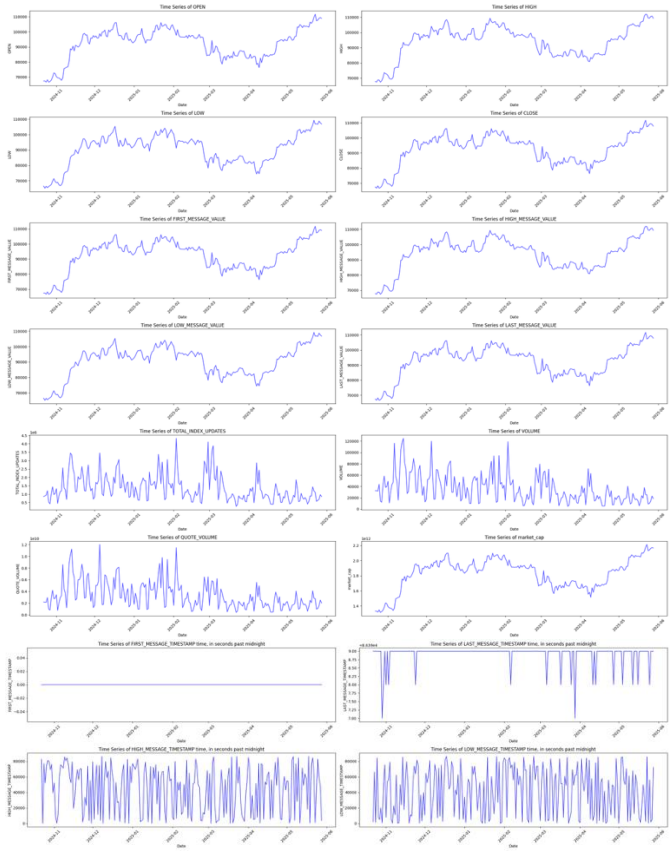
To view day-to-day price dynamics, we plot a T−2 detrend of the target variable, along with a T-2 Fibonacci detrend using a window size of 8. The T−2 detrend exhibits high volatility, as is typical of detrended data. In contrast, the Fibonacci causal detrend produces a smoother series that may better capture overarching directional trends in daily price movements. We utilise the latter as our final target variable during modelling.

4. Volume Patterns



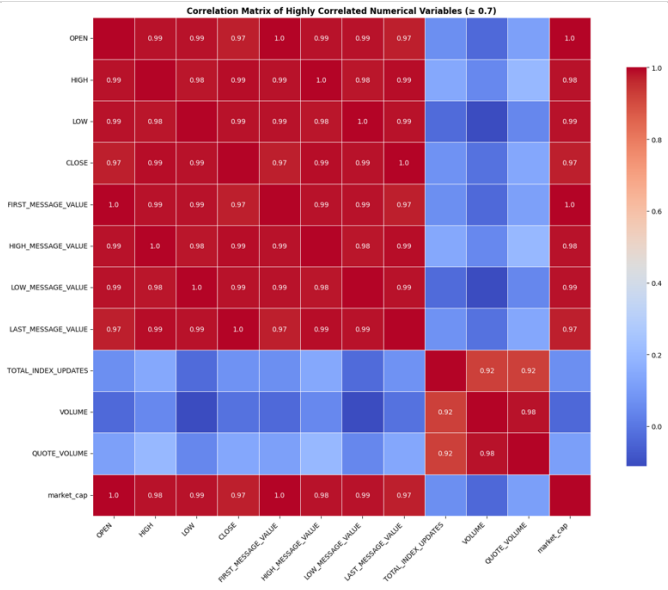
Daily Bitcoin trading volumes exhibit significant volatility throughout the training period. Notably, a reduction in variability becomes apparent around mid-March 2025, suggesting a possible shift in trading dynamics or market participation during that time. This pattern indicates that Bitcoin's market activity may be cyclical, characterized by alternating periods of heightened trading intensity and relative quiet, reflecting phases of increased market attention followed by consolidation.

5. Trends and Strong Correlations of Predictors

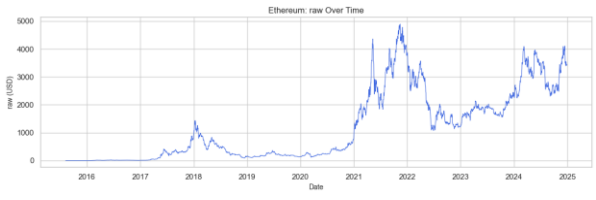
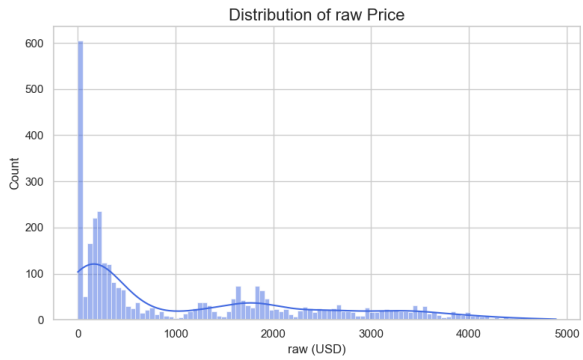


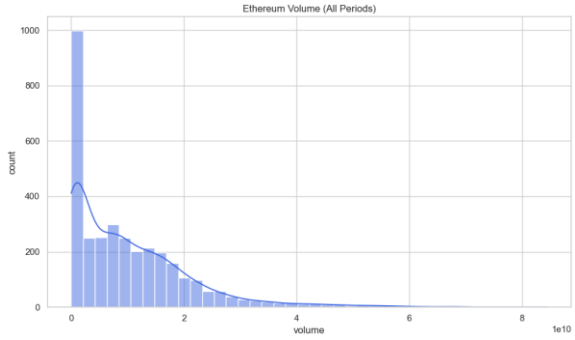
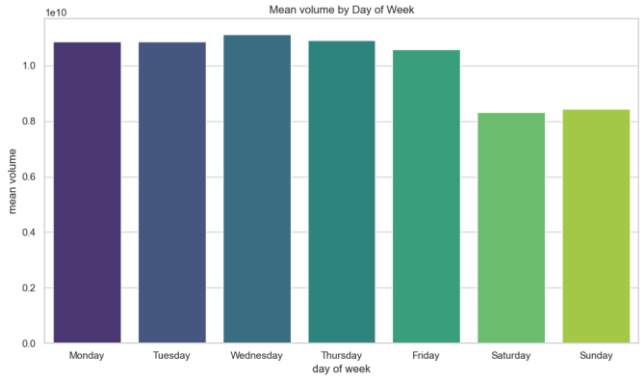
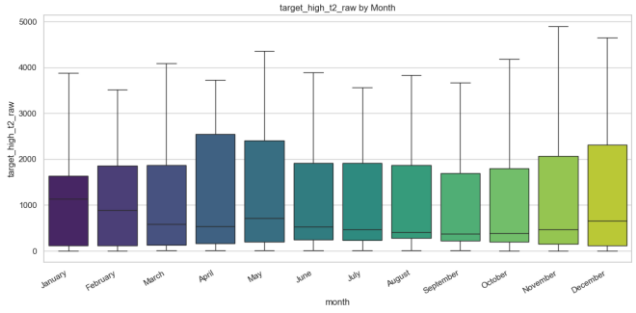
To explore each predictor feature, we plot all variables over time and generate a confusion matrix of strongly correlated features (any correlations ≥ 0.7).

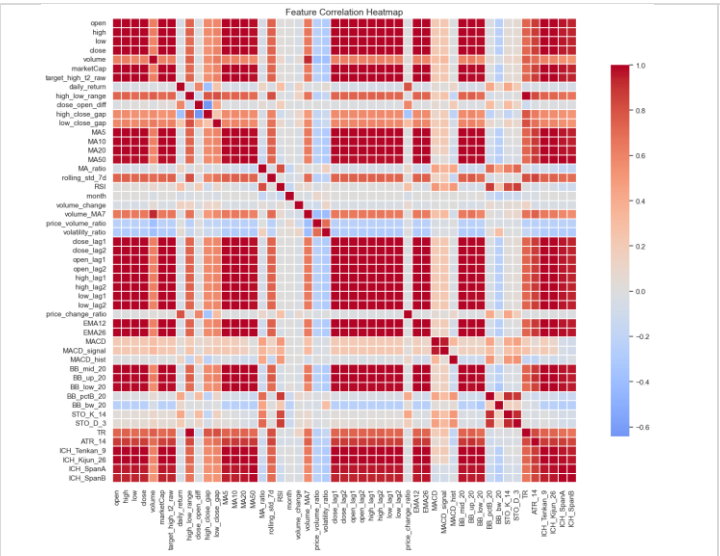
When plotting all numerical features over time, it is evident that OPEN price, HIGH price, LOW price, CLOSE price, first message value, high message value, low message value, last message value, and market capitalization closely follow the overall trend of the raw HIGH price—and, by extension, the target variable. This observation is supported by the very high positive coefficients in the plotted correlation matrix of all variables, indicating strong linear relationships and thus multicollinearity among these variables. Similarly, total index updates and quote volume, like volume, exhibit highly volatile behaviour, with a a clear reduction in variability beginning around mid-March 2025. This similarity is also reflected in their strong positive correlations with one another. Since multicollinearity among these two groups of variables—price indicators and volume indicators—may cause convergence or performance issues for certain algorithms, we intentionally

	select CatBoost as our modelling approach, as it is designed to handle multicollinearity effectively.

3.3.2 Ethereum (ETH)

Graph	Key Insights
1. Ethereum Price Trend (2015–2025) 	The plot shows Ethereum's price movement over the past decade, highlighting major growth cycles and market corrections. The data reveals three significant bull runs, in late 2017, early 2021, and late 2023, followed by notable drops reflecting high market volatility. Overall, the long-term trend demonstrates strong upward momentum despite short-term fluctuations, suggesting Ethereum's growing market adoption and resilience over time.
2. Distribution of Ethereum Prices 	The distribution of Ethereum prices is highly right-skewed, with most values concentrated below USD 1,000. A few extreme peaks above USD 3,000 reflect the sharp price surges during major bull markets. This skewness indicates that Ethereum's price tends to stay low for long periods but occasionally experiences rapid growth, showing the coin's volatility and speculative nature.

3. Distribution of Ethereum Trading Volume 	The distribution of Ethereum's trading volume is heavily right-skewed, meaning most trading days had relatively low activity, while only a few days showed extremely high volumes. These spikes likely occurred during major market rallies or crashes when trading activity surged. This pattern reflects typical market behaviour where trading volume intensifies during high volatility periods but remains low in stable conditions.
4. Mean Ethereum Trading Volume by Day of the Week 	Ethereum's trading activity is higher during weekdays, with Wednesday and Thursday showing the highest average volumes. In contrast, trading volume drops slightly on weekends, especially on Saturday. This pattern suggests that trading participation is more active during business days, possibly due to institutional trading behaviour and higher overall market engagement.
5. Monthly Distribution of Ethereum High Prices 	The monthly boxplot shows that Ethereum's high prices vary across the year, with slightly higher medians and wider spreads around April, May, and December. These months show greater volatility and potential for price spikes, possibly influenced by seasonal trading trends or external market events. Overall, Ethereum prices remain unstable year-round, reflecting consistent market fluctuation without a strong seasonal pattern.
6. Feature Correlation Heatmap	The correlation heatmap shows strong positive relationships between most price-based features such as open, high, low, close, and moving averages (MA5, MA10, EMA12, EMA26). These variables move closely together, reflecting the overall price trend.



The top 20 most correlated features with the target variable (target_high_t2_raw) are mainly lagged prices, short-term averages, and trend indicators. This confirms that Ethereum's short-term future price is highly dependent on its recent price history and moving averages, suggesting strong temporal continuity in price movement.

3.3.3 Ripple (XRP)

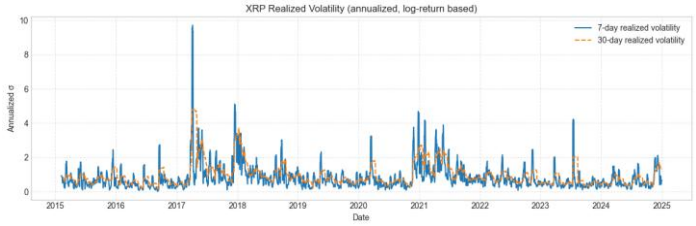
Graph	Key Insights

1. Summary of data
`Data columns (total 13 columns):`
`#  Column  Non-Null Count  Dtype`
`0  timeopen  3653 non-null  object`
`1  time_close  3653 non-null  object`
`2  time_high  3653 non-null  object`
`3  time_low  3653 non-null  object`
`4  open  3653 non-null  float64`
`5  high  3653 non-null  float64`
`6  low  3653 non-null  float64`
`7  close  3653 non-null  float64`
`8  volume  3653 non-null  float64`
`9  market_cap  3653 non-null  float64`
`10  timestamp  3653 non-null  object`
`11  date  3653 non-null  datetime64[ns, UTC]`
`12  source_file  3653 non-null  object`
`dtypes: datetime64[ns, UTC](1), float64(6), object(6)`
`memory usage: 399.5+ KB`
 	
count mean std min 25% 50% 75% max missing_%
open 3653.0 4.079195e-01 3.921454e-01 4.090780e-03 1.775820e-01 3.418074e-01 5.480213e-01 3.363570e+00 0.0
high 3653.0 4.243492e-01 4.170230e-01 4.189240e-03 1.825990e-01 3.530790e-01 5.672934e-01 3.841940e+00 0.0
low 3653.0 3.909608e-01 3.680295e-01 4.041380e-03 1.715492e-01 3.303050e-01 5.297881e-01 3.117340e+00 0.0
close 3653.0 4.084958e-01 3.931195e-01 4.090070e-03 1.778820e-01 3.420822e-01 5.480446e-01 3.377810e+00 0.0
volume 3653.0 1.788268e+09 3.360717e+09 2.481880e+04 8.840800e+07 9.766888e+08 1.889112e+09 5.172338e+10 0.0
market_cap 3653.0 1.925848e+10 1.935655e+10 1.371705e+08 7.083526e+09 1.525609e+10 2.824906e+10 1.548332e+11 0.0
	The Ripple (XRP) dataset consists of 3,653 complete daily records (2015–2024) across 13 well-structured columns, including time, price, volume, and market capitalization. No missing values were found, confirming excellent data quality. Numerical fields (open, high, low, close, volume, market_cap) are stored as float64, while time attributes are properly formatted as datetime objects. Descriptive statistics show an average daily price around USD 0.40 (std = 0.39), indicating high volatility. Trading volumes and market capitalization vary substantially — peaking at 5×10^{10} units and 1.5×10^{11} USD — confirming strong suitability for modelling dynamic price behaviour.

2. Price Trend	XRP's high-price trend exhibits long periods of stability interrupted by speculative surges — most notably during late 2017–early 2018 and mid-

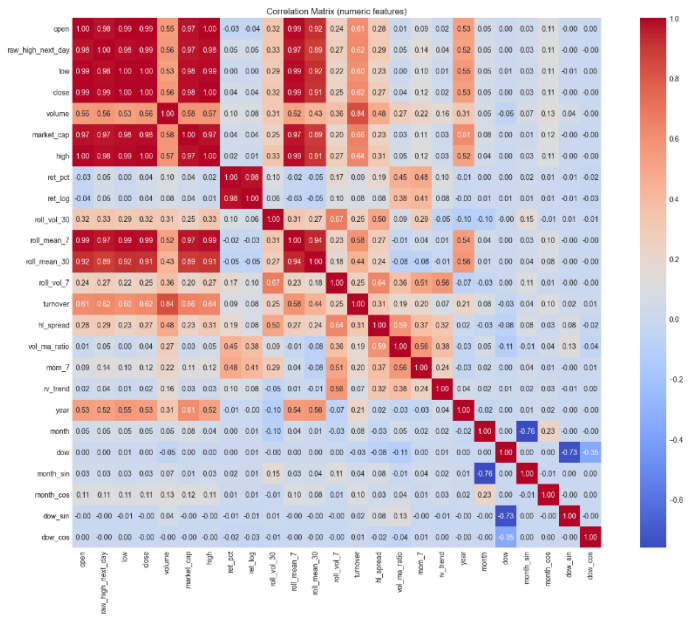
2021, followed by sharp corrections. The early period (2015–2017) remained relatively flat, suggesting limited predictive value and could be excluded from training. Post-2022, prices stabilized within a narrower range, reflecting a maturing market with lower volatility.

3. **Volatility**



Realized volatility displays recurring short-term spikes, with extreme peaks during major rallies (2017–2018, 2021). The 7-day volatility is more reactive to sudden swings, while the 30-day series captures longer trends. Since 2022, overall volatility has declined, suggesting increased market stability.

4. **Correlation**



The heatmap shows strong positive correlations (> 0.95) among price features (open, high, low, close, raw_high_next_day), reflecting shared market movement. Rolling mean and volatility features (0.3–0.7) add complementary information on trend and risk. Volume and turnover correlate moderately with prices, while cyclical date features (month_sin, dow_cos) remain largely independent, confirming seasonal diversity in the data.

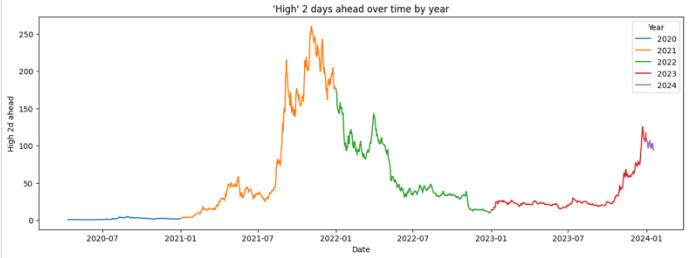
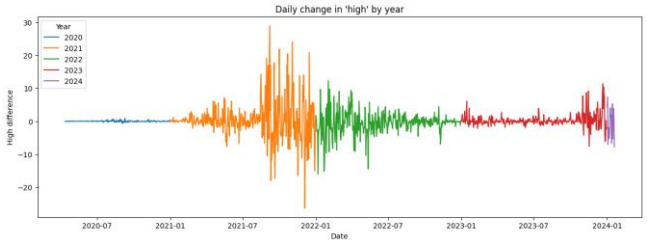
5. **Volume Patterns**

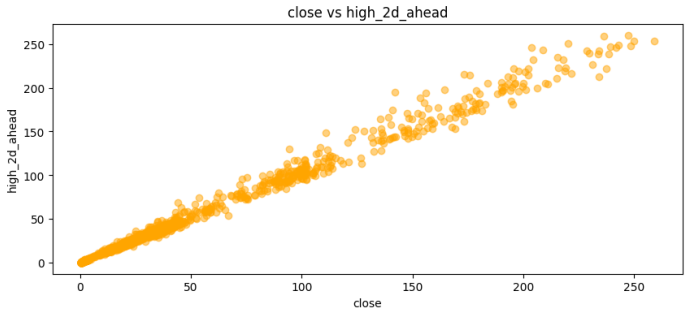
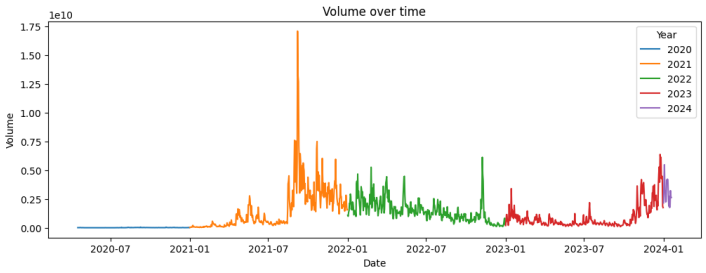
Trading volume was stable during 2015–2017, then surged during bull markets (2017–2018, 2020–2021, late 2024), aligning with major price rallies. Each spike was followed by a sharp drop, indicating short-lived speculative phases driven by investor sentiment. Overall, volume and price movements remain tightly linked during high-volatility periods.

3.3.4 Solana (SOL)

Graph	Key Insights

1. **Summary of data**
`Data columns (total 12 columns):`
`#  Column  Non-Null Count  Dtype`
`0  timeOpen  1726 non-null  object`
`1  timeClose  1726 non-null  object`
`2  timeHigh  1726 non-null  object`
`3  timeLow  1726 non-null  object`
`4  name  1726 non-null  int64`
`5  open  1726 non-null  float64`
`6  high  1726 non-null  float64`
`7  low  1726 non-null  float64`
`8  close  1726 non-null  float64`
`9  volume  1726 non-null  float64`
`10  marketCap  1726 non-null  float64`
`11  timestamp  1726 non-null  object`
`dtypes: float64(6), int64(1), object(5)`
	The Solana dataset contains 12 columns and 1,726 rows, spanning data from April 2020 to 2024. It includes time-related columns (timeOpen, timeClose, timeHigh, timeLow, timestamp) stored as objects, numeric price features (open, high, low, close), trading activity (volume), and market capitalisation (marketCap). There are no missing values, and the data types are a mix of numeric (float64, int64) and object types for timestamps. The object-type timestamp columns will need to be converted to datetime format for time-based analyses and feature engineering.

2. High 2 days ahead (Target) Trend Over Time 	From its launch in 2020 to early 2021, prices stayed low before rising sharply between mid-2021 and early 2022, marking the most volatile period. Prices then declined through 2022, remained stable during 2023, and began increasing again in late 2023–2024. This historical trend emphasises the importance of capturing both rapid fluctuations and longer-term patterns when forecasting Solana prices.
3. High Diff Over Time 	The chart highlights a dramatic increase in Solana's volatility, peaking in 2021. 2020 was marked by extremely low daily price swings, but this quickly transitioned into 2021 (orange), which shows the largest, most frequent daily differences, reflecting its major growth phase and high turbulence. In contrast, 2022 and 2023 exhibit a

	noticeable return to moderate volatility, with daily movements substantially smaller than the 2021 peak.
4. Close against High 2 days ahead (Target) 	The scatter plot clearly visualizes a strong positive linear relationship between Solana's closing price and its high price two days later, with the data points clustering tightly along a diagonal line. This tight clustering demonstrates that the current closing price is a highly accurate and strong predictor of the short-term future high price. Specifically, as the current price of SOL increases, the expected high price two days into the future reliably increases as well.
5. Volume over time 	Solana's trading volume has experienced major fluctuations over time. Volume was minimal during the early stages in 2020, before surging in 2021 (orange) to a peak of around 17.5 billion. This was followed by a sharp decline and a period of cooling throughout 2022 (green). Trading activity remained subdued for most of 2023 (red), but saw a notable late-year rebound that continued into 2024 (purple), although volume levels have yet to reach the 2021 peak.

4. Data Preparation

This section outlines the data preparation process for all four cryptocurrency assets — Bitcoin (BTC), Ethereum (ETH), Ripple (XRP), and Solana (SOL).

Each dataset underwent data cleaning, transformation, and feature engineering tailored to its characteristics before being split chronologically for model training and evaluation.

4.1 Bitcoin (BTC)

1. Data Cleaning

Bitcoin data was sourced from CoinDesk and CoinGecko then combined into a unified dataset for modelling. Data integrity of is verified through checks for missing values, duplicate records, and appropriate datatype casting. Although the intended business target is the next-day (T + 1) HIGH price, the model uses the two-day ahead (T + 2) HIGH price as the operational target variable. This adjustment is necessary because the HIGH price for a given day is only known after the trading day concludes. For example, the HIGH price for October 30th becomes available only on October 31st. Therefore, when generating a next-day prediction for October 31st, the most recent complete HIGH data corresponds to October 29th. By using this target, the model remains causally valid and operationally realistic, ensuring that predictions are based only on data that would have been available at the time of inference.

2. Data Transformation

To prepare our data for modelling, we standardise numerical variables and encode time-based features. We create a pipeline to wrap our numerical and time feature transformations, creating and saving a preprocessor object that streamlines data transformation in production.

We standardize numerical features to improve model convergence and predictive stability. Each feature is evaluated across multiple normality transformations (log, square root, Box–Cox, Yeo–Johnson, quantile, or none) to identify the transformation that maximizes a composite normality score. This score combines statistical normality (D'Agostino's K-squared test p-value) with penalties for skewness, kurtosis, and outlier fraction. The selected transformation is applied to each feature before final standardization. Time-based features are transformed into a numerical format, seconds since midnight, to allow models to train on this critical information. We sine and cosine transform these into cyclical variables to capture the contiguous nature of the lowest and highest values of these features.

3. Feature Engineering

Several features were created to capture nuanced aspects of Bitcoin market dynamics:

- **Price:** Price-based indicators such as interval volatility, net price change, normalized return, and normalized volatility were calculated to represent short-term price movements and variability.
- **Momentum:** To account for price momentum, upper and lower pressure indicators were created to reflect the distance between closing prices and the daily extremes.
- **Messaging:** Message-based indicators were also developed to capture intra-period messaging behaviour, including message volatility derived from high and low message values and message trend derived from first and last message values. Message activity duration and message velocity indicators are created, which capture the timing and rate of message-driven activity throughout the trading day.
- **Volume and Liquidity:** volume and liquidity indicators were constructed to represent trading activity and market conditions, including average trade price, turnover ratio, market depth, market valuation normalized by high price, and the ratio of total index updates to volume.

Together, these engineered indicators provide a richer feature set that integrates price dynamics, message behaviour, and market liquidity signals. Since past values of predictor and target variables are likely to contain critical information for future values, we create lags of each feature. We vary lag periods across experiments to optimize model performance. Since Fibonacci ratios are often referenced in market trading decisions for identifying price supports and resistance levels, we create a Fibonacci causal lookback of our target variable that detrends the series over a fixed window (in this case, from T–10 to T–2) using Fibonacci-weighted values. This approach removes the trend component influenced by recent price movements while weighting past observations according to the naturally occurring Fibonacci sequence. Because Fibonacci-based investment strategies are commonly reflected in market behaviour, detrending with a Fibonacci-weighted series may help capture these trading dynamics as they are embedded in Bitcoin price movements. This choice of detrending is further supported by our EDA, which showed that the Fibonacci-detrended series exhibited smoother daily price movements, potentially allowing the model to better capture overarching directional trends in Bitcoin's daily price behaviour.

4. Data Splitting

A chronological data splitting strategy was employed to prevent lookahead bias.[1] This approach ensures that all model evaluation simulates a realistic forecasting scenario, where future values are never used to inform past predictions. Unlike random splits commonly used in cross-sectional datasets, time-based splits preserve the integrity of temporal dependencies and autocorrelation structures present in the Bitcoin price series. The dataset is divided sequentially into training (22 October 2024 to 28 May 2025), validation (29 May to 9 August 2025), and test (10 August to 21 October 2025) subsets, maintaining the natural progression of time. Specifically, 60% of the earliest observations are used for model training, 20% for validation and hyperparameter tuning, and the remaining 20% for final out-of-sample testing.

4.2 Ethereum (ETH)

1. Data Cleaning

The dataset underwent a thorough cleaning process to ensure data quality and reliability before modelling. First, null values were checked across all columns to confirm data completeness. No missing values were detected, indicating that every feature contained valid entries suitable for training. This validation step helps avoid potential model errors and ensures accurate statistical calculations.

Next, duplicate records were identified and removed to prevent data redundancy and bias. In time-series data such as cryptocurrency prices, duplicates can distort trends and lead to overfitting. Fortunately, no duplicate rows were found, confirming the dataset's uniqueness and integrity.

Overall, this step verified that the dataset is clean, consistent, and ready for further transformation and feature engineering.

2. Data Transformation

The dataset was transformed to make time-based and numerical features suitable for model learning while avoiding data leakage.

Cyclical time encoding. Month and day-of-week columns were converted into cyclical features using sine and cosine transformations (month_sin, month_cos, dow_sin, and dow_cos). This approach represents time as a continuous cycle, ensuring that December and January are treated as consecutive rather than distant values. It allows the model to capture recurring seasonal and

[1] Experiments with rolling and expanding windows exhibited model underperformance, suggesting that temporal resampling did not enhance predictive stability or accuracy for this dataset.

weekly trading patterns more accurately. Raw columns such as month and day_of_week were removed to maintain consistency in the feature list.

Removal of time columns. Columns containing timestamps (timeOpen, timeClose, timeHigh, timeLow, timestamp) were dropped before model training. These columns only represent chronological order and could lead to data leakage if future timestamps influenced the learning process. Removing them ensures that the model focuses on meaningful numerical features derived from historical prices and technical indicators.

Feature scaling and saving. All features were standardised using StandardScaler, which was fitted on the training set and then applied to validation and test sets. The fitted scaler was saved as standard_scaler.pkl to maintain consistent data scaling during model deployment. Using the same scaling parameters ensures stability and accuracy when processing new input data in production.

3. Feature Engineering

The dataset was enhanced with multiple categories of features to capture market behaviour and temporal dynamics.

Price-based features:

Metrics such as daily_return, high_low_range, and close_open_diff were created to describe short-term price movements and daily volatility.

Trend and momentum features:

Moving averages (MA5, MA10, MA20, MA50) and rolling volatility (rolling_std_7d) were added to reflect market direction and strength. The RSI indicator was included to measure buying and selling pressure.

Technical indicators:

Advanced indicators like MACD, Bollinger Bands, Stochastic Oscillator, ATR, and Ichimoku Cloud were generated to capture momentum shifts, volatility, and support–resistance zones.

Time-based and lag features:

Features such as day_of_week, month, is_weekend, and lag values for open, close, high, and low prices were used to represent seasonal trends and sequential dependencies.

Volume-related features:

Variables like volume_change, volume_MA7, and price_volume_ratio were introduced to connect trading activity with price dynamics.

After removing missing values from rolling calculations, the final dataset contained 59 engineered features, providing a rich foundation for model training.

4. Data Splitting

The dataset was divided using a time-aware splitting strategy to preserve chronological order and prevent information leakage.

Holdout test set:

The last 300 records (around 9% of the data) were reserved as a final test set using the holdout_tail function. This set remained completely unseen during model training and validation, ensuring an unbiased evaluation of model performance.

Rolling cross-validation:

A walk-forward validation approach was implemented through the time_cv_splits function. The data was split into 30 folds, each consisting of 1,200 training samples and 60 validation samples, with a gap of two days to protect the forecast horizon. This method mimics real trading conditions by training on past data and validating on future data.

Final split extraction:

The most recent fold was selected for model training and evaluation, producing input shapes of (1200, 24) for training, (60, 24) for validation, and (300, 24) for testing. This structured setup ensures temporal integrity and reliable model generalisation on unseen future data.

4.3 Ripple (XRP)

1. Data Cleaning

All 20 XRP CSV files were merged into a single dataset of 3,653 daily records (2015–2024). Duplicates were removed using the date key, and variables were standardized into correct data types (float64 for prices and volumes, datetime for time fields). No missing values were present in the raw data.

The target variable was defined as the two-day-ahead high price (t+2), allowing the model to learn realistic future trends while stabilizing variance through log transformation. Rows containing incomplete rolling-window values were dropped to avoid bias.

2. Data Transformation

To address the heavy right-skew and outliers typical of cryptocurrency markets, key features (close, volume, turnover, and rolling averages) were log-transformed. All numerical features were then standardized using a RobustScaler, which scales data based on the interquartile range, making it resilient to volatility spikes.

A unified ColumnTransformer pipeline was implemented to apply these transformations consistently across training, validation, and testing datasets, ensuring reproducibility and preventing data leakage.

3. Feature Engineering

A diverse set of features was engineered to capture multiple market dimensions—trend, volatility, and liquidity:

- **Trend:** roll_mean_7, roll_mean_30 (short- and medium-term moving averages)
- **Volatility:** roll_vol_7, roll_vol_30 (annualized rolling standard deviations of log returns)
- **Liquidity and Range:** turnover = volume × close, hl_spread = (high − low)/close
- **Momentum and Regime Indicators:** mom_7, rv_trend = roll_vol_7 / roll_vol_30
- **Cyclical Calendar Features:** sine–cosine encodings of month and weekday to capture periodic trading patterns.

Feature selection combined correlation and mutual-information analysis to minimize redundancy. The final feature set used for modelling included: 'close', 'roll_mean_7', 'roll_mean_30', 'turnover', 'volume', 'roll_vol_7', 'roll_vol_30'.

4. Data Splitting

To preserve temporal integrity and mimic real-world forecasting, the dataset was split chronologically:

- **Training:** 2018–2022 (1,826 records)
- **Validation:** 2023 (365 records)
- **Testing:** 2024 (366 records)

This setup ensures that the model learns from past market behaviour, tunes hyperparameters on recent data, and evaluates on unseen future data, thus replicating realistic trading conditions.

4.4 Solana (SOL)

1. Data Cleaning

The dataset contained no erroneous values and no duplicate rows were found, so minimal cleaning was required. Time-related columns were stored as objects and converted to datetime format for analysis, while numeric columns were converted to float to ensure consistency.

2. Data Transformation

No explicit transformation was needed, as most features were numeric and XGBoost does not require scaling. However, creating lagged features and target as high price two days ahead introduced NaN values, which were removed before training.

3. Feature Engineering

To improve the model's predictive capability, a set of lag-based and derived features was engineered from the original Solana dataset.

The final features included: 'close', 'high', 'close_lag1', 'close_lag2', 'high_lag1', 'high_lag2', 'high_diff', 'close_diff', 'close_ma3', 'volume', 'volume_lag1', 'volume_lag2', and 'volume_ma3'.

The inclusion of lag features (e.g., close_lag1, high_lag2) enables the model to learn temporal dependencies and short-term price trends from previous days. The difference features (high_diff, close_diff) capture day-to-day fluctuations and momentum, offering insight into recent market direction. Moving average features (close_ma3, volume_ma3) help smooth out short-term volatility and provide a broader trend perspective. Finally, volume-related features and their lags (volume_lag1, volume_lag2) were incorporated to reflect trading activity, as sudden changes in volume often precede major price movements (Podobnik et al., 2009) [9].

4. Data Splitting

In preparation for model training, data from 2021 onwards was used. Solana's early price data from 2020, shortly after its launch show abnormally low prices that may introduce unwanted noise (Section 3.4.4). Excluding this period ensures a more stable and representative dataset for learning meaningful price patterns. The dataset was then split into 90% training and 10% testing, providing a model with enough data to learn, while maintaining a reserved portion for performance evaluation on unseen data.

4.5 Temporal Prediction Structure

To ensure temporal integrity, all models adopt a two-day-ahead prediction structure, where yesterday's complete data (Day) is used to forecast the high price two days ahead (Day + 2).

Since current day market data (Day + 1) is incomplete until market closure, this approach avoids information leakage and aligns with real-world trading conditions.

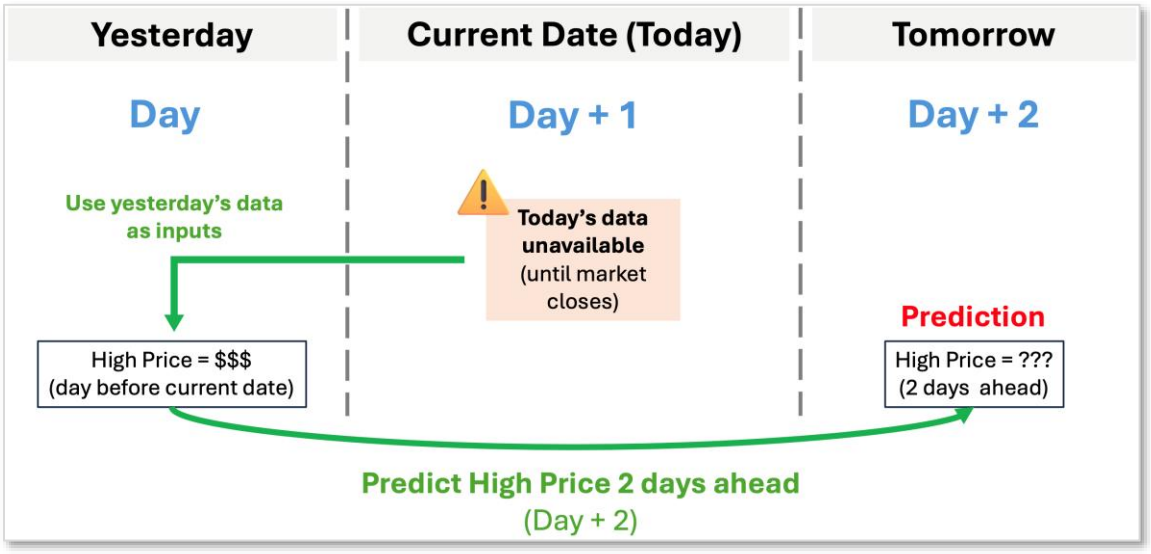


Figure 1: Rationale for Using Day +2 as the Prediction Target

5. Modelling

a. Approach 1: CatBoost for Bitcoin High Price Prediction

Model Selection Rationale

We select the powerful gradient boosting algorithm CatBoost for our Bitcoin prediction. CatBoost delivers best-in-class prediction speed and performance by implementing ordered boosting and in-built categorical handling (Dorogush et al., 2017 [5]; Dorogush et al., 2023 [6]). CatBoost can natively handle multicollinearity, non-linear relationships, internal regularization, and missing values, making it particularly appropriate for this use case given the observed multicollinearity of numerical variables and a maximum information feature selection strategy where all features are utilised. CatBoost's built-in regularization mechanisms, including parameters such as bagging temperature, leaf penalties, and random split scoring strength, enable the model to effectively identify and prioritize the most influential predictors while mitigating overfitting risks.

Modelling Workflow

We begin our modelling workflow by performing the feature transformation and engineering steps outlined in Section 4.1: creating our target variable, creating engineered features and lags, standardizing numerical variables and encoding time-based features, and splitting the data according to the described temporal split strategy. We select all available predictors for comprehensive maximum information feature strategy that maximizes the amount of information from our dataset and saving time under the project's time constraints, leveraging the strength of CatBoost's native handling of multicollinearity and built-in regularization mechanisms.

We then start experimentation by varying the target and predictor lag periods to identify the optimal lag configuration for model performance. Next, we perform Bayesian hyperparameter optimization using Hyperopt, refining parameter spaces and adjusting upper limits to balance model complexity with deployability requirements. Finally, the optimal model is trained on the combined training and validation datasets and evaluated against the test set to determine final model performance.

Model performance was primarily evaluated using Mean Absolute Error due to its direct interpretability and balanced sensitivity to prediction errors. MAE emphasizes average short-term predictive accuracy without disproportionately penalizing extreme deviations, as would occur with Root Mean Squared Error (RMSE). We additionally report RMSE for a holistic comparison between models, Mean Absolute Percentage Error (MAPE) for scale-independent comparison, and R-squared (R^2) to quantify the proportion of variance explained by models. Finally, SHAP analysis was

conducted to evaluate feature contributions, providing insights to guide future experiments and supporting interpretability during model inference.[2]

Specific Feature Engineering

We utilise the features described in section 4.1.3, to capture nuanced price, momentum, messaging, and volume elements of the Bitcoin market. Features utilised in the model are as follows:

Price-based features

- **Interval volatility** = Daily high price − daily low price
- **Net price change** = Closing price − opening price
- **Normalized return** = Closing price ÷ opening price
- **Normalized volatility** = Daily high price ÷ daily low price

Price momentum / pressure indicators

- **Upper pressure** = Daily high price − closing price
- **Lower pressure** = Closing price − daily low price

Message-derived features

- **Message volatility** = Highest message value − lowest message value
- **Message trend** = Last message value − first message value
- **Message activity duration** = Time of last message − time of first message
- **Message velocity** = (Last message value − first message value) ÷ (time of last message − time of first message)

Volume / liquidity features

- **Average trade price** = Trading volume ÷ quoted trading volume
- **Turnover ratio** = Trading volume ÷ market capitalisation
- **Market depth** = Market capitalisation ÷ trading volume
- **Valuation vs. high** = Market capitalisation ÷ daily high price
- **Message-trading ratio** = Total index updates ÷ trading volume

Hyperparameter Tuning

We experiment with different lag windows to determine the optimal predictor and feature lag periods. Then, Catboost hyperparameters are tuned using Hyperopt on the cross-validation set,

[2] SHAP results, presented as a Beeswarm plot illustrating the feature importance for correct predictions of the top-performing model, are provided in Appendix A.

testing several hyperparameter spaces to optimize our model. We list the hyperparameters tuned, their descriptions, and the hyperparameter search space that produced the top-performing model in the table below:

Hyperparameter	Description/Impact	Top Performing Search Range
Iterations	Maximum number of trees in a forest. Tradeoff between training duration and performance.	**100 to 1000**, steps of 50.
Depth	The maximum depth of individual decision trees. Tradeoff between training duration and performance.	Integer between 3 **and 10**, steps of 1.
Learning rate	Learning rate of weights. Tradeoff between training duration and performance.	Log-Uniform sample of float values between **0.01 and 0.3.**
Leave L2 regularization	L2 regularization applied to leaves of decision trees. Reduces overfitting and improves model generalization.	Log-Uniform sample of float values between **1 and 10.**
Bagging temperature	Adjusts Bayesian boosting and randomness in trees. Regularization: reduces overfitting and improves generalizability.	Uniform sample of float values between **0 and 1.**
Random strength	Adjusts randomness in decision splits. Regularization: reduces overfitting and improves generalizability.	Uniform sample of float values between **1 and 20.**

The top-performing model hyperparameters were selected as follows:

Hyperparameter	Value
Lookback	7 days
Evaluation metric	MAE
Iterations	100
Depth	4
Learning rate	0.10224695330819393
Leaf L2 regularization	2.498604814642479

Bagging temperature	0.7401894148400094
Random strength	4.65385901310944
Verbose	0
Random seed	42

b. Approach 2: LinearRegression for Ethereum High Price Prediction

Model Selection Rationale

Three regression algorithms were selected to model cryptocurrency price prediction: **Lasso Regression**, **Random Forest Regressor**, and **AdaBoost Regressor**. Each model offers different strengths in handling complex financial time-series data as below rationale.

Model	Type	Rationale for selection
Linear Regression	Linear, parametric	Chosen for its simplicity, interpretability, and strong overall performance that outperformed the baseline.
Random Forest Regressor	Ensemble of decision trees (bagging)	Selected to test non-linear relationships and feature interactions that linear models might miss.
AdaBoost	Ensemble of weak learners (boosting)	Included to evaluate how boosting improves prediction by focusing on difficult-to-learn samples.

Modelling Workflow

The modeling workflow followed a structured and reproducible pipeline to ensure consistent training, validation, and testing across all models.

First, the cleaned and transformed dataset was split using a **time-aware strategy** to preserve the chronological order of cryptocurrency prices. The **holdout_tail** function reserved the last 300 records as the final test set, while the remaining data was used for rolling cross-validation. A **walk-forward validation** approach with 30 folds was applied using the **time_cv_splits** function, allowing the models to train on past data and validate on future unseen intervals. This mimics real-world trading scenarios where only past information is available for forecasting.

For each fold, features were standardized using the **StandardScaler** fitted on the training data. This scaler was saved as standard_scaler.pkl and reused for all models to maintain consistent scaling during evaluation and deployment.

Each model was trained on the same split structure:

1. **Train on historical data (1,200 samples)**
2. **Validate on the next time window (60 samples)**
3. **Evaluate final performance on the test set (300 samples)**

Evaluation metrics included MAE, RMSE, MAPE, and R^2, providing a balanced assessment of both accuracy and generalization.

This systematic workflow ensured fair model comparison, prevented data leakage, and aligned the evaluation with real-world forecasting conditions.

Specific Feature Engineering

Feature engineering and selection were designed to enhance model learning by capturing meaningful market patterns while reducing redundancy and multicollinearity. The process involved three main stages: feature creation, selection, and validation.

1. Feature creation:

A broad set of features was engineered to represent different market behaviours:

- **Price and trend features:** Variables such as daily_return, close_open_diff, and high_low_range described short-term movement and volatility.
- **Moving averages and volatility:** Indicators like MA5, MA10, MA20, and rolling_std_7d captured medium- and long-term trend strength.
- **Technical indicators:** Features such as **RSI**, **MACD**, **Bollinger Bands**, and **Ichimoku Cloud** highlighted momentum, volatility, and potential support–resistance zones.
- **Time-based and lag features:** Encoded cyclic time variables (month_sin, month_cos, dow_sin, dow_cos) and lag values for open, high, low, and close enabled the model to capture seasonal and temporal dependencies.
- **Volume-based metrics:** Features like volume_change, volume_MA7, and price_volume_ratio linked price dynamics with trading activity.

2. Feature selection:

Two statistical methods were applied independently — **correlation analysis** and **mutual information (MI)** — to assess the relationship between each feature and the target variable.

- The **top 20 features** from each method were combined using a **union approach**, ensuring that both linear and nonlinear relationships were represented.

- Highly correlated variables (correlation coefficient > 0.9) were removed to prevent redundancy.
- Finally, the **Variance Inflation Factor (VIF)** was calculated for each feature, and only those with **VIF < 10** were retained to ensure low multicollinearity and stable model coefficients.

After this multi-step filtering process, **22 features** were selected as the final input set. These features effectively balance richness and interpretability, allowing the models to generalise well while avoiding overfitting.

Hyperparameter Tuning

Hyperparameter tuning was performed to optimise each model's configuration and improve predictive performance while maintaining generalisation on unseen data. A randomised search approach was used for efficiency, exploring a wide range of parameter combinations without the high computational cost of grid search.

Tuning was conducted on the latest rolling split, using a single fold defined by the cv_split configuration to reflect real-world forecasting conditions. Model performance during tuning was evaluated using negative root mean squared error (−RMSE) as the scoring metric, ensuring consistent comparison across models.

1. Lasso and Ridge Regression:

The regularisation strength (**α**) was searched within a logarithmic range from $\mathbf{10^{-4}}$ **to** $\mathbf{10^{2}}$ using **RandomizedSearchCV** with 40 iterations. The **Lasso Regression** model was selected as the best performer with alpha = 1.35, offering effective feature selection and strong regularisation for high-dimensional financial data.

2. Random Forest Regressor:

Parameters such as the number of trees (n_estimators), maximum depth (max_depth), minimum samples per split and leaf, maximum features, and bootstrap options were tuned. The best-performing configuration included n_estimators = 200, max_depth = 15, min_samples_split = 5, min_samples_leaf = 2, max_features = 'sqrt', and bootstrap = False.

This setup balanced bias and variance while capturing nonlinear feature interactions.

3. AdaBoost Regressor:

Key parameters tuned included the number of estimators, learning rate, and the complexity of the base learner. The optimal configuration used n_estimators = 200, learning_rate = 0.01, and a base DecisionTreeRegressor with max_depth = 6. This combination allowed the model to iteratively improve weak learners and better capture subtle price trends.

Each model was refitted on the combined training and validation data using the best hyperparameters. This ensured stable, well-regularised models ready for evaluation on the unseen test set.

c. Approach 3: LightGBM for Ripple High Price Prediction

Model Selection Rationale

To forecast the t+2 high price of XRP, we compared three complementary approaches to balance interpretability, non-linearity, and controllable complexity:

Model	Type	Rationale for selection
Linear Regression	Statistical baseline	Provides a transparent benchmark to test the predictive strength of engineered features and assess linear relationships between market indicators and the target variable(ValueAdder, n.d.) [11].
Random Forest Regressor	Ensemble, non-linear baseline	Captures more complex interactions and non-linear effects that a linear model may overlook, especially useful in volatile financial data (BairesDev, n.d.) [1].
LightGBM Regressor	Gradient boosting – final approach	Chosen as the final model for its ability to model complex relationships efficiently, handle multi-scale numeric features, and offer strong regularization with explainable feature importance (Song & Chen, 2024)[10].

Modelling Workflow

All models were trained on the same preprocessed dataset to ensure fairness and comparability.

1. Consistent preprocessing (see Section 4.3): Apply log transforms to skewed features and RobustScaler to reduce outlier influence.
2. Preserve chronology: Split the data into Train (2018–2022), Validation (2023), and Test (2024) to prevent leakage and mirror real-world forecasting.
3. Baseline training: Fit a Linear Regression model as the benchmark.
4. Model tuning: Tune LightGBM using TimeSeriesSplit with RandomizedSearchCV with MAE as the objective.
5. Post-processing for evaluation (see Part 6): Convert predictions back to price space:

$$\text{High}_{t+2} = \text{High}_t \times e^{\hat{y}}$$

Specific Feature Engineering

All models used the same final feature set derived from earlier feature engineering steps in Section 4.3 to ensure consistency and comparability:

Feature Group	Features	Purpose
Trend indicators	roll_mean_7, roll_mean_30	Capture short- and medium-term market momentum.
Volatility measures	roll_vol_7, roll_vol_30	Represent realized volatility across time horizons.
Liquidity features	volume, turnover	Indicate market activity and investor participation.
Price anchor	Close	Represents the reference level for forecasting future highs.

Model-based importance (LightGBM gain)

To validate the design and distinctiveness of these signals, we inspected LightGBM's gain importance (Figure 5.1). Ranked drivers:

1) roll_vol_30, 2) roll_mean_30, 3) volume, 4) roll_mean_7, 5) close, 6) roll_vol_7, and 7) turnover

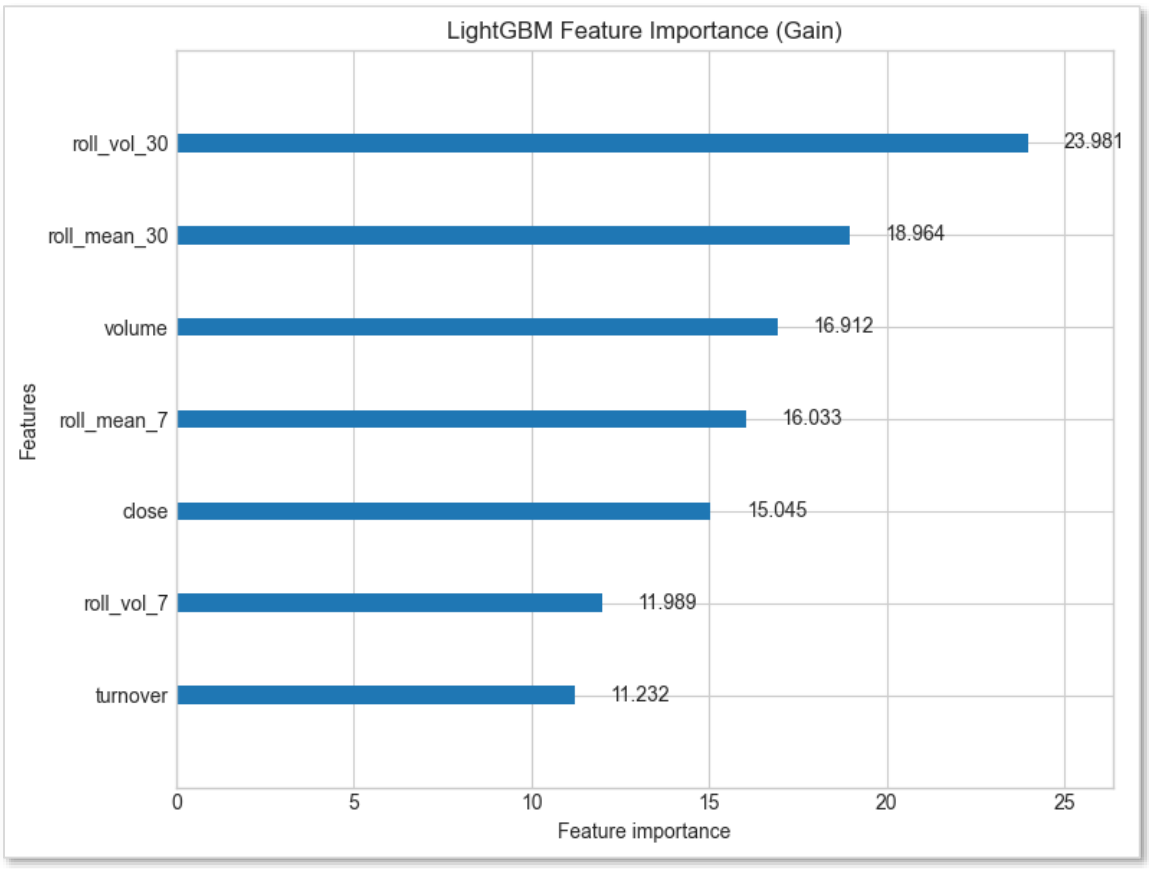


Figure 2: LightGBM Feature Importance

Hyperparameter Tuning

To balance complexity and generalisation over time, model tuning employed TimeSeriesSplit (5 folds) to preserve chronology and MAE as the optimisation objective. RandomizedSearchCV was used for LightGBM to efficiently search a broad, regularised parameter space.

Model 1 – Linear Regression (Baseline)

- **Type:** Ordinary Least Squares (no regularisation)
- **Preprocessing:** Shared pipeline (log transforms + RobustScaler)
- **Role:** Transparent benchmark of linear signal

Model 2 – Random Forest Regressor

- **Type:** Bagging ensemble of decision trees
- **Key setup:** n_estimators=300, max_depth=10, random_state=42
- **Role:** Non-parametric comparison to capture nonlinear interactions

Model 3 – LightGBM Regressor (Final Approach)

- **Type:** Gradient Boosting Decision Trees (histogram)
- **Tuning** (TimeSeriesSplit + RandomizedSearchCV; objective = MAE)
- **Search space (summary):**

Parameter	Range / Options	Select Configuration	Purpose
Num_leaves	15-50	35	Max leaf nodes per tree; finer splits capture more detail (too high can overfit).
Max_depth	-1, 4, 6, 8, 10	4	Caps tree depth to control complexity *(-1 = no limit)*
Learning_rate	0.005-0.05	0.005	Small step size per tree; slower but usually better generalisation.
N_estimators	500-2000	500	Number of boosting rounds; more trees compensate for the small learning rate.
Min_child_samples	10-100	100	Minimum samples in a leaf; prevents tiny, noisy splits.
Subsample	0.6-1.0	0.8	Row sampling per tree; reduces variance and overfitting.
Consample_bytree	0.6-1.0	0.8	Feature sampling per tree; decorrelates trees.
Reg_alpha (l1)	0.0-1.0	0.1	L1 regularisation; encourages sparsity in splits.
Reg_lambda (L2)	0.0-2.0	1.0	L2 regularisation; shrinks weights for stability.

d. Approach 4: XGBoost Regressor for Solana High Price Prediction

Model Selection Rationale

The XGBoost Regressor model was selected due to its effectiveness in handling time series regression problems over other state of art models (Zhang et al., 2021) [12]. XGBoost's gradient boosting framework allows it to capture nonlinear temporal relationships and complex feature interactions that often arise in financial data (Chen & Guestrin, 2016) [3]. Unlike traditional linear models, it can automatically account for variable importance, manage missing values and reduce overfitting through regularisation. Additionally, its scalability and efficiency make it suitable for iteratively tuning parameters and performing cross-validation on large datasets.

The baseline model is a simple naive predictor that assumes the next day's high price will be equal to yesterday's high price. This is considered a strong baseline in financial time series because prices often exhibit short-term autocorrelation, meaning the previous day's price is often a reasonable first approximation. Comparing XGBoost against this baseline allows us to quantify the added predictive value of the model beyond simple historical trends.

Modelling Workflow

To appropriately assess generalisation over time, the workflow outlined in Figure. 3 illustrates the combined use of Time Series Cross-Validation (TSCV) and Hyperparameter Optimisation (Hyperopt). TSCV was implemented with four folds (n=4) to evaluate the model's performance across multiple sequential train-validation splits avoiding any leakage of future data into the training process. This validation framework offered a consistent measure of the model's temporal performance. Hyperopt was then incorporated to efficiently search the hyperparameter spaces and identify the configuration that minimised the prediction loss, measured by RMSE. The optimisation process was conducted over 50 iterations, providing a balance between search depth and computational efficiency. Hyperopt was chosen for its Bayesian optimisation approach, which is more efficient than traditional grid or random search methods as it models the objective function and strategically explores promising regions of the parameter space (Bergstra et al., 2015) [2].

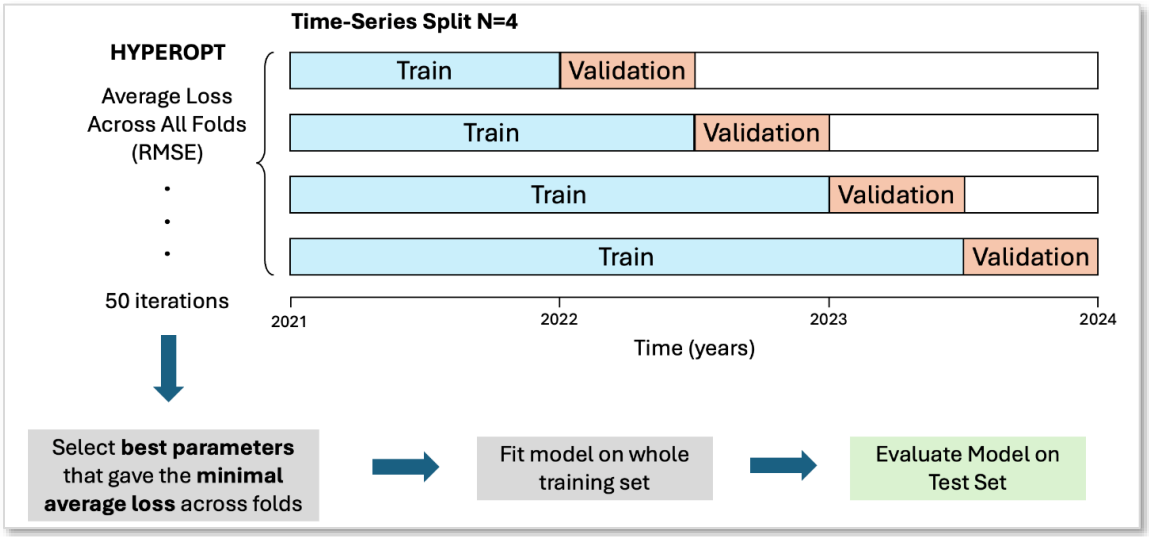


Figure 3: Hyperopt and Time-Series Cross Validation Hyperparameter Tuning Workflow

Specific Feature Engineering

The specific feature engineering details are already discussed in Section 4.4 of the report. It includes the creation of lag-based, difference, and moving-average features designed to capture temporal dependencies, market momentum, and broader trends. These engineered features aim

to improve the model's ability to learn from recent market behaviour and improve the accuracy of price predictions.

Hyperparameter Tuning

The following hyperparameters were tuned to optimise model performance using Hyperopt:

- n_estimators: Specifies the number of trees in the ensemble. A broad search range enables Hyperopt to find a balance between underfitting (too few trees) and overfitting (too many trees).
- max_depth: Defines the maximum depth of each tree. Shallower trees help prevent overfitting and promote generalisation, whereas deeper trees allow the model to capture more complex relationships in the data.
- learning_rate: Controls the contribution of each tree to the overall model. Lower values encourage slower but more stable learning, while higher values speed up convergence at the risk of overshooting optimal solutions.
- subsample: Determines the fraction of training samples used to grow each tree. Setting this value below 1 introduces randomness, which helps reduce overfitting and enhances generalisation.
- colsample_bytree: Specifies the fraction of features considered when building each tree. Randomly sampling features acts as a regularisation mechanism and prevents overfitting, especially when input features are correlated.
- gamma: Sets the minimum loss reduction required to make a split. Higher values make the model more conservative, avoiding unnecessary splits and reducing overfitting.
- reg_alpha: Corresponds to the L1 regularisation term, which encourages sparsity in leaf weights and helps control model complexity.
- reg_lambda: Corresponds to the L2 regularisation term, which penalises large leaf weights to improve model stability and generalisation.

6. Evaluation

6.1 Evaluation Metrics

The models were evaluated using MAE, RMSE, MAPE, and R^2, which collectively provide a balanced view of absolute error magnitude, percentage-based deviation, and explanatory strength. These metrics align with the business objective of accurately forecasting next-day high prices while minimizing financial risk from prediction errors.

Metric	Description	Relevance to Project Goals
MAE (Mean Absolute Error)	Measures the average magnitude of prediction errors in currency units.	Reflects practical error size in USD — crucial for price prediction.
RMSE (Root Mean Squared Error)	Penalizes larger errors more than MAE.	Highlights the model's sensitivity to high volatility or outliers.
MAPE (Mean Absolute Percentage Error)	Expresses average error as a percentage of actual price.	Intuitive measure for traders and analysts to interpret relative accuracy.
R^2 (Coefficient of Determination)	Measures how well the model explains variability in actual prices.	Indicates predictive power — closer to 1.0 = better model fit.

6.2 Results and Analysis

6.2.1 Individual Model Performance

Coin	Type	MAE	RMSE	MAPE(%)	R^2
BTC	Baseline – Naïve T-2 Lookback of high price	2354.50	3004.92	2.05	0.54
	Final Approach – CatBoost	**1951.93**	**2562.67**	**1.70**	**0.65**
	Interpretation	This model (MAE ≈ 1951.93) outperforms the naive baseline (MAE ≈ 2354.50) by approximately 402.57, representing a 17% improvement in predictive accuracy. It also outperforms the baseline across all other performance metrics (RMSE ≈ 2562.67, MAPE ≈ 1.70%, R^2 ≈ 0.65), indicating robust predictive outperformance and greater variation explainability. The final model therefore			

		demonstrates stronger performance across all evaluations. These results may be further improved by the inclusion of additional technical indicators such as RSI to further enhance model generalization. Augmenting the feature selection strategy may also yield stronger model results. SHAP analysis, plotted in Appendix A, highlights that features such as normalized return, day-1 lag of upper pressure, message velocity, lower pressure, and day-1 lag of message velocity contribute most strongly to accurate predictions. Conversely, features including first message value and day-7 lag open price exhibit comparatively low SHAP importance, indicating minimal contribution to model accuracy. These insights suggest that future iterations could benefit from a more selective feature inclusion strategy: excluding features with consistently low SHAP importance to reduce noise and further enhance model generalization.			
ETH	Baseline – High Price Prediction	100.08	137.25	3.13	0.92
	Random Forest Regressor	115.47	148.94	3.67	0.91
	AdaBoost	108.08	141.13	3.42	0.92
	Final Approach – Linear Regression	**92.25**	**126.09**	**2.87**	**0.94**
	Interpretation	The results show that the **Linear Regression model** performed the best among all tested models for predicting Ethereum's next two-day high price. It achieved the lowest MAE (92.25) and RMSE (126.09), with a MAPE of 2.87% and the highest R^2 of 0.94, indicating stronger predictive accuracy and a better fit than the baseline persistence model (MAE 100.08, RMSE 137.25, R^2 0.92). In contrast, both the **Random Forest Regressor** and **AdaBoost** performed worse than the baseline, suggesting they may have overfit or failed to generalize well to unseen data. Overall, the results imply that Ethereum's short-term price behavior is well captured by linear relationships between features, making Linear Regression the most reliable and interpretable model for this prediction task.			

XRP	Baseline – Linear Regression	0.0410	0.0927	4.11	0.97
	Random Forest Regressor	0.459	0.1091	4.31	0.96
	Final Approach – LightGBM	**0.383**	**0.0909**	**4.01**	**0.97**
	Interpretation	LightGBM performs best overall, indicating it captures useful non-linear relationships while maintaining strong generalisation. Linear Regression is a close second, showing that your engineered features already provide a strong linear signal. Random Forest trails, suggesting weaker generalisation and mild overfitting to mid-range regimes.			
SOL	Baseline – High Price Prediction	6.51	8.78	3.58	0.95
	Final Approach – XGBoost	**6.50**	**8.65**	**3.58**	**0.95**
	Interpretation	The XGBoost model provides a modest improvement over the baseline, reducing prediction errors and slightly increasing the proportion of variance explained. Overall, it demonstrates a small but consistent gain in predictive performance for Solana's next-day high price.			

6.2.2 Comparative Analysis

Across all four assets, each final model outperformed its baseline, though predictive gains differed according to the asset's volatility and data complexity. CatBoost and LightGBM achieved the strongest predictive accuracy, while Linear Regression and XGBoost demonstrated computational efficiency and stability.

For Bitcoin, the CatBoost model reduced MAE by approximately 17 % relative to the naïve baseline, confirming its ability to capture complex non-linear patterns in highly volatile markets (Dorogush, Ershov, & Gulin, 2018) [4].

For Ethereum, Linear Regression achieved the best overall fit ($R^2 = 0.94$) and lowest MAPE (2.87 %), validating its suitability for assets exhibiting near-linear short-term dynamics (James et al., 2021) [7].

For Ripple, LightGBM obtained the lowest RMSE (0.0909) and MAPE (4.01 %), balancing accuracy and speed through gradient-based one-side sampling and efficient histogram construction (Ke et al., 2017) [8].

For Solana, XGBoost produced modest yet consistent improvements, reflecting its strong regularisation and scalability that promote model stability under moderate volatility (Chen & Guestrin, 2016) [3].

Overall, CatBoost and LightGBM handled volatility most effectively due to their adaptive gradient-boosting mechanisms, whereas Linear Regression provided interpretability and fast computation, and XGBoost ensured reliable performance for deployment. These outcomes indicate that combining linear and boosting models may further enhance robustness and generalisation across diverse cryptocurrency conditions.

6.3 Business Impact and Benefits

The development of this cryptocurrency forecasting system demonstrates how advanced machine-learning techniques can transform raw market data into meaningful intelligence for decision-making. By integrating predictive analytics into a user-friendly web application, the project enables investors, analysts, and fintech stakeholders to act on data-driven insights rather than subjective sentiment.

Strategic Value and Stakeholder Relevance

Stakeholder Group	Business Value and Benefits
Retail and institutional investors	Gain data-supported forecasts that improve timing of trades, portfolio balancing, and exposure management in volatile markets.
Fintech platforms and analytics providers	Can embed the deployed APIs to deliver real-time forecasting features and enhance client engagement through interactive dashboards.
Academic and research communities	Obtain an open, reproducible framework for studying non-linear financial forecasting, model interpretability, and AI deployment practices.
Regulators and ethical-AI bodies	Access a transparent demonstration of responsible machine-learning application in financial domains, supporting governance and compliance.

Business Impact and Organisational Alignment

- **Improved Decision Efficiency:** Automated insights reduce human bias and enable faster reactions to market signals.
- **Operational Scalability:** The modular API architecture supports future integration with additional assets, indicators, or trading systems.
- **Risk Awareness:** Clear visibility of feature drivers (e.g., volatility, liquidity, momentum) assists in understanding and mitigating financial risk.
- **Knowledge Transfer:** The project bridges technical ML outputs and business understanding, creating a replicable template for other industries adopting AI-driven forecasting.

Recommendations

- **Institutional Integration:** Link predictive outputs to risk-management tools or trading simulators to quantify potential gains and losses under various scenarios.
- **Model Governance:** Establish regular model-performance reviews and retraining schedules to ensure continued reliability under changing market conditions.
- **Commercialisation Pathway:** Expand coverage to additional cryptocurrencies or asset classes, offering subscription-based analytical services for fintech clients.
- **Explainability and User Trust:** Incorporate visual feature-importance dashboards and uncertainty intervals to enhance stakeholder confidence and interpretability.

6.4 Data Privacy and Ethical Concerns

Ethical Data Governance

All datasets used in this project were obtained from publicly accessible market sources such as Kraken, CoinDesk, and CoinGecko. They contain aggregated trading information only and no personal, confidential, or sensitive user data. Consequently, privacy risk is minimal. Nonetheless, ethical data handling practices were followed to ensure transparency, fairness, and accountability throughout the modelling and deployment process.

Dimension	Ethical Practice and Mitigation Strategy
Privacy Protection	All data consist solely of aggregated market indicators (open, high, low, close, volume, market cap). No individual-level information is stored or shared.
Transparency and Reproducibility	All preprocessing, feature transformations, and model pipelines are documented in the GitHub repositories for peer verification.

Fairness and Bias Management	Models are trained independently for each cryptocurrency to prevent bias transfer between assets and to maintain equitable representation of market dynamics.
Responsible Use of Predictions	The Streamlit app includes disclaimers clarifying that predictions are educational and not financial advice. Users are encouraged to interpret results as probabilistic insights, not deterministic outcomes.
Environmental Responsibility	Efficient algorithms and lightweight containerised deployment reduce computational load and align with sustainable-AI practices.
Consideration of Indigenous and Local Communities	Although financial data are non-personal, the project recognises that speculative technologies can influence broader economic ecosystems. Responsible communication of results avoids perpetuating financial exclusion or misinformation that may disproportionately affect under-represented communities.

Recommendations for Responsible AI

1. **Ethical Communication:** Maintain explicit disclaimers and user education on prediction uncertainty to prevent misuse for speculative trading.
2. **Bias and Robustness Auditing:** Periodically test model stability across different market phases to ensure fair and consistent performance.
3. **Inclusive Design:** As the platform scales, consider accessibility features and outreach that support culturally diverse and Indigenous users engaging with financial technology.
4. **Sustainability Practices:** Optimise retraining frequency and infrastructure efficiency to minimise environmental impact of cloud-based computation.

7. Deployment

Overview

The trained machine learning models were deployed through a **containerized API-based architecture** to enable seamless real-time predictions. Each team member developed and deployed their own model using **FastAPI** as the serving framework. This approach allowed each model to run independently while maintaining a consistent API interface for integration with the main Streamlit dashboard.

The deployment pipeline begins with **model packaging**, where the trained model, feature scaler, and metadata are saved as serialized files (.pkl and .json). These artifacts are then loaded by FastAPI, which exposes a /predict endpoint that accepts preprocessed input features and returns predicted values in JSON format. Each API service is **containerized using Docker** to ensure portability and reproducibility across environments.

The APIs are hosted on **Render**, a cloud platform that automatically manages builds and scaling. This setup enables the **Streamlit** application to send HTTP requests to the appropriate endpoint based on the selected cryptocurrency. The modular design allows for easy model updates, isolated debugging, and scalability for future model additions or retraining pipelines.

Overall, this deployment strategy ensures that the trained models are not only accessible and consistent across users but also maintainable and adaptable for future enhancements.

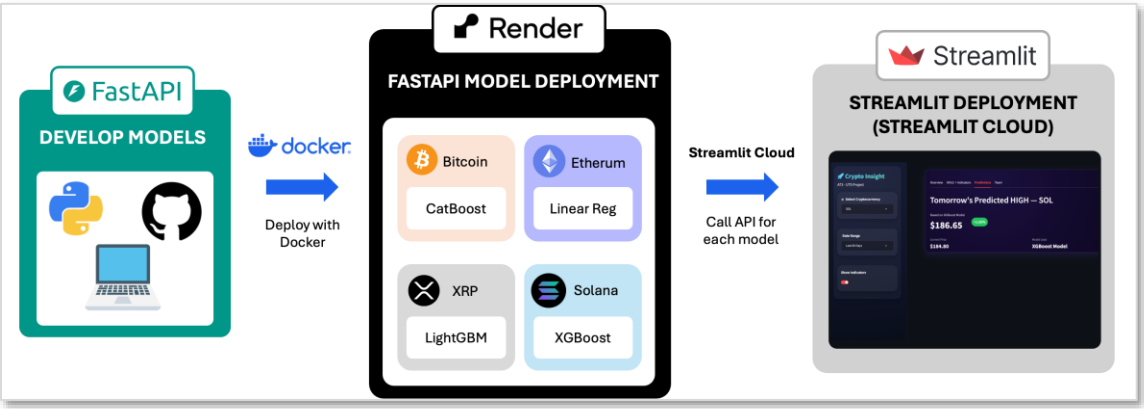


Figure 4: End-to-End Model Deployment Architecture Using FastAPI, Docker, Render, and Streamlit

a. Model Serving

Deployment Process

The deployment process followed a structured workflow to ensure reliability, scalability, and consistent integration across all models. Each trained model was independently deployed using FastAPI and Docker, then hosted on Render for public access.

1. **Model Packaging**
   After training, each model and its corresponding scaler were serialized using the joblib library and saved as .pkl files. A metadata file (model_meta.json) was also created to store model information, version details, and key performance metrics such as RMSE, MAE, MAPE, and R^2. These files allow each API service to load the model and scaler accurately for consistent prediction results across environments.
2. **API Development**
   Each model was wrapped in a FastAPI web service exposing a /predict/{Symbol} endpoint. The API accepts JSON input containing the latest OHLC and technical indicator values, applies preprocessing using the stored scaler, and outputs the predicted next-day high price in JSON format. A consistent response structure was maintained across all models for easy front-end integration.

Name	Asset / Model	Symbol for endpoint
Student A (Dylan)	BTC – CatBoost	/predict/bitcoin
Student B (Kittituch)	ETH – Linear Regression	/predict/ETHUSD
Student C (Ratticha)	XRP – LightGBM	/predict/xrp
Student D (Shawya)	SOL – XGBoost	/predict/SOLUSD

3. **Containerization**
   The FastAPI applications were containerized using Docker, ensuring each model runs in an isolated environment with fixed dependencies. This improves portability, reduces deployment errors, and simplifies updates or re-deployments when model retraining is performed.

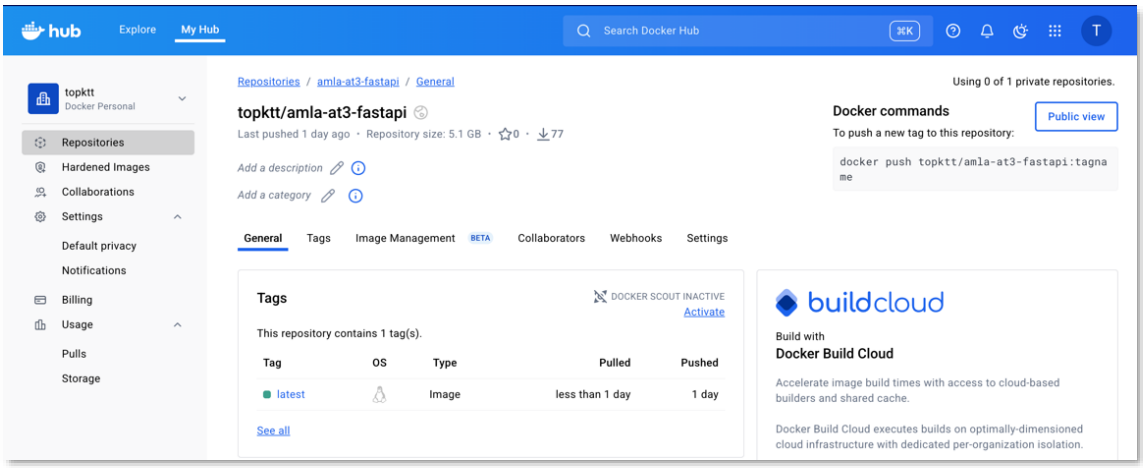


Figure 5: Docker Containerization of FastAPI Model

4. **Cloud Hosting on Render**

   Each containerized API was deployed to Render, a cloud platform that automates build and deployment steps. Environment variables and build commands were defined to start the FastAPI application automatically on each deployment. Render also provides HTTPS support, uptime monitoring, and scalable resources for handling user requests efficiently.

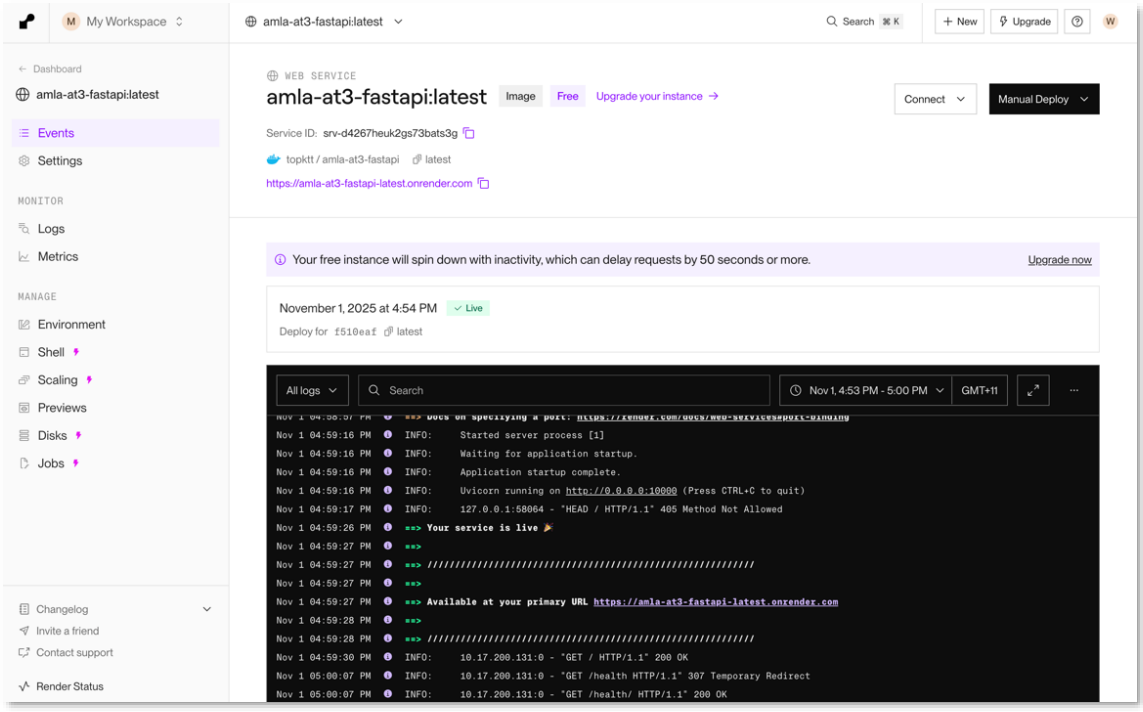


Figure 6: FastAPI Deployment on Render Cloud

5. **Integration with Streamlit App**

   The deployed model APIs were integrated into the Streamlit dashboard, which communicates with the endpoints via HTTP requests. When a user selects a cryptocurrency

(BTC, ETH, SOL, or XRP), the app calls the respective API endpoint (see Section 3.1.2), retrieves the prediction, and visualizes the result along with evaluation metrics and historical trends.

This deployment pipeline ensures a modular, reproducible, and maintainable system where each model can be updated, retrained, or replaced independently without affecting the overall architecture.

Integration with Web Application

Each cryptocurrency model has a dedicated FastAPI endpoint hosted on Render. Inside the Streamlit app, a mapping function (COIN_TO_ENDPOINT) links each coin to its corresponding API URL. When a user selects a coin and clicks **Refresh**, the application sends an HTTP POST request containing the latest OHLC and technical indicator values as a JSON payload to the model's /predict endpoint.

The FastAPI service receives this request, loads the appropriate model and scaler from the .pkl files created with joblib, and generates the **predicted next-day high price**.

To enhance performance and reliability, the integration also includes:

- **Caching with** @st.cache_data, which temporarily stores fetched data to reduce redundant API calls.
- **Error handling** for network timeouts or invalid responses, ensuring that the dashboard remains stable even during temporary outages.
- **Dynamic updates** using a refresh button that re-renders the entire page, allowing users to see the latest results from the APIs.

This integration combines Streamlit's interactivity with FastAPI's speed, creating a reliable and scalable deployment for real-time crypto forecasting.

Challenges, Considerations, and Future Improvements

During deployment, several technical challenges and considerations were identified. One key issue was API response latency when fetching predictions from Render, especially during network congestion or server cold starts. To address this, retry logic and loading indicators were added in the Streamlit app to improve responsiveness and user feedback.

Another consideration was dependency management across models, as each FastAPI service required consistent package versions for smooth execution. This was managed through Docker containerization, ensuring reproducibility and stable deployments.

Future improvements include implementing a CI/CD pipeline for automated testing and deployment, enabling continuous retraining of models using new market data, and introducing model versioning for easier updates. Additionally, hosting all models under a unified API gateway could further reduce latency and simplify scaling.

Overall, while the current deployment is stable and functional, these enhancements would strengthen performance, maintainability, and long-term scalability.

b. Web App

The Crypto Insight web application was developed to provide users with an interactive platform for real-time cryptocurrency forecasting and analysis. It serves as the front end for all deployed machine learning models, allowing users to explore historical market data, view technical indicators, and generate next-day price predictions across four cryptocurrencies: Bitcoin, Ethereum, Solana, and XRP.

The app is built with Streamlit, chosen for its simplicity and seamless integration with Python-based analytics. It connects directly to the FastAPI model endpoints hosted on Render. The application fetches live market data from the API – *Kraken, CoinDesk, and CoinGecko* to provide the most up-to-date inputs to display real-time charts. Since the API does not include technical indicators, the application calculates indicators such as RSI, moving averages, and Bollinger Bands locally before feeding the data into the models and visualizations.

Users can visualize price trends through interactive charts, compare model outputs, and track prediction accuracy over time. Overall, the web app bridges data science and user interaction, transforming complex model outputs into accessible, actionable insights for analysts, traders, and learners interested in crypto market behaviour.

Main Functionalities

The Crypto Insight web application provides several core features designed to make model predictions and data insights easy to access and interpret:

1. **Interactive Visualization**
   Historical OHLC data and technical indicators such as RSI and moving averages are displayed using Plotly charts, allowing users to explore patterns and trends visually.

Figure 7: Chart Visualization Tab

2. **Real-Time Prediction**
   Users can select a cryptocurrency and view the predicted next-day high price, generated by the model deployed through FastAPI

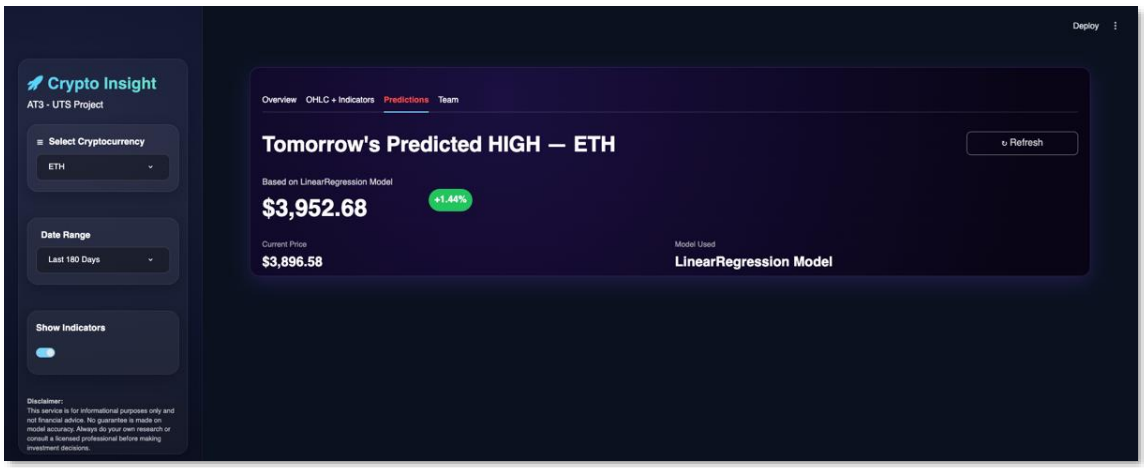


Figure 8: Prediction Tab

3. **Model Performance Dashboard**
   Each model's evaluation metrics—RMSE, MAE, MAPE, and R^2—are presented for transparent performance comparison across models.
4. **Team and Model Overview**
   A dedicated section lists each team member with their corresponding model, showing its performance metrics and role in the project.

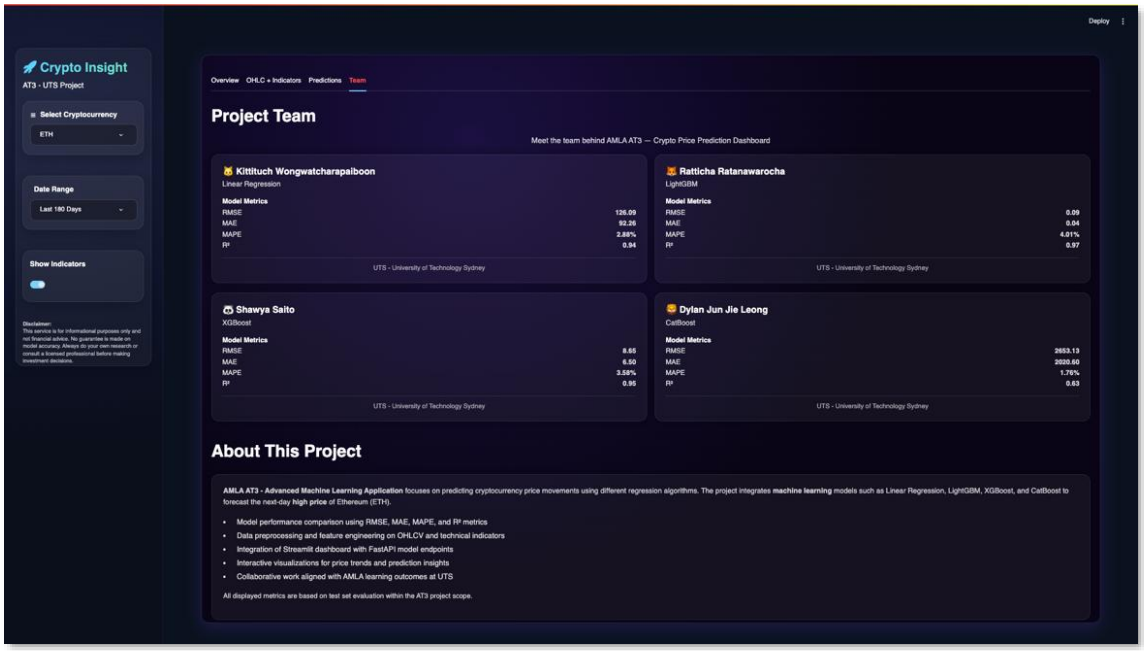


Figure 9: Model Performance, Team, and Model Overview Tab

5. **Data Refresh and Caching**
   The app includes a refresh button to update predictions and cached data using @st.cache_data, ensuring fast loading and reduced API calls.

Setup and Launch Instructions

The Crypto Insight web application is built with Python (3.11) and Streamlit, and all dependencies are managed through requirements.txt or Poetry. The following steps outline how to set up and launch the application:

1. **Clone the Repository**

   i. git clone <repository_link>
   ii. cd amla_group13_streamlit

2. **Install Dependencies**

   i. pip install -r requirements.txt

   or, if using Poetry:

   ii. poetry install

3. **Run the Application Locally**

```
i.  streamlit run app/main.py
```

4. **Access the Deployed Version**
   The web app is publicly accessible at:
   https://kittituchw-amla-group13-streamlit-appmain-irplhq.streamlit.app/

The application connects directly to the deployed FastAPI model endpoints on Render to fetch real-time cryptocurrency predictions and display model performance metrics.

Target Users and Use Cases

The **Crypto Insight** app is designed for **crypto traders, financial analysts, and data science students** who want quick, data-driven insights into cryptocurrency trends.

Users can view real-time forecasts, compare model performances, and explore technical indicators through interactive charts. Traders can use it to support short-term trading decisions, while students and researchers can study model behaviour and market dynamics.

Overall, the app makes complex predictive analytics accessible to both professionals and learners.

Benefits and Commercialisation Potential

The **Crypto Insight** app offers clear benefits by turning advanced machine learning predictions into an easy-to-use tool for real-time crypto analysis. It helps users make informed decisions, compare model performance, and understand market movements without needing coding or data science expertise.

For commercialisation, the app could evolve into a **subscription-based analytics platform**, offering premium features such as portfolio tracking, alert notifications, and integration with trading platforms. With further development, it could serve both **retail investors** and **financial institutions** seeking AI-driven insights for cryptocurrency forecasting.

Limitations and Future Improvements

While the current version of Crypto Insight is functional and user-friendly, several limitations remain. The app depends on API response times, which can occasionally cause delays during prediction requests. It also supports only four cryptocurrencies and a limited set of technical indicators.

Future improvements include expanding to more tokens, automating daily data updates, and improving scalability for higher traffic. Adding user accounts, personalized dashboards, and automated retraining pipelines would further enhance reliability and commercial potential.

8. Collaboration

a. Individual Contributions

Each member focused on one cryptocurrency asset and applied a distinct machine learning model. The table below summarizes the specific roles and key contributions of each member.

Student A - Dylan	
Asset / Model	BTC – CatBoost
Key Responsibilities	- Developed a CatBoost model for Bitcoin (BTC), covering the full pipeline from data acquisition, cleaning, transformation, and feature engineering to model training and evaluation through iterative experimentation cycles. - Developed FastAPI model service implementation for bitcoin model that serves model predictions independently and integrates with the project's Streamlit interface. - Developed FastAPI backend frameworks to enhance the data product's resource security, efficiency, and scalability. - Created data processing and transformation utilities to streamline experimentation workflows and accelerate model development cycles. - Reviewed the final report submission to ensure accuracy, reliability, and overall quality.
Key Contribution Achievements	- Successfully developed a Bitcoin prediction model that outperforms the baseline, adding measurable value to the overall data product. - Deployed a FastAPI model service that delivers Bitcoin T+1 predictions, inputs, and associated data. - Implemented and deployed an API key security framework compatible with Render, utilizing environment variables to ensure sensitive credentials remain protected and non–user-facing. - Implemented an API caching and automation system designed to make the data product backend more efficient, scalable, and cost-effective. Instead of fetching live data, cleaning it, and running model predictions every time a user requests a forecast, the system now performs these steps automatically once per day. This daily execution is aligned with the model's prediction frequency: predicting the next day's cryptocurrency high price.

A scheduled GitHub Actions workflow handles this process by:

- **Fetching raw market data** from external sources such as Kraken, CoinDesk, and CoinGecko.
- **Cleaning the raw data and running the predictive model** to generate forecasts for the upcoming trading day.
- **Caching the raw data, model inputs, and the prediction results** in the API storage layer. The cache is refreshed every 24 hours to minimize server usage while ensuring users receive the latest forecast.

Because the model's raw data requirements are relatively lightweight — typically 10–20 input features across fewer than 30 daily observations — the system can efficiently store and serve the entire prediction context without significant storage or performance overhead.

When users call the model prediction API endpoint, the service returns the cached prediction directly, eliminating redundant data retrieval, processing, and model execution. To further improve reliability, the API includes a fallback mechanism for user API calls: if the required data or predictions are missing locally, the endpoint automatically triggers a data pull, executes the full model prediction pipeline, and stores the updated results.

Overall, this caching and automated system reduces external API requests, lowers latency and compute costs, and improves scalability and reliability by allowing more users to utilise the service without increasing system load.

- Developed, unit tested, and implemented the following helper functions, made accessible via Test PyPI:
    - **Custom Scikit-learn Classes** for data transformation and encoding:
        - **TimeToSecondsTransformer**`: Custom scikit-learn transformer that converts datetime columns into a single numerical feature representing the total number of seconds elapsed since midnight.
        - `**CyclicalFeatures_CosSin**`: Custom scikit-learn transformer that converts cyclical numerical features (e.g., month, hour) into sine and cosine components for better representation in machine learning models.

	○ **Normality Scoring and Transformation** functions: ▪ `**normality_score**`: Evaluates how close a transformed feature is to a normal distribution by combining the D'Agostino (or Shapiro) p-value, absolute skewness, excess kurtosis, and outlier fraction into a composite score for selecting the best transformation method. ▪ `**best_numerical_transformation**`: Applies numerical transformations (Log, Sqrt, Box-Cox, Yeo-Johnson, Quantile) and returns the result of the one with the highest composite score for normality. ▪ `**summarize_numerical_transformations**`: Iterates through all numerical columns in a DataFrame, finds the best numerical transformation for each, and returns a summary of the recommended changes. Features that are already likely normally distributed (p-value ≥ alpha) are excluded from the summary. ○ **Fibonacci Detrending** functions: ▪ `**detrend_fib_causal**`: Performs causal detrending on a time series by subtracting a lookback trend component weighted by a Fibonacci sequence. ▪ `**reconstruct_fib_causal**`: Reconstructs the original time series by adding the Fibonacci-weighted lookback trend component back to the detrended residual component.
Student B - Kittituch	
Asset / Model	ETH – Linear Regression
Key Responsibilities	- Built the Linear Regression model for Ethereum (ETH), including data preparation, model training, and performance evaluation. - Set up and managed the team's GitHub repository to organise experiment notebooks, API code, and report files. - Created and maintained the Microsoft Teams communication channel to coordinate progress, updates, and deliverables. - Developed the Streamlit dashboard interface, designing an intuitive and visually engaging user experience with interactive charts, tabs, and model comparison features.

	- Contributed to the Deployment section of the report, documenting the end-to-end implementation process.
Key Contribution Achievements	- Successfully deployed the interactive Streamlit app integrating all individual APIs for real-time cryptocurrency forecasts. - Designed a clean, user-friendly Streamlit UI that visualises price trends, technical indicators, and prediction results effectively. - Streamlined collaboration through organised version control and communication platforms. - Produced a well-documented deployment process and clear technical instructions in the report. - Ensured smooth integration between model APIs and the web application for consistent user experience.
Student C - Ratticha	
Asset / Model	XRP – LightGBM
Key Responsibilities	- Developed the LightGBM model for Ripple (XRP), including end-to-end experimentation in Jupyter Notebook. - Built and deployed the FastAPI service to host the trained model and generate next-day high-price predictions. - Led report coordination by maintaining a consistent structure and formatting for all team sections. - Authored key report components, including Business Understanding, Data Understanding, Evaluation, Collaboration, and Conclusion sections.
Key Contribution Achievements	- Delivered a reproducible, high-performing LightGBM pipeline integrated with FastAPI for automated prediction. - Ensured report cohesion and clarity by standardising formatting and templates for team inputs. - Provided clear, professional writing that strengthened the overall readability and alignment of the final report. - Facilitated smooth collaboration by guiding section integration and quality review before submission.
Student D - Shawya	
Asset / Model	SOL – XGBoost
Key Responsibilities	- Developed the FastAPI code to fetch Solana OHLC data from Kraken and generate model features. - Integrated the pre-trained XGBoost model into the API for next-day high price prediction. - Contributed to report preparation including executive summary writing and creation of visual figures/diagrams.

Key Contribution Achievements	- Delivered a fully functional API capable of serving real-time predictions. - Produced clear, concise executive summary highlighting project objectives, methodology, and outcomes. - Created figures and diagrams that improved report readability and visual appeal.

b. Group Dynamic

The group maintained a strong and collaborative dynamic throughout the project, characterised by clear communication, mutual support, and shared accountability. Each member demonstrated initiative in managing their assigned tasks while remaining open to feedback and coordination. Regular checkpoints were held at each major project milestone to ensure progress alignment and equal contribution across all areas — data preparation, modelling, API development, deployment, and report writing.

Communication was seamless across multiple channels. WhatsApp served as the primary platform for day-to-day discussions and quick clarifications, while Microsoft Teams was used for file sharing, progress tracking, and report consolidation. The team also conducted a combination of online and in-person meetings, enabling members to collaborate efficiently, troubleshoot issues together, and stay aligned on goals. This balance of flexibility and consistency created a positive working environment and ensured that every member remained informed and engaged throughout the project.

c. Ways of Working Together

The team followed an iterative and milestone-based workflow, similar to the CRISP-DM framework, to structure the project from data understanding to deployment. Clear stages — including development, testing, and integration — allowed for incremental progress and early detection of challenges.

Weekly check-ins and ad-hoc meetings were conducted both online and face-to-face, depending on the task urgency and members' availability. Each meeting included progress updates, next-step planning, and allocation of upcoming deliverables. A shared GitHub repository was maintained as the central version-control system, ensuring that all code, documentation, and model artifacts remained synchronised and accessible.

Microsoft Teams was used for collaborative report writing and review, while WhatsApp facilitated fast decision-making for immediate questions. This combination of tools supported effective

teamwork, transparency, and alignment across all project components, from experimentation to API integration.

d. Issues Faced

While the group collaboration was highly effective overall, several challenges were encountered during the project. The main difficulty involved coordinating work across multiple platforms and time schedules, especially when integrating individual models into the shared Streamlit application. Minor version-control conflicts and formatting inconsistencies occasionally occurred when merging updates from different members.

These challenges were resolved through frequent checkpoints, in-person working sessions, and clear task ownership. Each member took responsibility for a defined area — ensuring that overlaps were minimised and integration proceeded smoothly. Another challenge was maintaining consistent report formatting and technical clarity across sections authored by different members, which was addressed by appointing one member to oversee report standardisation and final editing.

Lessons learned include the importance of establishing an early communication rhythm, documenting changes regularly on GitHub, and setting clear deadlines for integration phases. For future collaborations, the team recommends continuing this structured, transparent workflow and maintaining hybrid communication (both online and physical) to sustain cohesion and efficiency.

9. Conclusion

This project successfully developed an end-to-end machine learning data product for forecasting next-day high prices of major cryptocurrencies — Bitcoin, Ethereum, Ripple, and Solana. Through a consistent pipeline of data preparation, feature engineering, model training, and cloud deployment, each team member produced a functional predictive model integrated into the shared Crypto Insight web application. The system demonstrates how data-driven forecasting can enhance transparency and support decision-making in volatile financial markets.

The project achieved its goals by delivering reproducible ML workflows, interpretable model outputs, and a fully deployed FastAPI–Streamlit ecosystem. It met stakeholder expectations by offering practical, user-friendly insights that balance technical depth with business applicability. The collaboration showcased strong coordination, effective communication, and a shared commitment to technical excellence and ethical AI practices.

Future work should focus on improving model accuracy through advanced feature selection, expanding to additional cryptocurrencies or financial assets, and implementing continuous retraining pipelines for live data adaptation. Integrating explainability dashboards and uncertainty estimates would further enhance user trust and commercial readiness. Overall, this project provides a scalable and transparent foundation for applying responsible AI in financial forecasting.

10. References

[1] BairesDev. (n.d.). *Random decision forests in finance: A guide to predicting financial trends.* BairesDev Blog.

[2] Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). *Hyperopt: A Python library for model selection and hyperparameter optimization. Computational Science & Discovery, 8*(1), 014008. https://doi.org/10.1088/1749-4699/8/1/014008

[3] Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

[4] Dorogush, A. V., Ershov, V., & Gulin, A. (2018). *CatBoost: Gradient boosting with categorical features support* [Preprint]. arXiv. https://arxiv.org/abs/1706.09516

[5] Dorogush, A. V., Gulin, A., Gusev, G., Kazeev, N., Ostroumova, L., & Vorobev, A. (2017). Fighting biases with dynamic boosting. *arXiv*. https://arxiv.org/abs/1706.09516

[6] Dorogush, A. V., Gusev, G., Vorobev, A., Kazeev, N., Ostroumova, L., & Gulin, A. (2023). *CatBoost: A fast, scalable, high-performance gradient boosting on decision trees library*. GitHub. https://github.com/catboost/catboost

[7] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

[8] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree*. In *Advances in Neural Information Processing Systems (NeurIPS 2017)* (pp. 3149–3157). https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[9] Podobnik, B., Horvatić, D., Petersen, A. M., & Stanley, H. E. (2009). *Cross-correlations between volume change and price change. Proceedings of the National Academy of Sciences of the United States of America, 106*(52), 22079–22084. https://doi.org/10.1073/pnas.0911983106

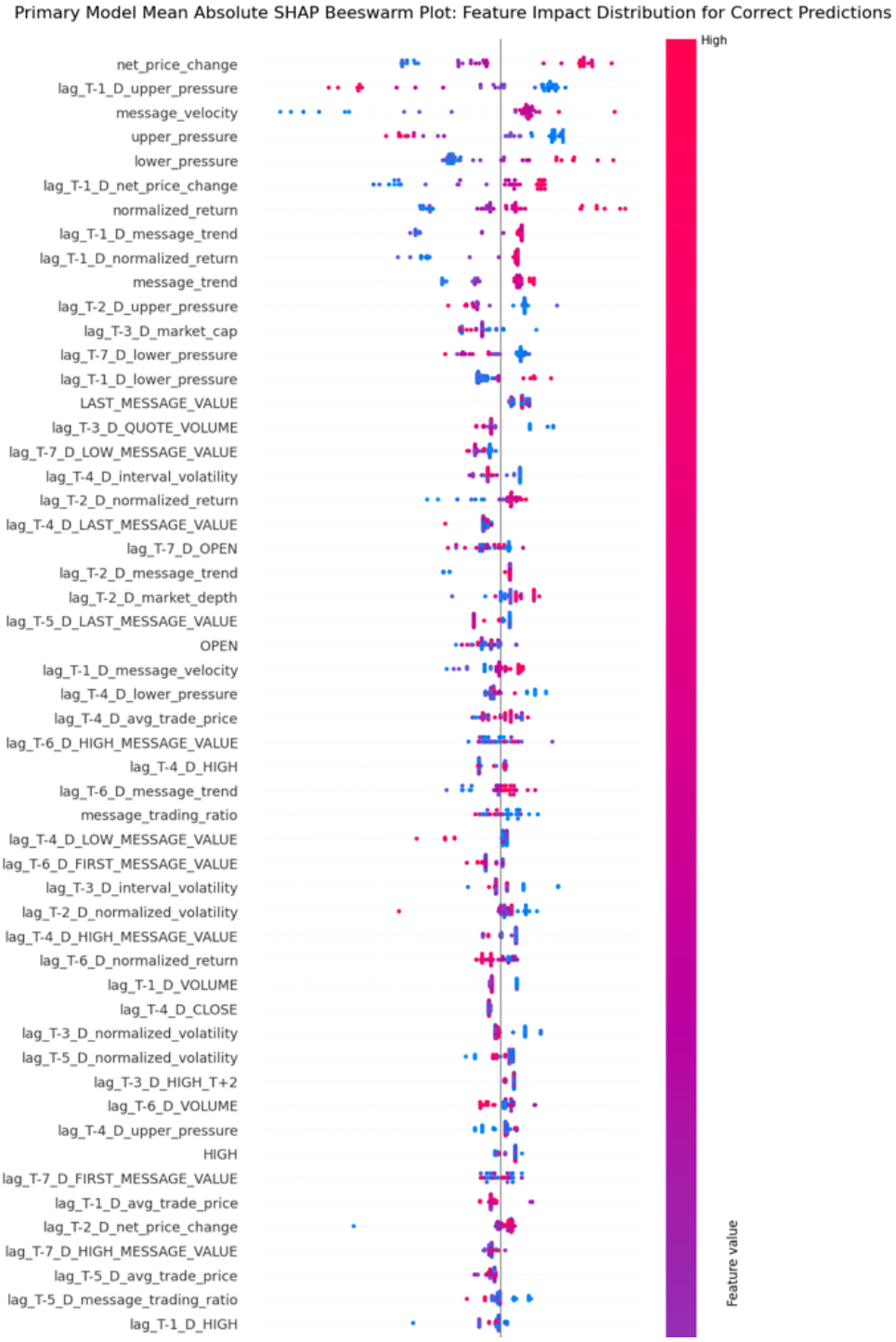
[10] Song, X., & Chen, Z. S. (2024). *Enhancing financial time series forecasting in the shipping market: A hybrid approach with Light Gradient Boosting Machine. Engineering Applications of Artificial Intelligence, 136*(Part A), 108942. https://doi.org/10.1016/j.engappai.2024.108942

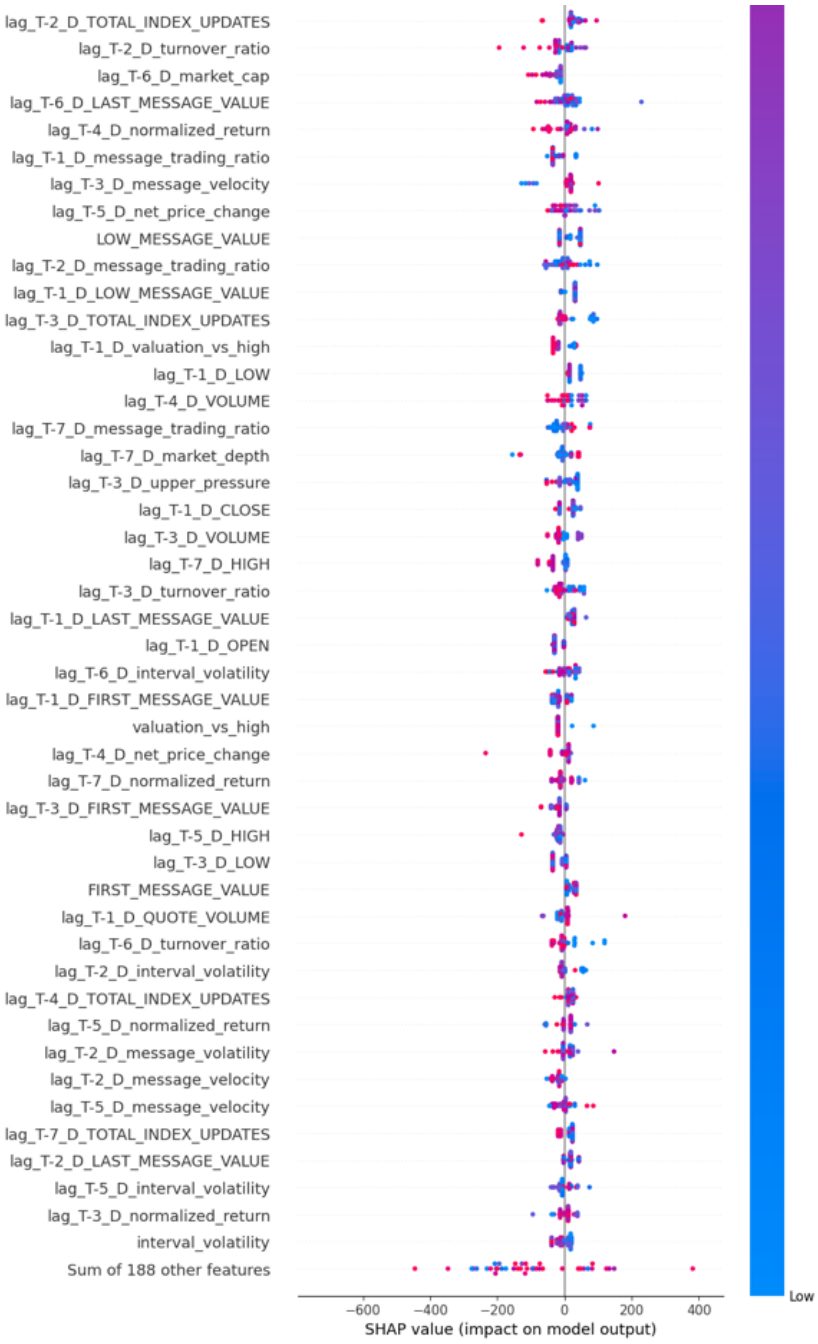
[11] ValueAdder. (n.d.). *Automating financial forecasts with linear regression.* ValueAdder Blog. https://www.valuadder.com/blog/automating-financial-forecasts-with-linear-regression/

[12] Zhang, L., Bian, W., Qu, W., Tuo, L., & Wang, Y. (2021). *Time series forecast of sales volume based on XGBoost. Journal of Physics: Conference Series, 1873*(1), 012067. IOP Publishing. https://doi.org/10.1088/1742-6596/1873/1/012067

11. Appendix

A. SHAP Analysis of Top-Performing CatBoost Bitcoin Model: Beeswarm Plot of Feature Impact Distribution for Correct Predictions




lag_T-2_D_TOTAL_INDEX_UPDATES
lag_T-2_D_turnover_ratio
lag_T-6_D_market_cap
lag_T-6_D_LAST_MESSAGE_VALUE
lag_T-4_D_normalized_return
lag_T-1_D_message_trading_ratio
lag_T-3_D_message_velocity
lag_T-5_D_net_price_change
LOW_MESSAGE_VALUE
lag_T-2_D_message_trading_ratio
lag_T-1_D_LOW_MESSAGE_VALUE
lag_T-3_D_TOTAL_INDEX_UPDATES
lag_T-1_D_valuation_vs_high
lag_T-1_D_LOW
lag_T-4_D_VOLUME
lag_T-7_D_message_trading_ratio
lag_T-7_D_market_depth
lag_T-3_D_upper_pressure
lag_T-1_D_CLOSE
lag_T-3_D_VOLUME
lag_T-7_D_HIGH
lag_T-3_D_turnover_ratio
lag_T-1_D_LAST_MESSAGE_VALUE
lag_T-1_D_OPEN
lag_T-6_D_interval_volatility
lag_T-1_D_FIRST_MESSAGE_VALUE
valuation_vs_high
lag_T-4_D_net_price_change
lag_T-7_D_normalized_return
lag_T-3_D_FIRST_MESSAGE_VALUE
lag_T-5_D_HIGH
lag_T-3_D_LOW
FIRST_MESSAGE_VALUE
lag_T-1_D_QUOTE_VOLUME
lag_T-6_D_turnover_ratio
lag_T-2_D_interval_volatility
lag_T-4_D_TOTAL_INDEX_UPDATES
lag_T-5_D_normalized_return
lag_T-2_D_message_volatility
lag_T-2_D_message_velocity
lag_T-5_D_message_velocity
lag_T-7_D_TOTAL_INDEX_UPDATES
lag_T-2_D_LAST_MESSAGE_VALUE
lag_T-5_D_interval_volatility
lag_T-3_D_normalized_return
interval_volatility
Sum of 188 other features
−600
−400
−200
0
200
400
SHAP value (impact on model output)
Low