Statistical Learning for Crypto Data: Kaggle Contest on G-Research Crypto Forecasting

Hangyu LIN
Department of Mathematics
HKUST

Abstract

Cryptocurrencies have revolutionized the financial landscape, creating new opportunities and challenges for investors and traders alike. As the cryptocurrency market operates 24/7 and exhibits high volatility, accurate forecasting and decision-making require sophisticated data analysis techniques. Machine learning, with its ability to extract patterns from complex data, has emerged as a powerful tool for analyzing cryptocurrency time series data. This work presents a comprehensive analysis of machine learning techniques applied to crypto time series data, mainly the G-Research Crypto Forecasting Contest. Firstly, we delve into data preprocessing techniques, including data cleaning, normalization, and feature engineering, to ensure the quality and relevance of the input data. Next, we delve into a range of machine learning algorithms, including ElasticNet, the tree-based boosting method LightGBM, LSTMs, and transformer-based methods, to assess their effectiveness in modeling cryptocurrency price movements. Preliminary results from our experiments indicate that the LightGBM method outperforms others, while deep learning approaches have not yielded the anticipated results. The code for the project can be found in https://github.com/avalonstrel/CryptoAnalysis.git

1 Introduction

Cryptocurrencies (1; 13), digital or virtual forms of currency that use cryptography for security, have become increasingly popular financial instruments. Unlike traditional currencies, they operate on decentralized platforms, typically blockchain technology, which records all transactions across a network of computers. The volatile nature of cryptocurrency prices, influenced by factors such as market demand, regulatory news, and technological developments, presents both challenges and opportunities for predictive analytics.

The G-Research Crypto Forecasting [1] initiative focuses on predicting the future movements of cryptocurrency prices. The challenge typically involves providing participants with a rich dataset that includes historical prices, volumes, and other relevant financial indicators of various cryptocurrencies. Participants are tasked with employing statistical and machine learning techniques to develop predictive models that can forecast future cryptocurrency price movements.

In this work, we will conduct a comprehensive analysis of different machine learning methods on the crypto forecasting task. In the realm of cryptocurrency forecasting, various machine learning methods are employed to tackle the challenges of time series analysis. Linear models like ARIMA (12) and its variants provide a straightforward approach but are limited by their assumption of linearity. Regularization techniques such as Lasso and ElasticNet (3) help prevent overfitting, which is crucial in managing the noisy data typical of cryptocurrency markets. Tree-based models, including XGBoost and LightGBM (4), utilize gradient boosting frameworks to effectively capture complex

[1]https://www.kaggle.com/c/g-research-crypto-forecasting

nonlinear relationships and are well-suited to handle diverse input features, aligning with the intricate dynamics of cryptocurrency prices. Deep learning approaches, particularly Long Short-Term Memory (LSTM) (5) units and other recurrent neural networks (RNNs) (9), excel in analyzing sequential data and identifying long-term dependencies critical for understanding price trends. Additionally, transformer-based models (14), originally designed for natural language processing, have been adapted to time series forecasting, using self-attention mechanisms to emphasize significant points in the data, thus enhancing prediction accuracy in certain scenarios. Typically, we will test some recent proposed and popular transformer methods for time series data including Informer (16), Reformer (8), iTransformer (10).

Besides the models to predict, another key component for crypto forecasting is feature design. Feature design, or feature engineering, is crucial for developing machine learning models in cryptocurrency forecasting, involving the creation of informative features that capture market complexities (2; 6). Price-based Features such as lagged prices, rolling statistics, and technical indicators (MACD, RSI, Bollinger Bands) capture trends and volatility. Volume-based Features like historical volumes and volume oscillators reflect market sentiment and momentum. Time features extract seasonal effects from timestamps and handle irregular series through elapsed time. Order book features such as spread and depth gauge liquidity and sentiment. Derived Statistical Features including returns, log returns, and volatility estimates normalize price changes and assess stability. External Factors incorporate market news sentiment and economic indicators affecting the market. Network and Transaction Features specific to cryptocurrencies, like transaction volume and active addresses, indicate network usage and growth. Effective feature engineering enhances predictive accuracy by normalizing data and using feature selection techniques like PCA (11) and L1 regularization to manage high-dimensional data, thus capturing essential market dynamics. We will evaluate various strategies that incorporate distinct features in the upcoming experiments.

In conclusion, we will conduct the following analysis:

- Feature Selection: We will utilize a range of indicators and features for cryptocurrency data prediction.
- Model Analysis: We will test various methods and provide a comparison of their performance.

2 Data Analysis

2.1 Dataset Description

First of all, we will introduce the data of G-Research Crypto Forecasting Competition. Among the thousands of cryptocurrencies that have emerged, notable ones such as Bitcoin (BTC), Ether (ETH), and Dogecoin (DOGE) are widely recognized. According to data from CryptoCompare as of July 25, 2021, cryptocurrencies are heavily traded across various exchanges, with an average daily trading volume of approximately \$41 billion over the past year. The price movements of these cryptocurrencies are closely interconnected; while Bitcoin often leads market trends, other cryptocurrencies also significantly influence price dynamics across the board. The challenge requires building machine learning models to simultaneously predict the movements of all 14 assets. The effectiveness of these models will be judged based on the correlation between the predicted and actual market data collected during a three-month evaluation period following the competition's conclusion. Since the contest is closed and we can not use the evaluation code, we will construct test datasets from the training data to evaluate different methods. Specifically, we give the features provided in the original dataset in Table 1 and show a plot of BTC in Figure 1.

This forecasting competition aims to predict near-future returns for prices P_a, for each asset a. For each row in the dataset, we include the target for prediction, *Target*. The *Target* is derived from logarithmic returns (R_a) over 15 minutes, defined as:

$$R_a(t) = \log\left(\frac{P_a(t+16)}{P_a(t+1)}\right)$$

Crypto asset returns are highly correlated, largely following the overall crypto market trends. To test the ability to predict returns for individual assets, we perform a linear residualization, removing the

Table 1: Dataset Features Description

Field	Description
timestamp	All timestamps are returned as second Unix timestamps (the number of seconds elapsed since 1970-01-01 00:00:00.000 UTC). Timestamps in this dataset are multiples of 60, indicating minute-by-minute data.
Asset_ID	The asset ID corresponding to one of the cryptocurrencies (e.g., Asset_ID = 1 for Bitcoin). The mapping from Asset_ID to crypto asset is contained in `asset_details.csv`.
Count	Total number of trades in the time interval (last minute).
Open	Opening price of the time interval (in USD).
High	Highest price reached during the time interval (in USD).
Low	Lowest price reached during the time interval (in USD).
Close	Closing price of the time interval (in USD).
Volume	Quantity of asset bought or sold, displayed in base currency (USD).
VWAP	The average price of the asset over the time interval, weighted by volume. VWAP is an aggregated form of trade data.
Target	Residual log-returns for the asset over a 15 minute horizon.

market signal from individual asset returns when creating the target. In more detail, if $M(t)$ is the weighted average market returns, the target is defined as:

$$M(t) = \frac{\sum_a w_a R_a(t)}{\sum_a w_a}, \quad \beta_a = \frac{\langle M \cdot R_a \rangle}{\langle M^2 \rangle}, \quad \text{Target}_a(t) = R_a(t) - \beta_a M(t)$$

where the brackets $\langle \cdot \rangle$ represent the rolling average over time (3750-minute windows), and the same asset weights w_a are used for the evaluation metric.

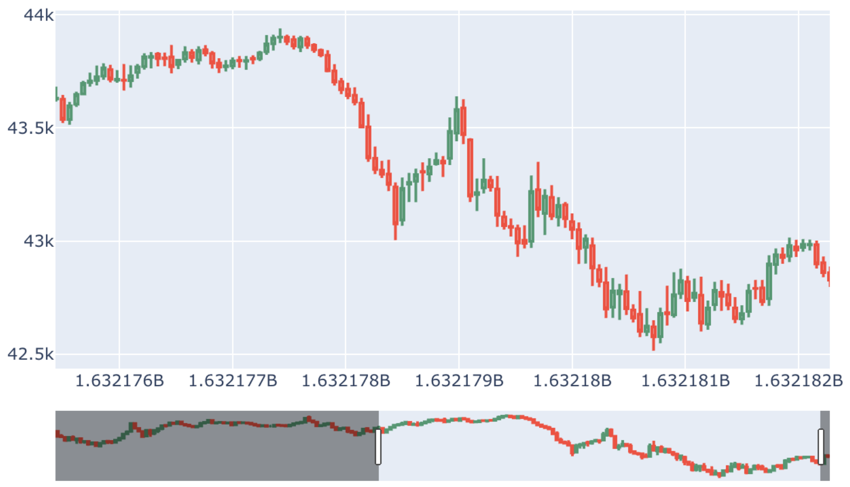

Figure 1: Slice of the Bitcoin prices including Open, High, Low and Close.

2.2 Feature Engineering

Technical analysis indicators are critical tools used by traders to predict future market movements based on historical data. TA-Lib which we mainly used in our project provides various functions that simplify the process of calculating these indicators. Here are some important ones:

1. **Simple Moving Average (SMA)**: The Simple Moving Average is an arithmetic moving average calculated by adding recent closing prices and then dividing that by the number of

time periods in the calculation average.

$$\text{SMA} = \frac{P_1 + P_2 + \cdots + P_n}{n}$$

where $P_1, P_2, \ldots, P_n$ are the closing prices of the asset over n periods.

2. **Exponential Moving Average (EMA)**: The Exponential Moving Average gives more weight to the most recent prices, thus reacting more significantly to price changes than the simple moving average.

$$\text{EMA}_{\text{today}} = (\text{Price}_{\text{today}} \times k) + (\text{EMA}_{\text{yesterday}} \times (1 - k))$$

where $k = \frac{2}{(N+1)}$, N is the number of periods.

3. **Relative Strength Index (RSI)**: The Relative Strength Index measures the speed and change of price movements. RSI oscillates between zero and 100.

$$\text{RSI} = 100 - \frac{100}{1 + \text{RS}}$$

where RS (Relative Strength) is the average of n days' up closes divided by the average of n days' down closes.

4. **Moving Average Convergence Divergence (MACD)**: This indicator is used to spot changes in the strength, direction, momentum, and duration of a trend in a stock's price.

$$\text{MACD} = \text{EMA}_{\text{fast}}(P) - \text{EMA}_{\text{slow}}(P)$$

where EMA_{fast} and EMA_{slow} are the exponential moving averages for shorter and longer periods, respectively.

5. **Bollinger Bands (BB)**: Bollinger Bands are volatility bands placed above and below a moving average, where volatility is based on the standard deviation, which changes as volatility increases and decreases.

$$\text{Upper BB} = \text{SMA}(P, n) + 2 \times \text{SD}(P, n)$$
$$\text{Lower BB} = \text{SMA}(P, n) - 2 \times \text{SD}(P, n)$$

where $\text{SD}(P, n)$ is the standard deviation of the last n prices.

The features described above represent only a subset of the essential elements incorporated into our final models. Following extensive cross-validation experiments, we selected a primary set of features encompassing various types of indicators, including those based on moving averages and momentum. For detailed information on these features, please refer to the Appendix 6.1.

3 Methods Review

In this section, we will review the methods we analyze in this project.

3.1 Elastic Net

Elastic Net (17) is a regularization technique used in linear regression that combines the benefits of Lasso (Least Absolute Shrinkage and Selection Operator) and Ridge Regression. This approach is particularly effective when dealing with highly correlated predictors or when the number of predictors exceeds the number of observations.

The Elastic Net optimization function is given by:

$$\text{Minimize: } \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1}{2}(1 - \alpha) \sum_{j=1}^{p} \beta_j^2 \right)$$

where λ is the regularization penalty parameter, and α is the mixing parameter between Lasso and Ridge.

3.2 Boosting Method

Gradient Boosting Machines (GBMs) are a powerful ensemble learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When trees are used as the base learners, this technique is often referred to as Gradient Boosted Decision Trees (GBDTs).

LightGBM (7), or *Light Gradient Boosting Machine*, is a high-performance, open source gradient boosting framework based on decision tree algorithms. It is designed to be distributed and efficient, with a focus on speed and network scalability. LightGBM achieves great efficiency and lower memory consumption through several advanced techniques, including Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

3.3 Deep Learning Method

Long Short-Term Memory networks (LSTMs) (5) are a special kind of Recurrent Neural Network (RNN) capable of learning long-term dependencies in sequence prediction problems. This is particularly valuable in time series analysis, where the prediction accuracy can significantly depend on previous data points, potentially separated by long time intervals.

LSTMs are designed to avoid the long-term dependency problem typical of standard RNNs. This is achieved through their unique structure, which includes memory cells that maintain information for long periods and three types of gates that regulate the flow of information:

- **Forget Gate:** Determines parts of the cell state to be thrown away.
- **Input Gate:** Updates the cell state by adding new information.
- **Output Gate:** Decides the next hidden state and the output information based on the current input and the memory of the cell.

Originally designed for tasks in natural language processing (NLP), the Transformer model, introduced by Vaswani et al (14). in their seminal paper "Attention is All You Need," has expanded its utility to the domain of time series analysis. The model's core mechanism, self-attention, facilitates the learning of dependencies across data points, making it adept at handling sequential data like time series. Typically, we introduce three types of transformers designed for time series applications, i.e., Reformer, Informer, iTransformer.

The **Reformer** (8), introduced to enhance memory efficiency, incorporates two main innovations:

- **Locality-Sensitive Hashing (LSH) Attention:** This approach reduces the computational complexity of the self-attention mechanism, handling sequences significantly more efficiently by hashing input features into buckets with similar items.
- **Reversible Residual Layers:** By allowing the activations to be reconstructed from the outputs, these layers minimize memory use during training, supporting deeper network architectures without increased memory overhead.

Targeted at long-sequence time-series forecasting, the **Informer** (16) optimizes the processing of long data sequences through several key features:

- **ProbSparse Self-attention:** This innovative attention mechanism selectively focuses on the most significant parts of the data, reducing computational requirements.
- **Self-attention Distilling:** The sequence length is progressively reduced across the layers of the model, focusing the model's capacity on the most informative segments of the input.
- **Enhanced Decoder:** Optimized for predicting multiple future points in time-series data, this decoder is particularly effective in multi-horizon forecasting tasks.

Considering the shortcomings of Transformer-based approaches in time series forecasting, **iTransformer** (10) examines the reasons why Transformers may underperform even basic linear models in this domain, despite their success in various other fields. We observe that the conventional structure of Transformer-based forecasters might not be ideally suited for multivariate time series forecasting. They find that data points from the same time step, which represent distinctly different physical phenomena captured through various measurement techniques, are amalgamated into a single token. This process erases the inherent multivariate correlations. Furthermore, tokens representing individual time steps may fail to convey useful information due to their overly localized receptive fields and the misalignment of events that occur simultaneously but are unrelated. Additionally, the use of permutation-invariant attention mechanisms in the temporal dimension, as noted by Zeng et al. (15), may not be appropriate. This can significantly hinder the Transformer's ability to accurately capture and represent essential time series dynamics and multivariate relationships, thus limiting its effectiveness and generalizability across diverse time series datasets.

4 Main Results

4.1 Experiments Setup

The start date for the training data varied significantly across different currencies. Given that my model calculates the average of changes for each currency, it was crucial to ensure consistency between the training and evaluation periods for all existing currencies. Anticipating minimal missing values for all currencies during the competition's evaluation period, I opted not to use the entire training dataset. Instead, I selected the training data from periods where there was a sufficient representation of currencies.

The choice of the starting date was initially based on cross-validation (CV) scores. In retrospect, this approach was flawed because it led to comparisons of CV scores across different datasets, which may not provide a consistent basis for model evaluation.

Moreover, each currency exhibited several extended periods with no data ('blank' periods). To address potential data gaps caused by these periods, I employed a forward fill strategy. However, I was concerned that applying forward fill indiscriminately throughout might degrade the data quality, especially during prolonged blank periods. Therefore, I implemented a limitation on the extent of forward fill. For the evaluation phase, the model was configured to use unlimited forward fill, which I deemed acceptable given the absence of extended blank periods during this phase.

As we mentioned in Section 2, we will follow a feature design as described in Appendix 6.1. Detailed parameters for each method will be found in args.py/trainer.py for each method. In the competition, your predictions will be evaluated on a weighted version of the Pearson correlation coefficient, with weights given by the Weight column in the Asset Details file.

4.2 Main Results

Table 2: Weighted Pearson correlation coefficient of Various Methods

Method	**Binance Coin**	**Bitcoin**	**Cardano**	**Weighted**
Linear Regression	−0.001 717 79	0.016 022 8	0.028 954 9	0.017 892 2
LightGBM	0.0194612	0.0237192	0.0282305	0.0265247
ElasticNet	0.001 749 64	0.019 110 7	0.002 301 69	0.005 990 41
Informer	−0.005 138 73	0.000 928 189	0.009 654 25	0.000 145 813
Reformer	−0.006 820 8	0.001 828 61	0.007 663 72	0.000 311 872
Transformer	−0.005 686 32	0.000 378 054	0.007 169 61	0.000 044 829

We conclude the results in Table 2 and Figure 2. The comparative analysis of various forecasting methods for cryptocurrencies highlights the robust performance of LightGBM, which significantly outshines other models. LightGBM achieves a weighted average performance of 0.0265247, marking a notable 48.35% improvement over Linear Regression, which has a weighted average of 0.0178922. This stark difference underscores the effectiveness of LightGBM in capturing the nuances of cryptocurrency data.

In contrast, more complex deep learning models such as ElasticNet, Informer, Reformer, and Transformer show considerable underperformance when compared to LightGBM. Specifically, ElasticNet's weighted performance is 0.00599041, which is a 77.4% decrease from LightGBM. The Informer and Reformer models perform even less effectively, with weighted performances of 0.000145813 and

0.000311872 respectively, translating to decreases of 99.45% and 98.82% from LightGBM's results. The Transformer model, with a minimal weighted score of 4.42829e-05, shows a near-total decrease of 99.83% compared to LightGBM.

These results not only demonstrate the superior predictive capability of LightGBM but also highlight the challenges faced by more sophisticated models in this particular application. The significant underperformance of deep learning models suggests that without extensive customization and tuning, they struggle to compete with more straightforward or specialized approaches like LightGBM in the realm of financial time series forecasting. This analysis points to the need for careful model selection and optimization, particularly when working with the volatile and unpredictable nature of cryptocurrency markets.

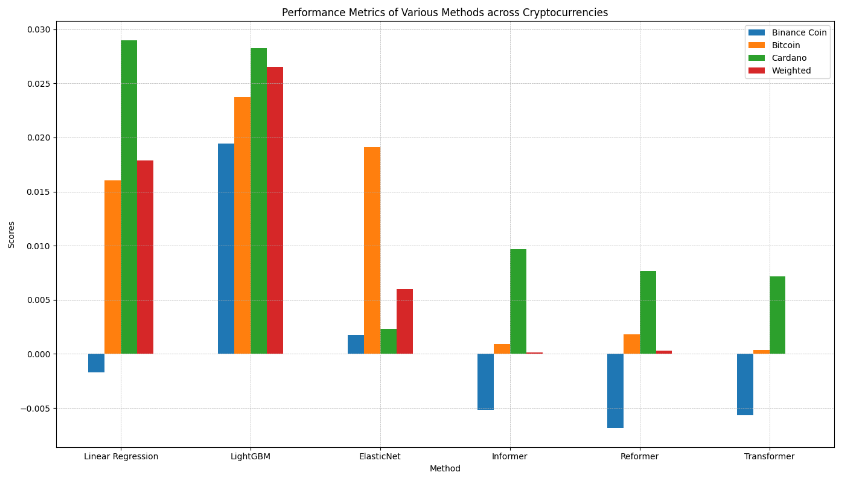

Figure 2: Performance of various methods test on 14 cryptocurrencies and the weighted performance

4.3 Analysis on LightGBM

In this section, we will have a close look at the performance of the LightGBM. From Figure 3, the plot has two lines representing the actual returns (in blue) and the predicted returns (in red dashed line). The actual returns show significant volatility, with sharp increases and decreases over short periods The predicted returns line appears to be smoother than the actual returns, suggesting that the model may not be capturing all the volatility present in the actual data. This is common in predictive modeling, where the aim is often to predict the trend rather than the noise. In summary, the prediction model seems to generally follow the direction of the actual returns, but with reduced volatility, indicating it may be smoothing out some of the short-term fluctuations.

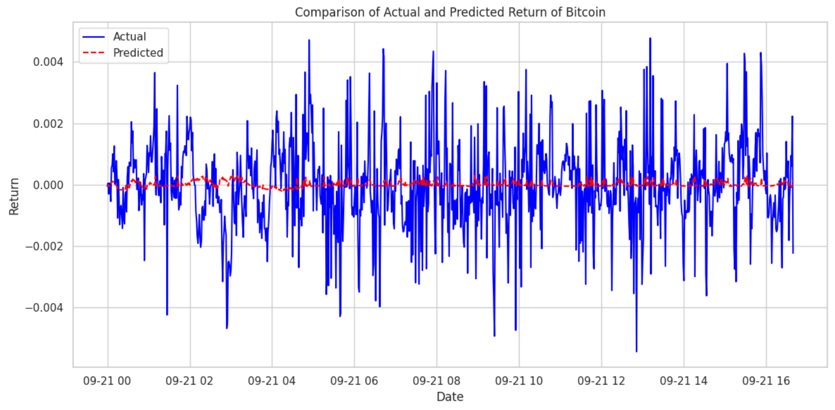

Figure 3: Predicted and actual return of Bitcoin from 2021-9-21.

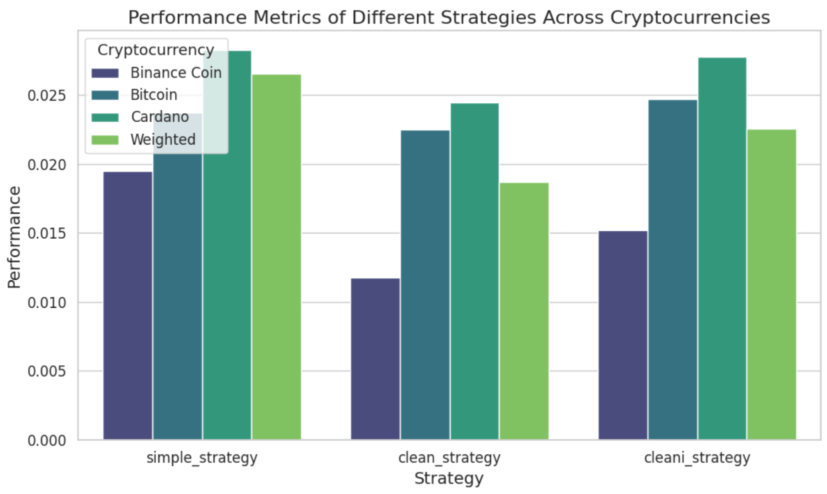

Figure 4: Performance of Investment Strategies on Cryptocurrencies, depicting metrics for Binance Coin, Bitcoin, Cardano, and a weighted average

The bar chart depicted in Figure 4 provides a comparative analysis of three distinct investment strategies applied to various cryptocurrencies. ReturnStrategy is what we employ, and others are two simplified strategies without the return parameters. We can observe the following:

- The *ReturnStrategy* shows the highest performance for Cardano, slightly less for Bitcoin, and the least for Binance Coin.
- The *CleanStrategy* has a reduced performance across all cryptocurrencies compared to the *simple_strategy*, with the most notable decrease in Binance Coin.
- The *CleanStrategy2* demonstrates an improved performance relative to the *clean_strategy*, nearing the levels of the *ReturnStrategy* for Bitcoin and Cardano, but still below for Binance Coin.

In conclusion, while the *simple_strategy* yields the highest overall performance, the *cleani_strategy* may offer a balance between risk management and return, as indicated by its performance recovery from the *clean_strategy*. The consistent underperformance of Binance Coin across all strategies suggests it may be less responsive to these investment strategies compared to Bitcoin and Cardano.

4.4 Importance of Features

We plot the feature importances learned from LightGBM in Figure 5. The figure includes feature importance plots for a number of cryptocurrencies: Binance Coin, Bitcoin, Bitcoin Cash, Cardano, EOS.IO, Ethereum, Ethereum Classic, IOTA, and Litecoin. We conclude the findings as follows,

- All cryptocurrencies seem to have a set of common features, which include technical indicators such as LRM 60, LCM 60, CCI, BOP, among others, with "60" possibly indicating a time period.
- Certain features are particularly important for specific cryptocurrencies. For example, ADX 60 is most important for Binance Coin, while CMO is most significant for Bitcoin.
- The importance scores vary significantly across different cryptocurrencies, suggesting that the predictive models for each cryptocurrency rely on a different range of feature importance values.

It is worth noting that the variability in scales across cryptocurrencies suggests that the importance for each feature is normalized differently or that the models are inherently different.

The analysis of feature importance scores for cryptocurrencies can provide insights into the market behavior or trading patterns associated with each cryptocurrency, potentially aiding in the development of targeted trading strategies.

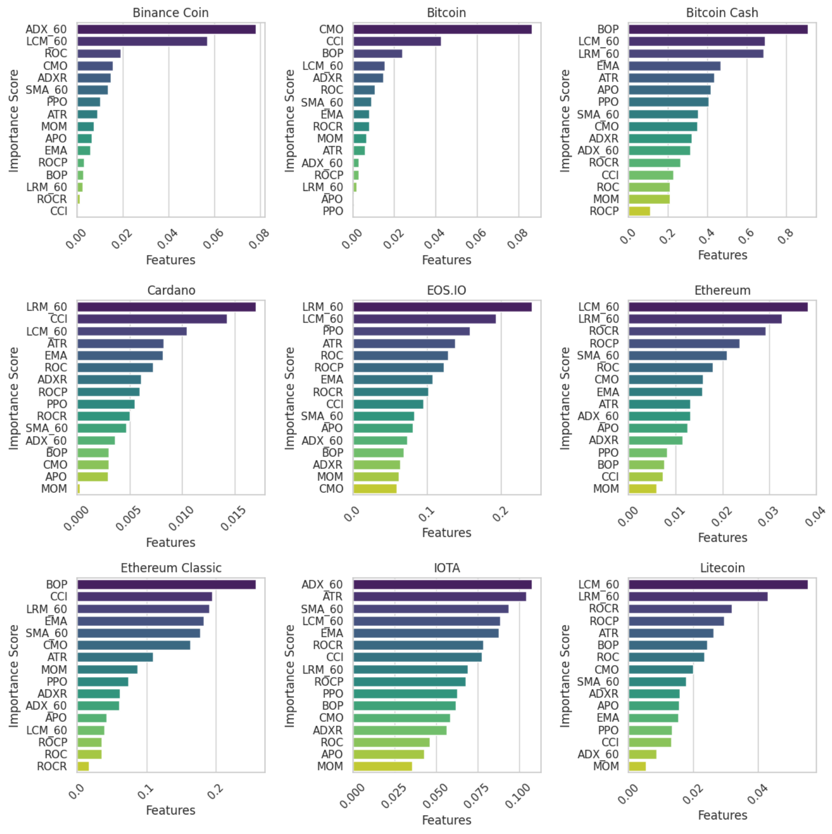

Figure 5: Feature importance learned from LightGBM.

4.5 Analysis on Deep Learning methods

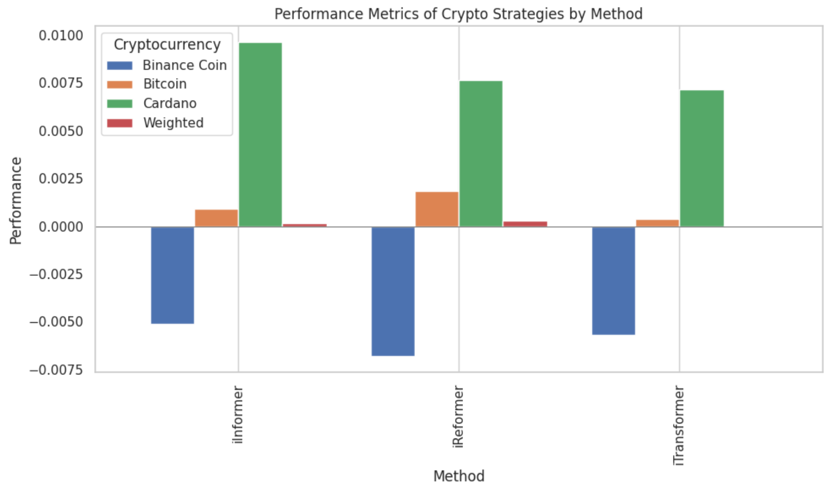

Figure 6: Performance of different deep models.

Based on the Figure 6, here's an analysis:

- All three methods resulted in a negative performance for Binance Coin, with the *iReformer* method showing the most significant negative impact.

- Bitcoin, on the other hand, showed a positive performance across all methods, with *iReformer* leading to the highest positive performance.
- The *iReformer* method seems to be the most volatile, showing the largest negative impact on Binance Coin but the highest positive impact on Bitcoin and a strong positive impact on Cardano.
- *iInformer*, while having a negative impact on Binance Coin, shows the best performance for Cardano and a moderate positive outcome for Bitcoin.

This analysis indicates that the choice of method could depend heavily on which cryptocurrency has a more dominant weight in an investor's portfolio or strategy preference. If the strategy is heavily weighted towards Bitcoin or Cardano, *iReformer* or *iInformer* might be favorable, respectively. However, if the strategy is to minimize losses on Binance Coin, *iTransformer* appears to be the least detrimental.

5 Conclusion

This study has explored the application of various machine learning techniques on cryptocurrency time series data, specifically within the context of the G-Research Crypto Forecasting Contest. Our exploration included rigorous data preprocessing methods such as cleaning, normalization, and feature engineering, which are crucial for enhancing the predictive quality of machine learning models. Among the tested algorithms—ElasticNet, LightGBM, and transformer-based methods—our findings reveal that LightGBM stands out for its effectiveness in modeling cryptocurrency price movements. In contrast, despite the advantages of deep learning models like transformers in handling sequential data, they have not performed as expected in our experiments.

These results highlight the dynamic and somewhat unpredictable nature of cryptocurrency markets, where simpler, more interpretable models like LightGBM can often outperform more complex models. This could be attributed to overfitting or the inability of deep learning models to generalize well in the highly volatile and noisy environment that characterizes cryptocurrency data.

References

[1] ANTONOPOULOS, A. M. *Mastering Bitcoin: Programming the open blockchain.* " O'Reilly Media, Inc.", 2017.

[2] DE PRADO, M. L. *Advances in financial machine learning.* John Wiley & Sons, 2018.

[3] GARETH, J., DANIELA, W., TREVOR, H., AND ROBERT, T. *An introduction to statistical learning: with applications in R.* Spinger, 2013.

[4] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. H., AND FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[5] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[6] JANSEN, S. *Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python.* Packt Publishing Ltd, 2020.

[7] KE, G., MENG, Q., FINLEY, T., WANG, T., CHEN, W., MA, W., YE, Q., AND LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (2017), NIPS.

[8] KITAEV, N., KAISER, Ł., AND LEVSKAYA, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[9] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *nature 521*, 7553 (2015), 436–444.

[10] LIU, Y., HU, T., ZHANG, H., WU, H., WANG, S., MA, L., AND LONG, M. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625* (2023).

[11] MAĆKIEWICZ, A., AND RATAJCZAK, W. Principal components analysis (pca). *Computers & Geosciences 19*, 3 (1993), 303–342.

[12] SHUMWAY, R. H., STOFFER, D. S., AND STOFFER, D. S. *Time series analysis and its applications*, vol. 3. Springer, 2000.

[13] SILVER, D., HUBERT, T., SCHRITTWIESER, J., ANTONOGLOU, I., LAI, M., GUEZ, A., LANCTOT, M., SIFRE, L., KUMARAN, D., GRAEPEL, T., ET AL. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[14] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[15] ZENG, A., CHEN, M., ZHANG, L., AND XU, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence* (2023), vol. 37, pp. 11121–11128.

[16] ZHOU, H., ZHANG, S., PENG, J., ZHANG, S., LI, J., XIONG, H., AND ZHANG, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (2021), vol. 35, pp. 11106–11115.

[17] ZOU, H., AND HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2 (2005), 301–320.

6 Appendix

6.1 Detailed Features Design

- Simple Moving Average (SMA) configurations:
  - SMA with time period 15
  - SMA with time period 60
- Exponential Moving Average (EMA) with default settings

  Momentum Indicators

  - Average Directional Index (ADX):
    - ADX with time period 30
    - ADX with time period 60
  - Balance of Power (BOP) with default settings
  - Average Directional Movement Index Rating (ADXR) with default settings
  - Absolute Price Oscillator (APO) with default settings
  - Commodity Channel Index (CCI) with default settings
  - Chande Momentum Oscillator (CMO) with default settings
  - Percentage Price Oscillator (PPO) with default settings
  - Momentum (MOM) with default settings
  - Log Close Mean (LCM):
    - LCM with time period 30
    - LCM with time period 60
  - Log Return Mean (LRM):
    - LRM with time period 30
    - LRM with time period 60
  - Rate of Change (ROC) with default settings
  - Rate of Change Percentage (ROCP) with default settings
  - Rate of Change Ratio (ROCR) with default settings
  - Average True Range (ATR) with default settings