G-Research Crypto Forecasting Project Report

Zezhen Ding
Department of Industrial Engineering and Decision Analytics, HKUST
Hong Kong SAR, China
zdingah@connect.ust.hk

Abstract

G-Research Crypto Forecasting is a competition held on Kaggle. This competition hopes that participants will use K-line data to predict the future return of crypto assets. There are a total of 14 types of crypto assets, with data from January 1, 2018 to September 21, 2021. This study uses this competition data to build a price prediction model and build a position strategy to evaluate whether the model can make a profit in the real market. This project combines time series characteristics, uses feature mining, feature sharing, data fusion, and data augmentation methods to process data, and uses the LightGBM fitting data to implement a profitable model.

1 Introduction

1.1 Backgroud

Crypto assets have become a rapidly developing new type of financial asset. Since 2017, the market value of major crypto assets like Bitcoin has expanded rapidly, and the global user base has also been steadily increasing. According to statistics, over 5,000 crypto assets have been developed worldwide currently. Despite the overall upward trend, cryptocurrency prices differ significantly from traditional financial assets such as stocks and futures. Crypto assets prices have price fluctuations that are large and volatility is strong, influenced more by external micro events, lack fundamental support, with prices mainly determined by market sentiment and speculative behavior. As an emerging asset class, the regulatory environment is not yet mature, with greater uncertainty factors. Accurately predicting and analyzing cryptocurrency price movements is highly meaningful for traders and investors. On one hand, it can provide reference for investment decisions to reduce risks; on the other hand, it helps to gain in-depth understanding of the various factors influencing prices, and provide basis for improving regulatory standards. In addition, developing prediction models can also test and verify economic and financial principles that impact price formation, to solve questions and propose new insights for related research fields.

1.2 Data overview

This dataset contains information on historic trades for several crypto-assetss, such as Bitcoin and Ethereum. Your challenge is to predict their future returns.
The train data set contains following features and target:

timestamp - A timestamp for the minute covered by the row.
Asset_ID - An ID code for the crypto-assets.
Count - The number of trades that took place this minute.
Open - The USD price at the beginning of the minute.
High - The highest USD price during the minute.
Low - The lowest USD price during the minute.
Close - The USD price at the end of the minute.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Volume - The number of crypto-assets units traded during the minute.
VWAP - The volume weighted average price for the minute.
Target - 15 minute residualized returns.

1.3 Data analysis

From above information, we counted the number of samples 24236807. There are 1956960 unique timestamp's, from January 1, 2018 to September 21, 2021. There are 14 crypto-assets, and their sample sizes are not exactly the same. Some crypto-assets are not covered until the end of time(Figure 1). This is because some crypto-assets may have been created at an intermediate time. In more detail, the data is one piece per minute, but there are also times when there is no data, that is, there is some discontinuous data. It can also be observed from Figure 2 that the price of crypto assets fluctuates greatly, with a 20-fold difference between the highest price and the lowest price.

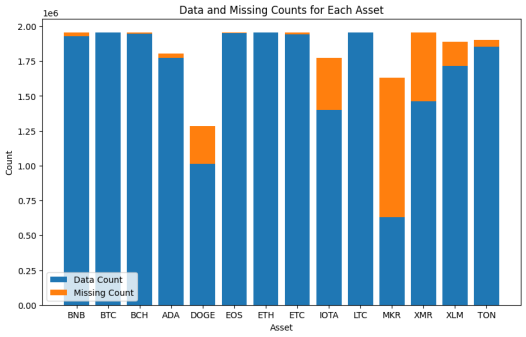

Figure 1: Data and Missing Counts for Each Asset.

Figure 2: Bitcoin's monthly candlestick.

1.4 Project target

Original Target The original target is provide in train data which is predict returns in the near future for prices P^a, for each asset a. For each row in the dataset, there include the target for prediction. Target is derived from log returns (R^a) over 15 minutes.

$$R^a(t) = log(P^a(t+16) \ / \ P^a(t+1)) \tag{1}$$

Crypto asset returns are highly correlated, following to a large extend the overall crypto market. As we want to test your ability to predict returns for individual assets, we perform a linear residualization,

removing the 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:

$$M(t) = \frac{\sum_a w^a R^a(t)}{\sum_a w^a}$$

$$\beta^a = \frac{\langle M \cdot R^a \rangle}{\langle M^2 \rangle}$$

$$\text{Target}^a(t) = R^a(t) - \beta^a M(t) \tag{2}$$

where the bracket $\langle . \rangle$ represent the rolling average over time (3750 minute windows), and same asset weights w^a used for the evaluation metric.

Ret900s However, we found that this Kaggle competition can no longer be submitted, which means that we cannot compete fairly with the contestants. And when we wanted to recreate Target based on (1) and (2), we discovered some problems as similar previous work(Recreating Target) show. First the $R^a(t)$ in the training data is calculate by (3), and when gap in data appears, the next 3750 rows $\beta^a(t) = 0$ and $\text{Target}^a(t) = R^a(t)$. This incorrect handling of the target resulted in a large difference between the first and second stage results of the competition. We think this is not a good target, at least β^a should skip missing data. In order to make the target goal more clear, this project finally chose to use R^a as the training target, and we called it Ret900s.

$$R^a(t) = (P^a(t+16)/P^a(t+1)) - 1 \tag{3}$$

1.5 Evalation

Weighted correlation The evaluation metric is (weighted correlation) as opposed to a weighted mean of correlation. The metric is defined as follows, where 'a', 'b' and 'weights' are vectors of the same length.

$$S = \sum_{i=1}^{n} w_i$$

$$\mu_a = \frac{1}{S}\sum_{i=1}^{n} w_i a_i \quad \mu_b = \frac{1}{S}\sum_{i=1}^{n} w_i b_i \quad \sigma_a^2 = \frac{1}{S}\sum_{i=1}^{n} w_i (a_i - \mu_a)^2 \quad \sigma_b^2 = \frac{1}{S}\sum_{i=1}^{n} w_i (b_i - \mu_b)^2$$

$$corr = \frac{1}{\sigma_a \sigma_b}\frac{1}{S}(\sum_{i=1}^{n} w_i a_i b_i - \mu_a \mu_b) \tag{4}$$

2 Pipline

The experiment adopted a relatively standard pipline as Figure 3. First, the data was mined for features, then the data of different crypto assets or features at the same time were fused, and finally the time series cross-validation method was used to evaluate the performance of the LightGBM model.

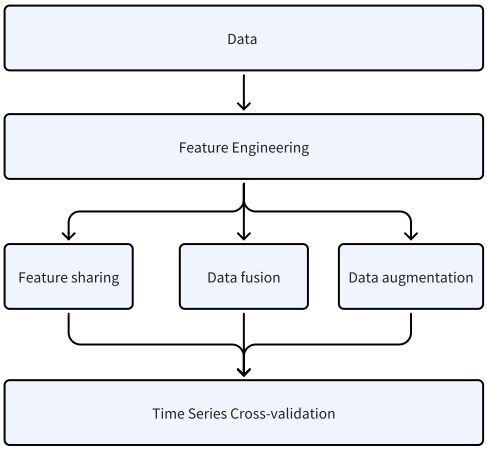

Figure 3: Project pipline.

2.1 Feature Engineering

We performed feature engineering by leveraging common technical indicators to construct the relevant features. We processed the features as symmetrically as possible around 0 to ensure that the features are not impacted by market conditions. Finally, 36 features were generated. The numerical distribution of features is shown in Figure4. Feature is generated through the following technical indicators.

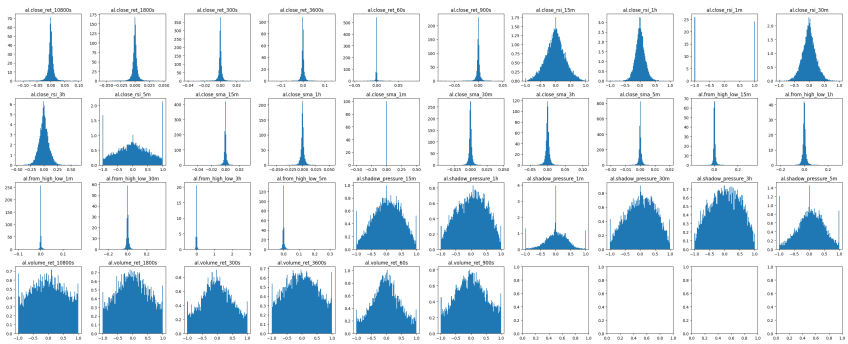

Figure 4: Feature distribution.

Simple Moving Average (SMA) SMA is a method used to smooth out price data by creating a constantly updated average price. It assigns equal weight to all values.

$$SMA = \frac{Sum\ of\ n\ periods}{n}$$

Exponential Moving Average (EMA) EMA is a type of moving average that gives more weight to recent prices, which can make it more responsive to new information compared to a simple moving average. The EMA is calculated by giving more weight to the most recent price.

$$EMA = (Close - Previous\ EMA) * (2/(n+1)) + Previous\ EMA$$

Moving Average Convergence Divergence(MACD) This is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price.

$$MACD = EMA_{n1} - EMA_{n2}$$

Relative Strength Index(RSI) This is a momentum oscillator that measures the speed and change of price movements. It is used to identify overbought or oversold conditions in a market.

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

Bollinger Bands This is a volatility indicator and commonly used in stock market trading. It consists of a Simple Moving Average (SMA) and two standard deviation lines, usually 2 standard deviations away from the SMA.

$$UpperBand = SMA + (StdDev * STD)$$
$$LowerBand = SMA - (StdDev * STD)$$

Return calculation

$$Return = \frac{Current\ Price - Previous\ Price}{Previous\ Price}$$

Shadow features

$$UpperShadow = High - max(Open, Close)$$
$$LowerShadow = min(Open, Close) - Low$$

2.2 Data Process

After generating features, we still need to do some data processing to improve data utilization.

data fusion Although crypto assets have different data ticksizes, they have similar data structures. We can try to share data to increase the amount of data and make the model less likely to overfit. The specific method is shown in Figure5.

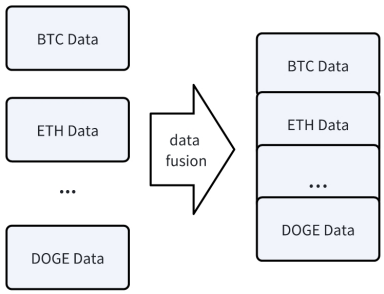

Figure 5: data fusion.

feature sharing In cryptocurrencies, the price changes of assets are often influenced by BTC and ETH. Therefore, we can consider adding their features to other assets in order to represent the state of the market. The specific method is shown in Figure6.

data augmentation We find that the bullish and bearish movements in the market are sometimes imbalanced. To achieve balanced uptrends and downtrends in the market, we take the negatives of all features and targets. This is also the reason why we ensure the features are symmetrically distributed around 0 when constructing them.

2.3 Time Series Cross validation

Time series cross validation is a technique used to evaluate predictive models on time series data when the goal is forecasting into the future. It helps avoid data leakage that can occur when using standard cross validation. In time series cross validation, the data is split based on time rather than

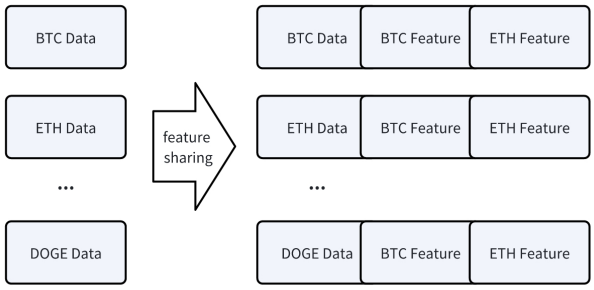

Figure 6: Feature sharing.

randomly. The training set will contain observations from the past, while the validation/test set will contain observations from the future. For example, observations from time periods 1 to t-1 may be used for training, while observations from time period t are used for evaluation.This process is then repeated by moving the validation set forward by one time step. So observations from time periods 1 to t would become the new training set, while observations from time period t+1 form the new test set.Common variants include rolling origin validation where multiple folds are created by sliding the validation window, and walk-forward validation where models are retrained on all historic data for each time step.The goal is to mimic the real world scenario of using only past observations to predict future ones, thus providing an unbiased assessment of a model's true predictive capability on time series data. It helps address issues like data snooping that standard cross validation cannot.
In our experiments, we have a window width of 3 years for the training set and a window width of 1 month for the test set, with a step size of 1 month. The specific method is shown in Figure7.

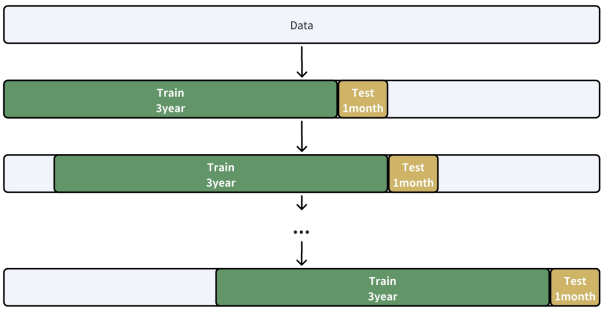

Figure 7: Time Series Cross validation.

3 Experiments

3.1 Model

Linear Regression (LR) is a fundamental algorithm in machine learning and statistics, designed for solving regression problems. It operates under the assumption of a linear relationship between the input features and the output values to be predicted. The model's coefficients and bias are determined by minimizing a certain "distance" or error, such as the least squares error, between the predicted linear model and the actual target values in the training set. Despite the potential limitations of its linear assumption in complex real-world scenarios, this model remains a popular choice due to its straightforward implementation and high interpretability.

Lasso Regression(LASSO) is a type of linear regression that uses shrinkage, where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This particular type of regression is well-suited for

models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

Light Gradient Boosting Machine (LGBM) is an efficient and light-scale gradient boosting framework based on decision tree algorithms. It adopts two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling. These two techniques address the limitations of histogram-based algorithms that were primarily used in all previous versions of Gradient Boosting Decision Tree models. Briefly, the model first fits a decision tree model on the training data, and then adjusts the weight of each instance in the training data based on the prediction results of the previous tree. Afterwards, it fits another new tree on the adjusted data and makes predictions using an ensemble of the new tree and the previous tree. This procedure is performed iteratively for a pre-specified number of steps, and the final model is an ensemble of all trees obtained during the training process.

3.2 Main result

In our experiment setup, we compared relatively overfitting-resistant models such as linear Regression, Lasso and LightGBM. The hyperparameters for Lasso were alpha=0.01, and for LightGBM they were $num\ of\ leaves = 10$, $max\ of\ depth = 5$, $num\ of\ estimators = 100$. The experiments showed that LightGBM performed best. The poor performance of the linear model may be because the features generated based on technical indicators are weak and contain a lot of noise, so the linear model cannot fit well.

Table 1: Performance of different model

Model	Weight Correlation(1e-2)
LR	4.20
LASSO	2.66
LGBM	**8.64**

3.3 Ablation of Method

When using LightGBM, we compared the performance improvements from different data processing methods, as shown in Table2. Rolling train refers to using time series cross validation, while not using Rolling train means only using data from the first 3 years as the training set and 2021 data as the test set. The experiments demonstrated that Rolling train, Data fusion, Feature sharing and Data augmentation all led to stable performance gains. Rolling train brought better performance by capturing evolving market patterns over time, as the training data was closer to the test period. Data fusion and Data augmentation increased model robustness to some extent. However, we found Data augmentation counteracted when used together with all methods, indicating the current approach is not ideal. Feature sharing provided more predictive features to the model, enhancing its performance.

Table 2: Performance of different method

Method	Choise							
Rolling train		✓				✓	✓	✓
Data fusion			✓			✓	✓	✓
Feature sharing				✓			✓	✓
Data augmentation					✓			✓
Weight Correlation(1e-2)	5.06	6.15	6.60	5.51	5.25	7.24	**8.64**	5.78

3.4 Simple backtest

Backtest Correlation does not assess whether a model can make money In the field of quantitative finance, backtest is a method used to evaluate the effectiveness of an investment strategy. It involves applying the strategy to historical market data and simulating actual trades to calculate performance metrics. Through backtest, investors can assess the performance of an investment strategy in the

past and make strategy improvements or decisions based on the backtest results. We formulate the following position strategy. Assume that the signal output by the model is α. We open a position with the value of α every minute. If α is positive, we will go long, and if α is negative, we will go short. The position will be closed after 15 minutes. At the same time, it is assumed that the handling fee for each opening and closing of a position is fee. Then we will be able to calculate Mean Profit and Mean Fee.

$$\text{Mean Profit} = \frac{\sum(\alpha \times \text{Ret900s})}{\sum|\alpha|} * 10000 \tag{5}$$

$$\text{Mean Fee} = \frac{\sum|\alpha_t - \alpha_{t+900s}| * fee}{\sum|\alpha|} * 10000 \tag{6}$$

From Table3, it is evident that the Mean Profit for each asset surpasses the Mean Fee, indicating that utilizing this strategy can yield profits for each asset. With the exception of BTC, XMR, and TON, which exhibit relatively weaker profitability, the other assets demonstrate similar levels of profitability. This suggests that our model is relatively stable. However, a closer examination of Figure8 reveals that the majority of profits are concentrated within specific time periods. This is not an ideal scenario because, outside of these concentrated profit periods, the strategy consistently incurs losses.

Table 3: Backtest of different assets

Asset	Mean Profit	Mean Fee	Asset	Mean Profit	Mean Fee
ADA	37.43	6.27	ETH	24.66	6.27
BCH	24.61	6.18	IOTA	25.64	6.31
BNB	29.48	6.20	LTC	28.22	6.22
BTC	10.91	6.10	MKR	26.73	6.12
DOGE	37.93	5.98	TON	15.25	6.11
EOS	27.97	6.14	XLM	34.97	6.19
ETC	34.40	6.05	XMR	16.88	6.10

Figure 8: Profit changes of BTC.

4 Conclusion and future work

Overall, this project has successfully implemented a profitable strategy through feature engineering and a series of data processing and training techniques. Currently, the strategy relies on capturing specific time periods to generate profits, while in ordinary times, the model consistently incurs losses. Based on this observation, future considerations could involve setting some model outputs with values close to zero to zero, reducing the frequency of opening positions and saving transaction fees. In terms of this approach, there is still much room for further exploration in feature engineering and data augmentation. In the context of the Kaggle competition itself, we have also observed some participants who forgo feature engineering and directly utilize neural networks to solve the problem, which is another direction worth exploring in the future.