**Use of XG Boost to Predict Kline Data of BTCINR**

**Methodology:**

1. I begin by loading Kline data from APIs and organizing it to facilitate an efficient split for training and testing.
2. From the dataset, I extract 1,500 data points and split them into training and testing sets in a ratio of 14:1, allocating 1,400 data points for training and 100 for testing.
3. To train my XGBoost model, I utilize five key features: open, close, high, low, and volume, ensuring a comprehensive representation of the market data.
4. I set the random\_state parameter to 42. This arbitrary integer is chosen to guarantee consistency and reproducibility in my model's results, enabling me to achieve reliable outcomes across multiple runs.
5. I implement GridSearchCV to optimize the hyperparameters of my XGBoost model, exploring various combinations of parameters such as n\_estimators, learning\_rate, max\_depth, and min\_child\_weight. This step allows me to identify the best configuration for my model.
6. Using the optimized model, I make predictions on all target variables: open, high, low, and close prices. I then train separate models for each target to ensure tailored predictions based on the specific characteristics of each price.
7. I create a comprehensive comparison DataFrame to evaluate the model's performance. This includes actual and predicted values for all four target variables, alongside corresponding timestamps and volume data.
8. I calculate error metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for each target variable to assess the accuracy of my predictions. These metrics provide insights into the model's performance and help identify areas for improvement.
9. Finally, I visualize the results using side-by-side candlestick charts for actual versus predicted values, providing a clear and intuitive representation of the model's predictive capabilities.

**Limitations of the Model:**

1. The model is trained on a relatively small dataset of 1,500 data points, which may not adequately capture the full range of market variability and trends.
2. With complex models like XGBoost, there is a significant risk of overfitting to the training data, particularly if the hyperparameters are not optimally tuned.
3. The model relies on a limited set of features—namely open, close, high, low, and volume—potentially neglecting other important predictors that could enhance its performance.
4. The model treats each data point independently, failing to account for the temporal dependencies and autocorrelation that are critical in time series analysis.
5. Additionally, the model does not consider market fluctuations driven by news events, which can lead to inaccuracies in its predictions.
6. A disadvantage of using Grid Search in my XGBoost methodology is that it can be computationally expensive and time-consuming, especially with a large hyperparameter space. Evaluating every possible combination of hyperparameters may lead to long processing times and increased resource usage. Additionally, Grid Search might miss optimal hyperparameters if they fall between the specified values, potentially resulting in suboptimal model performance.

**Potential Solutions:**

1. Increase Dataset Size: Gather more historical data from the API to enhance variability.
2. Prevent Overfitting: Use cross-validation to assess model performance and optimize hyperparameters.
3. Expand Feature Set: Incorporate additional technical indicators to capture more market dynamics.
4. Account for Temporal Dependencies: Utilize time series forecasting models such as LSTM to recognize trends over time.

Metrics Summary:

