Bitcoin Price Analysis using Plotly and Forecasting Using Real-Time Kraken API and LSTM

Introduction

In the era of digital finance, cryptocurrency stands at the forefront of market innovation, with Bitcoin leading the charge. This project focuses on analyzing and forecasting Bitcoin price movements using real-time and historical data from the Kraken API. The goal is to integrate real-time monitoring with statistical analytics and deep learning, culminating in a robust visualization and predictive modeling pipeline. The implementation involves fetching and preprocessing financial data, performing real-time tracking, generating interactive visualizations, and using an LSTM model to forecast future price points. All visualizations are rendered using Plotly to ensure interactivity and clarity. This document outlines the steps and methodology used throughout the project.

Real-Time and Historical Data Analysis

Data Retrieval and Preprocessing

The `BTCPriceTracker` class is responsible for extracting current and historical price data. Real-time prices are obtained via Kraken's Ticker endpoint, while the historical data is gathered using their OHLC (Open-High-Low-Close) interface for the past 10,000 minutes. All timestamps are converted to human-readable datetime formats, and the close prices are used for subsequent analysis. Once the real-time data stream concludes (typically by a KeyboardInterrupt), the combined historical and real-time dataset is cleaned and aligned based on time indices.

Analytics and Visual Insights

The merged dataset undergoes a series of analytics operations. First, a 5-point Simple Moving Average (SMA) is computed to highlight price trends. Next, volatility is calculated using a rolling standard deviation, and percent change is computed to identify short-term price momentum. These insights are visualized across three Plotly subplots:

1. **BTC Price with Simple Moving Average**: Displays the raw price and smoothed SMA line.
2. **Bitcoin Price Volatility Bands**: Shades ±1 standard deviation around the price.
3. **BTC Price % Change Over Time**: Plots the percentage change in value.

These interactive charts allow users to understand patterns in price movements, detect volatility spikes, and assess market behavior over time.

Class and Function Descriptions

- **`BTCPriceTracker.get_latest_price()`** retrieves the most recent BTC price from Kraken.
- **`BTCPriceTracker.get_historical_data()`** fetches OHLC data for a given time window and returns a DataFrame of closing prices.
- **`BTCForecaster.__init__()`** initializes the scaler and sequence length needed for the LSTM model.
- **`BTCForecaster.prepare_sequences()`** normalizes and slices the dataset into X and y sequences for supervised learning.
- **`BTCForecaster.train_model()`** defines the LSTM architecture, compiles it, and trains it on the provided data.
- **`BTCForecaster.forecast()`** generates predictions from the most recent window using the trained model.
- **`evaluate_predictions()`** calculates MAE and MSE between predicted and actual price points.
- **`plot_forecast_plotly()`** visualizes the LSTM model forecast against actual data.
- **`plot_combined_dashboard()`** creates a unified Plotly figure with all four major visualizations.

Forecasting with LSTM Neural Network

Model Architecture and Training

Forecasting was handled using the `BTCForecaster` class. A Sequential LSTM model was created, trained on sequences of normalized historical close prices. The input to the model consists of 60-minute sliding windows, and the target is the following 60 data points. The dataset was scaled using `MinMaxScaler` for compatibility with the LSTM's activation functions. The model comprises one LSTM layer with 64 units and a Dense output layer predicting the next 60 points. It was trained over 5 epochs with a batch size of 64.

Forecasting Results and Evaluation

After training, the model was used to forecast the next 60 minutes based on the most recent window. The actual recent prices and the forecasted values were plotted using Plotly for direct comparison. Model performance was evaluated using:

- **Mean Absolute Error (MAE):** 106.14
- **Mean Squared Error (MSE):** 14,219.69

This indicates that while the model captures overall directional trends, precise short-term forecasts remain challenging due to the noisy nature of financial time-series data.

Visual Results

1. **Simple Moving Average Plot**: The Plotly graph clearly shows how the SMA smooths out the noise in BTC price fluctuations, aiding in trend recognition.
2. **Volatility Bands**: These plots demonstrate areas where BTC prices deviate sharply from the mean, which is crucial for risk management.
3. **Percentage Change**: Highlights significant moments of sharp market activity and micro-trend reversals.
4. **Forecast Plot**: Displays actual vs predicted prices, helping users visually assess forecasting accuracy.

All plots were displayed interactively using Plotly, allowing for zoom, pan, and hover-tool insights.

Conclusion

This project demonstrates the integration of real-time cryptocurrency data analysis with advanced deep learning forecasting. The Kraken API provided reliable and high-resolution BTC price data, while the analytics section offered meaningful insights through moving averages, volatility, and rate of change. The LSTM model served as an effective tool for short-term forecasting, though its predictive accuracy remains influenced by the inherent volatility of Bitcoin. Future improvements may include experimenting with bidirectional LSTMs, attention mechanisms, or hybrid models combining statistical and neural approaches. Ultimately, this pipeline sets a solid foundation for further research in financial prediction and real-time AI analytics.

Additional Analysis and Future Scope

Importance of Real-Time Data

The use of real-time data not only enhances the relevance of analysis but also improves responsiveness to market shifts. In fast-moving markets like cryptocurrency, the value of real-time pipelines cannot be overstated. This project's real-time integration provides actionable insights for time-sensitive applications such as trading algorithms and market monitoring dashboards.

Role of Feature Engineering

While this project primarily focuses on close prices, additional features such as trade volume, high/low bands, or market sentiment indicators (e.g., social media trends) could be integrated. Incorporating such signals may improve model learning and enhance prediction stability.

Model Interpretability

Although LSTM models offer high performance, they operate as black boxes. Adding explainable AI (XAI) tools to interpret forecast drivers could foster trust in algorithmic decision-making, particularly in financial environments where transparency is critical.

Error Analysis and Resampling

The errors observed in MAE and MSE suggest opportunities for optimization. One possible approach includes varying the sampling rate. For instance, training models on 5-minute intervals rather than 1-minute could reduce noise while still capturing meaningful trends.

Extending Forecast Horizon

This project limits forecasts to one hour ahead. Future extensions could include multi-hour or daily forecasts using stacked LSTMs or temporal convolutional networks (TCNs). Such models could uncover long-term cyclical behavior in Bitcoin markets.

Deployment Considerations

The implemented pipeline can be wrapped into an API or web app using frameworks like Flask or Streamlit. This would enable user-friendly interfaces for non-technical stakeholders. Additionally, integration with alert systems could provide real-time signals based on forecast deviations or volatility anomalies.

Comparison with Baseline Models

Benchmarking LSTM performance against simpler models like ARIMA or linear regression could offer context to its effectiveness. If LSTM significantly outperforms these baselines, it justifies the added complexity and computational overhead.

Cross-Market Prediction

Incorporating data from related markets such as Ethereum or major financial indices (e.g., NASDAQ, S&P 500) could be explored. Multivariate time-series models could detect co-movements, leading to better risk management strategies.

Cloud and Edge Integration

For scalability, this system could be deployed on cloud platforms (e.g., AWS Lambda, Google Cloud Functions) or on edge devices to support decentralized finance applications. Real-time forecasting at the edge would be particularly useful for mobile or IoT-based trading tools.

Ethical and Economic Considerations

As with any financial AI system, ethical concerns must be addressed. Forecasts can influence trading behavior and market volatility. Responsible usage policies, model disclaimers, and fairness audits should be implemented before deploying any such model in production.