XRP Price Prediction using LSTM

Learning Issues and Rectifications for

# 1. Plan of Action

The goal was to forecast the next 5 minutes of XRP cryptocurrency prices using historical data and live streaming data. The project was divided into several stages: loading and preprocessing data, building a suitable LSTM model, performing predictions, saving forecasted results, and optimizing the model for real-time predictions.   
Throughout the development process, various issues arose during different phases of the project, requiring iterations, debugging, and rectifications. The final code addresses these problems and optimizes the pipeline for effective price predictions.

# 2. Learning Issues Faced

## 2.1 Data Loading and Format Issues

Initial Problem: Data from historical files and the live file were not consistently formatted. The Timestamp column was often misinterpreted due to time zone differences, and the data types (especially for XRP Price) sometimes weren't properly parsed, leading to misalignment in training inputs.  
Rectification: Ensured consistent date parsing (parse\_dates=['Timestamp']) across both historical and live files. Cast XRP Price as a float to maintain numerical consistency.

## 2.2 Data Preprocessing Challenges

Initial Problem: The LSTM model was not accepting the initial data shape. LSTM expects 3D input data with the shape (samples, time\_steps, features), but the original approach had mismatched dimensions.  
Rectification: Preprocessed the data by reshaping it to (samples, 1, 1). This format ensures each sample contains one time step with one feature (XRP Price).

## 2.3 Model Design and Overfitting

Initial Problem: The first versions of the model suffered from overfitting due to a simple model structure and insufficient data. Although the model performed well on training data, its predictions on live data were inaccurate.  
Rectification: To mitigate overfitting, the following steps were taken:  
- Batch Size: Reduced batch size to 32 to give the model more opportunities to learn from different subsets of the data.  
- Early Stopping: Introduced an EarlyStopping callback to halt training if the model’s performance did not improve, preventing overfitting.

## 2.4 Efficient Model Weights Management

Initial Problem: The model was retraining from scratch every time the script was run, which was inefficient and time-consuming.  
Rectification: Implemented checkpointing to save the model weights after each successful training iteration. On subsequent runs, the code checks if these weights exist and loads them if available, reducing the training time significantly.

## 2.5 Prediction for Live Data

Initial Problem: Predictions were initially based solely on historical data, but the requirement was to also incorporate real-time data from xrp\_rates\_live.csv. There were also issues with incorrect timestamps during forecasting, leading to misaligned predictions.  
Rectification: Modified the forecasting loop to include the latest available price from the live file. Used pd.date\_range() to ensure that the Timestamp for each forecasted minute aligns with real-world time.

## 2.6 Forecast File Overwriting Issue

Initial Problem: Forecast data was being overwritten in xrp\_forecast.csv after each iteration, erasing previous predictions.  
Rectification: Changed the approach to append new forecast results instead of overwriting them. Added a processed\_time column to store when each forecast was processed, making it easier to track updates.

## 2.7 Long Training Time and Efficiency

Initial Problem: Training the LSTM model with all the historical data (two years' worth) each time was inefficient and took several hours. The model was also slow to make predictions, especially when retraining on a continuous basis.  
Rectification: Split the tasks to handle historical data and live data separately. The model is now pre-trained on historical data and loaded from weights, focusing on making fast predictions using the latest live data, thus improving efficiency.

# 3. Final Code Changes

The final version of the code is designed to load data, preprocess it, build an LSTM model, and then forecast the next five minutes' worth of XRP prices based on live data. Predictions are appended to the forecast file along with a processed\_time column to record the timestamp of the predictions.

## 3.1 Model Training Optimization

- Pre-trained the model on historical data, saving the model's weights to avoid retraining from scratch.  
- Introduced ModelCheckpoint and EarlyStopping for efficient model management.

## 3.2 Handling Live Data and Appending Forecasts

Modified the code to incorporate predictions from live data stored in xrp\_rates\_live.csv. Appended predictions to the xrp\_forecast.csv file instead of overwriting it.

## 3.3 Forecasting Loop with Real-time Processing

The run\_forecasting\_loop function now continuously updates every 5 minutes, using the most recent XRP price to make predictions.

# 4. Future Enhancements

While the current version optimizes the prediction loop, further improvements could include:  
1. Advanced Model Architectures: Testing GRU (Gated Recurrent Units) or Transformer models to improve prediction accuracy.  
2. Hyperparameter Tuning: Fine-tuning the LSTM's hyperparameters, such as the number of units and layers, to find an optimal configuration.  
3. Incorporating More Features: Including other market indicators or features like trading volume, volatility, or technical indicators to enrich the model’s inputs and improve prediction quality.

# 5. Conclusion

The final implementation is the result of continuous iterations and refinements to handle challenges related to data preprocessing, model overfitting, training time, and real-time forecasting. By resolving each issue step by step, the solution has been optimized for making efficient predictions on XRP prices while leveraging both historical and live data sources.