Multi-Asset High-Frequency Trading Supervised Learning Project Plan

Aamer Jalan
Arin Baswana

Spring 2025

Abstract

This document presents a detailed plan for a university-level project that builds a comprehensive supervised learning pipeline using raw high-frequency trading (HFT) data from the Binance API. The goal is to predict short-term directional price movements for multiple crypto assets by leveraging cross-asset order book dynamics. All data processing and modeling steps are implemented using only `pandas` and `NumPy`.

1 Project Goal

Objective: Develop a deep neural network (DNN) based multi-class classifier to predict the short-term (e.g., 10–20 tick horizon) price movements (up, neutral, down) for several crypto assets simultaneously. The model will integrate not only individual asset features but also capture cross-asset influences, particularly how BTC's market dynamics affect altcoins such as ETH, BNB, XRP, and LTC.

2 Data Collection and Processing

2.1 Data Collection from Binance API

- **Target Assets:** BTCUSDT, ETHUSDT, BNBUSDT, XRPUSDT, LTCUSDT.
- **API Endpoint:** `https://api.binance.com/api/v3/depth?symbol={symbol}limit=1000`
- **Method:** Use asynchronous programming with Python's `asyncio` and `aiohttp` to concurrently fetch raw order book data.
- **Output:** Store the raw JSON responses in a file (e.g., `data/raw/raw_crypto_orderbooks.json`) along with associated timestamps and asset identifiers.

2.2 Data Processing and Cleaning

- **Parsing:** Convert the JSON file into `pandas DataFrames` for each asset.
- **Feature Extraction:**
  - **Mid-Price:** $\text{mid} = \frac{\text{bid}+\text{ask}}{2}$
  - **Spread:** $\text{spread} = \text{ask} - \text{bid}$
  - **Volume Imbalance:**

$$\text{imbalance} = \frac{\text{bid\_volume} - \text{ask\_volume}}{\text{bid\_volume} + \text{ask\_volume} + \varepsilon}$$

  - **Rolling Volatility:** Compute the standard deviation of the mid-price over a rolling window (e.g., 1-second window).
- **Alignment:** Synchronize timestamps across assets to construct a unified cross-asset feature matrix.
- **Storage:** Save the cleaned and processed data into a CSV file (e.g., `data/processed/processed_data`

2.3 Train-Test Split

- **Time-Based Split:** Use the first 70% of the data (chronologically) as the training set and the remaining 30% as the testing set.
- **Label Generation:** For each asset, compute the future percentage change in mid-price over the prediction horizon. Discretize the change into three classes:
  - `+1` if the change $> \alpha$ (price up)
  - `0` if $-\alpha \leq \text{change} \leq \alpha$ (neutral)
  - `-1` if the change $< -\alpha$ (price down)
- **Feature Matrix:** Each training example includes the features from its own asset and the corresponding features from BTC (to capture cross-asset influence).

3 Model Selection and Mathematical Foundation

3.1 Chosen Model: Deep Neural Network (DNN) for Multi-Class Classification

- **Rationale:** A DNN is well-suited to capture the non-linear relationships in HFT data, including the complex interdependencies between multiple assets.
- **Architecture:**
  - **Input Layer:** Accepts the combined feature vector for an asset (including cross-asset features).

- **Hidden Layers:** Multiple layers with non-linear activation functions (e.g., ReLU).
- **Output Layer:** Uses softmax activation to provide probability estimates for three classes.

3.2 Mathematical Details

Forward Pass:

$$a^{[l]} = W^{[l]} z^{[l-1]} + b^{[l]}, \quad z^{[l]} = \mathrm{ReLU}(a^{[l]}), \quad l = 1, 2, \ldots, L-1,$$

with $z^{[0]} = x$ (the input vector). For the output layer:

$$a^{[L]} = W^{[L]} z^{[L-1]} + b^{[L]}, \quad \hat{y} = \mathrm{softmax}(a^{[L]}),$$

where the softmax function is defined as:

$$\hat{y}_k = \frac{e^{a_k^{[L]}}}{\sum_j e^{a_j^{[L]}}}.$$

Loss Function: Categorical cross-entropy loss for m training examples:

$$J = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_{ik} \log(\hat{y}_{ik}),$$

where $K = 3$ (for up, neutral, and down).

Backpropagation and Weight Update: Gradients are computed for each layer:

$$W^{[l]} \leftarrow W^{[l]} - \alpha \frac{\partial J}{\partial W^{[l]}}, \quad b^{[l]} \leftarrow b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}},$$

with learning rate α.

4 Project Directory Structure

```
project/
 data/
    raw/
       raw_crypto_orderbooks.json
    processed/
        processed_data.csv
 src/
    __init__.py
    data_collection.py      % Asynchronous functions for Binance API collection
    data_processing.py      % Parsing, cleaning, and feature engineering
    train_test_split.py     % Script to split data into training and test sets
    model.py                % DNN model implemented with NumPy
```

```
   train.py                 % Script for training the model
   evaluate.py              % Script for evaluating the model and generating metrics/plo
results/
   figures/                 % Plots, confusion matrix, etc.
   logs/                    % Training logs and metrics
README.md
main.py                    % Main script to run the entire pipeline
```

5 Expected Outcomes and Conclusions

- **Predictive Performance:** The trained DNN will accurately classify short-term price movements, with performance evaluated via metrics such as accuracy, precision, recall, and F1-score.
- **Cross-Asset Influence:** By incorporating BTC's features into the model for altcoins, the project will quantify the influence of BTC on other cryptocurrencies.
- **Pipeline Validation:** A robust, end-to-end pipeline—from raw data acquisition through to model evaluation—demonstrates the feasibility of constructing a complete HFT system using fundamental Python libraries.
- **Market Microstructure Insights:** Analysis of feature importance and model outputs will provide insights into the dynamics of order book data and the interplay between different asset classes.

6 Conclusion

This project will implement a sophisticated supervised learning model using raw Binance API data to forecast short-term price movements across multiple crypto assets. The process involves building a complete pipeline that collects, cleans, and processes high-frequency order book data, creates a unified cross-asset feature set, and trains a deep neural network from scratch with NumPy. The outcomes will provide both quantitative performance metrics and qualitative insights into market microstructure dynamics, establishing a robust foundation for further research and real-time trading applications.

References