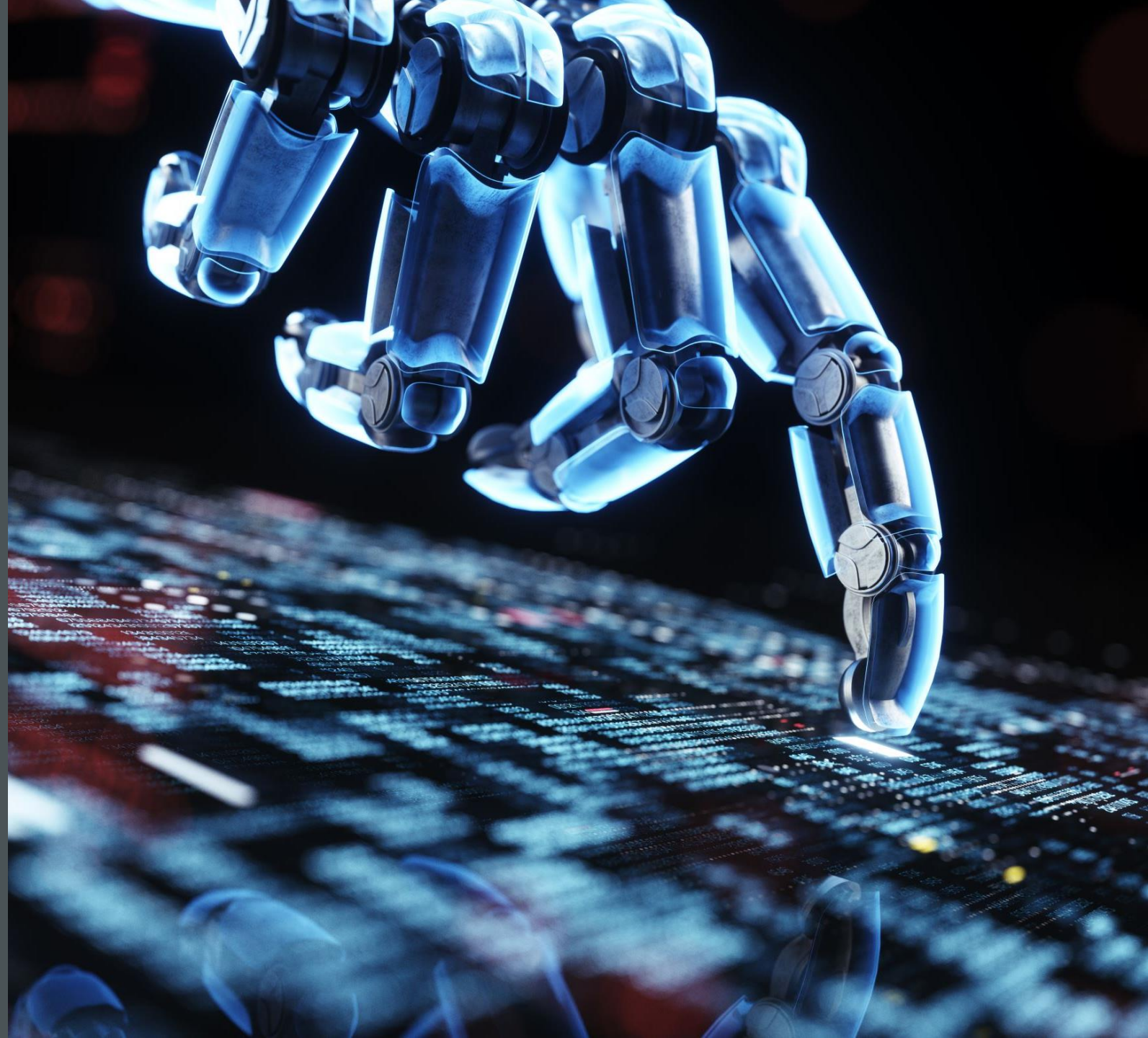
TRADE BOT REPORT

Zhaoyu Bai

2025 March



EXECTUTIVE SUMMARY

Introduction

- Addresses the opportunities and challenges of cryptocurrency trading, characterized by high volatility and rapid market changes.
- Highlights the limitations of manual trading, including the need for constant monitoring and high stress.
- Emphasizes the importance of a modular design for scalability and future upgrades.

Objective

- Develop a Python-based crypto trading bot that integrates with the Binance API.
- Incorporate key functionalities such as data handling, feature engineering, model development, signal processing, strategy and risk management, backtesting, and both live and mock trading.

EXECTUTIVE SUMMARY

Methodology & Implementation

- Leverages an event-driven architecture with real-time data flows for immediate market response.
- Uses a modular structure to simplify future expansions and allow easy modifications or replacements of individual components.
- Focuses on robust data processing, seamless integration of trading modules, and dynamic risk controls.

Results & Performance

- Demonstrated favorable performance metrics in backtesting, including ROI, drawdowns, and win rates.
- Validated the system's capability to manage risk effectively while executing trades in real time.

CONTENT

- Introduction
- Objectives
- Methodology & Implementation
- Results & Performance
- Future works

INTRODUCTION

- **Cryptocurrencies:** high volatility, rapid price movements, and continuous market development --- opportunities and challenges for short-term trading.
- **Manual Trading:** significant attention and constant market monitoring, workload and stress.
- **Scalability and Maintainability:** to handle future expansions, e.g. new trading strategies and upgrading existing components --- Modular design

OBJECTIVE

Develop a Python-based cryptocurrency trading bot integrating Binance API that does:

- ❑ **Data Handling**
- ❑ **Feature/Indicator Engineering**
- ❑ **Model Development**
- ❑ **Signal Processing**
- ❑ **Strategy Management**
- ❑ **Risk Management**
- ❑ **Backtesting**
- ❑ **Live and Mock Trading**
- ❑ **Logging and Reports**

The tradebot has been trough two major upgrade, the previous two does not integrate the Live and Mock trading functionality.

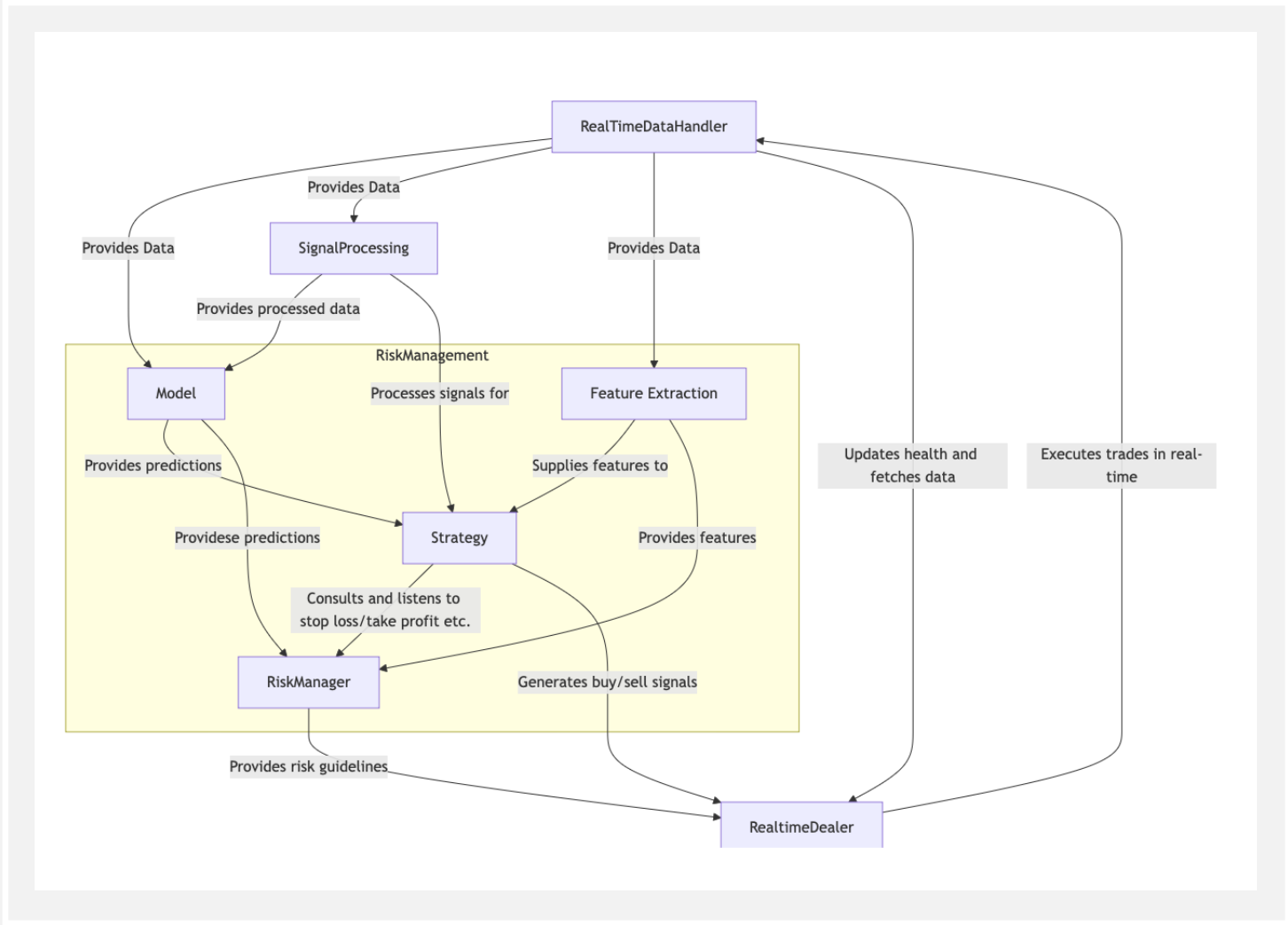
STRUCTURE OVERVIEW

Real-Time Data Flows

- Enables instant response to market changes
- Facilitates event-driven interactions across components

Modular Structure

- Increases scalability and flexibility
- Allows easy expansion, modification, or replacement of individual parts



METHODOLOGY & IMPLEMENTATION

DATA HANDLER

Data collection (historical/real-time), cleaning

METHODOLOGY: DATA HANDLER MODULE

- **Purpose & Objectives:**
  - Unified interface for managing, processing, and retrieving data (historical & real-time)
  - Designed for financial analysis and modeling
- **Core Principles:**
  - **Abstraction:**
    - Base class with common functionalities (e.g., configuration loading, URL construction, parameter management)
  - **Modularity:**
    - Separates core functionalities from specialized tasks in derived classes (HistoricalDataHandler, RealTimeDataHandler)
  - **Scalability & Flexibility:**
    - Adapts to various data sources and formats using dynamic configuration (JSON files)

METHODOLOGY: DATA HANDLER MODULE

- **Workflow Integration:**
  - **Configuration Management:**
    - Loads source URLs, endpoints, and parameters consistently
  - **Data Processing Pipeline:**
    - Standardized methods for data retrieval, error handling, and transformation
  - **Extensibility:**
    - Derived modules extend the base functionality for specific data types and operations

IMPLEMENTATION: DATA HANDLER MODULE

- **Core Components**
  - **Configuration Loader:**
    - Methods to load JSON configuration files for data sources, cleaning, and validation parameters
  - **URL Builder & Data Fetching:**
    - Constructs API endpoints and fetches data accordingly
  - **Utility Methods:**
    - Standardizes date formatting and file path generation for consistent data storage
- **Error Handling & Robustness**
  - Implements retry loops with delays for API request failures
  - Uses progress monitoring (via tqdm) for long-running data downloads

IMPLEMENTATION: DATA HANDLER MODULE

- **Extensions via Derived Classes**
  - **HistoricalDataHandler**
    - Loads pre-processed data from local files
    - Fetches data in chunks to manage API limits and memory usage
    - Integrates cleaning and rescaling routines (via DataCleaner and DataChecker)
  - **RealTimeDataHandler**
    - Streams live data feeds
    - Processes data in real time with rapid error handling and dynamic updates

FEATURE ENGINEERING

Technical/Customize Indicator calculation and selection

METHODOLOGY: FEATURE MODULE – EXTRACTION

- **Data Transformation:**
  - Convert raw input into a standardized format (e.g., Pandas DataFrame)
  - Clean, normalize, and structure data for uniform downstream processing
- **Modularity:**
  - Modular design allows extension with new extraction techniques
  - Encourages reusability and customization based on data type/requirements
- **Integration:**
  - Final output is a well-structured dataset for seamless integration with feature selection and modeling

METHODOLOGY: FEATURE MODULE – EXTRACTION

- **Technical Indicators:** Utilized the TA library for traditional indicators (e.g., moving averages, RSI, MACD)
- **Incremental Updates:** Developed an incremental update mechanism for technical indicators to efficiently process streaming or updated data
- **Custom Metrics:** Domain-specific features capturing unique patterns

METHODOLOGY: FEATURE MODULE – SELECTION

- **Evaluations:**
  - Assess relevance via statistical correlations
  - Use machine learning techniques
- **Dimensionality Reduction:**
  - Apply PCA or similar methods to reduce redundancy and feature space
- **Selection Criteria:**
  - Use variance thresholds and mutual information to decide which features to retain
- **Optimization:**
  - Produce a refined dataset focused on features with the highest predictive value
  - Reduce overfitting, lower computational cost, and enhance interpretability

IMPLEMENTATION: FEATURE MODULE

- **Modularity & Extensibility:**
  - Clear separation of extraction methods allows for easy extension
  - Users can add new feature computations without disrupting the existing pipeline
- **Pipeline Integration:**
  - Extracted features are formatted to integrate smoothly with subsequent modules

IMPLEMENTATION: FEATURE MODULE – EXTRACTOR

- **Pre-processing:**
  - Cleaning and normalization methods
- **Computation Routines:**
  - Functions for calculating statistical measures, technical indicators (leveraging the TA library), and custom metrics
- **Incremental Update:**
  - Implements incremental updates for traditional technical indicators to handle data updates efficiently
- **Structured Output:**
  - Organize computed features into a DataFrame for further use

IMPLEMENTATION (FUTURE FOCUS): FEATURE MODULE – SELECTION

- **Evaluation Methods:**
  - Compute statistical metrics and apply machine learning techniques for feature importance
- **Dimensionality Reduction Tools:**
  - Implement algorithms (e.g., PCA) to retain essential information (ongoing)
- **Selection Algorithms:**
  - Filter out less relevant features using dynamic or predefined thresholds
- **Output Optimization:**
  - Produces a dataset containing only the most informative features for improved modeling performance

MODEL DEVELOPMENT

Forcasting Model design and training

METHODOLOGY: MODEL MODULE

- **Unified Modeling Framework:**
  - Base Model provides common methods (train, predict, evaluate)
  - Specialized models to be developed for:
    - **Machine Learning:** Neural networks, decision trees, ensembles
    - **Physics:** Simulation-based, analytical models from physics inspiration
    - **Statistical:** Regression, time series, hypothesis testing
- **Model Trainer:**
  - Centralized module for model training, evaluation, and integration

IMPLEMENTATION (FUTURE FOCUS): MODEL MODULE

- **Current State:**
  - Machine learning/physics/statistical Modules are empty and serve as scaffolding
- **Planned Features:**
  - Define common interfaces and shared methods in base_model.py
  - Extend specialized models from the base model
  - Implement model trainer to manage training loops and evaluation metrics
  - Parameter management, scalability, and extensibility as key design principles

SIGNAL PROCESSING

Applications in time-series analysis, noise reduction

METHODOLOGY: SIGNAL PROCESSING MODULE

- **Design Principles:**
  - **Modularity:**
    - Separate modules for filtering, processing, and transformation
  - **Scalability & Flexibility:**
    - Designed to handle a wide range of signals and data formats
  - **Integration:**
    - Seamless pipeline from filtering through to feature extraction, and predictive models

METHODOLOGY: SIGNAL PROCESSING MODULE

Core Components:

- **Signal Processor (signal_processor.py):**
  - Coordinates the application of filters and transformations
  - Manages tasks like segmentation, normalization, and sequential processing
- **Filters (filters.py):**
  - Remove noise and unwanted frequency components
  - Implements various filtering techniques (low-pass, high-pass, band-pass)
- **Transformation (transform.py):**
  - Converts signals from the time domain to the frequency domain
  - Implements techniques such as Fourier and Wavelet Transforms for frequency analysis

IMPLEMENTATION: SIGNAL PROCESSING MODULE

- **Signal Processor Module (signal_processor.py):**
  - Supports processing of both batch and streaming data
- **Filters Module (filters.py):**
  - Designs filters with configurable parameters (e.g., cutoff frequencies, order)
  - Dynamic configurability allows easy tuning for different signal types
- **Transformation Module (transform.py):**
  - Provides functions for converting signals from time to frequency domain (e.g., FFT, Wavelet)
  - Enables feature extraction from transformed signals

RISK MANAGEMENT

To Mitigate risk and protect capital at both single asset and portfolio levels

METHODOLOGY: RISK MANAGEMENT MODULE

- **Single Asset Risk Management (single_risk.py):**
  - Sets stop-loss and take-profit levels based on volatility and risk/reward ratios.
  - Adjusts position sizes to limit risk per trade.
  - Supports incremental risk updates with evolving market data.
- **Central Risk Manager (risk_manager.py):**
  - Aggregates risk metrics across different strategies.
  - Provides a unified interface for monitoring and adjusting risk parameters.

METHODOLOGY: RISK MANAGEMENT MODULE

- **Portfolio Risk Management (portfolio_manager.py):**
  - Manages risk across multiple assets using diversification and rebalancing.
  - Monitors overall portfolio exposure and correlations.
- **Capital Allocation (capital_allocator.py):**
  - Allocates capital based on risk-adjusted returns.
  - Dynamically adjusts allocations in response to market conditions.
- **Design Considerations:**
  - Systematic and quantitative risk controls (e.g., using ATR for thresholds).
  - Customization via JSON configuration files for risk parameters.
  - Seamless integration with trading strategies and real-time data updates.

IMPLEMENTATION: RISK MANAGEMENT MODULE

- **Single Asset Risk Management:**
  - Implements functions for calculating stop-loss/take-profit levels.
  - Adjusts position sizes using predefined risk metrics.
  - Updates risk measures incrementally as new data arrives.
- **Central Risk Manager:**
  - Consolidates individual risk metrics.
  - Provides utilities for enforcing risk limits and making adjustments.
  - Incorporates logging and error-handling for robust risk monitoring.

IMPLEMENTATION (FUTURE FOCUS): RISK MANAGEMENT MODULE

- **Portfolio Risk Management:**
  - Evaluates asset correlations and implements diversification strategies.
  - Monitors overall portfolio exposure and triggers rebalancing when needed.
- **Capital Allocation:**
  - Uses risk-adjusted metrics to distribute capital among assets.
  - Dynamically modifies allocations based on real-time risk and market changes.
  - Integrates with central and portfolio risk managers for comprehensive control.

STRATEGY MODULE

Realization of trading strategy

METHODOLOGY: STRATEGY MODULE

- **Single Asset Strategy:**
  - Focus on one asset using technical analysis and tailored risk management
  - Use chosen model to forecast and make buy and sell signals.
- **Multi Asset Strategy:**
  - Aggregates signals from multiple assets for portfolio-level trading
  - Uses portfolio optimization, diversification, and dynamic rebalancing

METHODOLOGY: STRATEGY MODULE

Customization via JSON Files:

- Strategies can be customized using JSON configuration files (e.g., strategy.json)
- Example configuration includes parameters for:
  - **Model:** e.g., Base_RSIwADX_type0 with custom indicator thresholds and parameters
  - **Risk Manager:** Stop-loss, take-profit, and position sizing parameters
  - **Decision Maker:** Decision rules such as threshold levels
- Enables flexible, user-defined strategy adjustments for different assets

IMPLEMENTATION: STRATEGY MODULE

- **Single Asset Strategy (single_asset_strategy.py):**
  - Implements technical indicators and signal generation for one asset
  - Integrates risk management (stop-loss, take-profit) specific to single asset dynamics
  - Applies incremental updates to process evolving market data efficiently
- **Multi Asset Strategy (multi_asset_strategy.py):**
  - Aggregates signals from individual asset strategies
  - Performs portfolio-level analysis for optimal capital allocation
  - Dynamically rebalances the portfolio based on market conditions

IMPLEMENTATION: STRATEGY MODULE

- **Customization & Integration:**
  - Loads and applies strategy configurations from JSON files
  - Allows user-defined customization for models, risk management, and decision-making parameters
  - Seamlessly integrates with data handling and execution systems for live trading
- **Additional Considerations:**
  - Systematic signal generation through predefined rules and algorithms
  - Emphasis on modularity, enabling rapid iteration and testing of strategy rules

BACKTESTING MODULE

To Simulate historical trading performance to evaluate strategies/models

METHODOLOGY: BACKTESTING MODULE

- **Backtester (backtester.py):**
  - Simulates trade execution and virtual order management
  - Tracks performance metrics like cumulative returns and drawdowns
- **Model Evaluation (model_evaluation.py):**
  - Computes performance metrics (accuracy, ROI, Sharpe ratio)
  - Compares predictive power of different models
- **Strategy Evaluation (strategy_evaluation.py):**
  - Aggregates trade and model performance to assess overall strategy effectiveness
  - Supports optimization and parameter tuning

METHODOLOGY: BACKTESTING MODULE

- **Design Principles:**
  - Systematic simulation to mirror live market conditions
  - Modularity for independent development and testing of components
- **Customization via JSON Files:**
  - *strategy.json* for multi-asset strategies: Configures model methods, risk manager parameters, and decision rules (e.g., Base_MACD_type0)
  - *single_strategy.json* for single-asset strategies: Defines specific parameters like start date, interval, and indicator thresholds (e.g., Base_RSIwADX_type0)
- **Data Handling Customization:**
  - *data_h.json* sets data ingestion parameters such as file type, symbols, retry count, memory and log settings, and required data labels

IMPLEMENTATION: BACKTESTING MODULE

- **Workflow Integration:**
  - Use python notebook for quick analysis
  - Sequential processing: Data ingestion → Trade simulation → Model & Strategy evaluation
  - Customization via JSON files allows flexible configuration of models, risk management, and data parameters
- **Backtester:**
  - Loads and processes historical market data
  - Executes simulated trades based on generated signals
  - Records detailed performance statistics

IMPLEMENTATION: BACKTESTING MODULE

- **Model Evaluation:**
  - Calculates performance metrics to compare various predictive models
  - Provides statistical analysis for model validation
- **Strategy Evaluation:**
  - Aggregates trade-level and model-level results to gauge overall strategy effectiveness
  - Supports tuning of strategy parameters based on backtest outcomes

LIVE/MOCK TRADING MODULE

Execution of orders, tracking of accounts in real time

METHODOLOGY: REAL/MOCK TRADING MODULE

- **Key Concepts:**
  - **Unified Trading Engine:**
    - Shared core for order generation, risk controls, and connectivity
  - **Real-Time Trading:**
    - Executes live trades via Binance API
  - **Mock Trading:**
    - Simulates orders, account balance, and order status
    - Mimics Binance behavior for realistic testing
- **Customization:**
  - Configurable via JSON files (e.g., API keys, account settings, order parameters)

IMPLEMENTATION: REAL/MOCK TRADING MODULE

- **Real-Time Dealer:**
  - Connects directly to Binance for live market data and order execution
  - Implements order submission, modification, and cancellation
- **Mock Trading Modules:**
  - **Mock Real-Time Dealer:**
    - Simulates live trading by generating mock order responses
  - **Mock Order Manager:**
    - Manages simulated orders and virtual account details

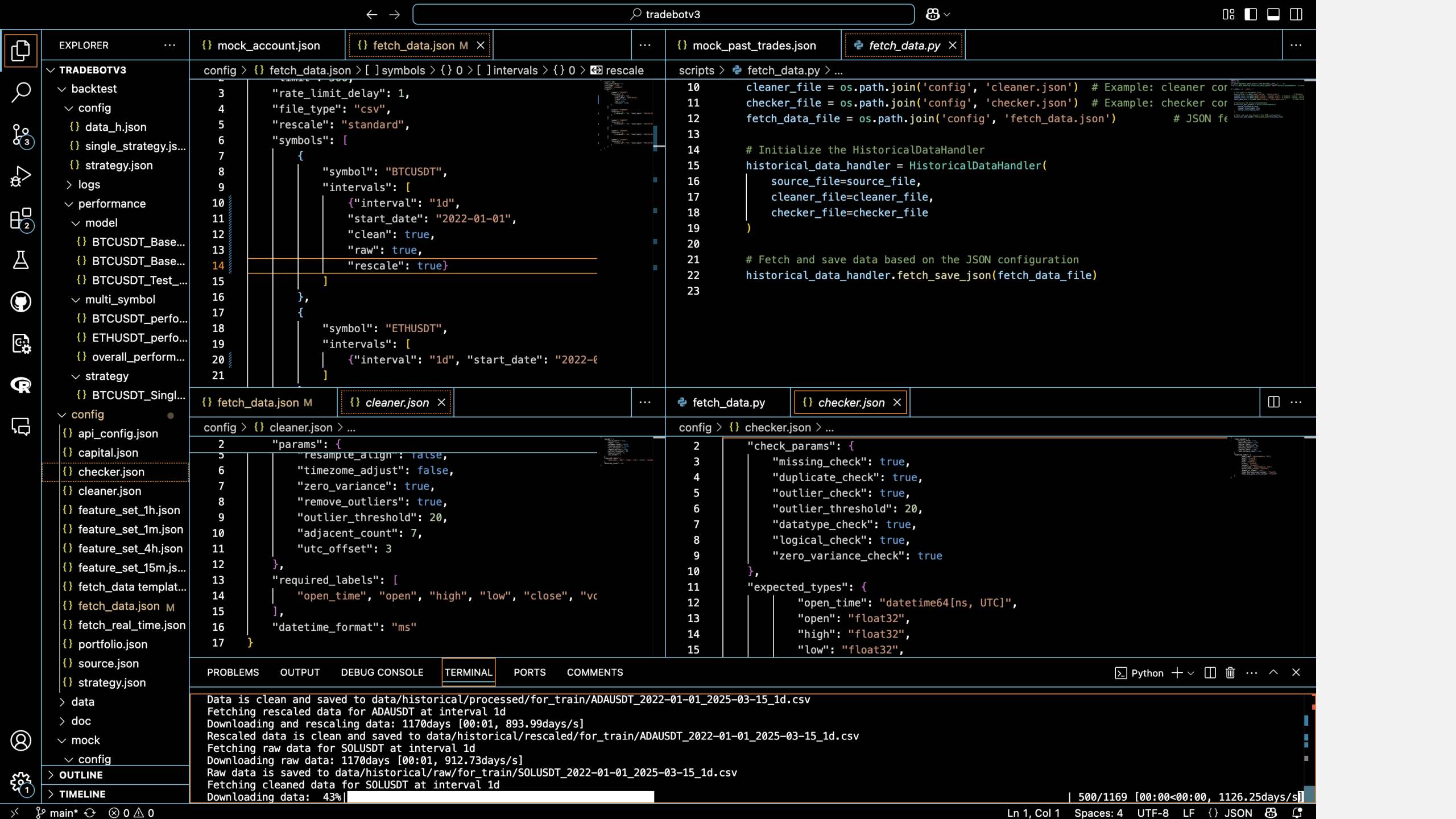
IMPLEMENTATION: REAL/MOCK TRADING MODULE

- **Common Features:**
  - Integrated risk management (stop-loss, take-profit, position sizing)
  - Logging, error-handling, and consistent API compatibility with Binance
- **Configuration & Flexibility:**
  - JSON-based settings allow easy switching and parameter tuning between real and mock environments

RESULTS & PERFORMANCE

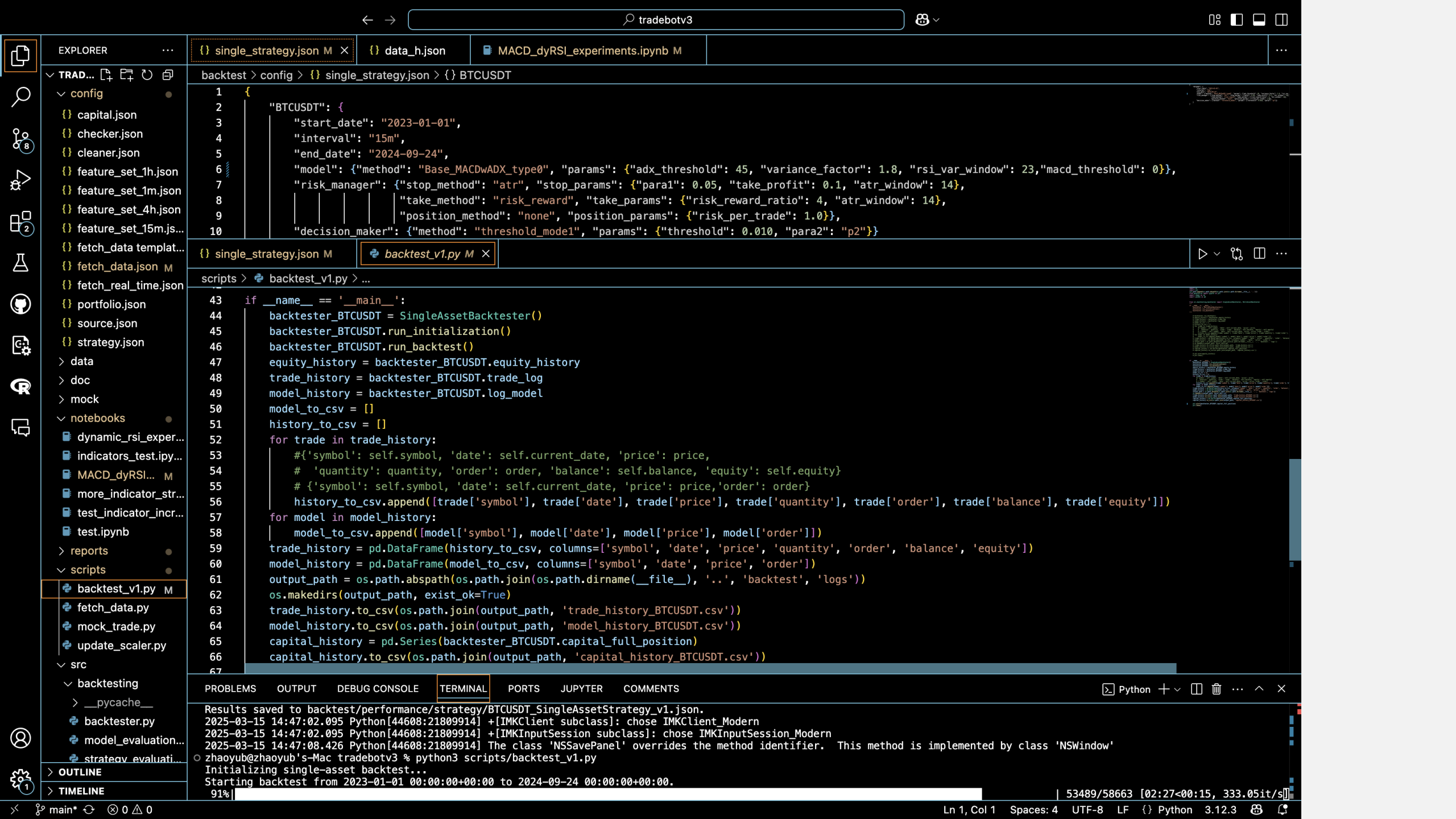
Code Snippets and performance demonstration

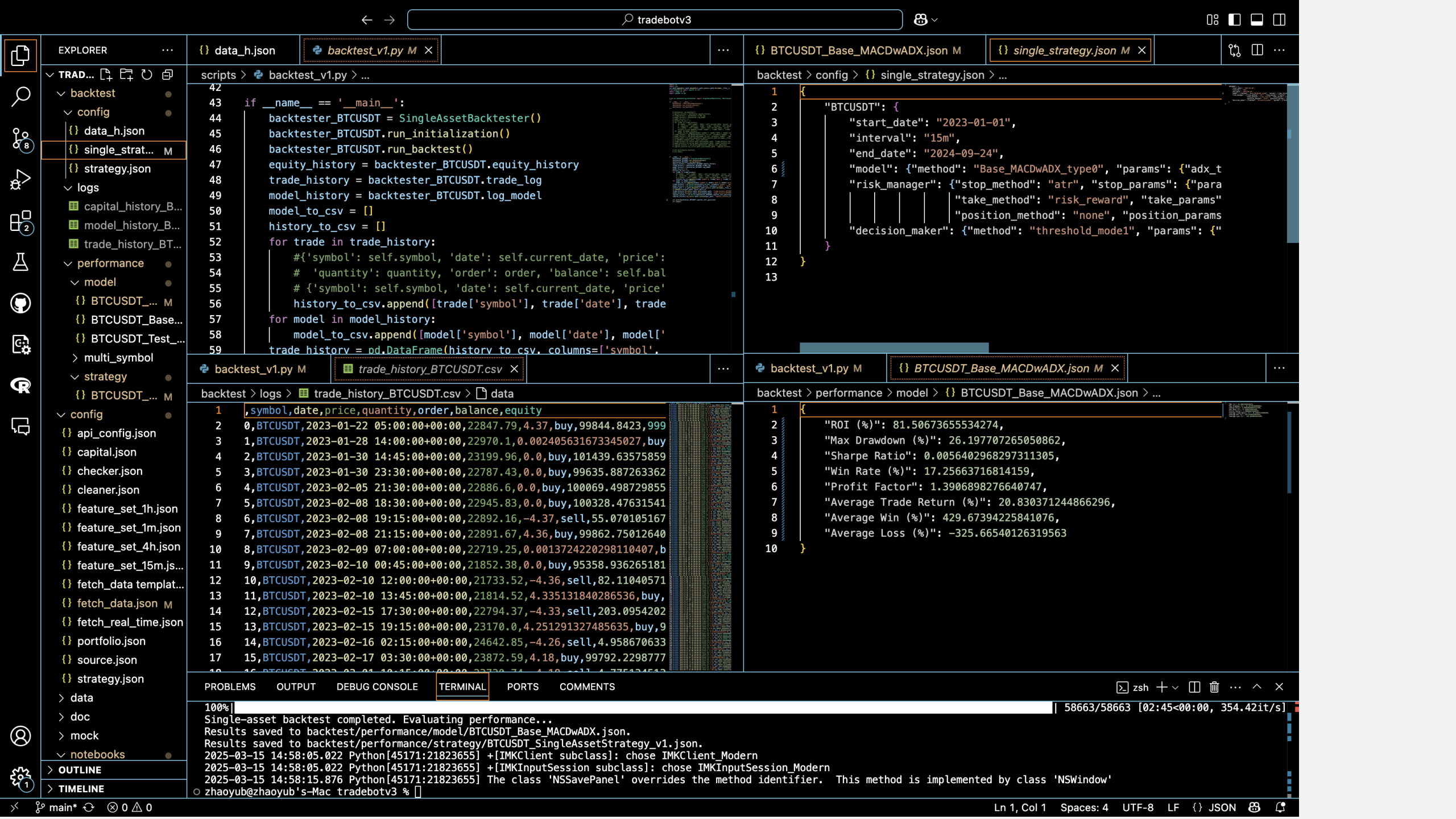
DATA RETRIVAL & CLEANING

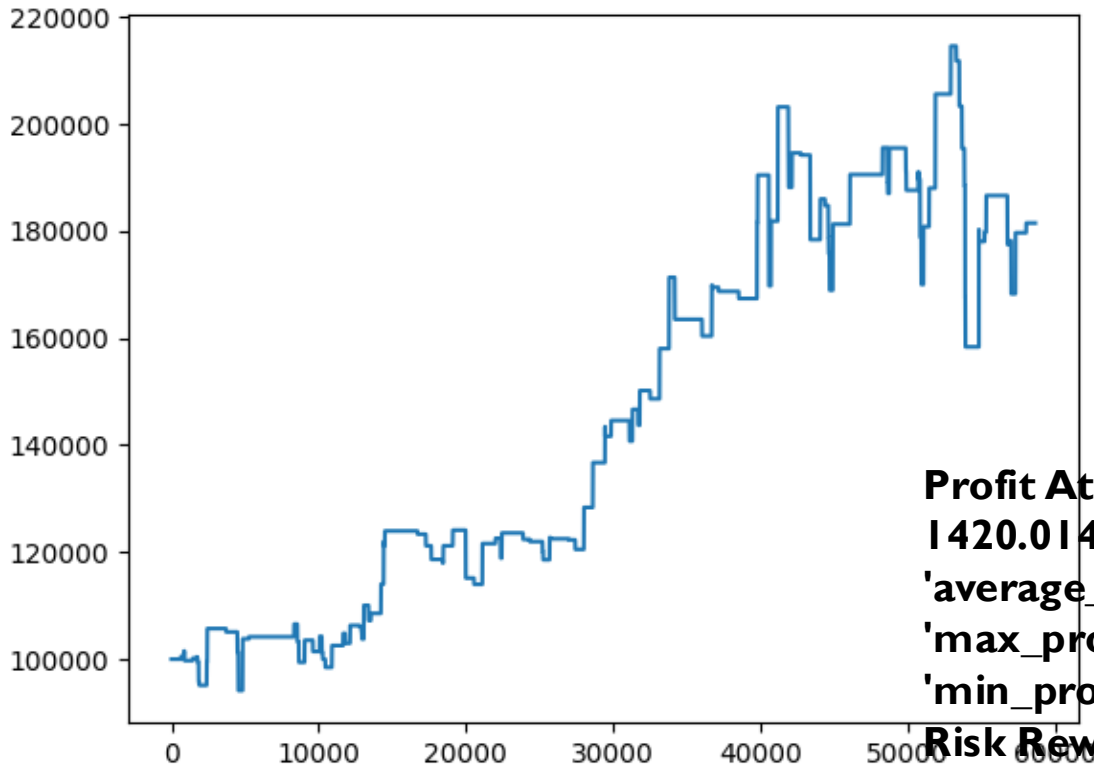

tradebotv3
EXPLORER
TRADEBOTV3
backtest
config
data_h.json
single_strategy.js...
strategy.json
logs
performance
model
BTCUSDT_Base...
BTCUSDT_Base...
BTCUSDT_Test_...
multi_symbol
BTCUSDT_perfo...
ETHUSDT_perfo...
overall_perform...
strategy
BTCUSDT_Singl...
config
api_config.json
capital.json
checker.json
cleaner.json
feature_set_1h.json
feature_set_1m.json
feature_set_4h.json
feature_set_15m.js...
fetch_data templat...
fetch_data.json M
fetch_real_time.json
portfolio.json
source.json
strategy.json
data
doc
mock
config
OUTLINE
TIMELINE
mock_account.json
fetch_data.json M
config > {} fetch_data.json > [ ] symbols > {} 0 > [ ] intervals > {} 0 > rescale
"rate_limit_delay": 1,
"file_type": "csv",
"rescale": "standard",
"symbols": [
"symbol": "BTCUSDT",
"intervals": [
{"interval": "1d",
"start_date": "2022-01-01",
"clean": true,
"raw": true,
"rescale": true}
"symbol": "ETHUSDT",
"intervals": [
{"interval": "1d", "start_date": "2022-
mock_past_trades.json
fetch_data.py
scripts > fetch_data.py > ...
cleaner_file = os.path.join('config', 'cleaner.json') # Example: cleaner cor
checker_file = os.path.join('config', 'checker.json') # Example: checker cor
fetch_data_file = os.path.join('config', 'fetch_data.json') # JSON fe
# Initialize the HistoricalDataHandler
historical_data_handler = HistoricalDataHandler(
source_file=source_file,
cleaner_file=cleaner_file,
checker_file=checker_file
)
# Fetch and save data based on the JSON configuration
historical_data_handler.fetch_save_json(fetch_data_file)
fetch_data.json M
cleaner.json
config > {} cleaner.json > ...
"params": {
"timezome_adjust": false,
"zero_variance": true,
"remove_outliers": true,
"outlier_threshold": 20,
"adjacent_count": 7,
"utc_offset": 3
"required_labels": [
"open_time", "open", "high", "low", "close", "vc
"datetime_format": "ms"
fetch_data.py
checker.json
config > {} checker.json > ...
"check_params": {
"missing_check": true,
"duplicate_check": true,
"outlier_check": true,
"outlier_threshold": 20,
"datatype_check": true,
"logical_check": true,
"zero_variance_check": true
"expected_types": {
"open_time": "datetime64[ns, UTC]",
"open": "float32",
"high": "float32",
"low": "float32",
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
Python
Data is clean and saved to data/historical/processed/for_train/ADAUSDT_2022-01-01_2025-03-15_1d.csv
Fetching rescaled data for ADAUSDT at interval 1d
Downloading and rescaling data: 1170days [00:01, 893.99days/s]
Rescaled data is clean and saved to data/historical/rescaled/for_train/ADAUSDT_2022-01-01_2025-03-15_1d.csv
Fetching raw data for SOLUSDT at interval 1d
Downloading raw data: 1170days [00:01, 912.73days/s]
Raw data is saved to data/historical/raw/for_train/SOLUSDT_2022-01-01_2025-03-15_1d.csv
Fetching cleaned data for SOLUSDT at interval 1d
Downloading data: 43%| | 500/1169 [00:00<00:00, 1126.25days/s]
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JSON


tradebotv3
EXPLORER
_account.json
{} fetch_data.json M
BTCUSDT_2022-01-01_2025-03-04_4h.csv
SOLUSDT_2022-01-01_2025-03-15_1d.csv
data > historical > processed > for_train > BTCUSDT_2022-01-01_2025-03-04_4h.csv > data
Col 5: close
1 open_time,open,high,low,close,volume
2 2022-01-01 00:00:00+00:00,46216.93,46949.99,46208.37,46813.2,3495.2
3 2022-01-01 04:00:00+00:00,46813.21,47555.55,46591.23,47194.73,3522.
4 2022-01-01 08:00:00+00:00,47194.73,47344.69,46715.39,46758.87,2729.
5 2022-01-01 12:00:00+00:00,46758.87,47491.14,46756.3,47219.04,3055.9
6 2022-01-01 16:00:00+00:00,47219.04,47954.63,47186.07,47329.78,4061.
7 2022-01-01 20:00:00+00:00,47329.79,47814.0,47222.22,47722.65,2739.1
8 2022-01-02 00:00:00+00:00,47722.66,47750.0,47269.47,47329.47,2387.9
9 2022-01-02 04:00:00+00:00,47329.47,47415.73,46770.23,47173.45,3018.
10 2022-01-02 08:00:00+00:00,47173.44,47499.45,46970.05,47235.5,2119.8
11 2022-01-02 12:00:00+00:00,47238.89,47447.0,47013.5,47340.47,2780.32
12 2022-01-02 16:00:00+00:00,47340.47,47990.0,46654.0,46907.68,5597.88
13 2022-01-02 20:00:00+00:00,46907.67,47469.34,46815.17,47286.18,2436.
14 2022-01-03 00:00:00+00:00,47286.18,47289.11,46722.39,46793.26,2994.
15 2022-01-03 04:00:00+00:00,46793.27,47211.09,46750.0,47051.37,2838.9
16 2022-01-03 08:00:00+00:00,47051.36,47431.67,46778.0,47378.0,4262.39
17 2022-01-03 12:00:00+00:00,47378.0,47570.0,46575.25,46693.45,5463.72
18 2022-01-03 16:00:00+00:00,46693.44,46790.0,46263.29,46360.01,5861.4
19 2022-01-03 20:00:00+00:00,46360.01,46576.94,45696.0,46446.1,6241.20
20 2022-01-04 00:00:00+00:00,46446.1,46549.19,45934.5,46313.97,3604.76
21 2022-01-04 04:00:00+00:00,46313.96,46600.21,45996.26,46550.0,5734.3
22 2022-01-04 08:00:00+00:00,46549.99,46772.41,46300.0,46733.77,4686.5
data > historical > processed > for_train > SOLUSDT_2022-01-01_2025-03-15_1d.csv > data
1 open_time,open,high,low,close,volume
2 2022-01-01 00:00:00+00:00,170.01,179.2,169.93,179.06,634599.4
3 2022-01-02 00:00:00+00:00,179.06,179.53,174.31,176.35,620087.4
4 2022-01-03 00:00:00+00:00,176.34,176.49,166.59,170.15,1031316.44
5 2022-01-04 00:00:00+00:00,170.14,173.93,166.47,167.72,1041332.6
6 2022-01-05 00:00:00+00:00,167.74,171.32,144.92,154.52,1659848.1
7 2022-01-06 00:00:00+00:00,154.48,154.8,145.98,150.26,1575071.2
8 2022-01-07 00:00:00+00:00,150.28,150.72,133.08,136.45,2719397.2
9 2022-01-08 00:00:00+00:00,136.43,148.11,133.0,142.14,2695415.5
10 2022-01-09 00:00:00+00:00,142.1,145.71,137.83,140.82,1435153.1
11 2022-01-10 00:00:00+00:00,140.82,143.86,130.0,135.86,1894512.5
12 2022-01-11 00:00:00+00:00,135.84,143.5,133.58,140.2,1328579.6
13 2022-01-12 00:00:00+00:00,140.19,153.39,138.22,151.51,1946788.8
14 2022-01-13 00:00:00+00:00,151.5,157.8,145.41,146.14,1984121.6
15 2022-01-14 00:00:00+00:00,146.13,150.56,141.84,146.17,1428145.2
16 2022-01-15 00:00:00+00:00,146.18,149.49,143.9,147.64,865014.75
17 2022-01-16 00:00:00+00:00,147.61,151.96,146.09,147.98,964270.1
18 2022-01-17 00:00:00+00:00,147.98,148.29,137.02,139.8,1297654.1
19 2022-01-18 00:00:00+00:00,139.81,142.77,134.39,141.22,1254337.1
20 2022-01-19 00:00:00+00:00,141.32,142.7,132.77,135.25,1175171.4
21 2022-01-20 00:00:00+00:00,135.23,143.83,126.01,127.54,1464594.0
22 2022-01-21 00:00:00+00:00,127.54,128.73,106.11,112.27,4523020.5
{} fetch_data.json M
ETHUSDT_2022-01-01_2025-03-15_1d.csv
data > historical > raw > for_train > ETHUSDT_2022-01-01_2025-03-15_1d.csv > data
1 open_time,open,high,low,close,volume,close_time,quote_asset_volume,number_of_trades,taker_buy_base_asset_volume,taker_buy_quote_asset_volume,ignore
2 2022-01-01 00:00:00+00:00,3676.22000000,3776.45000000,3673.46000000,3765.54000000,154143.89680000,2022-01-01 23:59:59.999000+00:00,574718328.45274400,
3 2022-01-02 00:00:00+00:00,3765.54000000,3857.44000000,3717.30000000,3828.27000000,154791.42630000,2022-01-02 23:59:59.999000+00:00,585374957.73046000,
4 2022-01-03 00:00:00+00:00,3828.11000000,3853.09000000,3680.00000000,3765.89000000,236245.85860000,2022-01-03 23:59:59.999000+00:00,892944105.55299400,
5 2022-01-04 00:00:00+00:00,3765.89000000,3900.73000000,3713.11000000,3785.11000000,288258.55490000,2022-01-04 23:59:59.999000+00:00,1098182607.92572600,
6 2022-01-05 00:00:00+00:00,3785.10000000,3848.00000000,3415.00000000,3540.63000000,397942.03630000,2022-01-05 23:59:59.999000+00:00,1464819255.09351100,
7 2022-01-06 00:00:00+00:00,3539.82000000,3550.43000000,3300.00000000,3406.81000000,496745.74130000,2022-01-06 23:59:59.999000+00:00,1698727637.36502600,
8 2022-01-07 00:00:00+00:00,3406.81000000,3416.76000000,3064.42000000,3199.24000000,669631.72620000,2022-01-07 23:59:59.999000+00:00,2149562773.32925700,
9 2022-01-08 00:00:00+00:00,3198.68000000,3250.00000000,3000.00000000,3080.95000000,400692.10610000,2022-01-08 23:59:59.999000+00:00,1252795163.48626900,
10 2022-01-09 00:00:00+00:00,3080.98000000,3211.72000000,3057.35000000,3151.97000000,275874.16430000,2022-01-09 23:59:59.999000+00:00,864505480.88173800,
11 2022-01-10 00:00:00+00:00,3151.97000000,3185.00000000,2928.83000000,3082.68000000,529818.38660000,2022-01-10 23:59:59.999000+00:00,1616550523.09720100,
12 2022-01-11 00:00:00+00:00,3082.87000000,3263.54000000,3051.80000000,3239.72000000,362446.70870000,2022-01-11 23:59:59.999000+00:00,1145852583.62298000,
13 2022-01-12 00:00:00+00:00,3239.73000000,3411.43000000,3206.20000000,3371.26000000,370465.80150000,2022-01-12 23:59:59.999000+00:00,1226038697.47290200,
14 2022-01-13 00:00:00+00:00,3371.60000000,3410.00000000,3233.43000000,3240.61000000,336234.87990000,2022-01-13 23:59:59.999000+00:00,1116293607.68792400,
15 2022-01-14 00:00:00+00:00,3240.61000000,3342.79000000,3188.00000000,3307.42000000,285326.35050000,2022-01-14 23:59:59.999000+00:00,931347172.76288100,
EXPLORER
TRAD...
data
historical
processed
for_train
SOLUSDT_202...
XTZUSDT_202...
from_real
raw
for_train
ADAUSDT_202...
BTCUSDT_202...
ETHUSDT_202...
SOLUSDT_202...
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
zsh
Fetching rescaled data for XTZUSDT at interval 1d
Downloading and rescaling data: 1170days [00:01, 901.50days/s]
Rescaled data is clean and saved to data/historical/rescaled/for_train/XTZUSDT_2022-01-01_2025-03-15_1d.csv
zhaoyub@zhaoyub's-Mac tradebotv3 %
main*
CSVLint
Query
Align
Col 1: open_time
Ln 1, Col 1
Spaces: 4
UTF-8
LF
CSV

BACKTESTING

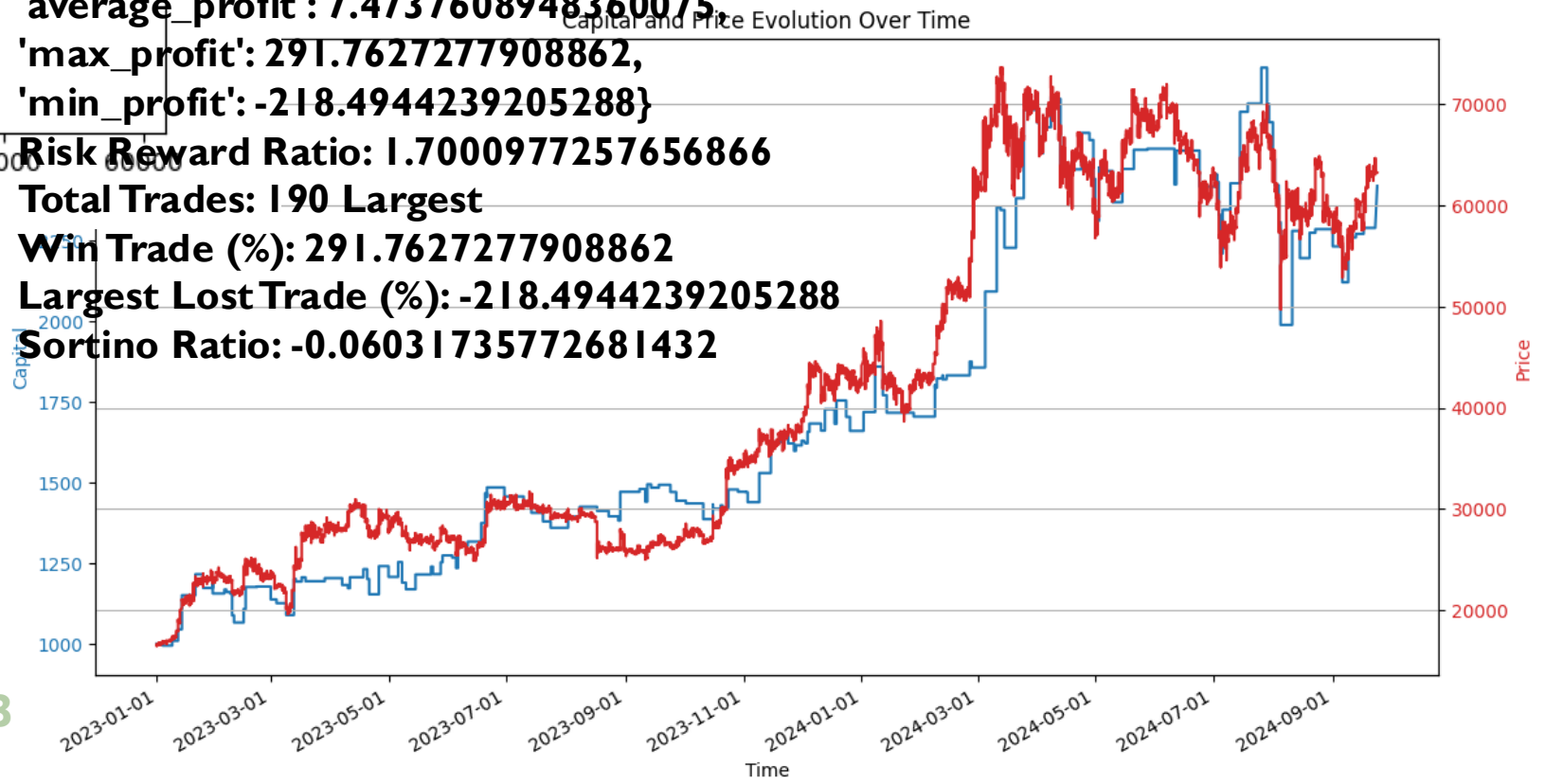

tradebotv3
EXPLORER
single_strategy.json M
data_h.json
MACD_dyRSI_experiments.ipynb M
backtest > config > single_strategy.json > BTCUSDT
{
"BTCUSDT": {
"start_date": "2023-01-01",
"interval": "15m",
"end_date": "2024-09-24",
"model": {"method": "Base_MACDwADX_type0", "params": {"adx_threshold": 45, "variance_factor": 1.8, "rsi_var_window": 23,"macd_threshold": 0}},
"risk_manager": {"stop_method": "atr", "stop_params": {"para1": 0.05, "take_profit": 0.1, "atr_window": 14},
"take_method": "risk_reward", "take_params": {"risk_reward_ratio": 4, "atr_window": 14},
"position_method": "none", "position_params": {"risk_per_trade": 1.0}},
"decision_maker": {"method": "threshold_mode1", "params": {"threshold": 0.010, "para2": "p2"}}
single_strategy.json M
backtest_v1.py M
scripts > backtest_v1.py > ...
if __name__ == '__main__':
backtester_BTCUSDT = SingleAssetBacktester()
backtester_BTCUSDT.run_initialization()
backtester_BTCUSDT.run_backtest()
equity_history = backtester_BTCUSDT.equity_history
trade_history = backtester_BTCUSDT.trade_log
model_history = backtester_BTCUSDT.log_model
model_to_csv = []
history_to_csv = []
for trade in trade_history:
#{'symbol': self.symbol, 'date': self.current_date, 'price': price,
# 'quantity': quantity, 'order': order, 'balance': self.balance, 'equity': self.equity}
# {'symbol': self.symbol, 'date': self.current_date, 'price': price,'order': order}
history_to_csv.append([trade['symbol'], trade['date'], trade['price'], trade['quantity'], trade['order'], trade['balance'], trade['equity']])
for model in model_history:
model_to_csv.append([model['symbol'], model['date'], model['price'], model['order']])
trade_history = pd.DataFrame(history_to_csv, columns=['symbol', 'date', 'price', 'quantity', 'order', 'balance', 'equity'])
model_history = pd.DataFrame(model_to_csv, columns=['symbol', 'date', 'price', 'order'])
output_path = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'backtest', 'logs'))
os.makedirs(output_path, exist_ok=True)
trade_history.to_csv(os.path.join(output_path, 'trade_history_BTCUSDT.csv'))
model_history.to_csv(os.path.join(output_path, 'model_history_BTCUSDT.csv'))
capital_history = pd.Series(backtester_BTCUSDT.capital_full_position)
capital_history.to_csv(os.path.join(output_path, 'capital_history_BTCUSDT.csv'))
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
COMMENTS
Python
Results saved to backtest/performance/strategy/BTCUSDT_SingleAssetStrategy_v1.json.
2025-03-15 14:47:02.095 Python[44608:21809914] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-15 14:47:02.095 Python[44608:21809914] +[IMKInputSession subclass]: chose IMKInputSession_Modern
2025-03-15 14:47:08.426 Python[44608:21809914] The class 'NSSavePanel' overrides the method identifier. This method is implemented by class 'NSWindow'
zhaoyub@zhaoyub's-Mac tradebotv3 % python3 scripts/backtest_v1.py
Initializing single-asset backtest...
Starting backtest from 2023-01-01 00:00:00+00:00 to 2024-09-24 00:00:00+00:00.
91%| | 53489/58663 [02:27<00:15, 333.05it/s]
main*
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.12.3


tradebotv3
EXPLORER
data_h.json
backtest_v1.py M
BTCUSDT_Base_MACDwADX.json M
single_strategy.json M
scripts > backtest_v1.py > ...
backtest > config > single_strategy.json > ...
backtest_v1.py M
trade_history_BTCUSDT.csv
backtest > logs > trade_history_BTCUSDT.csv > data
backtest_v1.py M
BTCUSDT_Base_MACDwADX.json M
backtest > performance > model > BTCUSDT_Base_MACDwADX.json > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
zsh
100%| | 58663/58663 [02:45<00:00, 354.42it/s]
Single-asset backtest completed. Evaluating performance...
Results saved to backtest/performance/model/BTCUSDT_Base_MACDwADX.json.
Results saved to backtest/performance/strategy/BTCUSDT_SingleAssetStrategy_v1.json.
2025-03-15 14:58:05.022 Python[45171:21823655] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-15 14:58:05.022 Python[45171:21823655] +[IMKInputSession subclass]: chose IMKInputSession_Modern
2025-03-15 14:58:15.876 Python[45171:21823655] The class 'NSSavePanel' overrides the method identifier. This method is implemented by class 'NSWindow'
zhaoyub@zhaoyub's-Mac tradebotv3 %
main*
Ln 1, Col 1
Spaces: 4
UTF-8
LF
JSON



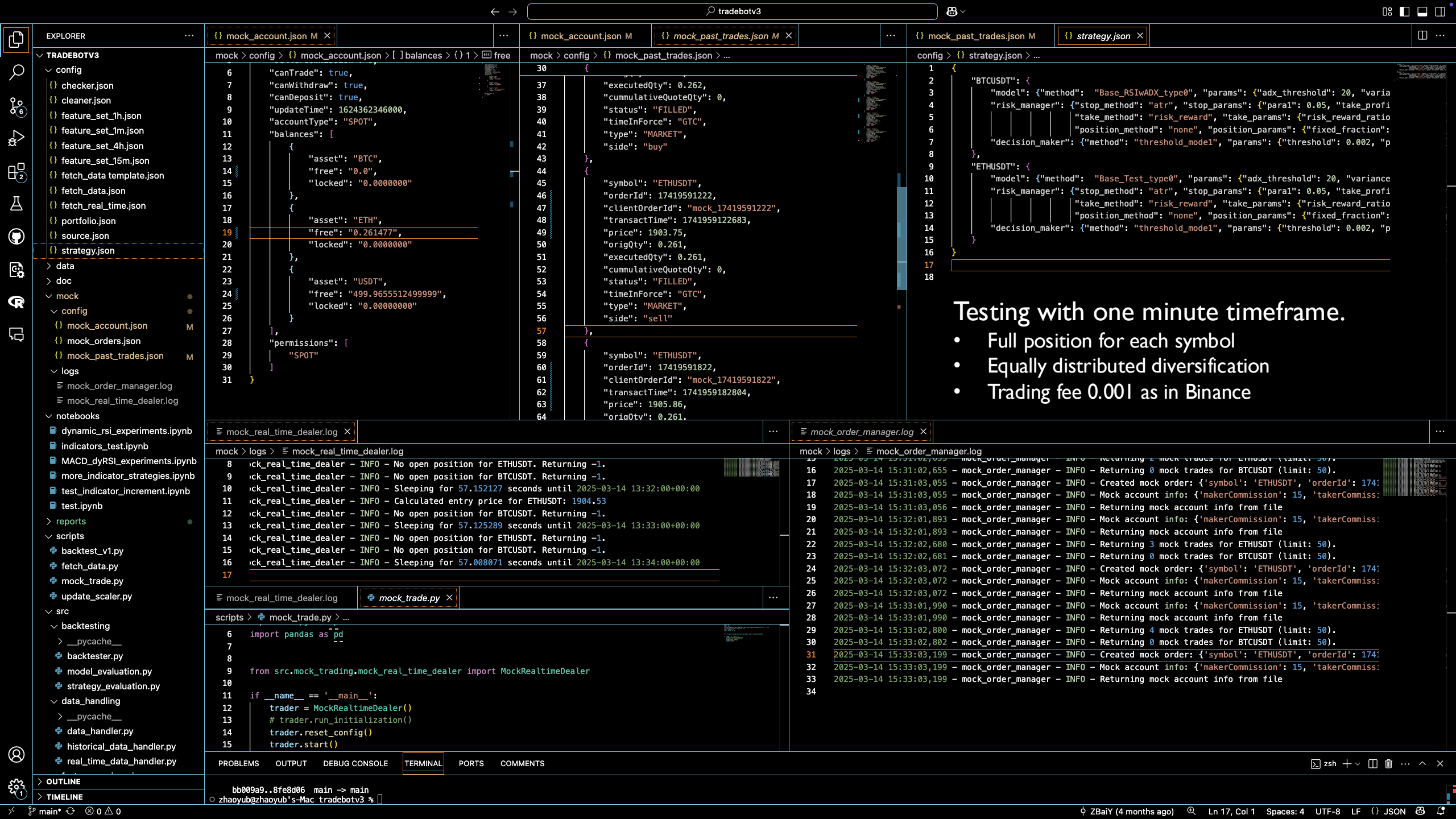
Performance Metrics:
Total ROI (%): 142.00145700188415
Max Drawdown (%): 28.636068837554156
Sharpe Ratio: 0.008170404219610626
Symbol ROI (%): 282.882555147515
Trade Efficiency (%): 15.8344654790018 58
Trade Distribution: {
'winning_trades': 85,
'losing_trades': 105,
'win_rate (%)': 44.73684210526316}

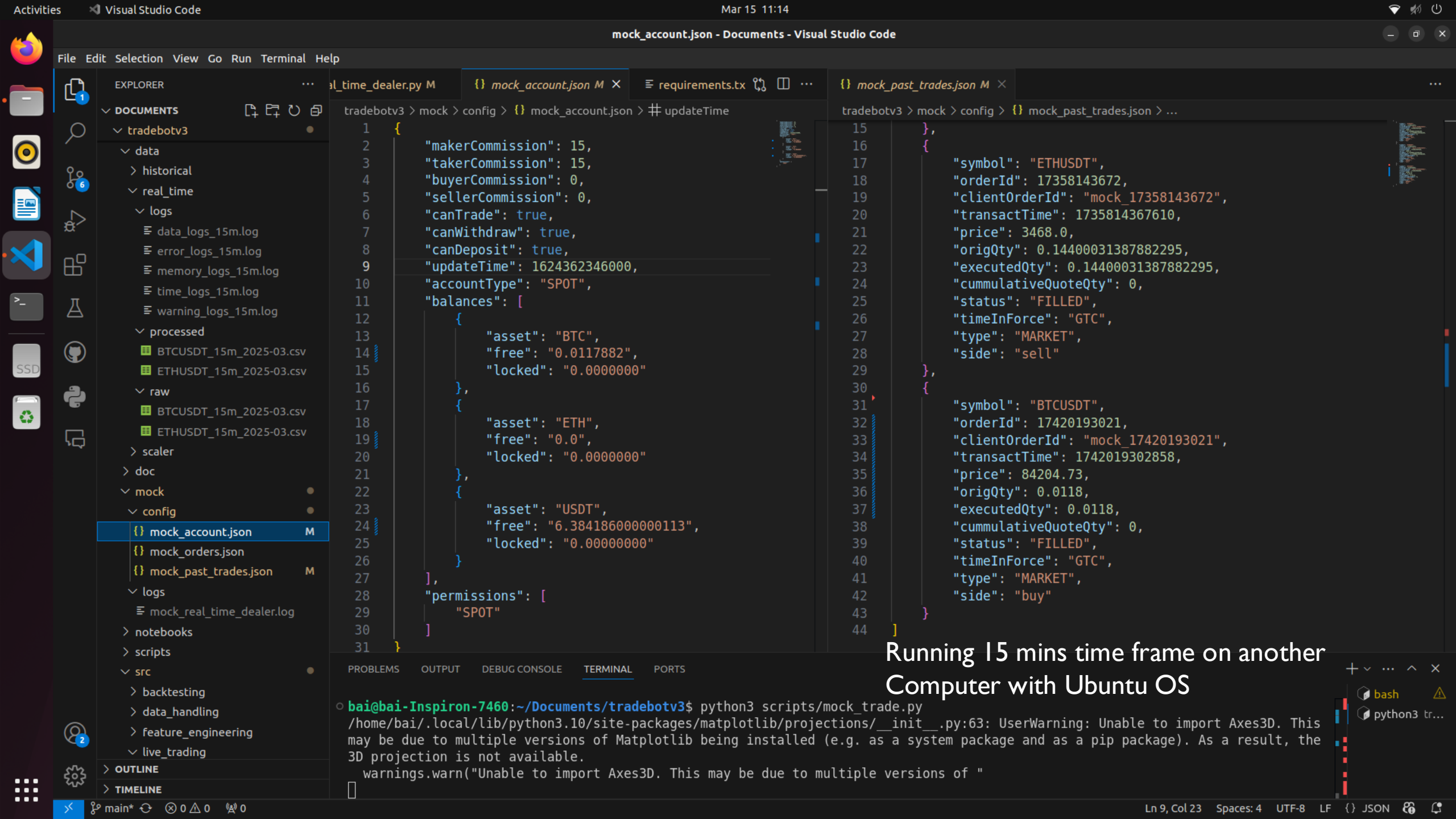
Profit Attribution: {'total_profit': 1420.0145700188414,
'average_profit': 7.473760894836007 5,
'max_profit': 291.7627277908862,
'min_profit': -218.4944239205288}
Risk Reward Ratio: 1.7000977257656866
Total Trades: 190 Largest
Win Trade (%): 291.7627277908862
Largest Lost Trade (%): -218.4944239205288
Sortino Ratio: -0.060317357726814 32



"ROI (%)": 81.50673655534274,
"Max Drawdown (%)": 26.197707265050862,
"Sharpe Ratio": 0.005640296829731 1305,
"Win Rate (%)": 17.256637168141 59,
"Profit Factor": 1.3906898276640747,
"Average Trade Return (%)": 20.830371244866296,
"Average Win (%)": 429.67394225841076,
"Average Loss (%)": -325.665401263 19563

MOCK TRADING


tradebotv3
EXPLORER
TRADEBOTV3
config
checker.json
cleaner.json
feature_set_1h.json
feature_set_1m.json
feature_set_4h.json
feature_set_15m.json
fetch_data template.json
fetch_data.json
fetch_real_time.json
portfolio.json
source.json
strategy.json
data
doc
mock
config
mock_account.json M
mock_orders.json
mock_past_trades.json M
logs
mock_order_manager.log
mock_real_time_dealer.log
notebooks
dynamic_rsi_experiments.ipynb
indicators_test.ipynb
MACD_dyRSI_experiments.ipynb
more_indicator_strategies.ipynb
test_indicator_increment.ipynb
test.ipynb
reports
scripts
backtest_v1.py
fetch_data.py
mock_trade.py
update_scaler.py
src
backtesting
__pycache__
backtester.py
model_evaluation.py
strategy_evaluation.py
data_handling
__pycache__
data_handler.py
historical_data_handler.py
real_time_data_handler.py
OUTLINE
TIMELINE
mock_account.json M
mock > config > mock_account.json > [ ]balances > {}1 > free
"canTrade": true,
"canWithdraw": true,
"canDeposit": true,
"updateTime": 1624362346000,
"accountType": "SPOT",
"balances": [
{
"asset": "BTC",
"free": "0.0",
"locked": "0.0000000"
},
{
"asset": "ETH",
"free": "0.261477",
"locked": "0.0000000"
},
{
"asset": "USDT",
"free": "499.96555124999999",
"locked": "0.00000000"
}
],
"permissions": [
"SPOT"
]
}
mock_account.json M
mock_past_trades.json M
mock > config > mock_past_trades.json > ...
{
"executedQty": 0.262,
"cummulativeQuoteQty": 0,
"status": "FILLED",
"timeInForce": "GTC",
"type": "MARKET",
"side": "buy"
},
{
"symbol": "ETHUSDT",
"orderId": 17419591222,
"clientOrderId": "mock_17419591222",
"transactTime": 1741959122683,
"price": 1903.75,
"origQty": 0.261,
"executedQty": 0.261,
"cummulativeQuoteQty": 0,
"status": "FILLED",
"timeInForce": "GTC",
"type": "MARKET",
"side": "sell"
},
{
"symbol": "ETHUSDT",
"orderId": 17419591822,
"clientOrderId": "mock_17419591822",
"transactTime": 1741959182804,
"price": 1905.86,
"origQty": 0.261,
mock_past_trades.json M
strategy.json
config > strategy.json > ...
{
"BTCUSDT": {
"model": {"method": "Base_RSIwADX_type0", "params": {"adx_threshold": 20, "varia
"risk_manager": {"stop_method": "atr", "stop_params": {"para1": 0.05, "take_profi
"take_method": "risk_reward", "take_params": {"risk_reward_ratio
"position_method": "none", "position_params": {"fixed_fraction":
"decision_maker": {"method": "threshold_mode1", "params": {"threshold": 0.002, "p
},
"ETHUSDT": {
"model": {"method": "Base_Test_type0", "params": {"adx_threshold": 20, "variance
"risk_manager": {"stop_method": "atr", "stop_params": {"para1": 0.05, "take_profi
"take_method": "risk_reward", "take_params": {"risk_reward_ratio
"position_method": "none", "position_params": {"fixed_fraction":
"decision_maker": {"method": "threshold_mode1", "params": {"threshold": 0.002, "p
}
}
Testing with one minute timeframe.
• Full position for each symbol
• Equally distributed diversification
• Trading fee 0.001 as in Binance
mock_real_time_dealer.log
mock > logs > mock_real_time_dealer.log
ock_real_time_dealer - INFO - No open position for ETHUSDT. Returning -1.
ock_real_time_dealer - INFO - No open position for BTCUSDT. Returning -1.
ock_real_time_dealer - INFO - Sleeping for 57.152127 seconds until 2025-03-14 13:32:00+00:00
ock_real_time_dealer - INFO - Calculated entry price for ETHUSDT: 1904.53
ock_real_time_dealer - INFO - No open position for BTCUSDT. Returning -1.
ock_real_time_dealer - INFO - Sleeping for 57.125289 seconds until 2025-03-14 13:33:00+00:00
ock_real_time_dealer - INFO - No open position for ETHUSDT. Returning -1.
ock_real_time_dealer - INFO - No open position for BTCUSDT. Returning -1.
ock_real_time_dealer - INFO - Sleeping for 57.008071 seconds until 2025-03-14 13:34:00+00:00
mock_order_manager.log
mock > logs > mock_order_manager.log
2025-03-14 15:31:02,655 - mock_order_manager - INFO - Returning 0 mock trades for BTCUSDT (limit: 50).
2025-03-14 15:31:03,055 - mock_order_manager - INFO - Created mock order: {'symbol': 'ETHUSDT', 'orderId': 1741
2025-03-14 15:31:03,055 - mock_order_manager - INFO - Mock account info: {'makerCommission': 15, 'takerCommiss
2025-03-14 15:31:03,056 - mock_order_manager - INFO - Returning mock account info from file
2025-03-14 15:32:01,893 - mock_order_manager - INFO - Mock account info: {'makerCommission': 15, 'takerCommiss
2025-03-14 15:32:01,893 - mock_order_manager - INFO - Returning mock account info from file
2025-03-14 15:32:02,680 - mock_order_manager - INFO - Returning 3 mock trades for ETHUSDT (limit: 50).
2025-03-14 15:32:02,681 - mock_order_manager - INFO - Returning 0 mock trades for BTCUSDT (limit: 50).
2025-03-14 15:32:03,072 - mock_order_manager - INFO - Created mock order: {'symbol': 'ETHUSDT', 'orderId': 1741
2025-03-14 15:32:03,072 - mock_order_manager - INFO - Mock account info: {'makerCommission': 15, 'takerCommiss
2025-03-14 15:32:03,072 - mock_order_manager - INFO - Returning mock account info from file
2025-03-14 15:33:01,990 - mock_order_manager - INFO - Mock account info: {'makerCommission': 15, 'takerCommiss
2025-03-14 15:33:01,990 - mock_order_manager - INFO - Returning mock account info from file
2025-03-14 15:33:02,800 - mock_order_manager - INFO - Returning 4 mock trades for ETHUSDT (limit: 50).
2025-03-14 15:33:02,802 - mock_order_manager - INFO - Returning 0 mock trades for BTCUSDT (limit: 50).
2025-03-14 15:33:03,199 - mock_order_manager - INFO - Created mock order: {'symbol': 'ETHUSDT', 'orderId': 1741
2025-03-14 15:33:03,199 - mock_order_manager - INFO - Mock account info: {'makerCommission': 15, 'takerCommiss
2025-03-14 15:33:03,199 - mock_order_manager - INFO - Returning mock account info from file
mock_real_time_dealer.log
mock_trade.py
scripts > mock_trade.py > ...
import pandas as pd
from src.mock_trading.mock_real_time_dealer import MockRealtimeDealer
if __name__ == '__main__':
trader = MockRealtimeDealer()
# trader.run_initialization()
trader.reset_config()
trader.start()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
zsh
bb009a9..8fe8d06 main -> main
zhaoyub@zhaoyub's-Mac tradebotv3 %
main*
0 0
ZBaiY (4 months ago)
Ln 17, Col 1
Spaces: 4
UTF-8
LF
JSON


Activities
Visual Studio Code
Mar 15 11:14
mock_account.json - Documents - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
DOCUMENTS
tradebotv3
data
historical
real_time
logs
data_logs_15m.log
error_logs_15m.log
memory_logs_15m.log
time_logs_15m.log
warning_logs_15m.log
processed
BTCUSDT_15m_2025-03.csv
ETHUSDT_15m_2025-03.csv
raw
BTCUSDT_15m_2025-03.csv
ETHUSDT_15m_2025-03.csv
scaler
doc
mock
config
mock_account.json M
mock_orders.json
mock_past_trades.json M
logs
mock_real_time_dealer.log
notebooks
scripts
src
backtesting
data_handling
feature_engineering
live_trading
OUTLINE
TIMELINE
l_time_dealer.py M
mock_account.json M
requirements.tx
mock_past_trades.json M
tradebotv3 > mock > config > mock_account.json > updateTime
{
"makerCommission": 15,
"takerCommission": 15,
"buyerCommission": 0,
"sellerCommission": 0,
"canTrade": true,
"canWithdraw": true,
"canDeposit": true,
"updateTime": 1624362346000,
"accountType": "SPOT",
"balances": [
{
"asset": "BTC",
"free": "0.0117882",
"locked": "0.0000000"
},
{
"asset": "ETH",
"free": "0.0",
"locked": "0.0000000"
},
{
"asset": "USDT",
"free": "6.384186000000113",
"locked": "0.00000000"
}
],
"permissions": [
"SPOT"
]
}
tradebotv3 > mock > config > mock_past_trades.json > ...
},
{
"symbol": "ETHUSDT",
"orderId": 17358143672,
"clientOrderId": "mock_17358143672",
"transactTime": 1735814367610,
"price": 3468.0,
"origQty": 0.14400031387882295,
"executedQty": 0.14400031387882295,
"cummulativeQuoteQty": 0,
"status": "FILLED",
"timeInForce": "GTC",
"type": "MARKET",
"side": "sell"
},
{
"symbol": "BTCUSDT",
"orderId": 17420193021,
"clientOrderId": "mock_17420193021",
"transactTime": 1742019302858,
"price": 84204.73,
"origQty": 0.0118,
"executedQty": 0.0118,
"cummulativeQuoteQty": 0,
"status": "FILLED",
"timeInForce": "GTC",
"type": "MARKET",
"side": "buy"
}
]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bai@bai-Inspiron-7460:~/Documents/tradebotv3$ python3 scripts/mock_trade.py
/home/bai/.local/lib/python3.10/site-packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g. as a system package and as a pip package). As a result, the 3D projection is not available.
warnings.warn("Unable to import Axes3D. This may be due to multiple versions of "
bash
python3 tr...
Running 15 mins time frame on another Computer with Ubuntu OS
main*
Ln 9, Col 23
Spaces: 4
UTF-8
LF
JSON

FUTURE WORKS

FUTURE WORKS

- Optimization of data storage
- Shorting Capabilities
- Expansion of strategy/indicator portfolio

Trade Bot version 4:

- Cleaner modular design
- Enhancement of Configuration File designs