**1. Data Retrieval and Constituent Handling (data\_loader.py)**

**Issues:**

* **API Key Stored in Plain Text:** The API key was hardcoded in the script, posing a security risk.
* **Incorrect Stock Selection Logic:** The filtering criteria for added/removed stocks were not strict enough, leading to survivorship bias correction failure.
* **Failure to Handle Special Stock Symbols:** Stocks like **BRK.B** were not converted to **BRK-B**, causing download failures.

**Fixes:**

* **Use Environment Variables:** Store the API key in environment variables to enhance security.
* **Strict Filtering Criteria:**

**Added stocks** must have a StartDate within the backtesting period.

**Removed stocks** must have an EndDate within the backtesting period.

* **Standardize Symbol Formatting:** Replace . with - to ensure compatibility with Yahoo Finance.

**2. Survivorship Bias Correction (feature\_engineering.py)**

**Issues:**

* **Fetching All Data at Once:** This may trigger API rate limits and does not handle missing stocks properly.
* **Loose Time Truncation Logic:** Did not account for stocks that had been delisted before the backtesting period, leading to index errors.

**Fixes:**

* **Batch Data Downloads:** Retrieve data in batches of 50 stocks per request to avoid API limits.
* **Strict Time Filtering:**

**Removed stocks:** Data is truncated at their EndDate.

**Added stocks:** Data is included only from their StartDate.

* **Remove Completely Empty Columns:** Prevents invalid data from affecting calculations.

**3. Momentum Factor Calculation and Model (ml\_pipe\_line.py)**

**Issues:**

* **Unreliable Risk-Free Rate Source:** The risk-free rate was scraped from a webpage, making it prone to failure and not compounded correctly.
* **Misaligned Rolling Window:** Missing values caused errors in the 12-month return calculation.
* **Lack of Error Handling in Stock Selection:** The strategy did not account for cases where fewer than five valid stocks were available.

**Fixes:**

* **Use FRED API for Reliable Risk-Free Rate Data:** Convert the annualized rate to a monthly compounded rate.
* **Improve Rolling Window Alignment:**
  + Use min\_periods=12 to ensure a full 12-month history.
  + Fill missing values (NaN) by assuming suspended stocks had zero returns.
* **Dynamic Stock Selection:** If fewer than five valid stocks are available, select as many as possible.

**4. Backtesting and Performance Evaluation (backtesting.py)**

**Issues:**

* **No Trading Cost Consideration:** Backtest results were overly optimistic.
* **Misaligned Time Series:** The strategy and benchmark returns had mismatched indices, leading to incorrect performance metrics.

**Fixes:**

* **Simulate Trading Costs:** Apply a **0.1% one-way transaction fee** to make results more realistic.
* **Force Time Index Alignment:** Ensure strategy and benchmark returns have **matching timestamps** before computing performance metrics.

**5. Main Execution Flow and Robustness (run\_strategy.py)**

**Issues:**

* **Lack of Exception Handling:** Network failures or missing data could crash the program.
* **No Logging Mechanism:** Debugging was difficult due to the absence of logs.

**Fixes:**

* **Implement try-except Blocks and Logging:** Catch and log errors instead of allowing the program to crash.
* **Use tenacity for API Request Retries:** Automatically retry failed API requests to improve robustness.

**Summary**

The revised code significantly improves strategy **accuracy and usability** through the following enhancements:

**Data Quality:**

* Strictly process constituent stock changes and special symbols to **avoid survivorship bias**.

**Model Robustness:**

* Use **authoritative data sources** and improve the rolling window & stock selection **error handling**.

**Backtesting Realism:**

* Introduce **trading costs** and ensure **time alignment** for a **realistic** simulation.

**Code Resilience:**

* Implement **exception handling, logging, and environment variable management** to **enhance maintainability**.