DSCI-560 Lab 4 – Algorithmic Trading Simulation

https://github.com/tianqi0301/DSCI560/tree/main/Lab4

Team Name: The Coach

Name	USC ID
Tianqi Qiu	1716906139
Shida Yan	5725964711
Herun Kan	7222919427

Project Structure

```
.
├── main.py
├── portfolio.py
├── indicators.py
├── strategies.py
├── metrics.py
├── data_stream.py
└── requirements.txt
```

How to Run the Project

Install Dependencies

```
pip install -r requirements.txt
```

Run the Simulation

```
python main.py
```

Project Overview

In this project, we built an algorithmic trading system that analyzes stock price data and simulates trading decisions using a mock portfolio. The goal of the project is not to predict the market perfectly, but to understand how trading algorithms work in practice and how portfolio performance can be evaluated.

Instead of relying on continuous live trading, which is difficult to manage in a classroom environment, we simulate trading over real historical intraday market data. This allows us to clearly observe buy and sell decisions, portfolio changes, and performance metrics in a reproducible way.

Data Source

- **Source:** Yahoo Finance
- **Library Used:** `yfinance`
- **Data Type:** Historical intraday stock data
- **Interval:** 5-minute bars
- **Time Period:** One full trading day

Each row of data represents a completed 5-minute trading interval. Using intraday data gives us realistic market behavior while avoiding the limitations of running a system continuously for an entire trading day.

Real-Time Data Collection

```
%%writefile data_stream.py
import yfinance as yf
from datetime import datetime

def stream_price(symbol="AAPL"):
    ticker = yf.Ticker(symbol)
    data = ticker.history(period="1d", interval="1m")
    if data.empty:
        return None
    return {
        "time": datetime.now(),
        "symbol": symbol,
        "price": float(data["Close"].iloc[-1]),
        "volume": int(data["Volume"].iloc[-1])
    }
```

Stocks Tracked

We selected 10 highly liquid stocks and ETFs to ensure stable data availability and realistic trading behavior:

```
SYMBOLS = [
    "AAPL", "MSFT", "GOOGL", "AMZN", "NVDA",
    "META", "TSLA", "AMD", "SPY", "QQQ"
]
```

This selection includes large-cap technology stocks, more volatile stocks, and market ETFs, providing diversification across different market characteristics.

Trading Algorithm

```
%%writefile strategies.py
def hybrid_strategy(df):
    if len(df) < 15:
        return 0
    price = df["price"].iloc[-1]
    sma_val = df["sma"].iloc[-1]
    rsi_val = df["rsi"].iloc[-1]
    if price > sma_val and rsi_val < 70:
        return 1
    if price < sma_val or rsi_val > 70:
        return -1
    return 0
```

Strategy Overview

We use a hybrid technical analysis strategy that combines the Simple Moving Average (SMA) and the Relative Strength Index (RSI). These indicators are commonly used in algorithmic trading and are easy to interpret, making them suitable for this assignment.

How the Strategy Works

- **Buy Signal (1):**
    - The stock price is above its moving average, indicating an upward trend
    - RSI is below 70, meaning the stock is not overbought
- **Sell Signal (-1):**
    - The stock price falls below the moving average or
    - RSI rises above 70, indicating potential overbought conditions

The buy condition is intentionally stricter than the sell condition. This design helps the strategy avoid entering weak trades while exiting positions early when market conditions become unfavorable.

Mock Trading Environment

```
%%writefile portfolio.py
class Portfolio:
    def __init__(self, initial_cash=100000):
        self.cash = initial_cash
        self.positions = {}
        self.history = []

    def buy(self, symbol, price, shares):
        cost = price * shares
        if self.cash >= cost:
            self.cash -= cost
            self.positions[symbol] = self.positions.get(symbol, 0) + shares

    def sell(self, symbol, price, shares):
        if symbol not in self.positions:
            return
        if self.positions[symbol] < shares:
            return
        self.cash += price * shares
        self.positions[symbol] -= shares
        if self.positions[symbol] == 0:
            del self.positions[symbol]

    def value(self, prices):
        total = self.cash
        for s, sh in self.positions.items():
            total += prices.get(s, 0) * sh
        return total
```

To evaluate the strategy, we use a simulated portfolio rather than real money.

Portfolio Setup

- **Initial Capital:** $100,000

- **Leverage:** None
- **Transaction Costs:** Not included

Position Sizing

We use an aggressive position sizing approach, allocating 30% of the available cash whenever a buy signal occurs. This allows the portfolio to meaningfully participate in market movements instead of holding most of the capital as cash.

At each time step, the system tracks:

- Current cash balance
- Shares held for each stock
- Total portfolio value (cash + value of all positions)

Performance Metrics

```
%%writefile metrics.py
import numpy as np
import pandas as pd

def calculate_metrics(values):
    returns = pd.Series(values).pct_change().dropna()
    total_return = (values[-1] / values[0]) - 1
    sharpe = 0
    if returns.std() != 0:
        sharpe = (returns.mean() / returns.std()) * np.sqrt(252)
    return total_return, sharpe
```

To evaluate the strategy, we use the following metrics:

Total Portfolio Return

This measures how much the portfolio gained or lost over the simulation period:

$$\text{Total Return} = \frac{\text{Final Portfolio Value} - \text{Initial Capital}}{\text{Initial Capital}}$$

Sharpe Ratio

The Sharpe ratio measures risk-adjusted performance, showing how much return was achieved relative to volatility:

$$\text{Sharpe Ratio} = \frac{\text{Average Return}}{\text{Standard Deviation of Returns}} \times \sqrt{252}$$

A higher Sharpe ratio indicates more stable and consistent returns.

Results

Simulation Configuration

- Time horizon: 1 trading day
- Data interval: 5 minutes
- Capital allocation per trade: 30%

Final Results

```
Total Return: 1.54%
Sharpe Ratio: 1.40
```

```
2026-02-06 20:59:00+00:00 | AAPL | Signal: -1 | Cash: $498,311.66 | Positions: {'MSFT': 1290} | Total Value: $498,311.66
2026-02-06 20:59:00+00:00 | MSFT | Signal: -1 | Cash: $1,015,356.56 | Positions: {} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | GOOGL | Signal: -1 | Cash: $1,015,356.56 | Positions: {} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | AMZN | Signal: -1 | Cash: $1,015,356.56 | Positions: {} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | NVDA | Signal: -1 | Cash: $1,015,356.56 | Positions: {} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | META | Signal: -1 | Cash: $1,015,356.56 | Positions: {} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | TSLA | Signal: 1 | Cash: $710,798.14 | Positions: {'TSLA': 741} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | AMD | Signal: -1 | Cash: $710,798.14 | Positions: {'TSLA': 741} | Total Value: $1,015,356.56
2026-02-06 20:57:00+00:00 | SPY | Signal: -1 | Cash: $710,798.14 | Positions: {'TSLA': 741} | Total Value: $1,015,356.56
2026-02-06 20:59:00+00:00 | QQQ | Signal: -1 | Cash: $710,798.14 | Positions: {'TSLA': 741} | Total Value: $1,015,356.56

Final Results
Total Return: 1.54 %
Sharpe Ratio: 1.4
```

Interpretation

The strategy achieved a **positive intraday return**, which is realistic for short-term trading. While the total return is modest, this is expected when trading over a single day without leverage. The Sharpe ratio above 1 indicates that the returns were achieved with relatively controlled risk and limited volatility.

Key Design Choices

- Historical intraday data was used instead of live streaming to ensure reproducibility.
- A hybrid SMA and RSI strategy was selected for transparency and interpretability.
- Fixed percentage-based position sizing was used to increase market exposure.
- No leverage or transaction costs were included to focus on strategy behavior.

Limitations and Future Improvements

- Transaction costs and slippage are not modeled.
- Stop-loss and take-profit rules are not implemented.
- Results are based on a single trading day and may vary under different market conditions.

Possible future improvements include:

- Multi-day backtesting
- Drawdown analysis
- More advanced position sizing
- Comparison with ARIMA or LSTM-based models

Conclusion

This project demonstrates a complete algorithmic trading workflow, from data collection to strategy execution and performance evaluation. The results show that even a simple, interpretable strategy can produce positive returns when combined with appropriate risk management and position sizing.

Team Meeting Notes

Day 1 – 2/2/2026

- Reviewed the Lab 4 assignment requirements and overall objectives
- Discussed possible algorithm choices and agreed to focus on technical analysis methods
- Divided tasks so each team member could research different approaches

Day 2 – 2/3/2026

- Shared findings on moving averages and smoothing methods
- Discussed more advanced models such as ARIMA and LSTM
- Agreed to start with a simpler strategy before testing advanced models
- Decided to focus on portfolio-based evaluation metrics

Day 3 – 2/4/2026

- Finalized the use of a hybrid SMA and RSI trading strategy
- Designed the mock trading environment, including cash tracking, positions, and portfolio valuation
- Began initial intraday testing using historical market data

Day 4 – 2/5/2026

- Reviewed early backtesting results and portfolio performance
- Discussed how position sizing affected overall returns
- Decided to adopt a more aggressive allocation strategy
- Improved portfolio valuation and execution logic

Day 5 – 2/6/2026

- Confirmed that the trading system was complete and functioning correctly
- Assigned final responsibilities for documentation and the video presentation
- Prepared all materials for submission