Hybrid Financial Intelligence System

ML-Based Stock Movement Prediction

Dalil ADIMI   Hassen BEN AMOR

Group I2-NEW DAI
Data Science Module

Supervisor: Stephany RAJEH
EFREI Paris

February 11, 2026

Outline

Project Overview

Objective

Build an end-to-end ML pipeline that predicts whether a stock's price will rise by >3.5% over the next 5 trading days (binary classification).

Target Stocks:

- SMCI (Super Micro Computer)
- CRSP (CRISPR Therapeutics)
- PLTR (Palantir Technologies)

Tech Stack:

- Python (pandas, scikit-learn, XGBoost)
- Plotly for visualization
- Streamlit dashboard
- FastAPI backend

Data Pipeline

Sources & APIs:

Source	Library / API
Yahoo Finance	`yfinance`
FRED	REST API
Finnhub	REST API (optional)

Dataset:

- 4,414 daily records (2020–2026)
- 3 tickers, OHLCV + macro
- No missing values, no duplicates

Preprocessing Steps:

1. Download OHLCV per ticker
2. Merge macro data (forward-fill)
3. Compute 12 technical indicators
4. Build 14-day rolling aggregates
5. Generate binary target labels
6. Drop NaN warm-up rows

Result

4,357 rows $\times$ 40 columns

EDA — Stock Prices

[Insert: Normalized Price Chart]

Observations:

- SMCI: extreme rallies & drawdowns
- PLTR: uptrend since late 2022
- CRSP: independent pattern
- Daily volatility: 3.8–5.2%

Target Distribution:

- Positive class (up $>$3.5%): 34%
- Negative class: 66%
- Handled via `scale_pos_weight`

EDA — Distributions & Correlations

[Insert: Return Distribution]

[Insert: Correlation Heatmap]

Feature Engineering

12 Base Indicators:

- RSI(14), MACD (line, signal, hist)
- Bollinger Band width
- Moving Averages (50, 200)
- ATR(14), ATR %
- Volume Z-score
- 10Y Treasury Yield

Rolling Window (14 days):

- For each indicator: **mean, std, last**
- $12 \times 3 = 36$ features total
- Captures recent trend + variability

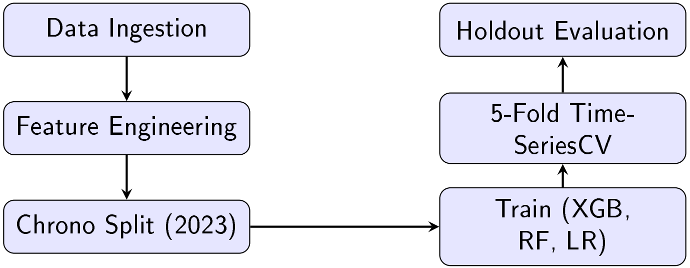
Target Variable:

$$y_t = \mathbb{1}\left(\frac{C_{t+5}}{C_t} - 1 > 0.035\right)$$

Why Rolling Windows?

Raw indicator values change scale over time. Rolling statistics normalize the features and improve generalization.

Pipeline Architecture



Train: before 2023-01-01 (2,038 samples)
Test: from 2023-01-01 (2,319 samples)

Scaling: MinMaxScaler
CV metric: Precision (5-fold)

Model Comparison — Results

Model	Accuracy	Precision	F1	ROC-AUC
XGBoost	53%	0.366	0.373	0.513
Random Forest	58%	0.390	0.303	0.518
Logistic Reg.	55%	0.399	0.406	0.506

[Insert: Confusion Matrix]

Key Takeaways:

- All models slightly above random (AUC > 0.50)
- Logistic Regression competitive with tree models
- Stock prediction is inherently noisy — these results are realistic

Backtest & Equity Curve

[Insert: Equity Curve]

Backtest Setup:

- Non-overlapping 5-day windows
- Long-only / cash strategy
- Signal: BUY if $P(\text{up}) \geq 0.50$

Metrics:

- Annualized return: 31.3%
- Max drawdown: -28.4%

Live Dashboard

[Insert: Dashboard Screenshot]

Conclusion

What we built:

1. End-to-end ML pipeline: ingestion $\rightarrow$ features $\rightarrow$ training $\rightarrow$ evaluation $\rightarrow$ deployment
2. Chronological train/test split (no data leakage)
3. 3 models compared (XGBoost, Random Forest, Logistic Regression)
4. Streamlit dashboard + FastAPI for serving predictions

Lessons learned:

- Stock prediction is hard — 53% accuracy is realistic for this domain
- More features $\neq$ better performance (we tested 40+ features, marginal gain)
- Temporal validation is critical to avoid overfitting
- Simple, interpretable pipelines are more valuable than complex ones

Future Work

Model Improvements:

- Add sentiment features (FinBERT)
- Try deep learning (LSTM, Transformers)
- Ensemble stacking
- Probability calibration

Engineering:

- Dockerized deployment
- Automated retraining pipeline
- Model monitoring & drift detection
- CI/CD for model updates

Thank You!

Questions?

Dalil ADIMI & Hassen BEN AMOR
Group I2-NEW DAI — EFREI Paris