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 12, 2026

Outline

What This Project Does (No Finance Needed)

In one sentence

We use **historical daily data** (prices + a few derived numbers) to train a model that answers: "**Will this asset go up by more than 3.5% in the next 5 days?**" Yes or no.

Why 3.5% and 5 days? We need a clear, binary target. 3.5% is a meaningful move for volatile stocks; 5 days is a short horizon so we have many examples. You can think of it as a **binary classification** problem: input = features today, output = class 0 (no) or 1 (yes). **Stocks we use:** SMCI, CRSP, PLTR (three tickers). They are volatile, so 3.5% moves happen often enough to learn from.

Project Overview

Objective

Build an end-to-end ML pipeline: **binary classification** — predict if price will rise $>3.5\%$ over the next 5 trading days.

Target stocks (3 tickers):

- SMCI, CRSP, PLTR

Tech stack:

- Python: pandas, scikit-learn, XGBoost
- Streamlit, FastAPI

ML pipeline in short:

1. Get data $\rightarrow$ build features
2. Split by **time** (train before 2023, test from 2023)
3. Train 3 models, pick one, tune threshold
4. Evaluate with metrics + backtest

ML in Plain Words (1/2)

What we do: *Supervised binary classification.*

- **Input (features):** For each day and each stock, we have $\sim$40 numbers (rolling means/stds of RSI, volatility, macro, etc.).
- **Output (label):** 0 or 1. 1 = "price went up more than 3.5% in the next 5 days."
- **Training:** We fit a model on past data (before 2023) so it learns which feature values tend to go with "1" vs "0."
- **Testing:** We evaluate on future data (from 2023) to see how well it *generalizes*.

ML in Plain Words (2/2)

Why split by time?

If we shuffled dates randomly, the model could "cheat" by using future information. We split by **time**: train on the past, test on the future $\Rightarrow$ no leakage, realistic performance.

Summary: Past data $\rightarrow$ train model $\rightarrow$ predict on unseen future dates $\rightarrow$ measure accuracy, precision, etc. on that holdout period.

Data Sources & Dataset

Where the data comes from:

Source	What we get
Yahoo Finance (`yfinance`)	Daily OHLCV per stock
FRED (API)	10-year US Treasury yield (one series)

Dataset size:

- 4,414 rows (one per date $\times$ ticker), 2020–2026
- 3 tickers: SMCI, CRSP, PLTR
- No missing values, no duplicates after preprocessing

Preprocessing Steps

What we do to the raw data (in order):

1. Download OHLCV per ticker; merge 10Y yield by date (forward-fill weekends).
2. Compute 12 derived series (RSI, MACD, Bollinger, MAs, ATR, volume Z-score, yield).
3. For each series: take **mean**, **std**, **last** over last 14 days $\Rightarrow$ 36 features.
4. For each row, define label: 1 if return over next 5 days $>$ 3.5%, else 0.
5. Drop rows with NaN (warm-up for indicators).

Result

Final matrix: **4,357 rows** $\times$ **40 columns** (features + id columns). Ready for ML.

Dataset & Features — Raw Columns

Raw dataset (one row per date $\times$ ticker):

Column	Meaning
OHLCV	Open, High, Low, Close (prices); Volume (trade count)
Macro	10Y US Treasury yield (FRED), forward-filled by date

From these we **derive** 12 time series (next slide). Then we apply a **rolling window**: for each series we take mean, std, and last value over the last 14 days, so the model sees “recent context” instead of a single day.

Dataset & Features — The 12 Base Features

12 base series (all numeric). Selected ones:

Feature	What it is (engineer-friendly)
RSI(14)	Momentum: 0–100 index from recent gains/losses
MACD (12,26,9)	Trend: fast vs slow price averages; signal & histogram
Bollinger width	Volatility: (upper − lower band) / price; 20-day $\pm 2\sigma$
MA 50 / MA 200	Smoothed price: 50-day and 200-day rolling mean
ATR(14), ATR%	Volatility: avg daily range; ATR% = ATR/price $\times$ 100
Volume Z-score	(volume − 50-day mean) / 50-day std
10Y yield	US gov bond rate (one series, merged by date)

Rolling step: mean, std, last over 14 days per series $\Rightarrow$ 36 features. Final: **4,357** $\times$ **40**.

Target & Macro Context

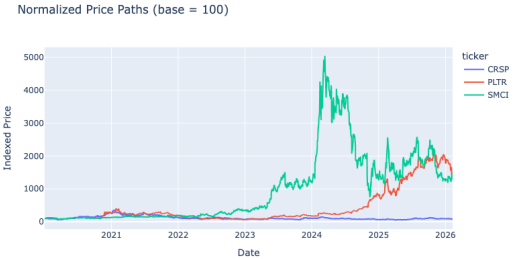
What we predict (plain language): At each day t, the label is: "Will this stock go **up by more than 3.5%** over the **next 5 trading days**?" $\Rightarrow$ binary (yes/no). Formally:

$$y_t = \mathbb{1}\left(r_{t,t+5} > 0.035\right), \quad r_{t,t+5} = \frac{\text{close}_{t+5}}{\text{close}_t} - 1$$

10-year Treasury yield: A single daily time series (US government bond rate from FRED). We use it as an **input feature** only: it encodes "macro regime" (e.g. low vs high rates). Merged by date and forward-filled so every (date, ticker) row has a value.

Chart: how that macro feature evolves over time. The model uses it as a numeric covariate.

EDA — Stock Prices



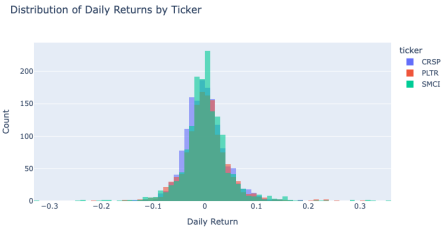
Observations:

- 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



Left: 5-day return dist.

Right: target balance (34% up $>$3.5%).

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

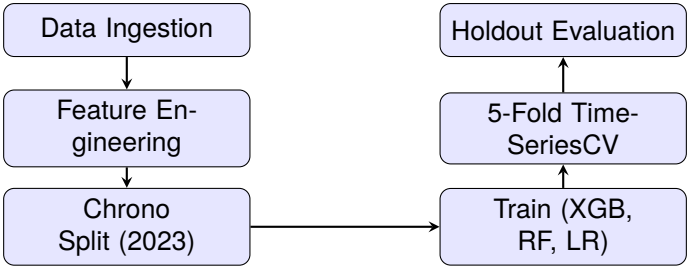
Target Variable:

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

Why Rolling Windows?

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

Pipeline Architecture

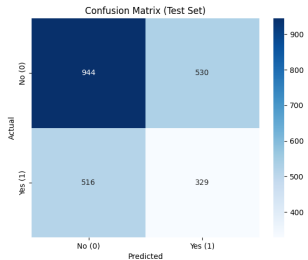


Train: before 2023-01-01 (2,038 samples)
Test: from 2023-01-01 (2,319 samples)

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

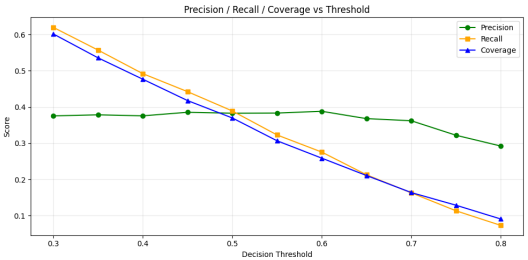


Key Takeaways:

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

Threshold Tuning (from Notebook)

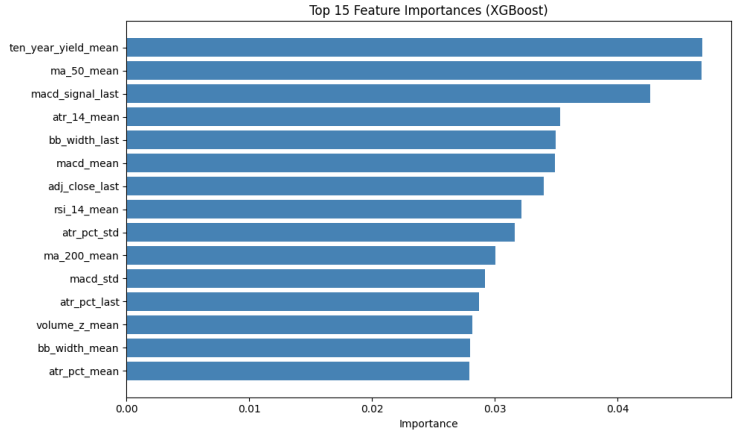
Problem: Default 0.5 may be suboptimal. We sweep thresholds and evaluate by *Sharpe ratio* on the backtest.



From notebook:

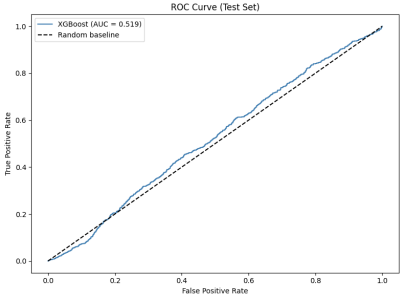
- Higher threshold $\Rightarrow$ fewer trades, often better precision
- Best threshold by Sharpe: **0.40**
- Trade-off: precision vs. coverage

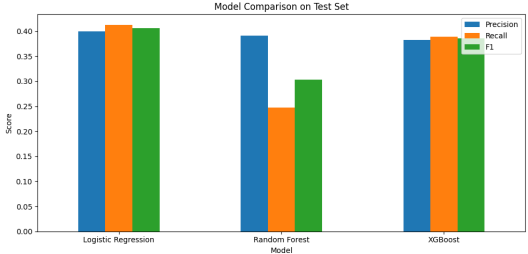
Feature Importance (XGBoost)



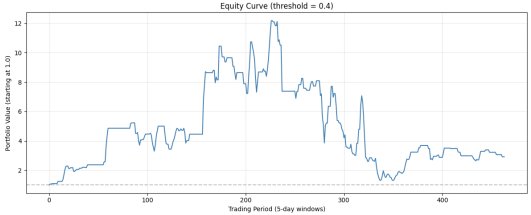
Rolling-window stats (mean/std/last) of RSI, ATR, MACD, 10Y yield, etc. drive predictions; ablation in the notebook confirms which feature groups help.

ROC & Model Comparison





Backtest & Equity Curve



Backtest Setup:

- Non-overlapping 5-day windows
- Long-only / cash strategy
- Signal: BUY if $P(\text{up}) \geq$ **0.40** (best by Sharpe)

Metrics (threshold 0.4):

- Annualized return, Sharpe, max drawdown from holdout
- Threshold chosen by risk-adjusted performance

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