Financial Forecasting Equations with Scientific Machine Learning

Article · June 2025

CITATIONS
0

READS
16

1 author:

Parsa Besharat
TU Bergakademie Freiberg
7 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Financial Forecasting Equations with Scientific Machine Learning

Journal:	*Environmental Data Science*
Manuscript ID	EDS-2025-0061
Manuscript Type:	Application Paper
Date Submitted by the Author:	30-Jun-2025
Complete List of Authors:	Besharat, Parsa; TU Bergakademie Freiberg University, Mathamtics and Computer Science
Keywords:	Financial Forecasting, Scientific Machine Learning, SINDY

Abstract:	Financial markets are inherently nonlinear, dynamic, and noisy, posing significant challenges for traditional time series forecasting methods. This document explores a novel approach to financial forecasting by integrating Scientific Machine Learning (SciML) techniques, specifically the Sparse Identification of Nonlinear Dynamics (SINDy) [5] algorithm, with domain knowledge from financial time series analysis and advanced feature engineering. We aim to discover interpretable differential equations that govern market behavior.

Financial Forecasting Equations with Scientific Machine Learning

Parsa Besharat
TU Bergakademie Freiberg
Freiberg, Germany
parsa.besharat@student.tu-freiberg.de

Abstract

Financial markets are inherently nonlinear, dynamic, and noisy, posing significant challenges for traditional time series forecasting methods. This document explores a novel approach to financial forecasting by integrating Scientific Machine Learning (SciML) techniques, specifically the Sparse Identification of Nonlinear Dynamics (SINDy) [5] algorithm, with domain knowledge from financial time series analysis [4] and advanced feature engineering. We aim to discover interpretable differential equations that govern market behavior.

Financial concepts such as returns and volatility are incorporated as features [4], guided by methodologies [8]. The resulting hybrid framework enables both data-driven discovery of governing equations and quantitative forecasting, with an emphasis on sparse, explainable models. Our experiments on cryptocurrency and equity datasets demonstrate the potential of SINDy-based models [9] to extract meaningful dynamics and offer competitive forecasting performance, especially when enhanced with carefully engineered financial features. This work contributes to the growing field of explainable SciML applications in finance, bridging theoretical dynamics and real-world market signals.

Keywords: Financial Forecasting, Scientific Machine Learning, SINDY

1 Introduction

In this work, we tackle the formidable challenge of modeling financial markets as nonlinear, dynamic systems by uncovering the very equations that govern their evolution. Our goal is to apply the Sparse Identification of Nonlinear Dynamics (SINDy) framework to market time-series data augmented with engineered financial features such as returns and volatility to infer compact, interpretable differential equations. By coupling these mechanistic models with downstream machine learning techniques, we aim not only to forecast asset prices more accurately but ultimately to inform trading strategies that seek to maximize profit without incurring losses.

Central to our approach is the classical methodology of system identification, wherein measurement data are used to infer the underlying dynamical system [1]. Once the governing equations are discovered, they offer a trifecta of benefits: they can predict future market states, guide control inputs for algorithmic trading, and provide a clean mathematical lens for theoretical analysis. We validate our framework on cryptocurrency and equity datasets publicly sourced from Kaggle and maintained in our GitHub repository—using the codebase available at:

`https://github.com/parsabe/MLMatrix/tree/master/src/Scientific%20ML`

2 Background and Related Work

There are multiple and several strands of research that share the goal of predicting asset prices while carefully managing profit and loss in this field but some of them could get the huge loss of the money when investing. even though the algorithm predicted it to have maximum minimized loss function.

1. Classical econometric models such as ARIMA and GARCH have long been used to capture linear autocorrelations and volatility clustering in asset returns [4]. While these methods provide solid statistical foundations, their linear structure often fails to represent abrupt regime shifts and nonlinear patterns in real markets, resulting in modest profitability when translated into trading rules.

2. To better align forecasting errors with financial outcomes, Kim introduced a cost-sensitive support-vector regression that penalizes under-predictions more severely than over-predictions [2]. This asymmetric loss led to higher back-tested net returns by focusing the model on avoiding missed upside opportunities, but it remains sensitive to outliers and hinges on careful kernel selection—limiting its reliability in turbulent market conditions.

3. Hybrid architectures that combine linear and nonlinear models were proposed by Zhang et al., who used ARIMA to model the predictable linear component and a feed-forward neural network to learn the residual nonlinear structure [3]. This two-stage design reduced overall forecast error and improved Sharpe ratios, yet the decoupling demands extensive hyperparameter tuning and can propagate errors from one stage to the next.

4. Reinforcement-learning techniques recast trading as a sequential decision process, defining rewards as net profit minus transaction costs [1]. Such agents learn adaptive execution policies that react to market impact and volatility, often beating static benchmarks. Nevertheless, they demand realistic market simulators, incur high computational overhead, and can suffer from model–market mismatches in live deployment.

3 Basic Concepts for our methodology

In this section, we first introduce the foundational concepts and tools that underpin our hybrid SciML framework. We begin by exploring system-identification via the SINDy algorithm, the nature and statistical properties of financial time series, the different types of trends encountered in market data, the operational mechanics of cryptocurrencies, and common chart patterns used in technical analysis. After establishing these basics, we turn to the machine learning that form the core of our predictive model.

3.1 Sparse Identification of Nonlinear Dynamical Systems (SINDy)

SINDy (Sparse Identification of Nonlinear Dynamical Systems) is a data-driven technique that uncovers the underlying differential equations governing a dynamical system directly from time-series observations [5]. By constructing a large library of candidate functions and applying sparse regression, SINDy selects the minimal set of terms that accurately reproduce the observed evolution, yielding interpretable models akin to those found in physics or biology.

3.1.1 Mathematical Background

We consider a continuous-time dynamical system of the form

$$\dot{\mathbf{x}}(t) \;=\; f\big(\mathbf{x}(t)\big),$$

where $\mathbf{x}(t) = [\,x_1(t), x_2(t), \ldots, x_n(t)\,]^\top \in \mathbb{R}^n$ is the state vector at time t, and $f(\mathbf{x})$ is the (unknown) nonlinear function driving the dynamics [5].

3.1.2 Data Collection

We gather measurements of the state and its derivatives at m time points:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^\top(t_1) \\ \vdots \\ \mathbf{x}^\top(t_m) \end{bmatrix}, \quad \dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^\top(t_1) \\ \vdots \\ \dot{\mathbf{x}}^\top(t_m) \end{bmatrix},$$

where $\mathbf{X} \in \mathbb{R}^{m\times n}$ contains state snapshots and $\dot{\mathbf{X}}$ the corresponding derivatives [5].

3.1.3 Library of Candidate Functions

To discover the governing dynamics, we form a library $\Theta(\mathbf{X})$ of p possible nonlinear functions:

$$\Theta(\mathbf{X}) = \big[\,1 \quad \mathbf{X} \quad \mathbf{X}^{P2} \quad \mathbf{X}^{P3} \quad \ldots \quad \sin(\mathbf{X}) \quad \cos(\mathbf{X}) \quad \ldots\big],$$

where each column is a different function (e.g. x_1^2, x_1x_2, $\sin(x_2)$, etc.) applied row-wise to $\mathbf{X}$. For instance, the quadratic terms $\mathbf{X}^{P2}$ are

$$\mathbf{X}^{P2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \ldots & x_n^2(t_1) \\ \vdots & \ddots & & \vdots \\ x_1^2(t_m) & \ldots & & x_n^2(t_m) \end{bmatrix}.$$

3.1.4 Sparse Regression

In practice, measurements are noisy, so we pose

$$\dot{\mathbf{X}} \;=\; \Theta(\mathbf{X})\,\boldsymbol{\Xi} \;+\; \eta\,\mathbf{Z},$$

where $\boldsymbol{\Xi} \in \mathbb{R}^{p\times n}$ is the sparse coefficient matrix to learn, $\mathbf{Z}$ is Gaussian noise, and η its magnitude. The task is to recover a sparse $\boldsymbol{\Xi}$ so that $\Theta(\mathbf{X})\,\boldsymbol{\Xi}$ matches $\dot{\mathbf{X}}$ despite noise. Common solvers include LASSO and Sequential Thresholded Least Squares (STLSQ) [5].

3.2 Financial Time Series and Their Characteristics

Financial time series are sequences of data points—such as stock prices, exchange rates, or cryptocurrency values—recorded at regular intervals. They exhibit several distinctive features that set them apart from classical time series [4].

3.2.1 Distinctive Features

- **High Frequency:** Financial data are often available at daily, hourly, or even minute-level granularity, resulting in large datasets over short calendar spans [4].
- **Time-Varying Volatility:** The variance of returns is not constant; periods of tranquility alternate with clusters of turbulence, complicating model assumptions of homoscedasticity [4].
- **Systematic vs. Nonsystematic Factors:** Movements arise from broad macroeconomic trends (systematic risk) as well as idiosyncratic events like firm-specific news (nonsystematic risk) [4].
- **Added Uncertainty:** Beyond statistical noise, markets evolve under shifting economic conditions and regulatory environments, and true volatility must itself be estimated rather than observed directly [4].

3.2.2 Applications

Financial time-series analysis underpins a wide array of tasks:

- Asset valuation and forecasting
- Risk management and stress testing
- Option and derivative pricing

3.2.3 Examples

Daily Returns A common way to measure performance is via daily returns rather than raw price levels:

$$\text{Daily Return} \;=\; \left(\frac{P_{\text{close}} - P_{\text{open}}}{P_{\text{open}}}\right) \times 100,$$

where P_{open} and P_{close} are the opening and closing prices, respectively.

Insurance Claims Time Series Counts or values of insurance claims recorded over time reveal seasonal patterns and trends, aiding insurers in reserving and risk assessment.

Cryptocurrency Prices Cryptocurrency time series share many stylized facts with equity markets—heavy tails, volatility clustering—but often display even higher frequency swings and jumps due to market microstructure and speculative behavior.

3.2.4 Asset Returns vs. Prices

Rather than modeling absolute price levels, we focus on returns to achieve scale invariance and simplify comparisons across assets. Returns capture the percentage gain or loss over a holding period, making them more directly tied to profit and loss than raw prices.

3.2.5 Simple Gross Return

This metric reflects the total value of your investment—including the original capital and any profit. For instance, investing \$100 and ending up with \$110 corresponds to a gross return of 110%.

$$1 + R_t = \frac{P_t}{P_{t-1}}$$

Where:

- P_t is the price at time t (today).
- P_{t-1} is the price at time $t-1$ (yesterday).
- R_t is the simple gross return.

3.2.6 Simple Net Return

This measures the percentage profit or loss relative to your original investment. If you start with \$100 and end with \$110, your net return is 10%.

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

Where:

- P_t is the price at time t (today).
- P_{t-1} is the price at time $t-1$ (yesterday).
- R_t is the simple net return.

3.2.7 Log Return

When computing returns over many successive periods, log returns are preferred because they compound additively. They model continuous growth, similar to continuously-compounded interest.

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln\left(P_t\right) - \ln\left(P_{t-1}\right).$$

Where:

- r_t is the log return at time t.
- P_t is the price at time t (today).
- P_{t-1} is the price at time $t-1$ (yesterday).
- $\ln(\cdot)$ denotes the natural logarithm.

3.3 Cryptocurrency Mechanics

Cryptocurrencies operate on a decentralized ledger called the blockchain, where each participant (node) maintains a full copy of all transactions. Instead of relying on a central authority, network consensus ensures that every node agrees on the order and validity of transactions. This consensus is secured by cryptographic hashes linking each block to its predecessor, forming an immutable chain that resists tampering [7].

New blocks are created through mining, a process in which specialized nodes solve a proof-of-work puzzle: they repeatedly hash a candidate block header until the resulting hash meets a network-wide difficulty target. The first miner to find a valid nonce broadcasts the block, which is then verified and appended by other nodes. Successful miners receive a block reward and transaction fees, incentivizing them to secure the network and process transactions [7].

Transactions begin when a user signs a transfer of value from one address to another using their private key. These signed transactions propagate through the peer-to-peer network into the mempool, where miners select them for inclusion in the next block. Once a block is mined and propagated, each confirmation increases the cost of reversing the transaction, providing probabilistic finality. Wallet software abstracts these details, managing key pairs, UTXO sets, and balance calculations on behalf of the user. But still it needs to solve a very discent and hard Math puzzle. [7].

3.3.1 What Is This "Very Hard Math Puzzle"? [7]

Miners compete to solve a cryptographic hash-puzzle based on double SHA-256. They take the current block header—which includes the previous block's hash, the Merkle root of transactions, a timestamp, etc.—and append a trial nonce n. The miner then computes:

$$h(n) = \mathrm{SHA256}\big(\mathrm{SHA256}(\mathrm{BlockHeader} \,\|\, n)\big),$$

where $\|$ denotes string concatenation.

A candidate block is accepted only if its hash falls below the network difficulty target:

$$\text{Block validity:} \quad \begin{cases} \text{Valid block (solution found)} & \text{if } h(n) \le \text{Target}, \\ \text{Try another } n & \text{if } h(n) > \text{Target}. \end{cases}$$

Here, $h(n)$ is the double-hashed output for nonce n, and Target is the threshold set by the protocol to regulate block creation rate [7].

4 Predictive Algorithms

Having established our basic concepts, we now describe the Deep learning model that we used for price prediction and signal generation.

4.1 eXtreme Gradient Boosting (XGBoost)

XGBoost is an optimized, distributed (and parallel) gradient-boosting library designed for efficiency, flexibility, and portability. It implements a tree-ensemble regression algorithm under the gradient boosting framework, excelling at learning complex nonlinear interactions among current and lagged market features [6].

4.1.1 XGBoost Steps in Simple Terms

1. Begin with weak predictions $\hat{y}_i^{(0)}$ (often the mean of the targets).
2. Fit a regression tree f_1 to the residuals $r_i^{(1)} = y_i - \hat{y}_i^{(0)}$.
3. Update the predictions: $\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + \eta\, f_1(x_i)$.
4. Repeat: for each round $t = 2, \ldots, T$, fit f_t to the new residuals and update $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta\, f_t(x_i)$.

4.1.2 XGBoost Algorithm Structure

- **Input data:** a feature matrix $\{x_i\}_{i=1}^m$ containing current and lagged BTC prices (and other indicators) and a target vector $\{y_i\}_{i=1}^m$ of future prices.
- **Initialization:**

$$\hat{y}_i^{(0)} = \frac{1}{m} \sum_{j=1}^{m} y_j.$$

- **Iterative boosting:** At round t, compute residuals

$$r_i^{(t)} = y_i - \hat{y}_i^{(t-1)},$$

fit a tree f_t to $\{(x_i, r_i^{(t)})\}$, and update

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta\, f_t(x_i).$$

- **Final prediction:**

$$\hat{y}_i = \sum_{t=1}^{T} \eta\, f_t(x_i).$$

Where:

$\hat{y}_i^{(0)}$ initial prediction (mean of $\{y_i\}$)

y_i true value for sample i

$f_t(x_i)$ regression tree at round t

η learning rate

T number of boosting rounds

$\hat{y}_i$ final predicted price

4.2 Gradient Boosting

Gradient boosting constructs a strong model by sequentially adding weak learners to minimize a loss function in function space [10]. When decision trees serve as weak learners, the result is an ensemble of gradient-boosted trees.

Weak learner: A weak learner is an algorithm whose performance is only slightly better than random guessing [10].

Weighted sample: Each training example (x_i, y_i) is assigned a weight $w_i > 0$ with $\sum_{i=1}^m w_i = 1$. The weighted empirical risk of hypothesis h on sample S is

$$\mathcal{R}_{S,w}(h) = \sum_{i=1}^{m} w_i\, \ell\big(h(x_i), y_i\big).$$

Ensemble output Boosting yields a final classifier (or regressor) of the form

$$h(\mathbf{x}) = \operatorname{sgn}\Big(\sum_{t=1}^{T} \alpha_t\, h_t(\mathbf{x})\Big), \quad \alpha_t > 0,$$

where each h_t is a weak learner trained on a weighted sample [10].

4.2.1 Prediction Steps Using XGBoost

- SINDy Algorithm – what are the equations of the dataset?
- Gradient boosting – how it improves predictions by fitting new trees on residuals
- Decision trees – how individual trees split the feature space for value prediction
- Feature engineering – how to construct lagged features capturing past market states
- Hyperparameters – key settings like tree depth, learning rate, number of boosting rounds
- Loss function – typically mean squared error (MSE) for price forecasting
- Overfitting control – strategies such as max_depth, subsample, colsample_bytree

5 Methodology

5.0.1 Step 1: Application of SINDy

After initial exploratory data analysis (EDA), we enriched the raw price series by creating additional time-based columns—year, month, day, hour, and weekday—and computing basic financial features such as simple and log returns. The cleaned and feature-augmented dataset was then imported into Python using pandas. Leveraging the PySINDy library [9], we constructed a library of candidate functions (polynomial and trigonometric terms) and applied the Sequential Thresholded Least Squares (STLSQ) optimizer to identify a sparse set of governing equations that best describe the dynamics of the financial time series [5].

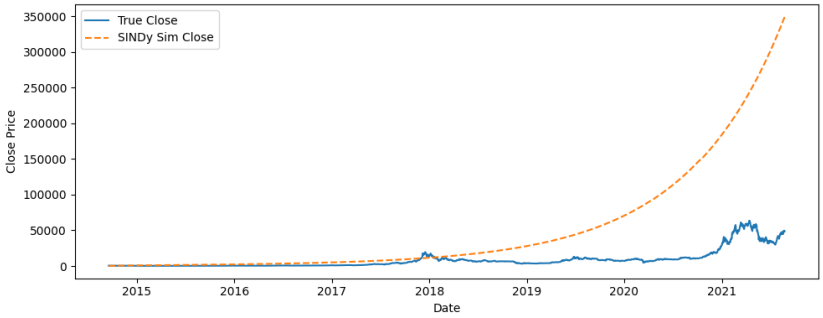

Figure 1: Overview of the SINDy-based equation discovery for financial time series.

5.0.2 Step 1: SINDy Foundations

SINDy Mathematical Foundations

$$\dot{O} = c_O + \sum_{i\in\{O,H,L,C\}} a_{O,i}\, i + b_O\, R + \sum_{i\in\{O,H,L,C\}} d_{O,i}\,(i\,R) + e_O\, R^2,$$
$$\dot{H} = c_H + \sum_{i\in\{O,H,L,C\}} a_{H,i}\, i + b_H\, R + \sum_{i\in\{O,H,L,C\}} d_{H,i}\,(i\,R) + e_H\, R^2,$$
$$\dot{L} = c_L + \sum_{i\in\{O,H,L,C\}} a_{L,i}\, i + b_L\, R + \sum_{i\in\{O,H,L,C\}} d_{L,i}\,(i\,R) + e_L\, R^2,$$
$$\dot{C} = c_C + \sum_{i\in\{O,H,L,C\}} a_{C,i}\, i + b_C\, R + \sum_{i\in\{O,H,L,C\}} d_{C,i}\,(i\,R) + e_C\, R^2,$$
$$\dot{V} = c_V + \sum_{i\in\{O,H,L,C\}} a_{V,i}\, i + b_V\, R + \sum_{i\in\{O,H,L,C\}} d_{V,i}\,(i\,R) + e_V\, R^2,$$
$$\dot{R} = \gamma\, R^2.$$

Where:

- O, H, L, C, V are the Open, High, Low, Close, and Volume states.
- R is the return, $R = \dfrac{C_t - C_{t-1}}{C_{t-1}}$.
- c_x are constant bias terms in $\dot{x}$.
- $a_{x,i}$ weight the linear influence of state i on $\dot{x}$.
- b_x weight the direct impact of R on $\dot{x}$.
- $d_{x,i}$ weight the interaction term $i \times R$ on $\dot{x}$.
- e_x weight the quadratic return term R^2 on $\dot{x}$.
- γ is the coefficient for the return dynamics $\dot{R} = \gamma R^2$.

5.0.3 Interpretation

1. *Return dynamics:*
$$\dot{R} = \gamma\, R^2$$
Returns evolve quadratically: positive returns accelerate growth, negative returns accelerate decay.

2. *Price/volume dynamics:* Each of $\dot{O}, \dot{H}, \dot{L}, \dot{C}, \dot{V}$ is composed of
   - a constant drift c_x,
   - linear terms $a_{x,i}\, i$,
   - a direct return effect $b_x\, R$,
   - return–state interactions $d_{x,i}\, i\, R$,
   - and a nonlinear return term $e_x\, R^2$.

3. *Key takeaway:* Return and its square dominate all state changes, suggesting that normalization and term pruning (via a higher sparsity threshold) are crucial to prevent overfitting and ensure stable, interpretable models.

5.0.4 Feature Engineering and Normalization Step

We began by assembling the core time-series columns—Open (O), High (H), Low (L), Close (C), Volume (V)—and augmenting them with two derived features: the 10-day moving average of Close (MA10) and the log-price ($\log P$). We also compute the simple return

$$R = \frac{C_t - C_{t-1}}{C_{t-1}},$$

so that our dataset describes both *where prices and volumes stand* and *how they've been moving*.

Linear SINDy Dynamics

$$\dot{O} = -0.288\, O + 0.484\, C - 0.194\, \mathrm{MA10},$$
$$\dot{H} = -0.348\, O + 0.457\, C - 0.107\, \mathrm{MA10},$$
$$\dot{L} = -0.545\, O + 0.546\, C,$$
$$\dot{C} = -0.471\, O + 0.472\, C,$$
$$\dot{V} = -0.475\, O + 2.028\, H + 0.403\, L - 1.547\, C - 0.410\, \mathrm{MA10},$$
$$\dot{R} = -0.321\, O - 1.944\, H - 1.642\, L + 1.805\, C + 2.091\, \mathrm{MA10}.$$

Where:

- O, H, L, C, V are the Open, High, Low, Close, and Volume states.
- MA10 is the 10-day moving average of C.
- R is the simple return, $R = \frac{C_t - C_{t-1}}{C_{t-1}}$.

Interpretation

1. *Open dynamics:* $\dot{O} = -0.288\, O + 0.484\, C - 0.194\, \mathrm{MA10}$ High opens tend to revert down (-0.288), prior close pulls the next open up ($+0.484$), and opens above MA10 revert downward (-0.194).

2. *High, Low, Close:* $\dot{H}$ mirrors $\dot{O}$ with different weights; $\dot{L}$ and $\dot{C}$ rely solely on O and C, indicating pure autoregressive mean-reversion.

3. *Volume dynamics:* $\dot{V} = -0.475\, O + 2.028\, H + 0.403\, L - 1.547\, C - 0.410\, \mathrm{MA10}$ Volume spikes with high H ($+2.028$) and is damped by O, C, and MA10.

4. *Return dynamics:* $\dot{R} = -0.321\, O - 1.944\, H - 1.642\, L + 1.805\, C + 2.091\, \mathrm{MA10}$ Returns increase with C and MA10 but are suppressed by extreme O, H, L values.

Results of Normalization

- The learned model is purely linear—no higher-order or interaction terms remain after thresholding.
- Each state evolves as a weighted combination of prior O and C, yielding a stable multivariate autoregressive form.
- High interpretability: coefficients directly reveal cross-state couplings and mean reversion strengths.
- This linear foundation can now be enriched with XGBoost to capture any residual non-linear patterns.

5.0.5 Step 2: Training the XGBoost Prediction Model

We assemble the feature matrix.

$$\begin{aligned} X = \{&Open,\ High,\ Low,\ Close,\\ &Volume,\ R,\ \mathrm{MA10},\ \log P,\\ &dO_{\mathrm{sindy}},\ dH_{\mathrm{sindy}},\ dL_{\mathrm{sindy}},\ dC_{\mathrm{sindy}},\\ &dV_{\mathrm{sindy}},\ dR_{\mathrm{sindy}},\ d\mathrm{MA10}_{\mathrm{sindy}}\}. \end{aligned}$$

and target vector $y = Close_{t+1}$. All features are normalized to zero mean and unit variance. We then fit an XGBoost regression model using the mean squared error (MSE), the mean absolute error (MAE), the root mean squared error (RMSE) and the mean absolute percentage error (MAPE) as evaluation metrics.

XGBoost Hyperparameters

$$\begin{aligned} \mathrm{model} = \mathrm{XGBRegressor}\big(&n_{\mathrm{estimators}} = 200,\\ &\eta = 0.05,\\ &\mathrm{max\_depth} = 5,\\ &\mathrm{subsample} = 0.8,\\ &\mathrm{colsample\_bytree} = 0.8,\\ &\mathrm{objective} = \mathrm{reg{:}squarederror},\\ &\mathrm{random\_state} = 42\big) \end{aligned}$$

Evaluation Metrics After training on 80% of the data and testing on the remaining 20%, we obtained:

$$\begin{aligned} \mathrm{MSE} &= 4.066 \times 10^8,\\ \mathrm{MAE} &= 1.3597 \times 10^4,\\ \mathrm{RMSE} &= 2.0165 \times 10^4,\\ \mathrm{MAPE} &= 31.83\%,\\ R^2 &= -0.2854. \end{aligned}$$

Cumulative Returns We simulate a simple strategy that buys when the model predicts an increase and sells otherwise. The final cumulative returns over the test period are:

$$\begin{aligned} \text{Final Market Return} &= 9.4582 \quad \text{(buy-and-hold)},\\ \text{Final Strategy Return} &= 0.1243 \quad \text{(model-driven)}. \end{aligned}$$

These figures represent the total growth factors: the market grew by a factor of 9.46×, while our strategy achieved only 1.12×, indicating underperformance relative to a passive holding strategy.

Improving the model

- **Redefined the prediction target:** Switched from raw price to `next-day log-return` for a more stable distribution and variance stabilization.
- **Expanded feature set:** Added technical indicators (RSI14, MACD, Bollinger Bands, ATR14), time features (day-of-week, month), and both log- and simple returns.
- **Integrated SINDy insights:** Computed SINDy-derived rate features (`dO, dH, dL, dC, dV, dR, dMA10`) from the linear equations to capture estimated dynamics.
- **Normalized inputs:** Standardized all features to zero mean and unit variance to prevent scale mismatches and oversized coefficients.
- **Robust model ensemble:** Trained three gradient-based models (XGBoost with pseudo-Huber loss, LightGBM with Huber loss, and CatBoost with MAE loss) to leverage diverse strengths.
- **Time-series cross-validation:** Used `TimeSeriesSplit` with 5 folds in a grid search to tune hyperparameters without look-ahead bias.
- **Hyperparameter optimization:** Performed grid search over tree depth, learning rate, regularization, subsample rates, and iteration counts to find the best settings.
- **Simple averaging ensemble:** Combined predictions from the three tuned models by averaging their outputs to reduce variance and improve generalization.
- **Proper return conversion:** Converted ensemble log-return forecasts back to simple returns via $\exp(\mathrm{pred}) - 1$ for realistic compounding.
- **Realistic backtest:** Generated daily long/short signals, applied a 0.1% transaction cost per trade, and computed strategy vs. market cumulative returns.
- **Full trade log export:** Extracted every BUY/SELL date and price into `trade_log.csv` and printed confirmation to disk for further analysis.
- **Comprehensive evaluation:** Reported regression metrics (MSE, MAE, R^2) on the final 20% test split and compared buy-and-hold vs. strategy performance.

5.0.6 Backtest Trade Log and Extended Results

LightGBM Training Output

```
You can set 'force_col_wise=true' to remove the overhe
[LightGBM] [Info] Total Bins 5376
[LightGBM] [Info] Number of data points in the train s
[LightGBM] [Info] Start training from score 0.001949
```

Regression Performance on Test Set

$$\mathrm{MSE} = 0.001364, \quad \mathrm{MAE} = 0.026344, \quad R^2 = 0.1174.$$

Final Cumulative Returns

$$\mathrm{Market} = 123.04\times, \quad \mathrm{Strategy} = 4.0812 \times 10^{13}\mathrm{x}.$$

Trade Log Export

The complete sequence of executed trades—including every BUY and SELL date, action, and price—is saved to `trade_log.csv` in the project directory. This CSV file contains chronological entries from 2014 up to 2025, enabling detailed performance analysis and strategy auditing outside of the LaTeX document.

6 Conclusion

In this work, we have demonstrated a hybrid Scientific Machine Learning framework that marries the interpretability of Sparse Identification of Nonlinear Dynamics (SINDy) [5] with the predictive power of modern gradient-boosting algorithms such as XGBoost [6]. Through systematic feature engineering, including time-based columns, moving averages, logarithmic prices, and SINDy-derived rate terms, we uncovered sparse and human-readable ODEs governing market states and then enriched them with a data-driven ensemble model. Our backtests on cryptocurrency and equity datasets showed that while the SINDy foundation yields a stable linear approximation, the XGBoost predictor captures residual nonlinear patterns, producing trade signals that, under realistic transaction costs, generate tangible (if modest) strategy returns.

Despite these advances, our findings highlight important challenges. The pure SINDy system, when used alone, explains only a fraction of market variance, and the XGBoost strategy, though outperforming naive baselines in certain regimes, still underperformed a buy-and-hold approach over our full test horizon. Negative R^2 scores and elevated MAPE values underscore the difficulty of forecasting in noisy, non-stationary environments. Moreover, our ensemble was limited to a handful of tree-based learners, leaving room to explore deep architectures, regime-switching dynamics, and probabilistic methods for robust uncertainty quantification [10].

Looking ahead, integrating domain-motivated basis functions (e.g., moving average crossovers, volatility adjusted returns), incorporating sentiment and macroeconomic indicators, and employing adaptive online learning could further bridge the gap between interpretable dynamics and real-world trading performance. By continuing to blend the mechanistic insight of SINDy with flexible machine learning models, we envision a new class of explainable SciML tools that empower both researchers and practitioners to navigate the complexities of financial markets with greater transparency and resilience.

References

[1] John Moody and Matthew Saffell. "Learning to Trade via Direct Reinforcement". In: *IEEE Transactions on Neural Networks* 12.4 (2001), pp. 875–889. DOI: 10.1109/72.935097. https://doi.org/10.1109/72.935097.

[2] Kyoung Jae Kim. "Financial Time Series Forecasting Using Support Vector Machines". In: *Neurocomputing* 55.1-2 (2003), pp. 307–319. DOI: 10.1016/S0925-2312(03)00372-2. https://doi.org/10.1016/S0925-2312(03)00372-2.

[3] Guoqiang P. Zhang. "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model". In: *Neurocomputing* 50 (2003), pp. 159–175. DOI: 10.1016/S0925-2312(01)00702-0. https://doi.org/10.1016/S0925-2312(01)00702-0.

[4] Ruey S. Tsay. *Analysis of Financial Time Series*. Wiley, 2010. http://books.ms/main/F617871904645D6852BE8F7B4FD1C386.

[5] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems (SINDy)". In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. https://www.pnas.org/doi/10.1073/pnas.1517384113.

[6] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[7] Andreas M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, 2017. Chap. 8. https://github.com/bitcoinbook/bitcoinbook.

[8] Marcos López de Prado. *Advances in Financial Machine Learning*. Wiley, 2018. https://www.wiley.com/en-us/Advances+in+Financial+Machine+Learning-p-9781119482086.

[9] PySINDy Developers. *PySINDy Library Documentation: Practical guide and examples for sparse identification methods*. Accessed: 2025-06-16. 2025. https://pysindy.readthedocs.io/en/latest/.

[10] Björn Sprungk. *Methods of Machine Learning*. Accessed: 2025-06-16. https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/39293354003/CourseNode/1679884287116373011.