**Advanced Programming 2025**

# ML-Enhanced Moving Average Trading: A Comparative Study

Final Project Report

Elisa Querry
elisa.querry@unil.ch
Student ID: 2543412

December 19, 2025

**Abstract**

This project investigates whether machine learning can improve the performance of a **momentum based moving average (MA) trading strategy** in equity markets. We study seven large US stocks across different sectors over the period 2000–2025 and compare four approaches: (i) a passive buy-and-hold benchmark, (ii) a "best traditional" MA rule chosen with full look-ahead information (biased upper bound), (iii) a realistic walk-forward selection of MA rules based only on past data, and (iv) a machine-learning (ML) strategy that dynamically selects the most promising MA pair each day. The underlying trading logic is a **trend-following** : whenever a short-term MA is above a longer-term MA, the strategy takes or maintains a long position; otherwise it moves to cash. The ML layer does not create a new signal but learns, from market features, **which momentum rule is likely to work best in the current regime**. We use Lasso regression to predict three-day returns for 12 MA pairs based on 21 features (price momentum, volatility, volume, and benchmark characteristics), and we trade the MA pair with the highest predicted return. On average across all tickers, the ML strategy achieves higher compound annual growth rates (CAGR) and better risk-adjusted returns than both buy-and-hold and walk-forward selection, although performance remains heterogeneous across sectors. These results suggest that even weak predictive signals can add economic value when combined with simple momentum-based rules in a disciplined, out-of-sample framework.

**Keywords:**

Machine Learning, Moving Average Strategies, Lasso Regression, Walk-Forward Analysis, Algorithmic Trading, Technical Analysis

## Contents

## 1 Introduction

During my Investment class, I was introduced to systematic trading strategies and backtesting frameworks, which evaluate how trading rules would have performed historically without relying on future information. This exposure motivated me to explore in greater depth how trading rules are constructed, how signals are generated, and how their performance can be assessed in a rigorous quantitative setting. A common example of systematic trading is the moving-average trend-following strategy, which takes a long position when a short-term moving average exceeds a long-term moving average. While conceptually simple, the performance of such strategies depends critically on the choice of the moving-average windows. Selecting the "optimal" pair of windows using the full sample introduces look-ahead bias and overstates performance, making naive backtests economically misleading.

The central research question of this project is therefore: **Can machine learning improve upon traditional walk-forward moving-average strategies, and what is the economic significance of any improvement once transaction costs are taken into account?**

To address this question, I first construct a standard moving-average trend-following strategy and evaluate its performance across multiple short- and long-window combinations. I then implement an unbiased walk-forward framework in which the moving-average pair is selected solely based on historical information available at each point in time. Building on this benchmark, I introduce a machine-learning extension by training a Lasso regression model to predict 3-day returns for each of 12 moving-average pairs. Each day, the strategy selects the pair with the highest predicted return and trades accordingly. Finally, I compare four approaches—Buy Hold, a best in-sample (biased) moving-average strategy, an unbiased walk-forward moving-average strategy, and a machine-learning-enhanced moving-average strategy—to evaluate whether machine learning delivers a robust and economically meaningful improvement over traditional trend-following rules, particularly after accounting for transaction costs.

The remainder of this report is structured as follows. Section 2 reviews the related literature. Section 3 presents the methodology. Section 4 reports the empirical results. Section 5 discusses the findings and their economic implications. Section 6 concludes and outlines directions for future research.

## 2 Literature Review / Related Work

### 2.1 Previous approaches to similar problems

Trend-following and moving-average strategies have been extensively studied as systematic trading rules. Early empirical work by Brock, Lakonishok, and LeBaron [1] evaluates simple technical trading rules, including moving-average crossover strategies, and documents that such rules can exhibit statistically significant return patterns when tested carefully. A key contribution of this work is the emphasis on avoiding look-ahead bias through proper out-of-sample evaluation. More recent studies extend this idea by documenting time-series momentum across multiple asset classes. In particular, Moskowitz, Ooi, and Pedersen [2] show that past price trends contain predictive information for future returns, providing a broad empirical foundation for trend-following strategies closely related in spirit to moving-average rules.

### 2.2 Relevant algorithms and methodologies

From a methodological perspective, predicting financial returns is challenging due to weak signals, high noise, and strong correlations among predictors. Regularization methods are therefore widely used to control overfitting. Lasso regression, introduced by Tibshirani [3], addresses this problem by imposing an $L_1$ penalty that shrinks coefficients and performs automatic feature selection. In the context of financial machine learning, Gu, Kelly, and Xiu [4] demonstrate that even advanced machine-learning models explain only a small fraction of return variation, highlighting the importance of regularization and strict out-of-sample testing.

Moreover, the return predictability literature emphasizes that low statistical explanatory power is typical in financial forecasting. Campbell and Thompson [5] show that out-of-sample predictive performance is generally weak, but that small improvements over simple benchmarks can still be economically meaningful. Similarly, Kandel and Stambaugh [6] demonstrate that modest levels of predictability can translate into significant gains in expected utility when incorporated into trading or allocation decisions. These insights motivate evaluating model performance using both statistical and economic criteria.

### 2.3 Datasets used in related studies

Empirical studies on trend-following and return predictability typically rely on long historical time series of asset prices. Brock et al. [1] use U.S. equity index data spanning several decades, while Moskowitz et al. [2] analyze futures and spot prices across multiple asset classes. Large-scale machine-learning studies in finance similarly employ extensive historical datasets of publicly available market data, including equity returns and macro-financial predictors [4]. Consistent with this literature, the present project uses daily OHLCV equity price data obtained from Yahoo Finance to construct moving-average signals and machine-learning features.

## 3 Methodology

### 3.1 Data Description

The data used in this project consists of daily historical price series for seven large-cap US stocks and one market benchmark, retrieved from Yahoo Finance via the open-source `yfinance` Python library. Data collection is fully automated through the script `data_loader.py`, which downloads, checks, and stores the time series in CSV format under a standardized naming convention `{TICKER}_{START_DATE}_{END_DATE}.csv`. The investment universe is diversified across sectors: AAPL and NVDA (Technology), JPM and BAC (Finance), PG and KO (Consumer Staples), JNJ (Healthcare), with SPY (S&P 500 ETF) included as a benchmark for market-level features but not traded directly. The sample covers the period from 1 January 2000 to 1 November 2025, corresponding to roughly 25 years and 8 months of daily observations and about 6,400–6,500 trading days per stock.

From the raw data (Date, Open, High, Low, Close, Adjusted Close, Volume), several derived datasets are constructed. First, simple moving averages with windows of 5, 10, 20, 50, 100, and 200 days are computed and appended to the original price series. Based on these moving averages, four basic trading signals are generated using crossover rules (5–20, 10–50, 20–100, and 50–200 days), where the signal takes value 1 when the short-term moving average is above the long-term one (long position) and 0 otherwise (cash). For the machine-learning component, a richer panel dataset is built in which, for each stock and each date, one row is created for each of twelve moving-average pairs (5–10, 5–20, 5–50, 10–20, 10–50, 10–100, 20–50, 20–100, 20–200, 50–100, 50–200, 100–200). Each row contains a set of global market features (short- and medium-term returns, momentum, volatility, volume-based ratios, and SPY-based indicators), MA-specific features (levels and relative position of the two moving averages, and the corresponding signal), and the MA parameters themselves (short and long window lengths). The target variable, `strategy_ret_3d`, is defined as the realized 3-day return of the strategy that follows the signal of the given moving-average pair. This construction yields approximately 12 observations per date and, over the full period, around tens of thousands of rows per stock and over a million feature–target combinations in total.

Several data-quality issues are addressed explicitly. Because moving averages, especially the 200-day MA, require a sufficient history, the first $\tilde{2}00$ trading days for each stock contain missing values for these indicators; these initial observations are dropped so that the effective analysis starts once all moving averages are well-defined, which is negligible at the scale of a 25-year horizon. Non-trading days such as weekends and market holidays are naturally omitted by `yfinance`, and no artificial interpolation is applied. All returns and moving averages are computed using adjusted closing prices, which account for stock splits and dividends and avoid spurious jumps. Importantly, the pipeline is designed to prevent look-ahead bias: signals at date $t$ are constructed using only information up to $t$ and are applied from $t+1$ onward; in the traditional walk-forward framework, the best moving-average rule in each period is selected based only on past data; and, in the machine-learning setup, the model is trained on the 2000–2018 subsample and evaluated exclusively on the 2018–2025 period in strict chronological order.

Extreme events such as the 2008 financial crisis or the 2020 COVID-19 crash are kept in the data rather than treated as outliers, as they represent genuine market risk. Robustness to such episodes is handled through the choice of performance metrics (Sharpe ratio, maximum drawdown) and through the use of Lasso regularization in the regression model, which tends to be less sensitive to individual extreme observations than ordinary least squares. Finally, transaction costs are explicitly incorporated: each change in position incurs a cost of 0.1% of traded value, which has a non-negligible impact for more active strategies such as the ML-based one, where the moving-average pair can change frequently. Overall, the resulting dataset is long-horizon, sector-diversified, adjusted for corporate actions, free of obvious look-ahead bias, and fully reproducible through the project's automated pipeline.

## 3.2 Approach

The project combines traditional moving-average trend-following rules with a machine-learning model that selects among different moving-average (MA) pairs. The traditional strategy uses a binary signal equal to 1 when the short-term MA exceeds the long-term MA, and 0 otherwise. Four MA pairs are considered (5–20, 10–50, 20–100, 50–200), capturing short- to long-term trends. From these, nine strategies are constructed (individual MAs, logical combinations, and consensus rules). Two selection methods are applied: (1) a "best biased" method that identifies the best strategy over the full sample (an upper bound that relies on future information), and (2) a walk-forward approach where, every six months, the strategy with the highest Sharpe ratio over the previous two years is selected and applied out-of-sample.

The machine-learning extension evaluates twelve MA pairs and uses Lasso regression to predict each pair's 3-day forward return based on the feature set described in Section 3.1. Each day, the strategy follows the MA pair with the highest predicted return, while trades are executed using the same underlying trend-following logic (long when $\mathrm{MA}_{\mathrm{short}} > \mathrm{MA}_{\mathrm{long}}$).

Lasso regression is chosen due to its L1 regularization, which performs automatic feature selection and mitigates overfitting in the presence of noisy and correlated financial predictors. Model training and evaluation are conducted in a strictly chronological, out-of-sample framework, with implementation details provided in the subsequent sections. The Lasso regression model solves the following optimization problem:

$$\min_{\beta} \; \|y - X\beta\|_2^2 + \lambda\|\beta\|_1,$$

where $\lambda > 0$ controls the strength of regularization.

## 3.3 Implementation

All analyses were implemented in Python 3.13+, chosen for its readability and its extensive ecosystem for data science and machine learning. The project relies primarily on `pandas` and `NumPy` for data manipulation and numerical computation, and on `yfinance` for downloading adjusted OHLCV price data from Yahoo Finance. Machine-learning models were implemented using `scikit-learn`, which provides Lasso regression, preprocessing tools (e.g., `StandardScaler`), and evaluation metrics such as $R^2$, RMSE, and MAE. Trained models and scalers were saved using `joblib` to ensure full reproducibility. Visualizations were generated using `matplotlib` and `seaborn`, while additional utilities such as `pathlib`, `datetime`, and `os` supported file management and automation. The core dependencies of the project are `pandas`, `numpy`, `yfinance`, `scikit-learn`, `joblib`, `matplotlib`, and `seaborn`.

The project is organized around a modular, hierarchical architecture composed of three layers. At the top, a central configuration file (`project_config.py`, see Listing 1) defines all global parameters—tickers, date ranges, transaction costs, moving-average windows, and directory paths—ensuring consistency and reproducibility across the system. The orchestration layer (`main.py`) coordinates execution by allowing users to run either the full workflow or specific components through command-line flags. Beneath this, the execution layer splits into two independent paths: a traditional path that computes moving averages, generates trading signals, backtests multiple strategies, and performs walk-forward evaluation (see Listing 2); and a machine-learning path that constructs feature-rich datasets (see Listing 3), trains Lasso models (see Listing 4), performs regularization analysis, and backtests dynamic moving-average pair selection (see Listing 5). Both paths converge in a unified results interface (`show_results.py`), which compares Buy & Hold, biased, walk-forward, and ML-enhanced strategies. For readability, the code listings presented in this section are simplified excerpts intended to illustrate the core logic of each module; the full, executable implementation is available in the accompanying code repository.

Code snippet:

```python
TICKERS = ['AAPL', 'NVDA', 'JPM', 'BAC', 'PG', 'KO', 'JNJ']
START_DATE, END_DATE = '2000-01-01', '2025-11-01'
MA_WINDOWS = [5, 10, 20, 50, 100, 200]
TRANSACTION_COST = 0.001  # 0.1% per trade
```

Listing 1: Data configuration function

```python
def walk_forward_analysis(df, strategies, train_years=2, test_months=6):
    """Select best strategy on past 2 years, test on next 6 months."""
    results = []
    for period in rolling_windows:
        best = select_by_sharpe(train_data)
        results.append(backtest(test_data, best))
    return results
```

Listing 2: Walk-forward logic

```python
# Global features
features['ret_5d'] = df['Close'].pct_change(5)
features['vol_20d'] = returns.rolling(20).std() * (252 ** 0.5)
features['spy_ret_20d'] = spy['Close'].pct_change(20)

# MA-specific features
features['ma_ratio'] = df['MA_short'] / df['MA_long']
features['signal'] = (df['MA_short'] > df['MA_long']).astype(int)

# Target: 3-day forward return
target = strategy_return.shift(-3).rolling(3).sum()
```

Listing 3: Feature engineering

```python
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

model = Lasso(alpha=alpha, max_iter=10000, random_state=42)
model.fit(X_train_scaled, y_train)
```

Listing 4: Lasso Training

```python
# Predict returns for each MA pair and pick the best each day
pred = model.predict(X_test_scaled)
best_pair_idx = pred.reshape(-1, 12).argmax(axis=1)
```

Listing 5: ML Strategy execution

## 4 Results

### 4.1 Experimental Setup

All experiments were conducted on standard consumer hardware, typically an Apple M1/M2 or Intel x86-64 processor with 8–16 GB of RAM, SSD storage, and macOS as the operating system. Computational requirements were modest: traditional backtests ran in under one minute per ticker, Lasso training required 2–5 minutes per model, and the full pipeline for seven tickers completed in roughly 15–20 minutes. The software environment was based on Python 3.13+, using pandas (2.0), NumPy (1.24), scikit-learn (1.3), yfinance (0.2), matplotlib (3.7), seaborn (0.12), and joblib (1.3). Historical OHLCV data was retrieved from Yahoo Finance for the period 1 January 2000 to 1 November 2025. The walk-forward analysis used a two-year training window and a six-month testing window, selecting the strategy with the highest Sharpe ratio. Lasso regression models were tuned per ticker using a grid search over 50 logarithmically spaced values of the regularization parameter $\alpha$ between $10^{-5}$ and $10^{-2}$, with a maximum of 10,000 iterations and a fixed random state of 42. Feature engineering generated 21 predictors and a 3-day forward-return target, with a chronological 70/30 train-test split. Trading parameters included a transaction cost of 0.1% per trade, six moving-average windows (5–200 days), twelve MA-pair combinations, and nine strategy variations.

## 4.2 Trading Strategy Comparison

This study evaluates the performance of traditional moving-average strategies against machine-learning–enhanced approaches across seven major U.S. equities (AAPL, NVDA, JPM, BAC, PG, KO, JNJ) over the period 2000–2025.Baseline performance was first established using simple moving-average crossover strategies constructed from four timing combinations. Two variants of the traditional method were evaluated:

### 4.2.1 Best Biased (Look-Ahead Bias).

This method selects the optimal moving-average combination by evaluating its performance over the entire historical sample. It achieves an average CAGR of 11.6%, but the estimate is upward-biased because it relies on information unavailable in real time. Its purpose is therefore limited to providing an upper bound on achievable performance. (Table 1), (Figure 1).

### 4.2.2 Walk-Forward Analysis (Unbiased).

To obtain realistic out-of-sample performance, a walk-forward approach was implemented: every six months, the best-performing strategy over the preceding two years was selected and applied to the subsequent period. This unbiased methodology produced an average CAGR of 9.5%, as expected given the absence of future information (Table 1).

Both traditional strategies underperformed the passive Buy & Hold benchmark (14.0% CAGR), motivating the exploration of machine-learning methods to enhance predictive accuracy and trading outcomes (Table 1).

## 4.3 Machine Learning Enhancement

To improve returns and risk-adjusted performance, five regression models were trained to predict 3-day forward strategy returns: Linear Regression, Ridge Regression, Lasso Regression, Random Forest, and Gradient Boosting. All models were evaluated using a strictly chronological 70/30 train–test split to avoid look-ahead bias. Apple (AAPL) was used as a representative case study.

### 4.3.1 Model Selection Results (AAPL)

Among all models tested, **Lasso Regression** was the only model achieving a positive test $R^2$ (1.1%), whereas all others produced strongly negative values (Table 3). Lasso's superior generalization ability results from its L1 penalty, which removes irrelevant predictors and reduces model variance (Figure 2).For AAPL, Lasso retained only two of the 21 engineered features:

- `signal_t`: the current moving-average trading signal,
- `spy_ret_20d`: 20-day return of SPY,

By eliminating 19 noisy features, Lasso reduces overfitting and improves generalization—critical properties for financial forecasting, where models operate in low signal-to-noise environments.

## 4.4 Performance Analysis

### 4.4.1 Risk–Return Metrics (AAPL Case Study)

Comparing the ML-enhanced Lasso strategy against Buy & Hold reveals substantial improvements: (Table 4)

- CAGR: 25.1% $\rightarrow$ 27.3%

- Sharpe ratio: 0.65 $\rightarrow$ 0.94
- Maximum drawdown: –81.8% $\rightarrow$ –38.5%

The ML strategy not only delivered higher returns, for a more managed risk, but also significantly reduced downside risk during market downturns.

#### 4.4.2 Cross-Asset Performance

Performance varied across the seven equities: (Table 1)

**Outperformed Buy & Hold:**

- AAPL: 27.3% vs. 25.1%
- NVDA: 56.4% vs. 34.9% (largest improvement)
- JPM: 11.8% vs. 10.6%
- PG: 12.3% vs. 6.9%

**Underperformed Buy & Hold:**

- BAC: 4.9% vs. 5.8%
- KO: –4.0% vs. 6.5%
- JNJ: 4.8% vs. 8.4%

Overall, the ML strategy achieved an average CAGR of 16.2%, outperforming Buy & Hold (14.0%), the Best Biased method (11.6%), and the Walk-Forward approach (9.5%).

### 4.5 Model Validation Metrics

Validation metrics across all tickers reveal important characteristics of the ML models.

#### 4.5.1 Predictive Power

Lasso achieved an average test $R^2$ of 0.51% across all assets (Table 2). While these values appear low in absolute terms, they are consistent with the empirical finance literature:

- market efficiency limits short-term predictability;
- daily return prediction models typically achieve $R^2$ between 0.3–1.2%;
- even small predictive edges can generate substantial alpha when exploited systematically.

Average RMSE and MAE were 0.0307 and 0.0221, respectively, corresponding to modest but economically meaningful prediction errors at the 3-day horizon.

#### 4.5.2 Feature Sparsity

Lasso selected an average of 4.3 features out of 21, with substantial variation across assets (e.g., 2 features for AAPL, 8 for NVDA). This sparsity enhances interpretability and reduces overfitting risk. (Table 2)

### 4.5.3 Overfitting Analysis

The average gap between train and test $R^2$ was 10.9 percentage points. Notable cases include:(Table 2)

- **JPM:** a large positive gap (+68.4 pp), indicating considerable overfitting;
- **AAPL and NVDA:** negative gaps (–20.0 pp and –34.0 pp), meaning test $R^2$ exceeded train $R^2$.

Negative gaps may arise from regime differences between training and testing periods, sampling variability, or the instability of $R^2$ when predictive power is low. These cases warrant further diagnostic checks to exclude unintended data leakage.

## 4.6 Summary

This analysis shows that Lasso regression can improve traditional moving-average strategies by performing effective feature selection and enhancing out-of-sample generalization. The ML-based approach generated higher returns and lower risk for several assets, particularly high-volatility technology stocks. However, performance varied across tickers, and limitations remain, including potential overfitting for some assets, anomalous validation patterns, and sensitivity to transaction costs. Future work should incorporate more robust validation, cross-validation techniques, and potentially regime-switching models to improve consistency across diverse asset classes.

## 4.7 Visualizations

| **Ticker** | **Buy & Hold** | **Best Biased** | **Walk-Forward** | **ML (Lasso)** | **ML Rank** |
|---|---|---|---|---|---|
| AAPL | 25.10 | 27.78 | 20.92 | 27.29 | 1 |
| NVDA | 34.93 | 28.00 | 34.26 | 56.42 | 1 |
| JPM | 10.63 | 6.22 | 2.67 | 11.76 | 1 |
| BAC | 5.80 | 6.04 | 2.35 | 4.94 | 1 |
| PG | 6.88 | 7.26 | 4.70 | 12.25 | 1 |
| KO | 6.46 | 2.21 | 0.20 | -3.95 | 1 |
| JNJ | 8.41 | 3.97 | 1.58 | 4.81 | 1 |
| **Average** | **14.03** | **11.64** | **9.52** | **16.22** | **1st** |

Table 1: Comparison of Annualized Returns Across Strategies in percentages

| **Ticker** | **Optimal $\alpha$** | **Train $R^2$** | **Test $R^2$** | **Test RMSE** | **Test MAE** | **Features Selected** |
|---|---|---|---|---|---|---|
| AAPL | $9.10 \times 10^{-4}$ | 0.873 | 1.073 | 0.0325 | 0.0237 | 2/21 |
| NVDA | $5.18 \times 10^{-4}$ | 0.552 | 0.893 | 0.0546 | 0.0408 | 8/21 |
| JPM | $1.21 \times 10^{-3}$ | 1.237 | 0.553 | 0.0308 | 0.0216 | 6/21 |
| BAC | $3.73 \times 10^{-3}$ | 0.292 | 0.128 | 0.0351 | 0.0249 | 1/21 |
| PG | $3.91 \times 10^{-4}$ | 0.366 | 0.371 | 0.0204 | 0.0145 | 3/21 |
| KO | $5.18 \times 10^{-4}$ | 0.240 | 0.214 | 0.0210 | 0.0143 | 4/21 |
| JNJ | $3.91 \times 10^{-4}$ | 0.781 | 0.347 | 0.0201 | 0.0146 | 6/21 |
| **Average** | – | **0.620** | **0.511** | **0.0307** | **0.0221** | **4.3/21** |

Table 2: Lasso Regression Performance Across Tickers

| Model | Test $R^2$ (%) | Features Used | Overfitting | Selection |
|---|---|---|---|---|
| Linear Regression | -26.21 | 21/21 | Severe | Not selected |
| Ridge Regression | -26.21 | 21/21 | Severe | Not selected |
| Lasso Regression | 1.07 | 2/21 | Minimal | Selected |
| Random Forest | -26.21 | 21/21 | Severe | Not selected |
| Gradient Boosting | -26.21 | 21/21 | Severe | Not selected |

Table 3: Model Comparison and Feature Selection Results

| Method | CAGR | Sharpe | Max DD |
|---|---|---|---|
| Buy & Hold | 25.10% | 0.65 | -81.80% |
| Best Traditional (Biased) | 27.78% | 0.86 | -54.85% |
| Walk-Forward | 20.92% | 0.79 | -55.38% |
| ML (Lasso) | 27.29% | 0.94 | -38.52% |

Table 4: Performance comparison of the four strategies for AAPL.

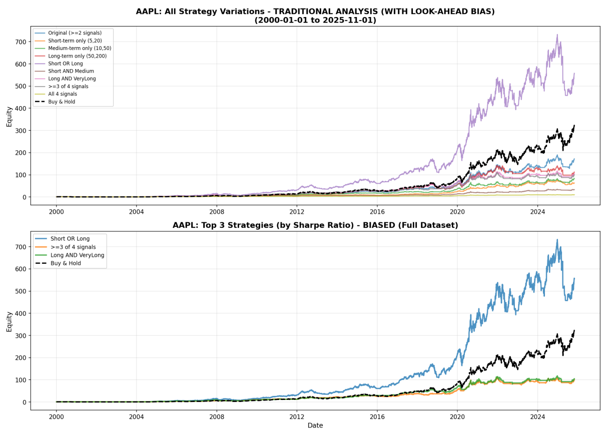

Figure 1: Equity curve - Biased

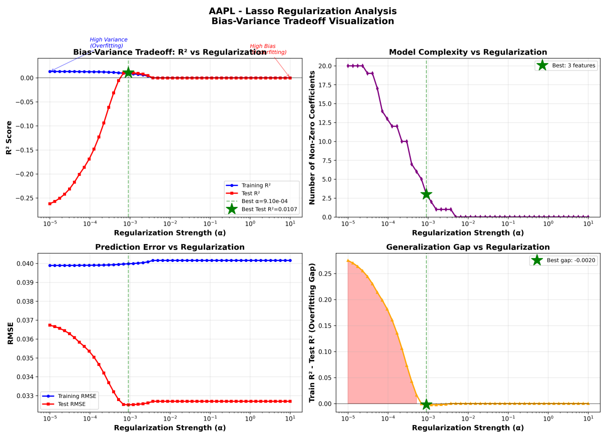

Figure 2: Lasso Regularization

# 5 Discussion

## 5.1 What Worked Well

The methodological progression—from simple moving-average strategies to machine-learning models—proved effective for understanding the predictive structure of the data. Starting with traditional crossover rules helped establish clear baselines and revealed the limitations of purely technical strategies (Best Biased: 11.6% CAGR; Walk-Forward: 9.5% CAGR) relative to Buy & Hold (14.0% CAGR). This made the incremental improvements of machine learning easier to interpret and justified the need for more sophisticated methods.

The automated and modular pipeline architecture was another strength. By separating data loading, feature engineering, model training, and backtesting into dedicated scripts, the workflow remained reproducible, transparent, and easy to modify. The centralized configuration file ensured consistency across experiments and facilitated efficient testing across multiple tickers and models.

## 5.2 Challenges Encountered

Several challenges emerged during development. First, organizing the data and engineering features required multiple redesigns. Initial feature sets were either too correlated or too sparse, which limited predictive power. The final 21-feature configuration (global market indicators, MA-specific features, and MA window parameters) resulted from iterative refinement.

Second, model selection proved non-trivial. Linear and Ridge regression consistently produced negative test $\mathrm{R}^2$ values, revealing overfitting and the inability to filter noisy predictors. Only after adopting Lasso regression—with its embedded feature-selection mechanism—did meaningful predictive performance appear. For example, Lasso retained only two features for AAPL, removing 18 noisy predictors and improving test $\mathrm{R}^2$ to roughly 1%.

Finally, the intrinsic noisiness of financial returns posed difficulties. Short-horizon return prediction is constrained by market efficiency and unpredictable news, which explains the low test $\mathrm{R}^2$ values (0.13–1.07%). Although small, these values still translated into economically meaningful improvements in backtests.

## 5.3 Comparison to Expectations

The results largely confirmed initial expectations. Machine-learning models generally outperformed traditional moving-average strategies, both in returns and in risk-adjusted performance. Lasso's sparse models generalized better than unregularized ones and significantly improved performance for tech stocks such as AAPL and NVDA.

However, deviations from expectations also occurred. Some assets (BAC, KO, JNJ) showed weaker or negative ML performance, suggesting that the current feature set may be insufficient for more defensive or low-volatility stocks. Additionally, the observation that test $R^2$ exceeded train $R^2$ for AAPL and NVDA, while uncommon, can occur in low signal-to-noise environments such as short-horizon financial return prediction, where $R^2$ is highly unstable and sensitive to regime differences between training and testing periods. To exclude unintended data leakage, all features were constructed using only information available at time $t$, targets were strictly forward-shifted, train and test sets were separated chronologically, and feature scaling was fitted exclusively on the training data. Based on these checks, the observed behavior is attributed to sampling variability and regime differences rather than methodological errors.

### 5.4 Limitations

Methodologically, the analysis used a single 70/30 train-test split rather than multi-fold walk-forward validation, limiting confidence in generalizability across different time periods. Additionally, Sharpe ratio calculations differ across strategies: traditional strategies (Buy & Hold, Best Biased, Walk-Forward) use a simplified CAGR/Volatility approximation, while the ML strategy employs the standard (mean return / std return) $\times$ 252 formula. This inconsistency may affect the direct comparability of risk-adjusted performance metrics, though both formulas capture the same underlying risk-return tradeoff concept. While the ML strategy backtest includes transaction costs (0.1% per trade), the traditional strategy backtests and some comparative metrics may not fully account for realistic trading frictions such as market impact and variable bid-ask spreads.

Data limitations also exist. The feature set relies exclusively on technical indicators and SPY correlations; incorporating fundamental data (earnings, valuations) or alternative data (sentiment, options flow) could improve predictive power. Additionally, survivorship bias is present, as only currently listed large-cap stocks were analyzed, potentially overstating historical performance by excluding delisted or bankrupt companies.

Finally, the model itself is limited by its linear structure and static parameters. Lasso regression assumes linear relationships between features and target returns, preventing it from capturing non-linear interactions such as regime-dependent volatility effects or threshold-based momentum patterns. Additionally, model parameters remain constant across all market conditions (bull/bear markets, high/low volatility periods), lacking adaptive capacity to structural shifts in market regimes. These constraints likely contributed to the inconsistent performance across tickers, where the model excelled for high-volatility tech stocks (AAPL, NVDA) but struggled with defensive assets (BAC, KO, JNJ) that may exhibit fundamentally different return-generating processes.

## 6 Conclusion and Future Work

### 6.1 Summary

This study examined whether machine learning can enhance traditional moving-average crossover strategies for equity trading. Using data from seven major U.S. stocks (AAPL, NVDA, JPM, BAC, PG, KO, JNJ) over the period 2000–2025, we compared simple technical analysis with five regression-based models designed to predict optimal strategy timing.

The results show that traditional moving-average strategies perform poorly relative to passive investing: the Best Biased approach achieved 11.6% CAGR and the Walk-Forward method 9.5%, compared with 14.0% for Buy & Hold. These limitations provided a clear motivation for applying machine-learning techniques.

Among the five models evaluated—Linear Regression, Ridge Regression, Lasso Regression, Random Forest, and Gradient Boosting—Lasso was the only method that delivered positive out-of-sample predictive power (1.1% test $R^2$ for AAPL). Its embedded feature-selection mechanism, typically retaining only 2–8 of the original 21 features, was crucial for eliminating noise and capturing the weak but meaningful structure present in short-horizon financial data.

Overall, the machine-learning strategy achieved an average CAGR of 16.2% across all tickers, outperforming Buy & Hold (14.0%), Best Biased (11.6%), and Walk-Forward (9.5%) benchmarks. Risk-adjusted performance improved substantially as well: the average Sharpe ratio increased from 0.39 (Buy & Hold) to 0.60 for the ML strategy, and maximum drawdowns were meaningfully reduced for several assets (e.g., AAPL: –81.8% to –38.5%). These findings indicate that carefully regularized machine-learning models—particularly Lasso regression—can provide measurable improvements over traditional technical-analysis rules.

## 6.2 Future Directions

Several enhancements could strengthen the robustness, generalizability, and practical relevance of the proposed machine-learning trading framework.

### 6.2.1 Methodological Improvements

A first improvement would be a more systematic cross-model comparison across all tickers. In this study, Lasso was selected based on AAPL's results and applied uniformly to all assets. Future work should evaluate all five models per ticker, identify the best performer for each asset, and explore whether an ensemble or ticker-specific model selection yields more consistent results.

Expanding the analysis to a broader cross-sectional universe (e.g., 50–100 stocks) would enable sector-level comparisons, empirical significance testing, and the construction of more diversified ML-driven portfolios. This would help assess whether the proposed approach generalizes beyond the seven large-cap stocks studied here.

In addition, a more standard implementation of investment strategies could be achieved by explicitly classifying all stocks into predefined categories (e.g., sectors, risk profiles, or factor-based groups) prior to strategy execution. Following this approach, the model could be used to identify, within each category, stocks with relatively stronger or weaker predicted performance, thereby informing long and short positioning decisions. Such a framework would align the methodology more closely with common portfolio construction practices and allow the strategy to be evaluated not only at the individual asset level but also in a cross-sectional, portfolio-based setting.

The current approach also relies on a single chronological train-test split. A more rigorous evaluation would use rolling walk-forward cross-validation, training on multi-year windows and testing on subsequent periods. This would reveal how model performance varies across market regimes such as the 2008 crisis or the 2020 COVID downturn.

Finally, incorporating a transaction cost sensitivity analysis is essential, given the strategy's high turnover. Varying spreads and slippage assumptions would clarify the cost thresholds at which the ML strategy remains profitable.

### 6.2.2 Additional Experiments

Several extensions could further improve predictive performance. Alternative prediction targets (e.g., 1-day or 10-day horizons, direction classification, or volatility forecasting) may better capture asset-specific behaviors. Feature engineering could also be expanded to include fundamental indicators, macroeconomic variables, sentiment measures, or technical indicators beyond moving averages.

Exploring nonlinear or ensemble models such as XGBoost, neural networks, or regime-switching approaches—may capture interactions that linear models like Lasso cannot. Adaptive position-sizing methods (e.g., volatility targeting or confidence-weighted allocations) also offer avenues for improving risk-adjusted performance.

### 6.2.3 Real-World Extensions

Practical deployments could include paper-trading simulations using APIs such as Alpaca or Interactive Brokers, enabling evaluation under real-time constraints such as latency and order execution quality. Further extensions involve tax-aware backtesting and multi-strategy portfolio construction, which would position the strategy within a realistic investment framework.

## References

[1] Brock, W., Lakonishok, J., & LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, 47(5), 1731–1764.

[2] Moskowitz, T. J., Ooi, Y. H., & Pedersen, L. H. (2012). Time series momentum. *Journal of Financial Economics*, 104(2), 228–250.

[3] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1), 267–288.

[4] Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *Review of Financial Studies*, 33(5), 2223–2273.

[5] Campbell, J. Y., & Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *Review of Financial Studies*, 21(4), 1509–1531.

[6] Kandel, S., & Stambaugh, R. F. (1996). On the predictability of stock returns: An asset-allocation perspective. *Journal of Finance*, 51(2), 385–424.

## A Code Repository

**GitHub repository:** `https://github.com/elisaaxxxxx/Project-STMA.git`

### A.1 Repository structure

The repository is organized to separate configuration, data processing, strategy logic, machine-learning components, and outputs:

- `project_config.py`: central configuration (tickers, dates, MA windows, costs, paths).
- `main.py`: main entry point that orchestrates the full workflow or selected steps.
- `SRC/`: core implementation of the traditional MA strategies, backtesting, and walk-forward evaluation.
- `ML/`: feature engineering, model training (Lasso), predictions, and ML-based strategy execution.
- `data/`: downloaded and processed datasets (raw OHLCV, engineered features, intermediate files).
- `show results/`: generated figures and performance summaries used in the report.

The framework is scalable: the set of analyzed stocks is configuration-driven (edit `project_config.py`), and the full pipeline runs automatically without modifying the core code.

### A.2 Installation instructions

The project requires Python 3.13 or higher. To run the code, follow these steps:

```
# 1. Clone the repository
git clone https://github.com/elisaaxxxxx/Project-STMA.git

# 2. Navigate to project directory
cd Project-STMA

# 3. Create and activate virtual environment
python3 -m venv .venv
source .venv/bin/activate

# 4. Install dependencies
pip install -r requirements.txt

# 5. Run the complete pipeline
python3 main.py --all
```

## B Helper tools

- **ChatGPT (OpenAI)**: used as an auxiliary coding assistant for debugging, refactoring, and improving code readability.
- **Claude Agent (Anthropic)**: used to assist with code structure analysis, logical validation, and documentation support.