Artificial Intelligence for Adaptive Investment Strategies

Master's Thesis in Computer Science (Artificial Intelligence) at the Université du Québec à Montréal

Master's program in Computer Science – Artificial Intelligence (2284)

Master Thesis at the Université du Québec à Montréal (UQAM), Faculty of Science on November 2nd 2025, by:

Michel DOUCERAIN

Thesis Supervision

Prof. Petko VALTCHEV
Ph.D., Professor

Thesis Supervisor

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor, Professor Petko Valtchev, for providing exceptional support, supervision, and guidance throughout the development of this thesis. I am deeply thankful for his detailed explanations of the FGC-Stream algorithm, as well as for introducing and clarifying the main theories, algorithms, and core concepts in the fields of association mining, artificial intelligence, data mining, machine learning, formal concept analysis, and Boolean functions.

He generously shared his extensive expertise in related topics such as closed patterns, maximal patterns, key or generator patterns, and algebraic structures including Boolean lattices, equivalence classes, Galois correspondences, and concept lattices. This thesis has drawn heavily upon these concepts and the related research in association mining, artificial intelligence, and data mining.

Abstract

This research aims to explore various machine learning, data mining, and association rule extraction methods to detect bullish or bearish movement signals within the context of portfolio management and algorithmic trading, in order to support decision-making for buy and sell transactions.

First, a review of the state of the art presents different studies applying machine learning techniques to investment. The main categories of technical indicators are covered, including trend indicators (ADX, AROON), momentum indicators (MACD, RSI, TSI, etc.), volatility, and overlays [86]. Then, several specialized studies focusing on association rule extraction are evaluated, particularly those related to predicting global stock indices [52].

In reviewing the related research on association rule extraction, it is observed that algorithms for detecting and extracting frequent patterns in data streams (such as ADWIN [10], Moment [21], NewMoment [62], CICLAD [70], and FGC-Stream [72]) aim to address concept drift and the high complexity of incremental stream processing. They rely on compact data structures (trees, tries, inverted lists, bit vectors) designed to optimize memory and computation time while preserving relevant frequent itemsets. Thanks to these approaches, it becomes possible to extract concise, robust, and scalable association rules, even from large and dynamic data streams. The most advanced algorithm selected for this study is FGC-Stream.

The FGC-Stream algorithm enables continuous extraction of frequent closed itemsets (FCIs) and their generators (FGIs) using a dynamic sliding window that expands or contracts based on data evolution. It employs a trie-based structure to efficiently manage the expansion and contraction of equivalence classes, allowing it to track changes related to concept drift and to identify emerging (jumping) or declining (falling) patterns. This approach provides a compact and expressive representation of frequent itemsets while ensuring incremental updates and the construction of relevant association rules within financial data streams.

The proposed architecture is as follows: after extracting 10 years of historical prices and volumes for the 500 U.S. equities comprising the S&P 500 index, the data are transformed into technical indicators covering momentum, volatility, volume, trend, relative performance against the index, liquidity, fundamentals, and market regime detection (low/high volatility, bullish or bearish) using Markov-Switching Regression (MSR), Hidden Markov Model (HMM), and Kaufman's Adaptive Moving Average (KAMA), as demonstrated in Pomorski and Gorse [81]. Following a rigorous feature evaluation process using machine learning models such as Random Forest, LightGBM, LSTM, and XGBoost, the top 30 features are selected to form the dataset. This dataset is then used to extract the most significant association rules according to the lift measure within the FGC-Stream algorithm.

Finally, a backtesting framework is implemented to validate the profitability of the extracted association rules. The backtest also incorporates a triple-barrier method [82] as a risk-control mechanism against estimated maximum loss using the GARCH model and for profit capture. In conclusion, the financial performance, in terms of risk-adjusted return, exceeds that of the benchmark S&P 500 index, with a cumulative return of 503.39% for the model compared to 333.98% for the S&P 500—an outperformance factor of approximately 1.52 over a period of about 10 years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Portfolio management and investment face increasing complexity and volatility in financial markets, making the prediction of market movements particularly challenging. Unpredictable fluctuations and complex interactions within markets require rigorous analysis, precise decision-making, and continuous updating of available information [33]. Artificial intelligence (AI) has emerged as a powerful tool to support investment strategies and optimize portfolio management [73].

Traditionally, the Efficient Market Hypothesis argued that asset prices reflect all available information and that all transactions occur at fair values, thereby limiting the possibility for investors to achieve above-average returns [27]. However, since the 1980s, the appearance of market anomalies and their exploitation have demonstrated the existence of inefficiencies in the processing of financial information [88]. These findings paved the way for the use of advanced data mining and machine learning techniques to identify investment opportunities and recurring patterns from market data.

The application of AI to continuous financial data streams enables the automatic extraction of relevant patterns and the anticipation of emerging trends. This approach improves forecasting quality, enhances portfolio stability, and allows models to learn more rapidly and adaptively in response to market dynamics [43].

1.2 Problem Statement

This research aims to identify relevant patterns to improve the quality of investment decision-making. In a context characterized by massive and continuous data streams and the rapid evolution of dynamic interactions among financial signals, it becomes particularly complex to distinguish reliable, robust, and truly informative signals from noisy or erroneous ones.

How can one distinguish meaningful patterns from noisy or misleading signals? How can one extract reliable patterns and demonstrate their stability over time? How can these patterns be presented in an interpretable and parsimonious way to make decision-making simple and actionable in a portfolio management context, allowing financial experts to validate the underlying rationale of these signals?

1.3 Objectives

This study aims to identify frequent, reliable, and potentially robust patterns from financial data to support efficient and profitable decision-making in portfolio management. It also proposes the development of a systematic methodology to conduct backtesting while integrating sound risk management principles.

The extraction of key features from a dataset exceeding 150 million data points, using advanced artificial intelligence techniques, seeks to improve both transparency and model explainability. An exhaustive coverage of major technical indicators is conducted across multiple time horizons and temporal sequences.

Finally, the backtesting results will evaluate the relevance of this approach in terms of performance relative to financial risk-adjusted measures.

1.4 Hypotheses

The first hypothesis assumes the presence of inefficiencies in how investors process information in financial markets, representing a potential source of gains through the systematic extraction of meaningful patterns associated with bullish stock market movements.

The second hypothesis posits that the analysis of price and volume movements of stocks in the U.S. equity market reflects not only information assimilation but also financial news, corporate fundamentals, sentiment, and the collective psychology of investors.

For example, during an uptrend characterized by increasingly higher highs, analysts anticipate that the upward movement will continue as long as the trend remains intact. Conversely, during a downtrend marked by lower lows, analysts expect selling pressure to dominate until a technical breakout occurs.

When prices oscillate within a horizontal range, the market is in a neutral or consolidation phase, where accumulation often precedes a rise and distribution often precedes a decline.

Historically, through the emergence of technical indicators, Charles Dow, founder of the Dow Jones Industrial Average, formulated the Dow Theory: markets move in trends, and these trends reflect the aggregation of all available information.

The first tools were quite simple: price charts, moving averages, and the drawing of support and resistance levels. During the 1960s–1980s, mathematical indicators were formalized and developed to smooth or interpret price and volume data, such as:

- Moving averages (SMA, EMA), useful for identifying trend reversals.
- RSI (Relative Strength Index, Welles Wilder, 1978), which measures the relative strength of a movement.
- MACD (Moving Average Convergence Divergence, Gerald Appel, 1979), which helps identify momentum crossovers.

In the modern era, from the 1990s to the present, a proliferation of tools has emerged: stochastic oscillators, Bollinger Bands, Ichimoku, and ATR (Average True Range, a measure of volatility). With computerization came instant calculations, large-scale backtests, and algorithmic trading systems.

Technical analysis has thus evolved closer to applied statistics and artificial intelligence.

Technical analysts operate on a key assumption: all relevant information (economic, fundamental, and psychological) is already reflected in price. Supporting this argument is the self-fulfilling nature of technical indicators: if many investors watch the same signals (e.g., a 200-day moving average crossover), their reactions actually generate the expected price movement.

Furthermore, based on crowd psychology theory, indicators condense collective sentiment and reveal extremes (overbought or oversold conditions).

Empirical experimentation has shown that certain signals perform statistically better than random chance over long periods.

Thus, it is reasonable to hypothesize that well-documented technical indicators are relevant and predictive signals for detecting buying and selling pressure trends or momentum in stock prices—signals that could prove effective and profitable if identified early enough to act before upward movements occur.

Another plausible hypothesis of this research is that techniques such as association rule mining applied to financial technical indicators and features selected by the most promising machine learning models can detect reliable bullish or bearish trend signals in financial markets. Once integrated into an algorithmic portfolio management system, these rules should improve investment decision-making and generate risk-adjusted performance superior to that of the benchmark index (S&P 500).

It is further assumed that these combinations of parsimonious association rules prevent combinatorial explosion and avoid over-complex decision systems that would otherwise produce unintelligible business rules for human financial experts.

These association rules are expected to be sufficiently reliable and robust to provide portfolio managers with a simple, interpretable, and manageable decision-making framework, suitable for communication with various stakeholders involved in financial risk management.

In other words, it is hypothesized that the combination of streaming frequent pattern mining algorithms, feature selection models, and a robust backtesting framework can exploit market inefficiencies and deliver superior, risk-adjusted returns that are both competitive and measurable.

1.5 Approach

The methodology adopted in this research is divided into several steps:

- **Data collection and preparation:** extraction of prices, volumes, and fundamentals of the 500 stocks composing the S&P 500 over a 10-year period.
- **Feature construction:** generation of technical indicators (momentum, trend, volatility, volume, liquidity, relative performance to the index) as well as market regime variables (using models such as HMM, MSR, and KAMA).
- **Feature selection:** application of several machine learning models (Random Forest, LGBM, LSTM, XGBoost) to retain the 30 most relevant features.
- **Association rule extraction:** application of the FGC-Stream algorithm to identify robust rules from continuous financial data streams, measuring their quality with the *lift* metric.
- **Backtesting:** evaluation of the profitability of the extracted rules using a protocol including the triple-barrier method and risk modeling through GARCH.
- **Comparative evaluation:** comparison of the proposed model's performance with the S&P 500 benchmark index over the 10-year study period.

1.6 Key Contributions and Results

The major contributions of this research are as follows:

- First study of its kind to develop an integrated methodology combining machine learning, market regime detection, and frequent pattern mining through association rule extraction.
- Adaptation and application of the FGC-Stream algorithm to the financial domain to extract dynamic association rules that are robust, reliable, transparent, incremental, and explainable.
- Implementation of a rigorous backtesting framework with integrated risk management (triple-barrier method and GARCH).
- Empirical demonstration that the extracted rules achieved a cumulative performance of 503.39% compared to 333.98% for the S&P 500, representing an outperformance factor of 1.52 over a 10-year period.

These results confirm that the proposed approach enhances the profitability and robustness of investment strategies against market volatility and conceptual drift.

1.7 Scope

The scope of this research is limited to U.S. equities from the S&P 500 index and a historical period of 10 years. Although the methodology is designed to be generalizable, its effectiveness may vary depending on:

- asset classes (currencies, commodities, bonds),
- time horizons (intra-day, weekly, monthly),
- and market conditions (financial crises, prolonged bull markets).

This study therefore represents a robust and extensible proof of concept, but further validation would be required for universal application. The development of an adaptive investment strategy responsive to market regimes could provide better protection against financial losses for investors' savings and wealth, leveraging AI models that are both powerful—through the use of massive financial indicator datasets—and interpretable, parsimonious, and explainable through association rule mining.

For professionals in the investment industry, such an approach would enable better systemic risk management, more competitive and higher-performing products, and reinforce client trust.

1.8 Structure of the Thesis

The thesis is structured as follows:

- **Chapter 1 – Introduction:** presentation of the context, research problem, objectives, hypotheses, and methodology.
- **Chapter 2 – Literature Review:** state of the art on machine learning in finance, technical indicators, frequent pattern mining algorithms, and existing association rule detection approaches.
- **Chapter 3 – Methodological Framework:** detailed description of data collection, construction of technical indicators, machine learning models (RF, LGBM, XGBoost, LSTM) for feature selection, and association rule extraction with the FGC-Stream algorithm.
- **Chapter 4 – Experimentation and Results:** presentation of empirical results, extracted rules, backtesting methodology, and comparison with the benchmark index.
- **Chapter 5 – Conclusion and Perspectives:** summary of contributions, practical implications, and future research directions.

Chapter 2

Literature Review and State of the Art

2.1 Machine Learning (ML) in Investment

"An Artificial Neural Network-based Stock Trading System Using Technical Analysis and Big Data Framework" (2017)

The study by Sezer, Ozbayoglu, and Dogdu [86] (2017) integrates several technical analysis indicators, particularly trend-based ones such as:

- **MACD (Moving Average Convergence Divergence):** This indicator computes the difference between two exponential moving averages (EMA) of different periods:

$$\text{MACD}_t = \text{EMA}_{12}(P_t) - \text{EMA}_{26}(P_t).$$

- **RSI (Relative Strength Index):** A measure of the historical strength or weakness of price movements:

$$\text{RSI} = 100 - \frac{100}{1 + RS}, \quad \text{where} \quad RS = \frac{\text{average gains}}{\text{average losses}} \quad \text{over } n \text{ periods.}$$

  It detects overbought (RSI > 70) or oversold (RSI < 30) conditions.

- **Williams %R:** A momentum indicator designed to identify overbought or oversold stocks:

$$\%R = 100 \times \frac{H_n - C}{H_n - L_n}$$

The analysis is applied to stocks within the Dow Jones Industrial Average (DJIA 30) index, covering the period from 1997 to 2006. The chosen model is a multilayer perceptron (MLP) neural network implemented within the Apache Spark Big Data framework. The model achieved an accuracy of 65.52%. For each DJIA component, portfolio returns and final values were compared against a passive investment strategy. Only 11 out of the 30 stocks outperformed the passive benchmark when using the MLP-based model. In conclusion, the authors recommend the use of deep neural networks to further improve results.

"Stock picking with machine learning" (2023)

Wolff and Echterling [96] (2023) examine the application of machine learning to weekly stock selection, based on an initial universe of all S&P 500 components between January 1999 and March 2021. The task is formulated as a binary classification problem aiming to predict whether a stock will outperform or underperform the market median return in the following week.

The authors use a wide range of fundamental factors as input variables — including size, value, profitability (EPS, ROIC), growth, and sector — together with technical indicators such as moving averages, historical risk or volatility, trading volume, and momentum. These variables are recalculated weekly to reflect current market conditions. The dataset contains approximately 1.3 million observations.

Several machine learning models are evaluated, including regularized logistic regression (LASSO and Ridge), random forests, deep neural networks (DNN), and recurrent LSTM networks. The training dataset includes 500 stocks with a 3-year lookback window, to account for structural breaks and evolving relationships in financial market data. To avoid survivorship bias, the analysis relies on the complete historical list of all 1,164 stocks that have ever been part of the S&P 500, considering only those present in the index at the end of each training period. While the classification accuracy of all models remains modest — around 50.4% — the authors demonstrate that these performances are statistically significantly higher than the non-informative 50% baseline.

Performance is assessed using annualized return, Sharpe ratio, information ratio, and Jensen's alpha relative to an equally weighted benchmark portfolio. Results show that machine learning-based strategies systematically outperform the benchmark, with significantly higher Sharpe ratios and more stable weekly returns. For example, ensemble methods achieve an annualized return of 20.8%, compared to 6.60% for the S&P 500.

A factor attribution analysis of ML-based strategies reveals that, except for the PCA (Principal Component Analysis) strategy — a dimensionality reduction technique that preserves as much variance as possible — all other strategies exhibit a positive and statistically significant exposure to the value factor ("HML") and the betting-against-beta factor ("BaB"), along with a significant negative exposure to the quality factor ("RMW").

From a portfolio management perspective, the methodology proves robust to geographic diversification, as demonstrated by reproducing the results on the STOXX Europe 600 index. These findings suggest that integrating machine learning–based signals into an active management process can enhance the return–risk tradeoff while maintaining acceptable interpretability.

"Forecasting Significant Stock Market Price Changes Using Machine Learning: Extra Trees Classifier Leads" (2023)

The study "Forecasting Significant Stock Market Price Changes Using Machine Learning: Extra Trees Classifier Leads" by Pagliaro [76] (2023) presents an approach based on an *Extra Trees Classifier* to forecast significant stock price variations over a 10-trading-day horizon. Using a sample of 120 stocks from various sectors, the author employs several **technical indicators** as explanatory variables, including:

- **Momentum**: MACD, RSI, TSI (True Strength Index), TSI Slope, RVGI, STC (Staff Trend Cycle), STC rate of change, Williams %R, and CFO (Chande Forecast Oscillator)
- **Overlay indicators**: rate of change of VWMA (Volume Weighted Moving Average), FWMA (Fractal Weighted Moving Average)
- **Trend indicators**: ADX, AROON
- **Volatility indicators**: Bollinger Bands, RVI (Relative Volatility Index), Donchian price, and rate of change of Donchian price

- **Exponential smoothing**: STC, STC rate of change, A50 and A23 rate of change to give more weight to recent observations, along with a seasonality component using Holt–Winters exponential smoothing

The model aims to predict the percentage difference between the current closing price and the price after 10 trading days (two calendar weeks). The target variable was discretized into three classes using predefined thresholds to ensure each interval contributes equally—a process known as stratified sampling.

The *Extra Trees Classifier* was chosen for its training speed, robustness to noisy data, and ability to handle a large number of variables. It uses a random subset of training data and random splits, which can reduce overfitting and improve generalization. The model was compared against Random Forest, Bagging, NuSVC, XGB Classifier, KNeighbors Classifier, and LGBM Classifier. Results show a **classification accuracy of 86.08%** on test data, significantly outperforming the benchmark model.

Extra Trees specifically overcome the limitations of Random Forests by offering faster training and better robustness to noise, making them particularly suitable for complex financial datasets.

Regarding evaluation methods, the article mentions the use of accuracy metrics to measure prediction quality. It also reports simulated trading experiments based on the model's buy/sell signals for 15 stocks, yielding a strategy return of 19.09%, compared to 4.74% for a buy-and-hold approach.

The use of a classification approach (rather than regression) proves relevant when dealing with a large feature space while limiting overfitting. The *Extra Trees Classifier* demonstrates practical superiority over classical methods, and the obtained accuracy (86.1%) confirms the relevance of this approach for short-term market movement forecasting.

"Improving stock trading decisions based on pattern recognition using machine learning technology" (2021)

Lin et al. [64] (2021) propose PRML (Pattern Recognition with Machine Learning), a systematic framework that automates the recognition and validation of candlestick patterns combined with technical indicators, and then transforms these validated patterns into investment signals for the Chinese stock market over the 2000–2020 period.

The authors adopt a pragmatic approach: formalizing 13 basic daily patterns, constructing all possible two-day and three-day combinations, computing a rich set of technical indicators and relative positions, testing several supervised learning methods, and retaining only those patterns whose predictive power exceeds a defined threshold. Experimental results show that, within their framework and for a one-day horizon, certain two-day combinations deliver attractive financial performances, measured by annualized return, Sharpe ratio, and Information ratio—even after accounting for transaction costs.

The method proceeds in three linear stages: (1) pattern definition and extraction, (2) feature vector construction and model training, and (3) pattern filtering and dynamic portfolio construction. The 13 daily patterns cover alternative candle forms (bullish/bearish bodies, long/short shadows, relative positions). Combining these forms yields 169 possible two-day and 2197 three-day patterns. For each historical pattern occurrence, the feature set associates candlestick structure, *loc* (relative position in the window), and a series of standard technical indicators (MA, EMA, MOM, ROC, CCI, ATR, OBV, Chaikin A/D, etc.) computed over short windows (5–10 days).

The training set covers 2000–2014, and validation is performed over 2015–2020. Data is split (80% training / 20% validation) to measure model accuracy for each pattern. Patterns with test accuracy above 55% are retained in the trading pool; various sub-pools (All, Adjust, TOP10, TOP5, TOP3) are defined according to frequency and precision. The signals are then translated into equally weighted (long-only, due to Chinese market constraints) portfolios with variable investment horizons $N \in \{1..10\}$ days.

The authors test multiple classical and deep learning methods to assess the robustness of the PRML framework with respect to the learning algorithm. Understanding their roles helps interpret the results.

Logistic Regression (LR): A linear classification model estimating the probability of an up/down movement through a sigmoid transformation of a linear combination of features. Its simplicity makes it a good baseline benchmark.

K-Nearest Neighbors (KNN): A non-parametric method predicting a class by the majority of the k nearest neighbors in feature space. KNN relies on a local similarity hypothesis: similar historical patterns yield similar outcomes.

Random Forest (RF): An ensemble of decision trees trained on subsamples and subsets of features. Each tree captures a nonlinear partition of the space; aggregation by voting reduces variance and detects complex interactions among technical indicators. In practice, RF is robust to noise and relatively insensitive to outliers.

Restricted Boltzmann Machine (RBM): A *Restricted Boltzmann Machine* (RBM) is a stochastic artificial neural network capable of learning a probability distribution over its inputs. Liang et al. [63] used an RBM to predict short-term stock market trends.

In the empirical phase, a logistic regression layer was connected to the RBM output for classification. Different parameter combinations led to varying classification performance.

Deep Networks (MLP, LSTM): MLPs (Multilayer Perceptrons) learn stacked nonlinear transformations; LSTM (Long Short-Term Memory) networks are designed to capture long sequential dependencies through memory cells controlled by input, forget, and output gates. LSTMs have proven useful for financial time series in several studies (Fischer & Krauss, 2018)[28]. In PRML, MLP and LSTM are used to test the framework's *dependence* on deep models and to verify whether a sequential representation (LSTM) improves signals derived from discrete patterns.

The main conclusions are as follows: on average, two-day combinations provide the best predictive power for the one-day horizon. Portfolios constructed from the TOP10/TOP5 two-day patterns achieve high annualized returns (e.g., **TOP10: 36.7% annual** in one configuration) and favorable Sharpe and Information Ratios, while showing drawdowns lower than the benchmark index over the same period. These results remain valid when including a transaction cost of 0.2% (a realistic value for the Chinese market according to the authors).

Several methodological and practical points must be understood before interpreting these results as a guarantee of operational success.

- **Statistical impact of patterns:** The number of patterns exceeding the 55% accuracy threshold decreases significantly as the forecasting horizon increases, especially for two-day based forecasts. Lower prediction accuracy and more frequent trades can cause a significant drop in returns, as shown in the 3-day and 7-day horizon results.

- **Performance measurement:** The financial metrics used (Sharpe, max drawdown, IR) are relevant, but lack evaluation under realistic conditions (size effect, liquidity/illiquidity, variable slippage, intraday execution). Accounting for a fixed 0.2% transaction cost is useful but partial.

- **Comparison with alternative techniques:** López Gil, Duhamel-Sebline, and McCarren [66] show in "An Evaluation of Deep Learning Models for Stock Market Trend Prediction" that modern models (Transformers, xLSTM-TS, TCN, N-BEATS, TFT, N-HiTS, and TiDE [66]) can offer alternative trade-offs.

"XGBoost-Based Multi-Factor Stock Selection Model for Rotational Trading" (2023)

Praphutikul and Limpiyakorn [83] (2023) propose using XGBoost, a gradient boosting decision tree algorithm, to construct stock selection signals based on twenty-seven heterogeneous factors (value, growth, momentum, liquidity, quality, dividend, size) and to drive a *rotational trading* strategy at monthly and quarterly horizons within the universe of Thai large- and mid-cap equities. The central idea is both simple and powerful: to use an algorithm to learn the nonlinear combination of factors that predicts the probability of belonging to the Top30% (high future return) versus Middle40% and Bottom30%. The models are trained on 2012–2019 data and robustness is tested in backtests over 2020–2021, a period including the pandemic's high volatility, with realistic transaction costs and a slippage model tied to intraday volatility.

The project follows a pipeline consisting of time window selection, universe definition, data cleaning, transformation, training, and portfolio construction. The universe includes the 200 largest capitalizations as of end-2019. Data covers 2012–2021 and is divided into 80% for training (2012–2019) and 20% for validation, with 2020–2021 used for backtesting. The twenty-seven selected factors are ranked and transformed into deciles (0–9). The target variable is a three-class classification: 0 (Bottom30%), 1 (Middle40%), and 2 (Top30%), according to the future return over 20 days (1 month) for the monthly model or 60 days (one quarter) for the quarterly model. This construction allows formulating the selection problem as probabilistic classification.

XGBoost is an industrial-grade implementation of decision-tree *gradient boosting*, designed for performance and scalability. At its core, boosting builds an additive predictor:

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

where each f_t is a weak decision tree that corrects the residuals left by the ensemble of previous trees. The algorithm minimizes a loss function plus a regularization term to control complexity:

$$\mathcal{L} = \sum_i \ell(y_i, \hat{y}_i) + \sum_t \Omega(f_t).$$

XGBoost's optimizations include cache-aware and parallelized implementations, making it both fast and robust on real-world datasets. For multi-class classification, XGBoost outputs class membership probabilities, allowing selection of stocks most likely to fall into the Top30%. These features explain XGBoost's popularity in quantitative finance, where relationships are often nonlinear and noisy.

XGBoost can be thought of as a team of experts (the trees) voting together, with each new expert focusing on the mistakes of the previous committee. Built-in regularization (penalties on tree complexity and number of leaves) prevents the algorithm from overfitting historical noise—a major risk in finance. XGBoost also provides feature importance measures that help interpret which factors the algorithm considers most predictive.

Validation ROC curves and AUCs show that the quarterly model has better discriminative capacity than the monthly model on validation data. Intuitively, fundamental signals evolve more slowly and thus suit a quarterly horizon, while technical (momentum) indicators capture short-term dynamics better exploited by monthly rotation. Variable importance analysis reveals that for monthly selection, momentum factors (e.g., 52-week high distance, 250-day rate of change) dominate, while for quarterly selection, fundamental factors (YoY revenue growth, QoQ EBIT, leverage ratios) weigh more heavily. This distinction provides valuable insight for practitioners: reallocation frequency should align with the nature of the exploited signals.

In backtests over 2020–2021, a period of high volatility due to the pandemic, monthly portfolios achieved higher CAGR and better risk-adjusted ratios (Sharpe, MAR) than quarterly portfolios when ignoring transaction costs—mainly because more frequent rotation captures more transient opportunities. However, per-trade analysis shows that the quarterly strategy yields higher trade expectancy and that average holding duration (Avg Bar Held) aligns with the intended horizons (≈ 20 days for monthly rotation, ≈ 60 days for quarterly rotation).

When the authors add realistic market frictions—0.15% commission per trade and dynamic slippage calculated based on intraday amplitude (explicitly detailed in the paper)—the relative advantage of monthly rotation decreases sharply. Specifically, the annual CAGR of the monthly-selected portfolio drops from about 48% without costs to $\approx 29\%$ with costs, while the quarterly portfolio declines more moderately (from $\approx 31\%$ to 27%). This result illustrates a fundamental principle: the higher the trading frequency, the greater the erosion of returns due to transaction costs and slippage. Realistic modeling of costs is therefore essential to assess the viability of an algorithmic strategy.

"Integrated Long-Term Stock Selection Models Based on Feature Selection and Machine Learning Algorithms for China Stock Market" (2020)

The article "Integrated Long-Term Stock Selection Models Based on Feature Selection and Machine Learning Algorithms for China Stock Market" (2020) examines the construction and evaluation of integrated stock selection models for the Chinese A-share market, combining *feature selection* methods with nonlinear machine learning algorithms. Yuan et al. [98] start from the premise that classical linear models (CAPM, APT, factor models) may be insufficient to capture market complexity, nonlinearity, and dynamics. They propose a complete methodological chain: defining an initial set of 60 explanatory variables from financial statements and daily stock prices; data cleaning and normalization; variable subset selection using methods such as SVM-RFE and Random Forest importance; hyperparameter tuning via time-based validation (sliding window); and prediction of excess return direction (binary classification) using SVM, Random Forest (RF), and Artificial Neural Networks (ANN). Finally, robustness is tested through backtests and construction of long-only and long-short portfolios. Experiments cover the period from January 1, 2010, to January 1, 2018, excluding ST stocks and recently listed securities (less than three months old).

The central task is a binary classification of an asset's future relative performance: the authors forecast each stock's monthly *excess return* relative to the benchmark index (Shanghai Composite). To avoid predictions being dominated by general market trends (overall up/down), the target variable is defined as the relative return:

$$r_{i,t}^{\text{exc}} = r_{i,t} - r_t^{\text{index}},$$

where $r_{i,t}$ is the return of stock i over the next month and r_t^{index} is the index return over the same period. Each month, stocks are ranked by r^{exc}: the top third is labeled $+1$, the bottom third -1, and the middle third ignored during training (to focus on the most distinct cases and reduce noise).

Data Preprocessing

To improve data quality before using them in learning models, several preprocessing steps are applied to limit the influence of extreme values, correct missing data, neutralize sector biases, and standardize numerical scales.

Outlier treatment. Financial data may contain abnormal values that can negatively affect models. To mitigate this, the authors use a median-based winsorization method. Let x_i be the i^{th} observation, x_m the median of the series, and DMAD the median absolute deviation $|x_i - x_m|$. The corrected value $x_{i,\text{new}}$ is defined as:

$$x_{i,\text{new}} = \begin{cases} x_m + n \times \text{DMAD}, & \text{if } x_i \geq x_m + n \times \text{DMAD}, \\ x_m - n \times \text{DMAD}, & \text{if } x_i \leq x_m - n \times \text{DMAD}, \\ x_i, & \text{otherwise.} \end{cases}$$

The parameter n controls the tolerance band around the median and limits the influence of extreme values without removing observations.

Missing value imputation. Due to missing financial reports or calculation errors, some variables may contain missing values that could bias analyses. To minimize this, missing values are replaced with industry averages. When a factor is missing for a stock, it is imputed by the average of the same factor across other stocks within the same industry.

Neutralization of size and industry effects. In the A-share market, returns and factors can be influenced by market capitalization and industry membership. For example, the EP (Earnings-to-Price) factor tends to be higher for banks than for Internet companies. To eliminate such effects and make factors comparable across firms, a multiple linear regression-based neutralization procedure is applied.

Standardization of variables. Features such as market value and the EP ratio have very different units and magnitudes. To prevent one variable from dominating the others, each factor is standardized using the z-score transformation:

$$x_{i,\text{new}} = \frac{x_i - \mu}{\sigma},$$

where μ is the mean and σ the standard deviation of the corresponding variable. This ensures that all features contribute evenly to model training regardless of their original scale.

Feature Selection

Feature selection is crucial when starting from a vector of 60 features, among the following:
Two feature selection approaches are compared:

Table 2.1: Input features for the stock market dataset

Category	**Description of features**
Valuation factors	Net profit / total market value; Net profit after exceptional items (TTM) / total market value; Net profit (TTM) / total market value; Operating revenue (TTM) / total market value; Net operating cash flow (TTM) / total market value; Net assets / total market value; Operating cash flow (TTM) / total market value.
Growth factors	Net profit growth rate (TTM); Operating revenue growth rate (TTM); Net profit growth after exceptional items (TTM); Growth rate of operating cash flow compared to the same period last year.
Financial quality	ROE (TTM); ROA (TTM); Gross margin (TTM); Net margin (TTM); Ratio of net profit to losses (TTM/OTD); Net profit (TTM/OTD) to operating revenue; Operating revenue (TTM/OTD) to total assets; Operating cash flow (TTM/OTD) to total assets and to net profit.
Leverage factors	Debt ratio (assets / liabilities); Current ratio; Quick ratio.
Size factors	Market capitalization in circulation; Total market value; Total assets; Total liabilities; Total equity; Operating revenue (TTM/OTD); Net profit (TTM/OTD); Total profit (TTM/OTD); Differences between assets, liabilities, and equity.
Turnover factors	Inventory turnover rates 1, 2, and 3.
Momentum factors	1-month return; 3-month return; 6-month return; 12-month return; Daily turnover × daily return (1 month); Daily turnover × daily return (3 months); Daily turnover × daily return (6 months); Daily turnover × daily return (12 months).
Volatility factors	Max price / min price ratio (1 month); Max price / min price ratio (3 months); Max price / min price ratio (6 months); Max price / min price ratio (12 months); Standard deviation of daily returns (1, 3, 6, 12 months).
Liquidity factors	Cash ratio; BLAS(20).
Quantitative / technical factors	RSI(6); RSI(12); BLAS(20h).

Note: OTD denotes the last quarter; *TTM* (Trailing Twelve Months) indicates the past four quarters.

SVM-RFE (Recursive Feature Elimination based on SVM) SVM-RFE proceeds iteratively: a linear SVM is trained on all variables, the importance of each variable is computed as the square of its associated linear SVM weight, then the least important variable is removed, and the process is repeated. Formally, if the linear SVM gives a weight vector $w = (w_1, \ldots, w_n)$, the importance of the ith feature is:

$$\text{score}_i = w_i^2.$$

Iterations continue until the desired number of features is reached. In this study, the authors retain the top 80% of features (48 out of 60) after ranking.

Random Forest-based selection Random Forest feature selection leverages the *out-of-bag* (OOB) concept. For each tree in the forest, observations not drawn in the bootstrap sample form the OOB sample. The importance procedure is as follows:

1. Compute the initial out-of-bag error OOB_error1_i of each tree on its OOB observations.

2. For each feature j, randomly permute its values in the OOB sample, add a certain level of noise to disrupt the signal, and recompute the error OOB_error2_i.

3. The importance of feature j for tree i is $\text{OOB\_error1}_i - \text{OOB\_error2}_i$. Averaging across all trees (n trees):

$$\text{Importance}_j = \frac{1}{n}\sum_{i=1}^{n}\Big(\text{OOB\_error1}_{i,j} - \text{OOB\_error2}_i\Big).$$

The variables are then ranked by importance, and—consistent with SVM-RFE—the top 80% are retained.

Prediction Algorithms

The predictive models used include SVM (with RBF kernel), Random Forest (classification), and a fully connected ANN (three layers).

Support Vector Machine (SVM) SVM seeks the hyperplane with the maximum margin. For a linearly separable dataset (x_i, y_i), with $y_i \in \{-1, 1\}$, the normalized distance of a point to the hyperplane $w^\top x + b = 0$ is:

$$D = \frac{|w^\top x + b|}{\|w\|}.$$

The primal optimization problem (without soft margin) is:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^\top x_i + b) \geq 1, \ \forall i.$$

The above is a quadratic programming problem solvable via standard optimization techniques. To allow margin violations (soft margins), a penalty parameter C is introduced, and the dual formulation (via Lagrange multipliers α_i) becomes:

$$\max_{\alpha \geq 0} \min_{w,b} L(w, b, \alpha)$$

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \Big(\text{label}_i (w^\top x_i + b) - 1\Big)$$

where label_i is the class label and α_i is the Lagrange multiplier.
After simplification, the dual form is:

$$\max_{\alpha} \ \sum_{i=1}^{n} \alpha_i \ - \ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \, \text{label}_i \, \text{label}_j \, x_i^\top x_j$$

subject to:

$$\alpha_i \geq 0, \qquad \forall i = 1, \ldots, n, \qquad \text{and} \qquad \sum_{i=1}^{n} \alpha_i \, \text{label}_i = 0.$$

The chosen RBF (Gaussian) kernel is:

$$K(x_i, x_j) = \exp\Big(-\gamma \|x_i - x_j\|^2\Big),$$

with $\gamma > 0$ as a hyperparameter. An advantage of SVM over logistic regression (LR) is its ability to handle non-linearity via kernel functions. Generally, data may not be linearly separable in low-dimensional space but can become separable when projected into a higher-dimensional space. The kernel function computes inner products in low-dimensional space while implicitly operating in high dimensions—allowing transformation and reduced computational cost. The decision function becomes:

$$f(x) = \text{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\Big).$$

Hyperparameters (C, γ) are selected through temporal cross-validation.

Random Forest (RF) RF is an ensemble of decision trees (Bagging). Each tree is trained on a bootstrap sample, and at each node, the best split is chosen among a random subset of features (reducing correlation among trees). For classification, the final prediction is the majority vote of the trees; for regression, it is the average of outputs. Key hyperparameters include: number of trees N, maximum number of features tested per split, depth or stopping conditions (minimum internal node size s, minimum leaf size l). The authors set $N = 100$, adopt practical simplifications to reduce complexity, and tune parameters using temporal validation.

Artificial Neural Network (ANN) The authors use a fully connected three-layer network (input → two hidden layers → output). For a given layer, the transformation is:

$$\text{hidden}_j = f\Big(\sum_i x_i W_{ij}^{(1)} + b_j^{(1)}\Big), \quad y_k = f\Big(\sum_j \text{hidden}_j W_{jk}^{(2)} + b_k^{(2)}\Big),$$

where f is the chosen activation function (sigmoid: $f(z) = 1/(1+e^{-z})$ in the presentation). The quadratic error used is:

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2,$$

and optimization is performed by backpropagation (gradient descent or quasi-Newton optimizer `lbfgs`). The partial derivatives presented by the authors for the weight updates are:

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = (y_j - t_j)\, y_j(1-y_j)\, \text{hidden}_i,$$

followed by the update $W_{ij}^{(2)} \leftarrow W_{ij}^{(2)} - \eta \frac{\partial E}{\partial W_{ij}^{(2)}}$, and for the previous layer:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \Big(\sum_k \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial \text{net}_k^{(2)}}\frac{\partial \text{net}_k^{(2)}}{\partial \text{hidden}_j}\Big) \cdot \text{hidden}_j(1-\text{hidden}_j)\cdot x_i,$$

Temporal Validation — Time Sliding Window Cross-Validation

The authors emphasize the inadequacy of classical k-fold cross-validation for financial time series, as it can mix future observations into the training set. They adopt a *time window slicing* strategy: for 12 months of data used for hyperparameter tuning, the period is divided into 12 consecutive temporal blocks; at each iteration, the first four blocks are used as the training set and the next block as the validation set — then the time window is slid forward to cover the entire training period. This procedure respects temporal order and simulates a realistic usage (never using the future to predict the past).

Chosen Parameters and Implementation

- For SVM: RBF kernel, grid search for (C, γ), evaluation via time window slicing.
- For RF: $N = 100$ trees, a maximum limited number of features tested per split (suggested as the square root of the total number of features), thresholds s and l to stop splitting.
- For ANN: 3-layer network, first hidden layer 20 neurons, second hidden layer 10 neurons, `lbfgs` optimizer, L2 regularization (parameter α), iteration limit `max_iter`.

For feature selection, SVM-RFE and RF each retain the top 80% of features (48/60) to ensure a coherent comparison across methods.

Evaluation: Metrics and Backtest

Model performance is evaluated on two levels:

1. **Predictive ability (classification)** — standard confusion matrix (TP, FP, FN, TN), precision, recall, accuracy, and especially AUC (area under the ROC curve), which is robust to threshold choice.

Table 2.2: Prediction Accuracy of Different Models

Model	No Selection	SVM-RFE	RF
SVM	51.7287%	51.7710%	51.8295%
RF	52.9331%	52.7187%	52.7804%
ANN	52.3216%	52.3497%	52.3204%

Note: "No Selection" means no feature selection method was applied.

2. **Economic performance (investment strategy)** — monthly backtests between January 2011 and January 2018, constructing long-only portfolios by taking stocks predicted as "+1" and then selecting the top 1%, 3%, 5% according to the predicted probability of being in the +1 class. Calculated indicators: annualized return, Sharpe ratio, maximum drawdown, win rate, profit/loss ratio, etc. The general formulas mentioned by the authors are the usual ones: annualized return R_p derived from total return P and duration n in days, volatility σ_p, etc. (the authors define these aggregates conventionally).

Results

The results show that:

- From a purely predictive perspective (AUC, accuracy), the performances of the three models (SVM, RF, ANN) are close, but RF has a slight overall advantage.
- In economic backtests, SVM and RF outperform the traditional linear APT; ANN is generally less profitable than SVM/RF in these experiments.
- The best pipeline is **RF-RF**: Random Forest used both for feature selection and prediction. This pipeline provides the best results in terms of annualized return, Sharpe ratio, and win/loss ratios.
- Selecting the top **1%** of stocks (most confident according to RF-RF) yields an annualized return of up to **29.51%** over the test period, according to the paper. A long-short construction (buy group 1 and sell group 10 simultaneously) produces an annualized portfolio return of **21.92%** with a **max drawdown of 13.58%**, **Sharpe $\approx$ 2.86** and a total return of **285%**, showing a significant reduction in maximum volatility compared to a fully long-only position (where max drawdown could exceed 45%).

Interpretation and Methodological Comments

Several methodological and practical points emerge:

- **Importance of feature selection** — filtering variables reduces noise and complexity, lowers overfitting risk, and facilitates relative interpretability; RF provides a robust importance measure via OOB and permutations.
- **Temporal validation** — the time sliding window strategy is essential to avoid data leakage caused by classical cross-validation mixing future and past.
- **Economic vs. statistical robustness** — a model may show reasonable AUC without translating into economically profitable strategies; backtesting (and ideally additional robustness tests) remains indispensable.

- **Advantage of ensemble models** — RF-RF combines ensemble resilience (variance reduction) with a natural importance method for noisy financial data, partly explaining its observed superiority.
- **Risk control** — adopting a long-short strategy reduces maximum drawdown and improves Sharpe even if absolute return is slightly lower than extreme long-only selections.

Identified Limitations and Future Directions

The authors acknowledge several limitations and open future directions:

- Tests limited to the Chinese A-share market — generalization to other markets (US, UK) should be validated.
- The choice of retaining 80% of features is heuristic; a systematic method to determine the optimal number of features (e.g., validation based on economic performance or information criteria) remains to be developed.

Table 2.3: Summary Matrix of Existing Studies — Machine Learning and Stock Prediction

Ref.	**Question / Hypothesis**	**Method**	**Data (period)**	**Size / Scale**	**Key Results**	**Limitations**	**Relevance to Thesis**
Sezer, Ozbayoglu, and Dogdu [86] (2017)	Prediction of returns via technical indicators	MLP under Apache Spark	DJIA 1997–2006	30 stocks	Accuracy 65.52%; 11 out of 30 stocks outperform passive strategy	Shallow model, limited horizon	Empirical validation of neural networks on real data
Wolff and Echterling [96] (2023)	Weekly outperformance classification	RF, DNN, LSTM, LASSO	S&P500 (1999–2021)	1.3M obs.	Annualized return 20.8% vs 6.6% (S&P 500); increased Sharpe	Moderate accuracy ($\approx$ 50%)	Shows advantage of ML ensembles in active management
Pagliaro [76] (2017)	10-day price forecasting	Extra Trees Classifier	120 US multi-sector stocks 1995–2022	Daily data	Accuracy 86.08%; return 19.09% $\gg$ buy-and-hold (4.74%)	Low interpretability, short horizon	Confirms superiority of extreme random forests for short-term signals
Lin et al. [64] (2021)	Validation of candlestick patterns using ML	PRML (pattern learning) + RF/LSTM	China 2000–2020	2197 patterns over 20 years	TOP10 annual return: 36.7%, high Sharpe	Chinese data, simplified transaction costs	Shows effectiveness of pattern recognition
Praphutikul and Limpiyakorn [83] (2023)	Stock selection via gradient boosting	XGBoost (multi-class)	Thailand 2012–2021	200 stocks	CAGR 48% without costs, 29% with costs	Sensitive to slippage	Demonstrates robustness of XGBoost for multifactor selection
Yuan et al. [98] (2020)	Integrated long-term model (A-Share)	SVM-RFE, RF, ANN	China 2010–2018	2000 stocks	RF-RF: annualized return 29.5%, Sharpe 2.86	Single market, 80% features retained	Key reference for integrated methodology (feature selection + ML)

2.2 Association Rule Mining (ARM) in Investment

Stock index prediction remains a central problem in quantitative finance. Financial data exhibit complex characteristics: high noise, non-linearity, dynamic dependencies, and *concept drift* phenomena (i.e., changing statistical relationships over time). In this context, the use of interpretable and robust methods is crucial.

Association rules, originally developed for market basket analysis, provide a methodological framework to identify frequent and significant relationships between variables. The idea is to automatically extract frequent patterns from large transactional or temporal databases, then use them as signals for investment decisions.

An association rule is represented as:

$$X \Rightarrow Y$$

where X and Y are disjoint sets of items (called *itemsets*). X is the antecedent and Y the consequent of the rule. The quality of a rule is evaluated through several statistical measures:

- **Support**: proportion of occurrences of $X \cup Y$ in the database.

  $$\text{Support}(X \to Y) = \frac{\sigma(X \cup Y)}{|D|}$$

  Support measures the *frequency* of the rule, interpreted as the fraction of transactions containing both X and Y.

- **Confidence**: measures the frequency of Y in transactions that already contain X.

  $$\text{Confidence}(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

  The confidence value indicates how reliable the rule is.

- **Lift**: Lift measures the contribution of antecedent X to the consequent Y.

  $$\text{Lift}(X \Rightarrow Y) = \frac{\text{Confidence}(X \Rightarrow Y)}{\text{Support}(Y)}$$

  A lift greater than 1 indicates a positive association between X and Y. Lift above 1 means they appear together more often than expected, while lift below 1 indicates they appear together less often than expected. High lift values thus signal a stronger association.

Rule discovery is performed in two steps:

1. Extraction of **frequent itemsets** using algorithms such as *Apriori* or *FP-Growth*, imposing a *minimum support.*

2. Generation of rules from these itemsets, followed by filtering based on confidence and lift thresholds, to isolate *interesting* relationships.

Challenges related to financial data remain:

- **Noise**: price series contain a large proportion of random fluctuations.

- **Non-linearity**: dependencies between technical indicators and price movements are not linear.

- **Concept drift**: statistical relationships change over time due to economic cycles, exogenous shocks, crises, or volatility regimes.

These characteristics make direct use of linear models difficult and highlight the relevance of adaptive and robust methods such as association rules.

"Assessing efficacy of association rules for predicting global stock indices" (2022)

The study "Assessing efficacy of association rules for predicting global stock indices" by Kaur and Dharni [52] (2022) examines the effectiveness of association rules for predicting global stock indices (Japan, United States, United Kingdom, India, Brazil, China, South Africa). The authors use 12 years of daily data across several global indices and apply technical indicators transformed into discrete variables (*bullish*, *bearish*, *neutral*).

Technical indicators used The main indicators considered include:

- **Relative Strength Index (RSI)**: momentum measure based on average gains/losses.
- **Stochastic Oscillator %K and %D**: overbought/oversold indicator.
- **Disparity 5 and 10 days**: relative gap between price and moving average.
- **Rate of Change (ROC)**: relative price change over a given period.
- **Commodity Channel Index (CCI)**: measures how far the current price deviates from the average price over the past n days.
- **Momentum**: measures the magnitude of price movement over a given period.
- **LW%R (Larry Williams %R)**: momentum measure evaluating overbought and oversold conditions.

To convert continuous price data of technical indicators into categorical data usable by the Apriori algorithm in Weka, the authors perform the following data transformations:

1. Discretize technical indicators into signal categories (buy / sell / hold).
2. Apply the rule mining algorithm with support and confidence thresholds.
3. Analyze the profitability of investment signals produced by these rules.

- Association rules derived from a few key indicators (RSI, stochastic oscillator, disparity) are sufficient to generate profitable strategies. For example, for the Dow Jones index, the annualized return is 32.30% compared to 7.225% for the passive approach.
- The approach offers **parsimony**: few rules but interpretable and actionable.
- Effectiveness varies across markets, reflecting the heterogeneous nature of concept drift. Reported annualized returns are: FTSE 73.76% vs 4.793%, IBOVESPA 43.581% vs 10.718%, JALSH 41.635% vs 1.660%, NIFTY 39.876% vs 4.004%, NIKKEI 38.250% vs 4.445%, SSE 65.318% vs 15.887%.

This study demonstrates the potential of association rules in stock index prediction. By combining parsimony, robustness, and performance, they provide an attractive alternative to complex models, especially for investors seeking simple and explainable signals.

“Stock market prediction using weighted inter-transaction class association rule mining and evolutionary algorithm” (2022)

Chen, Mo, and Zhang [20] (2022) presents a method for predicting stock movements that combines Weighted Inter-Transaction Class Association Rules (WICAR) with an evolutionary algorithm based on Genetic Network Programming (GNP). The goal is to extract rules with predictive power for stock price changes, taking into account the dynamic importance of transactions (e.g., volume), to improve both forecast accuracy and investment strategy profitability.

The association rule extraction method with GNP relies on two key concepts: the **Judgment Node Chain (JNC)** and the **sliding window**.

Judgment Node Chain (JNC)

A JNC consists of a series of judgment nodes linked together. Each node allows a decision to continue either along the *yes-side* or the *no-side* branch. A JNC is limited in the number of nodes, constraining the depth of generated rules. As in evolutionary algorithms, the genetic operators used are: *selection*, *crossover*, and *mutation*. The evolutionary process proceeds as follows:

1. Initialize a random population.
2. Evaluate the fitness function of each individual.
3. Generate new individuals via selection and genetic operations.
4. Replace the old population with the new one.
5. Repeat until the stopping condition is met.

Each JNC corresponds to a transaction, and the judgment nodes determine the attributes related to stock price changes. For example, A_1 may represent the direction of stock 1 on a given day, while A_2 corresponds to stock 2 on the same day.

Sliding window

To obtain inter-transaction association rules, the concept of a **sliding window** is introduced, which groups multiple JNCs into a multi-transaction. The window width corresponds to the number of JNCs combined. At each window shift, a new JNC is considered to generate rules.

Candidate rule extraction

The search process follows attributes within the sliding window:

- If an attribute condition is satisfied, follow the *yes-side* branch.
- Otherwise, follow the *no-side* branch to a new starting node in the next transaction.

This produces rules of the type:

$$A_1^0(\text{up}) \cap A_2^1(\text{down}) \cap \cdots \cap A_m^w(\text{up}) \quad \Rightarrow \quad C(\text{up}).$$

Superscripts indicate the temporal lag in days relative to the starting day. The window then slides to the next transaction, and the process repeats.

Weighted criteria

Each judgment node has a weight, considered during traversal. This allows defining **weighted support** (wsp) and **weighted confidence** (wconf):

$$wsp(r) = \frac{AC_k([A_1][A_2]\dots[A_n])}{Ante[0][0]\dots[0] - S_{\max}(r)},$$

$$wconf(r) = \frac{AC_k([A_1][A_2]\dots[A_n])}{Ante([A_1][A_2]\dots[A_m])}.$$

Here, $Ante[0][0]\dots[0]$ is the total number of sliding windows in the database; $S_{\max}(r)$ is the maximal interval covered by the set of attributes in rule r, used to revise support in inter-transaction association rule extraction, since $Ante[0][0]\dots[0]$ does not represent the total number of transactions.

Additionally, a χ^2 criterion is introduced to measure dependence between antecedent and consequent:

$$\chi^2(r) = \frac{N'(z' - x'y')^2}{x'y'(1-x')(1-y')}.$$

Fitness function and genetic operators

The fitness function combines dependence strength (χ^2), the number of attributes in the antecedent, and a reward for new rules:

$$F = \sum_{r\in R} \left(\chi^2(r) + 10 \cdot (n_{ante}(r) - 1) + \alpha_{new}(r)\right).$$

The applied genetic operators are:

- **Crossover**: exchange of nodes and connections between individuals.
- **Mutation**: modification of node functions and connections.

Thus, a large population and genetic operators generate a large number of association rules, stored in *rule pools*.

Empirical evaluation uses 30 Dow Jones stocks, with training data (2015–2018) and test data (2018–2019). Results show that the weighted version (WICAR+GNP) outperforms the conventional unweighted method in terms of prediction accuracy and simulated returns (average and maximum returns reported per sliding window and per classifier). In general, the weighted association rule method helps investors improve profits. Using a one-day sliding window, the weighted method produced an average return of 14.7% in the test period, while the conventional method achieved 13.4%. Similar results were obtained using 2- and 3-day sliding windows.

A 2-day sliding window with the weighted method provided an average return of 19.5%, compared to 15.3% for a 3-day window. For the conventional method over the same period, average returns were only 15.5% and 13.9% for 2- and 3-day windows, respectively.

In conclusion, the GNP+WICAR combination provides a framework to integrate temporal structure (inter-transactions), dynamic weighting, and evolutionary optimization (GNP) into rule mining for finance. The methodology can be applied beyond markets (e.g., intrusion detection, traffic prediction) by adapting judgment functions, weights, and evolutionary rewards.

Table 2.4: Summary matrix of studies on association rule mining and stock prediction

Ref.	**Question / Hypothesis**	**Method**	**Data (period)**	**Size / Scope**	**Key Results**	**Limitations**	**Relevance for Thesis**
Kaur and Dharni [52] (2022)	Effectiveness of association rules for predicting trends of international stock indices	Apriori algorithm with discretized technical indicators in Weka	Daily data 2008–2020 (Japan, USA, UK, India, Brazil, China, South Africa)	12 years of multi-market observations	Annualized returns from 32.30% to 73.76% depending on the index; parsimonious and interpretable approach	Sensitive to support/confidence thresholds	Demonstrates the viability of an explainable rule-mining approach for simple technical signals
Chen, Mo, and Zhang [20] (2022)	Improving stock prediction via weighted rules and evolutionary optimization	WICAR (Weighted Inter-Transaction Class Association Rules) + GNP (Genetic Network Programming)	Dow Jones 2015–2019 (daily stocks)	30 stocks over 4 years (train/test)	Annual returns of 14.7% (1-day), 19.5% (2-day) vs 13.4–15.5% for conventional method; improved prediction accuracy	Complex model; sensitive to weight and genetic operator tuning	Illustrates combining weighted association rules and genetic evolution for temporal pattern detection

2.3 Related research in association rule mining

The ADWIN (Adaptive Sliding Window) algorithm is an exemplary concept drift detector [10]. It dynamically adjusts the size of its sliding window online, depending on the degree of variation detected in the data it contains [10]. In streaming scenarios, the combinatorial explosion occurs when attempting to maintain a data structure capable of representing frequent itemsets while updating incrementally. On one hand, memory constraints prevent exhaustive storage of all frequent itemsets; on the other, any omission could prevent detection of itemsets that become frequent later. Therefore, a compact structure retaining essential information about frequent items in the current window is necessary [21]. Efficient memory management is also imperative to avoid overflows or errors in continuous processing [70]. Additionally, complexity management is essential when detecting interesting itemsets—whether frequent [61], rare [46], closed [21], maximal [50], or generators [32]—to extract concise and robust association rules [55].

The MOMENT algorithm relies on a sliding window to extract *Frequent Closed Itemsets* (FCI). It uses a *Closed Enumeration Tree* (CET), a compact memory structure that tracks FCIs and itemsets at the boundary between FCIs and other itemsets—allowing minimized computational load while maintaining true real-time update capability [61]. Incremental updates are performed by merging nodes representing non-closed but promising itemsets, with support maintained via intersections of tidsets (transaction identifiers), whereas decrements rely on partitioning computations to detect and remove obsolete itemsets [21, 70]. The NEW-MOMENT extension [62] improves performance by encoding tidsets as bit vectors, reducing both memory and processing time.

Sequential patterns are extracted using the PREFIXSPAN algorithm [78], which applies prefix-projection: only prefixes are traversed, and their postfix subsequences are projected and explored from the projected databases. This strategy builds local frequent patterns while controlling complexity. Two projection variants are proposed to optimize efficiency: level-wise projection and bi-level projection.

The MFPAS algorithm (Maximal Frequent Pattern with Apache Spark) [5] leverages Spark for dynamic streams (DDS) and large databases (TDB). It uses prime number encoding to accelerate arithmetic operations and builds an in-memory structure called the *ASP-tree*, more compact than FP-GROWTH trees [41] or lattices. Efficient pruning, respecting partial downward closure, reduces search space and the number of frequent candidates [50, 3]. A major strength of this approach is native scalability via Spark's parallelism.

Finally, the CICLAD algorithm [70] offers a notable compromise between memory and streaming efficiency. It relies on inverted lists for each element to quickly access Closed Itemsets (CI), and stores evolving intersections as tries defining equivalence classes. Obsolete CIs are identified via a demotion mechanism that adjusts or removes their support. When dynamic intersections produce new CIs, only the minimal canonical element (intersection of the class) is kept. During updates, navigating the trie identifies relevant intersection nodes via cardinality tests; an intersection smaller than a reference value results in inserting the new CI into the inverted list of elements along the `path(n)`. Promoting a CI simply increases its support. Compared to MOMENT [21], CICLAD shows significantly higher performance—up to 40 times faster on retail data, with even larger gains for datasets containing rare data [21].

The FGC-STREAM algorithm [72, 71] outperforms MOMENT [21] and STREAM-GEN [32] on several datasets, including retail, both in execution time and memory usage. It uses a contraction/expansion principle for the sliding window to automatically evict obsolete transactions, improving memory management. Furthermore, it outputs Frequent Generator Itemsets (FGI), useful for classification of association rules, thereby improving classification accuracy in categorical data streams [32].

2.3.1 Operation of the FGC-Stream Algorithm

We use the *FGC-Stream* algorithm to perform data mining on a continuous (*stream*) flow of financial data transformed into technical financial indicators. This algorithm extracts both Frequent Closed Itemsets (FCIs) and their generators (FGIs), providing a more compact and expressive representation of frequent itemsets. *FGC-Stream* operates using a dynamic sliding window, which can expand or contract, thereby optimizing the expansion procedure derived from Formal Concept Analysis (FCA).

When the window contracts, the algorithm explores updated results following the evolution of classes. Unlike batch-processing methods, stream mining algorithms only have access to a subset of transactions limited to the current window. This sliding window keeps only recent transactions in memory and evicts older ones during expansion or contraction.

A major challenge in stream data mining is concept drift, where the data distribution changes over time. This project aims to extract frequent closed, maximal itemsets as well as their generators to construct relevant association rules.

Formally, FCIs are defined via an equivalence relation induced by the set of transactions supporting an itemset. These FCIs correspond to the distinct maximal elements of the underlying equivalence classes, whereas FGIs represent their minimal elements. Moreover, a frequent itemset X is maximal if no frequent itemset Y exists such that $X \subset Y$.

FCIs and FGIs play a crucial role in applications such as compiling trading rules to exploit upward movements of financial assets.

FGC-Stream uses a trie structure storing the upper set by: (1) distinguishing types of frequent classes and extracting new FCIs via intersections, and (2) discovering emerging frequent classes through local pattern enumeration.

A frequent itemset (FI) extraction problem is based on a set of items $I = \{a_1, a_2, \ldots, a_n\}$.

A transactional database D is defined over I. Each transaction $t = (tid, X)$ associates an identifier *tid* with a set of items $X \subseteq I$. An itemset is a set of items, while a tidset denotes a set of transaction identifiers.

The quality of an itemset is measured by its support, $\sigma(X) = |\tau(X)|$, where $\tau(X)$ is the set of transactions containing X. A minimum support threshold ς (or `min_supp`) is used to discriminate frequent from infrequent itemsets.

Thus, $\wp(I)$, the power set of I, is partitioned into the frequent set $F(D, \varsigma)$ and its infrequent complement $iF(D, \varsigma)$. The anti-monotonicity of τ guarantees that any subset of a frequent itemset is frequent, and that $F(D, \varsigma)$ is closed.

The function τ induces an equivalence relation on $\wp(I)$: two itemsets X and Z are equivalent ($X \sim Z$) if they have the same image $\tau(X) = \tau(Z)$.

Each equivalence class $[X]_D$ has a unique maximum, called the Closed Itemset (CI), as well as a set of minima, also called its Generators (GIs). In FCA, closed itemsets, referred to as *intents*, are defined by $X = \kappa(X)$ where $\kappa = \tau \circ \iota$. Here, κ is a closure operator since τ and ι form a Galois connection.

The closure of an itemset X is the smallest closed superset containing X.

From an evolutionary perspective, a holistic approach considers that global changes result in transitions between partitions of $\wp(I)$. From a coarse partition associated with D, the system migrates to a refined partition related to D^+.

In D^+, each new FCI X splits its class $[X]_D$ into two parts. The closure $Y = \kappa_D(X)$ corresponds to the maximum of $[X]_D$, called the donor class, which subdivides into a new class

$[X]_{D+}$ containing X and its subsets, and a residual class $[Y]_D$ with the remaining members. This division results from support increases induced by new transactions t_n, while the residual class retains unchanged supports.

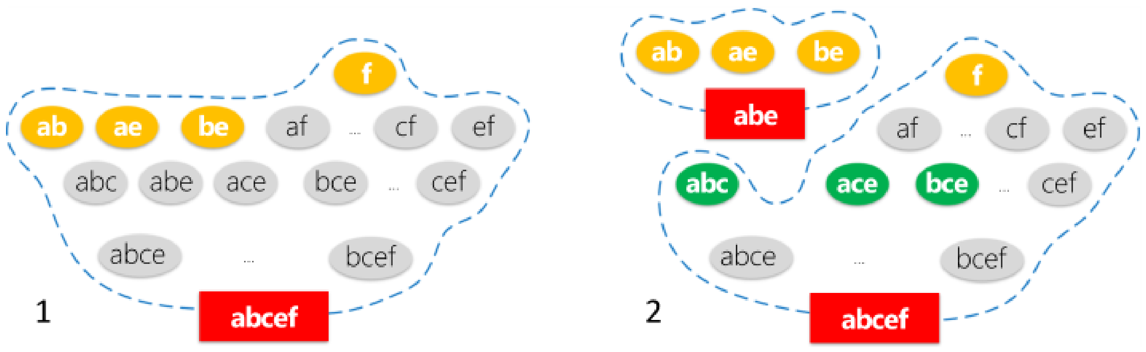


Figure 2.1: FGC-Stream: Donor Class

When a donor class (Figure 2.1) loses some of its generator itemsets (GIs), referred to as "stripped," the resulting residual class may retain some of these original GIs or not. These are called *residual generator itemsets*.

Moreover, support evolution no longer strictly follows the initial pattern because a new category of classes within D must be considered. So far, attention has focused only on the duality between donor and non-donor classes, the latter being defined as those containing no non-empty subset of members belonging to transaction t_n. However, among these non-donor classes, some require special handling based on their support. Specifically, this category is split into two subsets: on one hand, classes composed exclusively of subsets of t_n, called *promoted*, and on the other, remaining classes containing no subsets of t_n.

In a dual contraction context, removing an obsolete transaction t_o leads to a new set $D^- = D - t_o$. The evolution of closed (CI) and generator (GI) sets between D and D^- follows a pattern similar to that observed during expansion. All CIs and GIs in D included in t_o have their support adjusted accordingly, which may result in losing their minimal or maximal status within the corresponding classes in D^-.

Symmetrically to expansion where obsolete CIs are replaced by new ones, during contraction, downgraded CIs evolve oppositely to promoted CIs.

So far, the threshold ς, set to 1, has not been a crucial parameter in analyzing significant itemset evolution. However, when ς takes higher values, class-centered evolution becomes more heterogeneous. A major concern involves itemsets crossing the so-called "waterline" determined by ς.

It is also noteworthy that in the expanded window D^+, new classes appear by splitting from larger classes in D, resulting in supports slightly higher than the original donor class. Therefore, the donor–derived class pair may straddle this waterline. Some promoted classes then move from infrequent to the relevant portion of the class collection. These newly created or promoted classes from the infrequent zone are called *jumper classes*. Their respective FCIs and FGIs in D^+ are also referred to as "jumper ones."

Conversely, during window contraction D^-, some classes may see their frequency drop, going from frequent to infrequent. These classes are called *diving classes* (or *divers*).

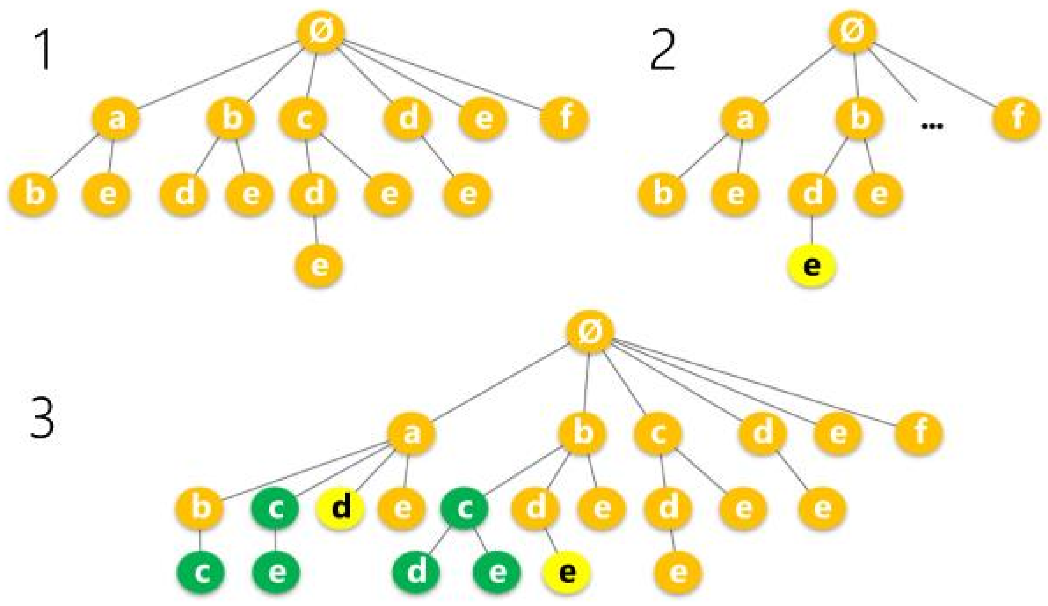


Figure 2.2: FGC-Stream: Trie progression

The progression of the trie (Figure 2.2) for the closed generator itemsets (FGIs) occurs in three distinct phases: first, the initial state before any update; second, the point at which the first *jumper* is identified; and finally, the final state after the update is complete. In the associated illustrations, FGIs corresponding to *jumpers* are shown in yellow, newly created FGIs appear in green, while other FGIs remain orange.

The contraction process in FGC-Stream is divided into two major steps: the first involves a recursive descent through the FGI trie, aimed at marking the classes encountered as *diving*, *downgraded*, or *annexed*, respectively. The second step, post-processing, includes the removal of classes identified as *diving* and the handling of *annexed* and *absorbing* class pairs.

Regarding the algorithmic complexity analysis of FGC-Stream+, three main cost components can be identified: (1) generation of FGI candidates and their validation against residual FGIs from the donor class, (2) detection of FGIs classified as *jumpers*, and (3) computation of closures for the classes associated with these *jumpers*.

2.3.2 Market Regime Detection

Modeling market regimes is a central issue in quantitative finance. Financial markets alternate between calm and unstable phases, bullish or bearish trends, and low or high volatility. The ability to identify these *regimes* in real-time and ideally anticipate them provides a decisive advantage for portfolio management and systematic investing. Pomorski and Gorse [81] (2023) propose an integrated approach combining three powerful but distinct instruments:

- the **Markov Switching Regression (MSR)** model from econometrics,
- the **Hidden Markov Model (HMM)** similar to machine learning approaches,
- the **Kaufman Adaptive Moving Average (KAMA)** from technical analysis.

The goal is threefold: (1) capture the **statistical volatility dynamics** (MSR), (2) represent **stochastic transitions between states** (HMM), and (3) integrate a measure of **trend directionality and efficiency** (KAMA). The text highlights the limitations of each tool when used in isolation and demonstrates how their combination significantly improves regime detection and classification.

The MSR, introduced by Hamilton [40] (1989), aims to model financial time series subject to abrupt structural changes. Unlike classical models (ARMA, GARCH), which assume stationary dynamics, the MSR allows model parameters to vary according to a **latent state** governed by a Markov chain [54].

Thus, the returns r_t of an asset are represented as:

$$r_t = \mu_{S_t} + \phi_{S_t} r_{t-1} + \sigma_{S_t}\varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0,1), \tag{2.1}$$

where $S_t \in \{1, 2, ..., K\}$ denotes the latent regime at time t. Each regime has its own mean (μ), autoregressive coefficient (ϕ), and standard deviation (σ).

Hidden Markov Chain

Regimes follow a first-order Markov law:

$$P(S_t = j \mid S_{t-1} = i) = p_{ij}, \tag{2.2}$$

with p_{ij} the transition probability between state i and j. The transition matrix P must be stochastic (each row sums to 1).

Economic Interpretation

In a financial context:

- a regime may correspond to a **calm period**: low volatility, moderately positive returns;
- another regime may represent a **crisis**: high volatility, negative returns;
- extensions allow for intermediate or specific regimes (bull, bear, stagnation).

The author emphasizes MSR's ability to **reproduce the persistence of market phases**, but also its limitations: it mainly identifies regimes **ex-post**. For example, during the March 2020 crash, the model confirmed the regime change only after several days of losses had already occurred.

This approach is robust for statistically characterizing regimes and is well-grounded in academic literature (Guidolin [39], 2011), but may involve detection latency and difficulty capturing directionality (bull vs bear).

Kaufman Adaptive Moving Average (KAMA)

Traditional moving averages pose a dilemma:

- a short moving average reacts quickly but produces many false signals,
- a long moving average filters noise but reacts too slowly.

Kaufman [51] (1995) proposes an adaptive solution that adjusts its reaction speed according to the **trend strength**.

The central idea is to compute an **efficiency ratio** (ER):

$$ER_t = \frac{|P_t - P_{t-n}|}{\sum_{j=1}^{n} |P_t - P_{t-j}|}, \tag{2.3}$$

with n a time window. This indicator measures the **linearity of movement**:

- $ER \approx 1$ if the price moves monotonically (strong trend),
- $ER \approx 0$ if the price oscillates without direction (noisy market).

The **smoothing factor** is then adjusted:

$$SC_t = \left[ER_t(SC_{fast} - SC_{slow}) + SC_{slow}\right]^2, \tag{2.4}$$

where SC_{fast} and SC_{slow} are constants related to short and long periods. Finally:

$$KAMA_t = KAMA_{t-1} + SC_t(P_t - KAMA_{t-1}). \tag{2.5}$$

KAMA adds a directional dimension absent from MSR:

- if KAMA rises, a bullish phase is identified;
- if it falls, a bearish phase is detected;
- if it stagnates, there is no exploitable trend.

This flexibility avoids false signals during ranges and reacts faster to genuine trends.

Hidden Markov Model (HMM)

The HMM generalizes MSR: instead of assuming regimes follow a Markov law directly observable via process parameters, observations are assumed to be generated by a probabilistic process conditional on an unobserved latent state.

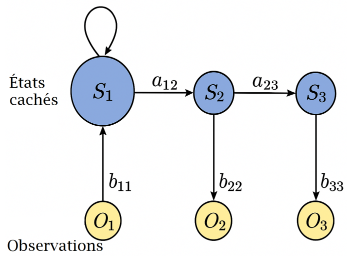


Figure 2.3: HMM structure

Formally, an HMM is defined by:

- a transition matrix $A = (a_{ij})$ between latent states,

- an emission matrix $B = (b_j(o_t))$, giving the probability of observing o_t given state s,
- an initial state distribution π.

Two essential algorithms:

- **Forward-Backward**: computes the probability of observation sequences.
- **Baum-Welch** (EM) [9] (1966): estimates A and B from data.
- **Viterbi** [**91**] (1967): identifies the most probable state sequence ex-post.

It offers high flexibility and can model non-Gaussian observation distributions. However, like MSR, HMM mainly detects regimes after their occurrence; its ex-ante predictive power remains limited.

MSR + KAMA: Toward an Enhanced Classification

The author proposes using MSR to identify **high vs low variance periods** and then refine this classification with KAMA to distinguish direction (bull vs bear). Thus, instead of two regimes, a four-regime typology is obtained:

1. Low variance, bullish trend (calm and upward market).
2. Low variance, bearish trend (slowly declining market).
3. High variance, bullish trend (volatile rebound).
4. High variance, bearish trend (persistent crisis).

KAMA parameters (window n, smoothing bounds SC_{fast} and SC_{slow}) are optimized via cross-validation and clustering (K-means) to minimize the **misclassification rate**, comparing detected regimes to known market phases.

- On equities and bonds: clear performance improvement, higher adjusted Sharpe ratio.
- On commodities and currencies: less pronounced but competitive performance.
- Transitional regimes (high variance/bull and low variance/bear) improve detection granularity but are not directly exploited in the strategy.

The HMM is used for comparison:

- It offers additional flexibility in modeling return distributions.
- However, it shares the same limitation as MSR: mainly ex-post detection.

The author demonstrates that **integrating a trend indicator like KAMA is essential** to move from purely statistical detection to an *operational* regime classification.

Key contributions:

- Combines econometric rigor (MSR) and technical flexibility (KAMA).
- Improves granularity: 4 regimes instead of 2.
- Greater operational relevance for trading strategies.

However, the approach remains primarily *nowcasting* rather than true ex-ante prediction. Integration with supervised learning algorithms (Random Forest, neural networks) is suggested, opening the possibility of exploiting transitional regimes via derivatives.

The combination of MSR, KAMA, and, for comparison, HMM represents a significant methodological advance in financial regime detection. MSR captures volatility dynamics, KAMA refines directionality, and HMM provides a machine-learning perspective. The integrated approach allows a shift from simple statistical identification to an operational classification directly applicable for asset management.

Market regime signals used as features in our dataset: Bearish with high variance, Bearish with low variance, Bullish with high variance, Bullish with low variance.

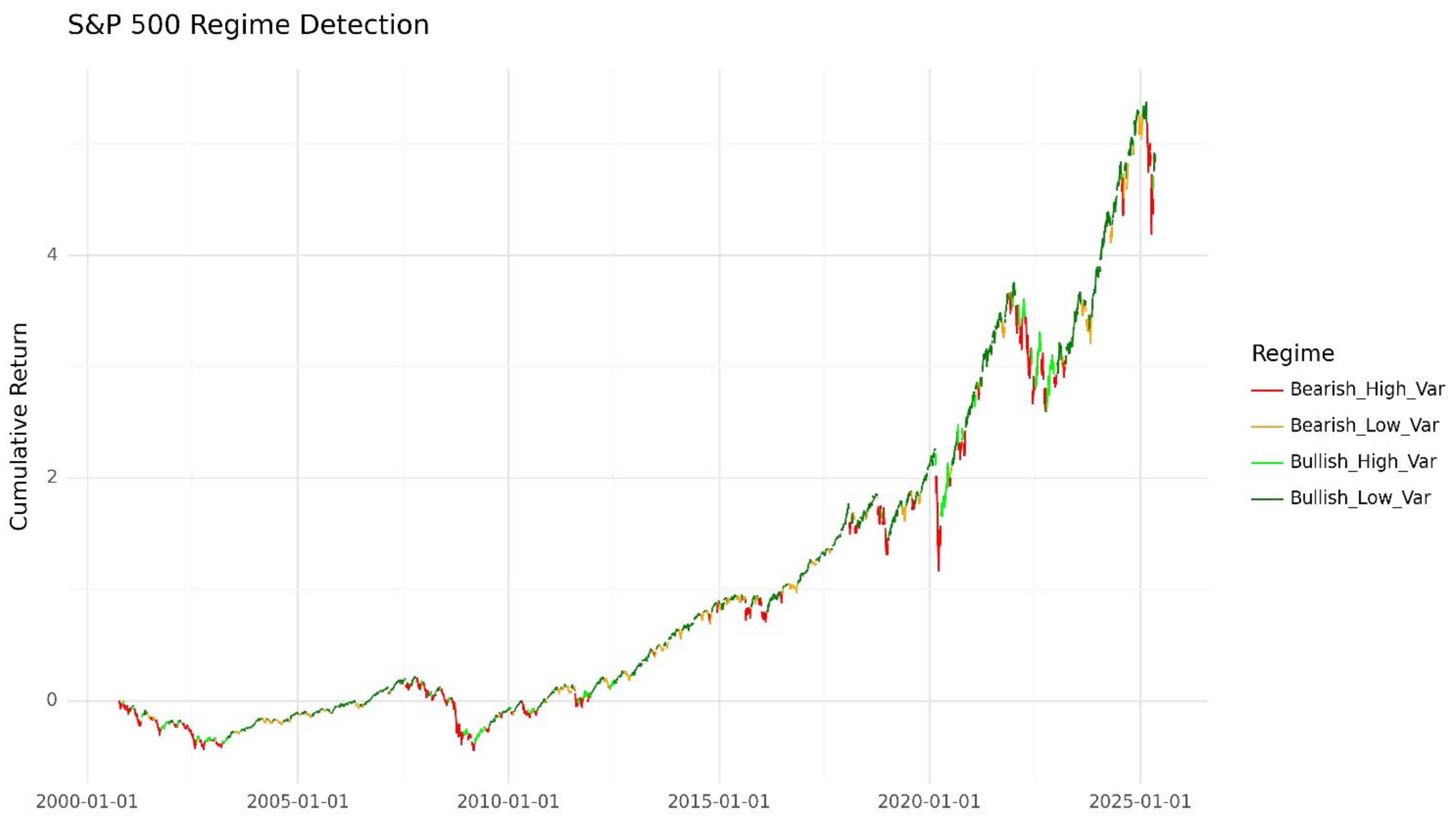


Figure 2.4: Market regime detection using MSR and KAMA

2.4 Critical Analysis

The analysis of existing work highlights several notable advances in applying machine learning and data mining methods in finance. Models such as Random Forest, LSTM, and Extra Trees have demonstrated a strong ability to extract relevant signals, with remarkable classification accuracy of 86.08% [76], and to improve risk-adjusted returns relative to benchmark indices. However, these approaches present several structural limitations.

First, most of these models suffer from a lack of interpretability. In a portfolio management context, it is essential to justify investment decisions to institutional investors or regulators. Deep neural networks or ensemble models such as XGBoost, although effective, often operate as "black boxes" that are difficult to explain.

Second, classical supervised classification approaches are highly dependent on the quality and stability of the selected features. Financial technical indicators can quickly lose relevance due to concept drift, a common phenomenon in constantly evolving markets. These methods therefore require frequent updates and complex recalibration to remain effective.

Third, research on association rules applied to financial markets has shown promising results, particularly in terms of parsimony and interpretability of extracted patterns. However, these methods have mostly been applied to aggregated historical data, and rarely in a continuous streaming context. Financial markets generate massive volumes of dynamic data, requiring algorithms capable of incremental learning and real-time updating. Moreover, existing incremental extraction methods (ADWIN, Moment, CICLAD) remain underexplored in finance, despite the theoretical appeal of the FGC-Stream algorithm for handling concept drift in real time.

2.5 Research Justification

These observations reveal a promising methodological avenue: high-performing ensemble models (Random Forest, LGBM, XGBoost) lack explainability; interpretable association rules extracted using the Apriori algorithm lack adaptability and have only been applied to a limited set of technical indicators (momentum, trend, and reversal) on stock indices in the study by Kaur and Dharni [52]; and streaming methods capable of handling concept drift have not yet been exploited in a financial context.

This research is therefore motivated by the goal of bridging this gap, by integrating incremental association rules (FGC-Stream) with more diverse and adaptive financial signals (momentum, trend, volatility, fundamentals, volume, liquidity, relative index performance, market regimes) on U.S. stocks composing the S&P 500. The objective is to reconcile interpretability, adaptability, and performance in an algorithmic portfolio management system, capable of adjusting to changing market conditions while offering improved robustness and enhanced explainability of investment decisions. Ultimately, the study aims to demonstrate that this approach delivers superior risk-adjusted performance.

Chapter 3

Methodology and AI System Design

3.1 AI System Architecture

The AI system architecture, illustrated in Figure 3.1, begins with the extraction of raw data on prices, volumes, and fundamental indicators of U.S. companies listed on the S&P 500. The data is then transformed into technical indicators, as explained later. The dataset is split into 80% for training and 20% for testing. Training is conducted on four different models—Random Forest, XGBoost, LightGBM, and LSTM—to extract the top 30 features. These features form a new dataset for training the FGC-Stream model to extract association rules for trading signals. These rules are ultimately used to produce the backtest and obtain financial performance and risk metrics.

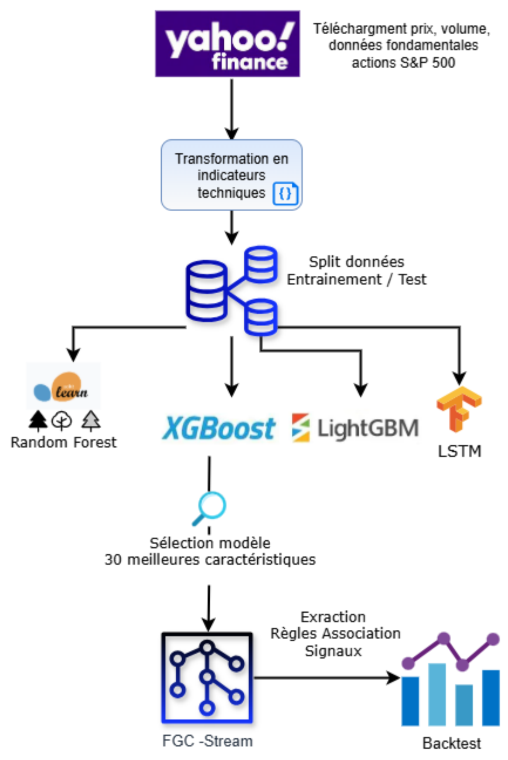

yahoo! finance
Téléchargment prix, volume, données fondamentales actions S&P 500
Transformation en indicateurs techniques
Split données Entrainement / Test
learn
Random Forest
XGBoost
LightGBM
LSTM
Sélection modèle 30 meilleures caractéristiques
Exraction Règles Association Signaux
FGC -Stream
Backtest

Figure 3.1: AI System Architecture

3.2 Data Description

Price, volume, and fundamental metric data for U.S. companies listed on the S&P 500 are downloaded from Yahoo Finance, covering a 10-year historical period.

Once the raw data is transformed into technical indicators, the continuous numerical data is normalized over its historical values, discretized, and ordered into quintiles or buy/neutral/sell signals to make it more suitable for classification models. The KBinsDiscretizer from scikit-learn is used to transform continuous variables (e.g., price, return) into discrete categories or classes. Variables are divided into bins so they are more interpretable for classifiers (e.g., LightGBM, XGBoost, Random Forest). The selected discretization strategy uses the k-means algorithm, producing irregular intervals not based on quantiles. The k-means algorithm identifies centroids representing the ”typical” points of the distribution. Bin boundaries are defined based on the influence zones of each centroid, which better captures natural data clusters, especially for highly skewed or multimodal distributions.

Some additional data, such as the VIX volatility index, is transformed to detect volatility anomalies. A VIX anomaly is defined as a Z-score exceeding 2.5 over a 90-day rolling window, in which case the binary signal becomes 1. The VIX is relevant because, following market-wide events (e.g., COVID-19, tariffs), most stocks tend to move in the same direction on that day.

For technical indicators, data is tracked up to 4 days back to capture past sequences. For fundamental indicators, sequences are captured over longer periods—5, 10, 30, 45, 60, and 90 days—since financial statements are typically released quarterly.

Additionally, the horizon for certain technical indicators is varied to capture short-term, medium-term, and long-term trends. For example, the Rate of Change (ROC) is calculated over 1-week, 1-month, 3-month, and 12-month intervals to detect persistent patterns across multiple time windows.

Data is also sampled to retain monotonic upward or downward movements over 3 consecutive days, capturing significant movements that are relevant for studying the technical signals or features triggering them. The target variable is the quintile class of S&P 500 returns over the following two weeks.

3.3 Preprocessing and Feature Engineering

3.3.1 Technical Indicators

The following technical indicators will be used, as reported in the research of Pomorski [80] (2024) and Kaur and Dharni [52] (2022).

Stochastic $\%K$

$$\%K = 100 \times \frac{C - L_n}{H_n - L_n}$$

where C is the current closing price, and L_n and H_n are the lowest and highest prices over the last n periods, respectively.

The Stochastic %K measures the relative position of the closing price within a price range over a given period. It indicates momentum strength and dynamics, helping to detect overbought or oversold conditions [60].

Stochastic $\%D$

$$\%D = \text{Simple Moving Average of } \%K \text{ over } m \text{ periods}$$

The Stochastic %D is a moving average of %K that smooths fluctuations and generates crossover signals, facilitating the detection of entry and exit points [60].

Stochastic Slow $\%D$

$$\text{Slow } \%D = \text{Simple Moving Average of } \%D \text{ over } p \text{ periods}$$

The Slow Stochastic %D is an even more smoothed version of %D, which further reduces market noise and improves signal reliability [60].

Commodity Channel Index (CCI)

$$CCI = \frac{P_t - SMA(P_t)}{0.015 \times \text{Mean Deviation}}$$

where $P_t = \frac{H+L+C}{3}$ is the typical price, and $SMA(P_t)$ is the simple moving average of P_t.

The CCI measures the deviation of the current price from its moving average, normalized so most values lie between -100 and +100. It identifies overbought or oversold conditions [58].

Relative Strength Index (RSI)

$$RSI = 100 - \frac{100}{1 + RS}, \quad RS = \frac{\text{Average Gains}}{\text{Average Losses}}$$

The RSI measures the relative strength of gains versus losses over a given period, oscillating between 0 and 100. It identifies overbought (>70) or oversold (<30) conditions [93].

Williams %R (William-R)

$$\%R = -100 \times \frac{H_n - C}{H_n - L_n}$$

Williams %R measures the closing price's position relative to the highest and lowest prices over n periods on a reversed scale (-100 to 0), signaling overbought and oversold zones [94].

Disparity Index - 5 days

$$\text{Disparity-5} = \frac{C}{SMA_5(C)} \times 100$$

The Disparity Index measures the percentage deviation of the closing price from its 5-day simple moving average, indicating temporary overvaluation or undervaluation relative to recent trends [75].

Disparity Index - 10 days

$$\text{Disparity-10} = \frac{C}{SMA_{10}(C)} \times 100$$

Similar to the 5-day version but using a 10-day average. It is smoother and reflects medium-term deviations from trend [75].

Oscillator Percentage (OSCP)

$$OSCP = \frac{MA_s - MA_l}{MA_s} \times 100$$

where MA_s is the short-term moving average and MA_l the long-term moving average l periods ago.

The OSCP measures the percentage difference between two moving averages, expressing price momentum as a simple oscillator [1].

Momentum

$$Momentum = C - C_n$$

Momentum measures the absolute difference between the current price and the price n periods ago. It indicates the speed of price movement and trend strength [**momentum**].

3.3.1.1 Ultimate Oscillator (UO)

Let:

- L = daily lowest price,
- H = daily highest price,
- C = daily closing price,
- C_q = previous day's closing price,
- $o_1 = 7$, $o_2 = 14$, $o_3 = 28$.

Define:

$$CQ = C - \min(L, C_q)$$

$$US = \max(H, C_q) - \min(L, C_q)$$

The ratios for each period are:

$$B_{o_1} = \frac{\sum_{u=1}^{o_1} CQ}{\sum_{u=1}^{o_1} US}, \quad B_{o_2} = \frac{\sum_{u=1}^{o_2} CQ}{\sum_{u=1}^{o_2} US}, \quad B_{o_3} = \frac{\sum_{u=1}^{o_3} CQ}{\sum_{u=1}^{o_3} US}$$

Finally, the Ultimate Oscillator is defined as:

$$UO = \frac{(B_{o_1} \times 4) + (B_{o_2} \times 2) + B_{o_3}}{7}$$

The Ultimate Oscillator combines three time horizons to reduce false signals. Values above 70 may indicate overbought conditions, while values below 30 suggest oversold conditions [95].

Percentage Volume Oscillator (PVO)

$$PVO = \left(\frac{\text{EMA}_{\text{short}} - \text{EMA}_{\text{long}}}{\text{EMA}_{\text{long}}}\right) \times 100$$

The PVO measures the difference between two exponential moving averages of volume, expressed as a percentage of the long EMA. It helps identify changes in volume momentum, confirming or contradicting price trends [89].

Rate of Change (ROC)

$$\begin{aligned}
&C = \text{closing price}, \quad C_{t-n_i} = \text{closing price } n_i \text{ days ago},\\
&n_1 = 21, \quad n_2 = 120, \quad n_3 = 252,\\
&ROC_{1m} = \frac{C}{C_{t-n_1}} - 1,\\
&ROC_{6m} = \frac{C}{C_{t-n_2}} - 1,\\
&ROC_{12m} = \frac{C}{C_{t-n_3}} - 1.
\end{aligned}$$

ROC measures the percentage change in price over a given period, indicating the speed of price movement. Positive ROC signals an upward trend, while negative ROC indicates a downward trend [90].

Average Directional Index Rating (ADXR)

$$ADXR = \frac{\text{Current ADX} + \text{ADX from n periods ago}}{2}$$

The ADXR is a moving average of the ADX, providing a smoother measure of trend strength. It helps assess whether a trend is strengthening or weakening [69].

Chande Momentum Oscillator (CMO)

$$CMO = \left(\frac{\sum_{\text{ups}} - \sum_{\text{downs}}}{\sum_{\text{ups}} + \sum_{\text{downs}}}\right) \times 100$$

The CMO measures the relative strength of upward versus downward movements over a period, oscillating between -100 and +100 to indicate overbought or oversold conditions [18].

Plus Directional Indicator (+DI)

$$+DI = 100 \times \left(\frac{\text{Sum of Upward Movements}}{\text{Sum of Total Movements}}\right)$$

The +DI measures the strength of upward movements within a trend. It is used together with the -DI to assess trend direction [48].

Minus Directional Indicator (-DI)

$$-DI = 100 \times \left(\frac{\text{Sum of Downward Movements}}{\text{Sum of Total Movements}}\right)$$

The -DI measures the strength of downward movements within a trend. It is used with +DI to determine the dominant market direction [48].

Triple Exponential Average (TRIX)

$$TRIX = \left(\frac{\text{EMA}_3(i) - \text{EMA}_3(i-1)}{\text{EMA}_3(i-1)}\right) \times 100$$

TRIX is a momentum indicator that filters out insignificant price movements by applying three successive exponential smoothings. It is used to identify trends and potential reversals [49].

3.3.1.2 Volatility

Average True Range (ATR)

$$ATR = \frac{1}{n}\sum_{i=1}^{n} TR_i$$

where $TR_i = \max(H_i - L_i, |H_i - C_{i-1}|, |L_i - C_{i-1}|)$ is the true range for period i, H_i is the high, L_i is the low, and C_{i-1} is the previous closing price.

ATR measures market volatility by accounting for price gaps over a given period. It is useful for assessing the risk level of an asset [92].

Bollinger Bands

$$\text{Upper Band} = MA_n + k \times \sigma_n, \quad \text{Lower Band} = MA_n - k \times \sigma_n$$

where MA_n is the n-period simple moving average, σ_n is the standard deviation over n periods, and k is a multiplicative factor (commonly $k = 2$).

Bollinger Bands measure volatility by plotting bands around the moving average, helping to identify overbought or oversold conditions [13].

Ulcer Index (UI)

$$\begin{aligned}
&H = \text{daily high}, \quad C = \text{closing price}, \quad n = 21,\\
&Max\_H = \max(H) \text{ over a period of } n \text{ days},\\
&PD = \frac{C - Max\_H}{Max\_H} \times 100,\\
&PD_s = \frac{\sum_{t=1}^{n} PD^2}{n},\\
&Ulcer = \sqrt{PD_s}.
\end{aligned}$$

The Ulcer Index measures downside risk by considering both the depth and duration of price declines. It is useful for assessing investment stability [47].

Mean-Std-Semi

$$\text{Semi-Deviation} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \mu)^2 \cdot \mathbf{1}_{r_i<\mu}}$$

where r_i is the return in period i, μ is the mean return, and $\mathbf{1}_{r_i<\mu}$ is an indicator function equal to 1 if $r_i < \mu$ and 0 otherwise.

Semi-deviation measures the volatility of returns below the mean, providing insight into downside risk.

Mean-Std-Month

$$\text{Monthly Average} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

where r_i is the return in period i for a month.

The monthly average provides an overview of asset performance over a monthly horizon.

Mean-Std-Qtr

$$\text{Quarterly Average} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

where r_i is the return in period i over a quarter.

The quarterly average provides insight into asset performance over three months.

Beta

$$\beta = \frac{\text{Cov}(R_e, R_m)}{\text{Var}(R_m)}$$

where R_e is the asset return, R_m is the market return, $\text{Cov}(R_e, R_m)$ is the covariance between the asset and market returns, and $\text{Var}(R_m)$ is the market return variance.

Beta measures an asset's volatility relative to the market. A beta greater than 1 indicates higher volatility than the market [**beta**].

Volatility Spike in VIX (Volatility Index)

$$\sigma^2 = \frac{2}{T}\sum_i \frac{\Delta K_i}{K_i^2} e^{RT} Q(K_i) - \frac{1}{T}\left(\frac{F}{K_0} - 1\right)^2,$$

T = time to expiration,

F = S&P 500 forward level derived from SPX option prices,

K_i = strike price of the i-th out-of-the-money option,

ΔK_i = average between strikes immediately above and below K_i,

K_0 = strike immediately below F,

R = risk-free rate,

$Q(K_i)$ = midpoint of bid-ask for strike K_i.

The VIX measures the market's implied volatility on the S&P 500 using option prices. It is often called the "fear index" [77].

3.3.1.3 Volume

On-Balance Volume (OBV)

$$OBV_t = OBV_{t-1} + \begin{cases} V_t, & \text{if } C_t > C_{t-1} \\ 0, & \text{if } C_t = C_{t-1} \\ -V_t, & \text{if } C_t < C_{t-1} \end{cases}$$

where C_t is the closing price at period t, and V_t is the volume.

OBV accumulates volume by adding when the price closes up and subtracting when it closes down. It helps detect buying or selling pressure by examining the volume-price relationship, useful for confirming trends [37].

Force Index

$$FI_t = (C_t - C_{t-1}) \times V_t$$

where C_t is the closing price and V_t is the volume.

The Force Index measures price movement strength by combining price change with volume. It reflects the power behind price moves, helping to identify trend strength or potential reversals [25].

Money Flow Index (MFI)

$$MFI = 100 - \frac{100}{1 + \text{Money Ratio}}, \quad \text{with Money Ratio} = \frac{\text{Positive Money Flow}}{\text{Negative Money Flow}}$$

where

$$\text{Typical Price} = \frac{H + L + C}{3}$$

and the money flow is calculated by multiplying the Typical Price by the volume for each period.

The MFI is an oscillator that uses both price and volume to detect overbought or oversold conditions. It is similar to RSI but incorporates volume, giving a more complete measure of buying or selling pressure [2].

Chaikin Oscillator

$$CO_t = EMA_3(ADL_t) - EMA_{10}(ADL_t)$$

where ADL_t is the Accumulation/Distribution Line at period t, and EMA_n is the exponential moving average over n periods.

The Chaikin Oscillator measures the momentum of money flow by analyzing the difference between two exponential moving averages of the Accumulation/Distribution Line. It is used to detect changes in momentum before price moves [17].

3.3.1.4 Trend

Moving Average Convergence Divergence (MACD)

$$MACD_t = EMA_{12}(P_t) - EMA_{26}(P_t)$$

where $EMA_n(P_t)$ is the exponential moving average of the price P_t over n periods.

MACD measures the difference between two EMAs of different periods to assess price momentum. It is used to identify trend changes and generate buy or sell signals via crossovers [6].

EMA Difference (EMA-Diff)

$$EMA\_diff_t = EMA_{short}(P_t) - EMA_{long}(P_t)$$

Similar to MACD, it is the difference between short-term and long-term EMAs.

This difference reflects convergence or divergence between short-term and long-term trends, used to detect potential reversal or continuation signals.

Know Sure Thing (KST)

$$\begin{aligned}
&RCMA\#1 = \text{Simple Moving Average (SMA) over 10 periods of ROC over 10 periods},\\
&RCMA\#2 = \text{SMA over 10 periods of ROC over 15 periods},\\
&RCMA\#3 = \text{SMA over 10 periods of ROC over 20 periods},\\
&RCMA\#4 = \text{SMA over 15 periods of ROC over 30 periods},\\
&KST = (RCMA\#1 \times 1) + (RCMA\#2 \times 2) + (RCMA\#3 \times 3) + (RCMA\#4 \times 4),\\
&\text{Signal Line} = \text{SMA over 9 periods of KST}.
\end{aligned}$$

KST combines several rates of change over different periods with weights, providing a trend signal and a signal line. It is useful to detect major trend reversals [59].

Detrended Price Oscillator (DPO)

$$DPO_t = P_{t-\frac{n}{2}+1} - SMA_n(P_t)$$

where $SMA_n(P_t)$ is the simple moving average over n periods.

The DPO removes long-term price trends to focus on short-term cycles. It helps identify cyclical highs and lows without the influence of the overall trend [11].

Vortex Indicator

$$\begin{aligned}
&+VM = |\text{Current High} - \text{Previous Low}|,\\
&-VM = |\text{Current Low} - \text{Previous High}|,\\
&+VM_n = \text{sum over n periods of } +VM,\\
&-VM_n = \text{sum over n periods of } -VM,\\
&\text{True Range (TR)} = \max\Big(\text{Current High} - \text{Current Low},\\
&\qquad |\text{Current High} - \text{Previous Close}|,\\
&\qquad |\text{Current Low} - \text{Previous Close}|\Big),\\
&TR_n = \text{sum over n periods of TR},\\
&+VI_n = \frac{+VM_n}{TR_n},\\
&-VI_n = \frac{-VM_n}{TR_n}.
\end{aligned}$$

The Vortex Indicator measures the strength of upward and downward movements by comparing movements between successive highs and lows. It helps identify the relative strength of dominant trends [23].

3.3.1.5 Performance Against Benchmark

Index Outperformance 1 Month (Index-outperf-1M)

$$\text{Outperf}_{1M} = \frac{R_{\text{index},1M} - R_{\text{benchmark},1M}}{|R_{\text{benchmark},1M}|}$$

where $R_{\text{index},1M}$ is the 1-month return of the index, and $R_{\text{benchmark},1M}$ is the 1-month return of the benchmark.

This metric evaluates the relative outperformance of an index compared to a benchmark over one month, allowing quick identification of whether the index has performed better or worse than the reference.

Index Outperformance 1 Week (Index-outperf-1W)

$$\text{Outperf}_{1W} = \frac{R_{\text{index},1W} - R_{\text{benchmark},1W}}{|R_{\text{benchmark},1W}|}$$

where $R_{\text{index},1W}$ is the 1-week return of the index.

This metric indicates the index's outperformance relative to a benchmark over one week, useful for short-term analyses and tactical adjustments.

Index Outperformance 1 Quarter (Index-outperf-1Q)

$$\text{Outperf}_{1Q} = \frac{R_{\text{index},1Q} - R_{\text{benchmark},1Q}}{|R_{\text{benchmark},1Q}|}$$

where $R_{\text{index},1Q}$ is the quarterly return.

This metric reflects relative outperformance over a quarter, allowing evaluation of medium-term trends.

Beta 1 Month (Beta-1M)

$$\beta_{1M} = \frac{\text{Cov}(R_{\text{asset},1M}, R_{\text{market},1M})}{\text{Var}(R_{\text{market},1M})}$$

where returns are calculated over 1 month.

Beta measures the sensitivity of an asset's return relative to the market over a monthly period, indicating its systematic risk.

Beta 1 Quarter (Beta-1Q)

$$\beta_{1Q} = \frac{\text{Cov}(R_{\text{asset},1Q}, R_{\text{market},1Q})}{\text{Var}(R_{\text{market},1Q})}$$

with quarterly returns.

Quarterly beta provides a view of systematic risk over a longer period, relevant for medium-term investment strategies.

Monthly Correlation (Monthly-correl)

$$\rho_{1M} = \frac{\text{Cov}(R_{A,1M}, R_{B,1M})}{\sigma_{R_A,1M}\sigma_{R_B,1M}}$$

where $R_{A,1M}$ and $R_{B,1M}$ are the monthly returns of two assets or indices.

Monthly correlation measures the linear relationship between two return series over one month, essential for diversification and risk management.

Quarterly Correlation (Quarterly-correl)

$$\rho_{1Q} = \frac{\text{Cov}(R_{A,1Q}, R_{B,1Q})}{\sigma_{R_A,1Q}\sigma_{R_B,1Q}}$$

with quarterly returns.

This correlation evaluates the linear dependence between assets over a quarterly period, helping to understand co-movement over longer horizons.

3.3.1.6 Liquidity

Amihud Illiquidity Measure

$$\begin{aligned}
&C = \text{closing price}, \quad V = \text{current volume}, \quad V_a = C \times V, \quad n = 21, \\
&C_p = \text{previous price}, \quad r = \frac{C}{C_p} - 1, \\
&EMA_1 = \left[V_a \times \frac{2}{1+n}\right] + EMA_{1p} \times \left[1 - \frac{2}{1+n}\right], \\
&EMA_2 = \left[V_a \times \frac{2}{1+n}\right] + EMA_{2p} \times \left[1 - \frac{2}{1+n}\right], \\
&Amih\_L = \frac{|EMA_1|}{EMA_2} \times 100000.
\end{aligned}$$

The Amihud illiquidity measure quantifies illiquidity by relating price changes to traded volume. A higher value indicates a less liquid market, where small volumes can move prices significantly. It is useful for assessing asset liquidity and the implicit transaction cost [4].

Kyle's Lambda (Kyle-L)

$$\begin{aligned}
&C = \text{closing price}, \quad V = \text{current volume}, \quad C_p = \text{previous price}, \\
&r = \frac{C}{C_p} - 1, \quad n = 21, \\
&S_t = \begin{cases} 1 & \text{if } r_t > 0, \\ -1 & \text{if } r_t < 0, \end{cases} \\
&V_a = S_t \times \ln(C \times V), \\
&R_{i,t} = \alpha_i + \beta_i V_{a,t} + \varepsilon_{i,t}, \\
&Kyle\_L = \text{moving average of } R_{i,t} \text{ over } n \text{ periods.}
\end{aligned}$$

Kyle's Lambda measures the sensitivity of price to traded volume, representing the impact of volume on price changes in financial markets. A high Kyle-L indicates a less liquid market, where buy or sell orders move prices strongly [57].

3.3.1.7 Corwin-Schultz Spread

$$
\begin{aligned}
&L = \text{daily low}, \quad H = \text{daily high}, \\
&n_1 = 5, \quad n_2 = 21, \quad H_{Lsq} = \ln\left(\frac{H}{L}\right), \quad c = 3 - 2\sqrt{2}, \\
&H_{\max} = \text{rolling max of } H_{Lsq} \text{ over } n_1 \text{ days}, \\
&L_{\min} = \text{rolling min of } H_{Lsq} \text{ over } n_1 \text{ days}, \\
&H_{\text{Lsqsum}} = \text{rolling sum of } H_{Lsq} \text{ over } n_1 \text{ days}, \\
&B = \text{rolling average of } H_{\text{Lsqsum}} \text{ over } n_2 \text{ days}, \\
&G = \ln\left(\frac{H_{\max}}{L_{\min}}\right)^2 - \left(\frac{\sqrt{2}-1}{\sqrt{2}}\right) B, \\
&A = \left(\frac{\sqrt{2}+1}{3}\right)\left(\frac{G}{c}\right), \\
&\hat{A} = \begin{cases} A & \text{if } A > 0, \\ 0 & \text{if } A < 0, \end{cases} \\
&CS = 2 \times \left(\frac{e^{\hat{A}} - 1}{e^{\hat{A}} + 1}\right).
\end{aligned}
$$

The Corwin-Schultz spread is a robust illiquidity measure based on the implicit bid-ask spread derived from extreme price variations over multiple days. It estimates implicit transaction costs while accounting for volatility and extreme prices, useful for assessing asset liquidity in high-frequency price data [22].

3.3.2 Fundamental Indicators

Earnings Per Share (EPS)

$$EPS = \frac{Net\ Income - Preferred\ Dividends}{Weighted\ Average\ Shares\ Outstanding}$$

where $Net\ Income$ is the company's net income, $Preferred\ Dividends$ are preferred dividends, and $Weighted\ Average\ Shares\ Outstanding$ is the weighted average number of common shares outstanding.

EPS measures the portion of net profit attributable to each common share. It is a key indicator to evaluate profitability per share and compare performance across companies [79].

Price-to-Earnings Ratio (PE)

$$PE = \frac{Market\ Price\ per\ Share}{Earnings\ per\ Share}$$

where $Market\ Price\ per\ Share$ is the current market price of a share.

The PE ratio indicates how much investors are willing to pay per unit of earnings. It is used to assess stock valuation: a high PE may indicate expectations of strong future growth, whereas a low PE may signal undervaluation or risk [8].

For example, technology companies, often characterized by high expected growth, generally have high PE ratios, reflecting investors' expectations of significant future earnings increases.

In contrast, consumer staples companies, such as food or household products, operate in mature sectors with more stable and predictable growth. These companies tend to have lower PE ratios, reflecting lower growth expectations.

Thus, a high PE in the tech sector may indicate a high valuation due to growth expectations, while a lower PE in the consumer staples sector reflects a more moderate valuation aligned with stable growth.

3.4 Machine Learning for Feature Extraction

3.4.1 Classification Evaluation Metrics

Accuracy

Accuracy measures the proportion of observations correctly classified out of the total number of observations. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

where TP represents true positives, TN true negatives, FP false positives, and FN false negatives. Accuracy is an intuitive and widely used metric, but it can be misleading when classes are imbalanced, as a model that always predicts the majority class could achieve high accuracy without properly detecting the minority class.

Specificity

Specificity measures the model's ability to correctly identify negative instances:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{3.2}$$

This metric complements recall and is essential when correctly identifying negatives is critical.

Balanced Accuracy

Balanced accuracy accounts for class imbalance by averaging sensitivity and specificity:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{3.3}$$

It allows fair comparison of models in contexts where some classes are much less represented than others.

Precision

Precision measures the proportion of correct positive predictions among all instances predicted as positive. It assesses the model's reliability when predicting the positive class and is defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.4}$$

where TP is the number of true positives and FP the number of false positives. High precision means that when the model predicts a positive event, it is highly likely to be correct. This metric is particularly useful in contexts where false alarms are costly, such as financial fraud detection, where a high number of false positives could lead to significant costs or unnecessary interventions.

F1 Score

The F1 score is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.5}$$

This metric is particularly useful when classes are imbalanced and one wants to balance precision and recall. It provides a single measure that reflects both the model's ability to correctly identify positives and to minimize false alarms.

In finance, evaluation metrics are essential to assess the performance of predictive models such as Random Forest, XGBoost, or LightGBM. For example, in predicting bank defaults or detecting fraud, special attention must be paid to false negatives, as missing an event can lead to significant losses. Using recall, precision, and the F1 score together provides a complete view of model performance [45].

A thorough understanding of evaluation metrics prevents misinterpretations. For instance, a model with high accuracy may mask poor performance on the minority class. It is therefore recommended to combine multiple metrics for a comprehensive assessment. Graphs such as the ROC curve or the confusion matrix complement these numerical measures and help visualize model performance across classes.

Evaluation metrics form the foundation for analyzing classification model performance. Accuracy, precision, recall, specificity, balanced accuracy, and the F1 score allow assessing different aspects of performance while accounting for class imbalance and the costs associated with errors.

3.4.2 Model Explainability

3.4.2.1 SHAP (*SHapley Additive exPlanations*)

SHAP is based on Shapley values [87] from game theory to fairly and locally exactly attribute a portion of a prediction to each feature. For a given instance, each feature is considered a player, and its average marginal contribution to the prediction is evaluated when joining all possible coalitions.

Shapley regression values provide a method to evaluate the importance of explanatory variables in linear models, particularly in the presence of multicollinearity. The approach involves retraining the model on different subsets of features, denoted F for the full feature set. An input feature has a high Shapley regression value when its effect on the model prediction is strong and independent of other features.

To measure this effect, one trains a model $f_{S\cup\{i\}}$ including the feature of interest i, and another model f_S without it. Predictions from these two models are compared, and the difference is attributed to feature i. Sets S and $S \cup \{i\}$ represent the feature values included in S and those obtained by adding feature i, respectively. This process is repeated for all possible subsets $S \subset F \setminus \{i\}$.

Shapley values are then calculated according to the classic Shapley formula:

$$\phi_i = \sum_{S\subset F\setminus\{i\}} \frac{|S|!\,(|F|-|S|-1)!}{|F|!}\left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right]$$

In the context of Shapley regression values [68], the function h_x maps the presence or absence of a feature in the original input space, where a value of 1 indicates the feature is included in the subset and 0 that it is excluded.

Shapley sampling values aim to explain any model by applying two principles: (1) use sampling approximations of the equation above, and (2) apply this method to any model while keeping the same approximations. This approach is called *model-agnostic* because it does not rely on the internal structure of the model. It allows estimating feature importance when retraining the model on all 2^n combinations would be computationally infeasible. The mathematical form remains the same as Shapley regression values, except that retraining at each iteration is unnecessary.

In summary, these methods provide a rigorous framework to fairly attribute feature importance in a model, even in the presence of strong correlations, while offering an approximate alternative when computational constraints exist.

SHAP formalizes attributions in an additive model [68]:

$$g(z') = \phi_0 + \sum_{k=1}^{N} \phi_k \, z'_k$$

with $z' \in \{0,1\}^N$ indicating feature presence, and ϕ_0 an effect for each explanatory variable. This formulation links SHAP to LIME while rigorously enforcing the three key properties, which LIME does not generally guarantee.

To make attribution calculation practical, SHAP proposes KernelSHAP, a model-agnostic method based on weighted regression using a kernel derived from Shapley combinatorial weights. KernelSHAP estimates ϕ_k without enumerating all coalitions but remains computationally expensive in high dimensions.

TreeSHAP [67] leverages tree-based models (forests, boosting) to compute Shapley values in polynomial time using dynamic programming along decision paths. TreeSHAP also enables computing feature interactions and detailed visualizations aggregated over attributions.

The choice between KernelSHAP and TreeSHAP depends on model type and input dimension: KernelSHAP is useful for non-tree black-box models, albeit slower, while TreeSHAP is highly recommended for tree-based models. KernelSHAP assumes feature independence, which may bias attributions when strong correlations exist.

3.4.2.2 LIME (*Local Interpretable Model-agnostic Explanations*)

LIME (*Local Interpretable Model-agnostic Explanations*) explains individual predictions by locally building a simple model, such as a linear regression, that closely approximates the complex model's behavior. Introduced by Ribeiro, Singh, and Guestrin [85], it helps understand why a black-box model made a decision for a particular feature, focusing only on its immediate neighborhood.

The underlying idea is intuitive: around a target instance x_0, perturbed versions z' are generated — for example by activating or deactivating words in text, or masking superpixels in images — and the complex model f is observed under these variations. A simple interpretable model g, often sparse linear regression (or shallow tree), is then fit to minimize:

$$\xi = \arg\min_{g \in G} \; \mathcal{L}(f, g, \pi_{x_0}) + \Omega(g),$$

where $\mathcal{L}$ measures local fidelity (typically squared loss) weighted by a kernel π_{x_0} that favors perturbations near the instance, and $\Omega(g)$ controls model complexity for interpretability.

This ensures g remains simple and understandable while locally faithful to f. LIME is model-agnostic and versatile, usable with any classifier or regressor — from deep neural networks to random forests.

In implementation, interpretable data x'_0 are binary vectors indicating feature presence or absence (words, superpixels), and h_x maps these elements to actual or averaged values depending on the domain.

Despite its simplicity, LIME has limitations: locality implies that neighboring instances can yield very different explanations, and random perturbation generation can create instability. Theoretically, for a linear model, g's coefficients correspond to gradients, but improper parameter choices may hide important features.

Nevertheless, LIME is widely used in sensitive domains such as healthcare or finance, providing transparency and confidence where traditional models are opaque. Python and R libraries support interactive visualizations facilitating communication between data scientists and domain experts.

3.4.2.3 Boruta

Boruta is a robust *feature selection* method designed to identify not only a minimal subset but all truly informative attributes. Introduced by Kursa, Jankowski, and Rudnicki [56], it leverages Random Forests while introducing a mechanism comparing features with *shadow attributes* — random reference features — to distinguish informative variables from random fluctuations.

The idea is simple yet powerful: for each original feature, a shadow copy is created with randomly shuffled values across observations, breaking correlation with the target. The enriched dataset is fed to a Random Forest, producing an importance measure (often a Z-score from permutation decrease in accuracy). Each real feature is compared against the maximum importance among shadows (MIRA). Features consistently outperforming shadows are deemed important; otherwise, they are removed.

The process is iterative and statistically rigorous: multiple runs with different permutations count "hits" where a feature exceeds MIRA, then a binomial test evaluates if this is significantly above or below chance, classifying each feature as *important*, *unimportant*, or *tentative*.

Boruta's strengths: *model-agnostic* (requires only an importance measure), *complete* (identifies all significant features), and *statistically rigorous*. Implementations exist in R and Python, with visual tools to track decisions through iterations.

Theoretically, Boruta addresses a weakness of classical raw importance approaches, which may overrate uninformative features by chance. Introducing shadows creates an internal random reference, avoiding arbitrary thresholds or assumed normality of Z-scores from forests.

Practically, Boruta is valuable in high-dimensional contexts — e.g., biomedical or genomic data — where identifying all relevant features, not just a small subset, is crucial for model interpretability and reliability.

3.4.3 Random Forest

The *Random Forest* algorithm is a supervised learning method based on an ensemble (*ensemble learning*) of multiple decision trees. It was introduced by Breiman [14] in 2001. The central idea is to combine the simplicity and interpretability of decision trees with the robustness and generalization ability provided by aggregation methods (*bagging*) and random feature selection.

A decision tree (*Decision Tree*) is a hierarchical structure composed of:

- **Decision nodes**: where a condition on an explanatory variable is tested;
- **Branches**: representing the possible outcomes of the test;
- **Leaves**: corresponding to a prediction (class or value).

The CART (*Classification and Regression Tree*) algorithm builds the tree top-down by recursively partitioning the feature space.

Two common impurity measures are often used when building a decision tree:

1. **Gini**:
$$GI(E) = 1 - \sum_{j=1}^{o} q_j^2$$
where q_j is the proportion of examples of class j in node E.

2. **Shannon Entropy**:

$$SE(E) = -\sum_{j=1}^{o} q_j \log_2 q_j$$

These measures assess node purity: the lower the value, the more homogeneous the node.

Information Gain

The *information gain* (IG) is defined as:

$$IG(E) = MI(E) - \Big(\frac{t_1}{t} MI(E_1) + \frac{t_2}{t} MI(E_2)\Big)$$

where MI is the chosen impurity measure, t the number of samples in E, and t_1, t_2 the numbers of samples in the child nodes.

Although simple and interpretable, CART trees suffer from:

- **Overfitting**: low bias but high variance;
- **Sensitivity to noise**: small variations in the data can produce very different trees.

Bagging: Bootstrap Aggregating

Bagging involves:

1. Drawing D bootstrap samples from the training set;
2. Training a tree on each sample;
3. Aggregating predictions (majority vote or mean).

This reduces variance but does not necessarily decorrelate the trees.

Random Forest improves bagging by introducing **random feature selection**:

- At each node, $R \ll O$ features are randomly selected from the O available;
- The best feature in this subset is chosen for the split.

This **decorrelates** the trees and improves overall performance.

1. For $i = 1$ to D:
   - Draw a bootstrap sample C_i;
   - Build a CART tree by selecting R random features at each node;
   - Add the tree to the forest.
2. To predict: aggregate the predictions of the D trees.

Key Hyperparameters

- `n_estimators`: number of trees;
- `max_depth`: maximum tree depth;
- `min_samples_split`: minimum samples to split a node;
- `min_samples_leaf`: minimum samples in a leaf;
- `max_features`: number of features considered at each split.

Random forests offer advantages such as high accuracy, less overfitting than single trees, and good handling of missing and categorical data. However, they are less interpretable than a single tree and can be computationally expensive for large datasets.

Random Forest strikes an effective balance between robustness, accuracy, and implementation simplicity. Its ability to reduce variance while maintaining low bias makes it a suitable algorithm for this research. Below are the results obtained after training.

3.4.3.1 Selected Random Forest Hyperparameters

The best hyperparameters obtained through cross-validation are listed below:

`n_estimators = 225` - Number of trees in the forest. Higher numbers reduce variance but increase computation time.

`max_depth = 11` - Maximum depth of a tree. Controls complexity: shallow limits overfitting, deep captures more detail.

`min_samples_split = 67` - Minimum samples required to split a node. Larger values yield more regular trees and reduce overfitting.

`min_samples_leaf = 84` - Minimum samples in a leaf. Prevents leaves that memorize noise.

`max_features = sqrt` - Number of features considered for a split. Lower values increase randomness and tree diversity.

`bootstrap = True` - Indicates sampling with replacement. Produces more diverse trees and reduces overall variance.

`class_weight = "balanced"` - Adjusts class weights inversely proportional to class frequency, correcting imbalance and giving more importance to rare classes.

`criterion = gini` - Split quality criterion. ”gini” or ”entropy”; ”gini” is the most common and fastest.

The Random Forest model achieves an overall accuracy of 63%, indicating satisfactory classification performance across the test set, considering that stock market data are noisy. However, the recall for class 3 is 49%, meaning it only detects 49% of actual positive instances in this class. This reflects difficulty in capturing this category correctly. In sensitive contexts, such a low recall is problematic as the majority of positive cases go undetected.

Class	Precision	Recall	F1-Score
0	0.76	0.59	0.66
1	0.61	0.59	0.60
2	0.60	0.72	0.66
3	0.66	0.49	0.56
4	0.91	0.50	0.64
Accuracy			0.63
Macro Avg	0.71	0.58	0.63
Weighted Avg	0.63	0.63	0.62

Table 3.1: Random Forest classification report after Boruta feature selection

3.4.4 XGBoost

The *eXtreme Gradient Boosting* (XGBoost) algorithm, proposed by Chen and Guestrin [19] in 2016, has established itself as one of the most influential and powerful supervised learning methods in artificial intelligence and machine learning. Based on decision trees and *gradient boosting*, XGBoost stands out for its ability to efficiently handle large volumes of data, its speed, and remarkable accuracy. To understand the importance of XGBoost, it is useful to first review the theoretical foundations of gradient boosting, then examine the innovations that make XGBoost a reference method, before discussing its specific applications in financial research and investment.

Gradient boosting is based on the idea of combining multiple weak models (often shallow decision trees) to form a strong and accurate model. Unlike *bagging*, exemplified by the *Random Forest* algorithm, boosting proceeds sequentially: each new tree is built to correct the errors of the previous trees. This approach relies on minimizing a loss function, typically by gradient descent, hence the name *gradient boosting*. Among the early foundational works are AdaBoost (Freund and Schapire [30], 1997) and the theoretical developments by Friedman, Hastie, and Tibshirani [31] (2000) that established the mathematical foundations of the algorithm.

XGBoost extends this principle and optimizes it by introducing several technical innovations: explicit regularization to limit overfitting, an efficient memory management method for large datasets, parallelization during tree construction, and intrinsic handling of missing values. These features give XGBoost a significant comparative advantage over classical gradient boosting algorithms.

Tree-based boosting methods combine multiple additive regression trees to obtain a more robust and accurate prediction than a single tree. In this type of model, the final prediction for an instance is the sum of contributions from several functions f_k, each f_k being a function from the space of regression trees (CART). Unlike decision trees that produce discrete labels, each leaf of a regression tree carries a continuous score. XGBoost formalizes the learning of this ensemble of trees through the minimization of a regularized objective and additive optimization based on local approximations of the loss term.

Model and Regularized Objective

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ with $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$, the model is an additive sum:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \in \mathcal{F},$$

where $\mathcal{F}$ is the space of regression trees defined by structures q (which map a vector x to a leaf index $j \in \{1, \dots, T\}$) and leaf weights $w \in \mathbb{R}^T$. Each tree function can be written as $f(x) = w_{q(x)}$.

To learn these functions, we minimize the regularized objective:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k),$$

with regularization of the form

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|_2^2,$$

where T is the number of leaves in the tree, Ω penalizes structural complexity, and λ controls the norm of leaf weights. The term $l(\cdot, \cdot)$ is a convex differentiable loss function (e.g., squared error or log-loss for classification). The regularized objective encourages simple trees (few leaves) and smoothed leaf weights, reducing overfitting.

Additive Learning and Second-Order Approximation

The parameter space is non-Euclidean (it contains functions/trees), so we proceed via additive learning. At iteration t, assume the first $t-1$ trees and current predictions $\hat{y}_i^{(t-1)}$ are known. We seek a function f_t that minimizes

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\Big(y_i,\ \hat{y}_i^{(t-1)} + f_t(x_i)\Big) + \Omega(f_t).$$

Expanding the loss with a second-order Taylor approximation around $\hat{y}_i^{(t-1)}$ yields:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i) + \Omega(f_t),$$

where

$$g_i = \partial l(y_i, \hat{y}_i^{(t-1)}), \qquad h_i = \partial^2 l(y_i, \hat{y}_i^{(t-1)}).$$

Here, g_i and h_i are the gradient and second derivative (curvature) of the loss at the current point; they summarize the information needed to locally adjust predictions. Removing constants independent of f_t, the approximated objective becomes:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \Big[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t(x_i)^2\Big] + \Omega(f_t).$$

Leaf-Wise Grouping and Closed-Form Weight Solution

Let $q(\cdot)$ be the tree structure and T the number of leaves. For each leaf j, define

$$I_j = \{i \mid q(x_i) = j\},$$

the indices of instances in leaf j. If the leaf weight is w_j, then $f_t(x_i) = w_{q(x_i)}$, and the approximated objective grouped by leaf is:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} \Big[\Big(\sum_{i \in I_j} g_i\Big) w_j + \tfrac{1}{2}\Big(\sum_{i \in I_j} h_i + \lambda\Big) w_j^2\Big] + \gamma T.$$

For a fixed structure q, the optimal w_j is obtained analytically by setting the gradient to zero:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}.$$

Substituting w_j^* into the objective yields an optimal score for the structure:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\tfrac{1}{2}\sum_{j=1}^{T} \frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

This score is used to compare competing tree structures: lower $\tilde{\mathcal{L}}^{(t)}(q)$ (or higher loss reduction gain in practice) indicates a better structure.

Split Gain Criterion

Tree construction is usually greedy: starting from a leaf, candidate splits are evaluated on a feature by their gain. For a leaf with set I split into I_L (left) and I_R (right), the gain is:

$$\text{Gain} = \tfrac{1}{2}\left(\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda} \right) - \gamma.$$

The $-\gamma$ term penalizes adding leaves, while the fractions reflect gradient and curvature compatibility. The split with the highest positive gain is chosen.

Exact and Approximate Split Search

For continuous variables, enumerating all splits is costly. The exact algorithm sorts feature values and computes cumulative g and h to evaluate gains. For efficiency, XGBoost uses approximate splits: candidate thresholds (percentiles or bins) are proposed, and sums of g and h are computed per bin. Gains are evaluated only at proposed thresholds, reducing complexity while maintaining good split quality. This is particularly useful for parallelization and large datasets.

Techniques to Limit Overfitting

Beyond explicit regularization γ and λ, XGBoost uses:

Shrinkage (learning rate) After constructing tree f_t with weights w_j^*, contributions are multiplied by $\eta \in (0,1]$ (learning rate): $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$. Lower η reduces each tree's influence, allowing more trees without overfitting.

Column Subsampling Similar to Random Forest, a random subset of features is used per tree (or per split). Column subsampling reduces correlation between trees and overfitting from dominant features.

Second-order formulation (using g_i and h_i) allows updates scaled by local loss curvature: high h_i reduces the contribution of an instance, stabilizing optimization. Combined with structure regularization (γ), weight regularization (λ), learning rate (η), and column subsampling, XGBoost offers a coherent set of levers to control bias/variance.

The greedy tree construction remains heuristic: only a small portion of the tree space is explored. Empirical quality comes from gradient and second-order statistics, regularization, and variance reduction via subsampling.

XGBoost formalizes tree boosting as a regularized optimization problem solved additively with second-order Taylor approximations. Key elements—structure score based on gradient and curvature sums, closed-form leaf weights, split gain criterion, shrinkage, and column subsampling—contribute to a powerful and flexible algorithm. Understanding these components aids hyperparameter tuning and interpretation on real datasets.

Application in Investment

One of the areas where XGBoost has garnered the most interest is quantitative finance and financial market forecasting. Several empirical studies demonstrate the power of this algorithm for handling complex and noisy financial data.

Dey, Kumar, Saha, and Basak (2016) [24] used XGBoost to predict stock price direction, showing that the algorithm often outperforms traditional linear methods. Carmona, Climent, and Momparler (2019) [16] applied XGBoost to predicting bank failures in the U.S. banking sector. Yazdani (2020) [97] explored its application for predicting U.S. recessions, although in this case XGBoost performed less well than random forests, highlighting that the algorithm's effectiveness strongly depends on the nature of the problem.

In portfolio management, XGBoost has been used to construct portfolios based on machine learning signals. Its ability to capture nonlinear interactions between financial variables makes it particularly suitable for extracting predictive signals from large datasets, including macroeconomic data, sentiment indicators, or order flow information.

The main strengths of XGBoost lie in its robustness, its ability to handle massive and heterogeneous datasets, and its learning speed, which allows rapid model recalibration according to market changes. These features explain its widespread adoption in Kaggle competitions, where it has often dominated leaderboards.

However, some limitations should be noted. First, although more regularized than other gradient boosting algorithms, XGBoost is still prone to overfitting, especially in financial contexts where noise is high and data stationarity is often absent. Second, interpretability remains limited, despite tools such as SHAP (SHapley Additive exPlanations), which aim to clarify the role of variables in predictions. Finally, its effectiveness varies by problem: for recession detection, some studies suggest that random forests offer better trade-offs between recall and precision.

3.4.4.1 Selected XGBoost Hyperparameters

The best hyperparameters obtained via cross-validation are:

`n_estimators = 222` Number of trees (or boosting iterations). Higher values can improve accuracy but increase the risk of overfitting.

`max_depth = 6` Maximum tree depth. Deeper trees capture complex relationships but may overfit.

`learning_rate (eta) = 0.08` Learning rate. Lower rates improve generalization but require more trees.

`subsample = 0.8` Fraction of samples used per tree. Values below 1 introduce randomness and reduce overfitting.

`min_child_weight = 5` Minimum sum of instance weight in a leaf. Higher values simplify the model and prevent fitting to noise.

`gamma = 2` Minimum loss reduction required to split a node. Higher γ makes the model more conservative.

`reg_lambda = 2` L2 regularization on weights. Reduces model variance.

`reg_alpha = 0.4` L1 regularization on weights. Encourages sparsity in variable usage.

`objective` Objective function. For multiclass classification, `"multi:softprob"` is used.

`num_class = 5` Number of classes.

`eval_metric = "mlogloss"` Evaluation metric used during training. Multiclass log loss penalizes confident but wrong predictions, improving robustness.

Class	Precision	Recall	F1-Score
0	0.75	0.63	0.69
1	0.62	0.58	0.60
2	0.61	0.72	0.66
3	0.65	0.51	0.57
4	0.82	0.53	0.65
Accuracy			0.63
Macro Avg	0.69	0.60	0.63
Weighted Avg	0.64	0.63	0.63

Table 3.2: XGBoost classification report

The XGBoost model achieves an overall accuracy of 63%, indicating satisfactory classification capability on the test set, considering the noisy nature of financial market data. The highest precision was achieved for class 5 at 82%.

From the confusion matrix of the XGBoost model (figure 3.2), most values are concentrated along the diagonal, representing correct predictions. However, between classes 1 to 3, classification errors occur, with these classes being confused about one-third of the time, indicating more difficulty distinguishing among them. Since the last performance quintile (class 4) is the target of interest, errors in classes 0 to 3 are less costly for this application.

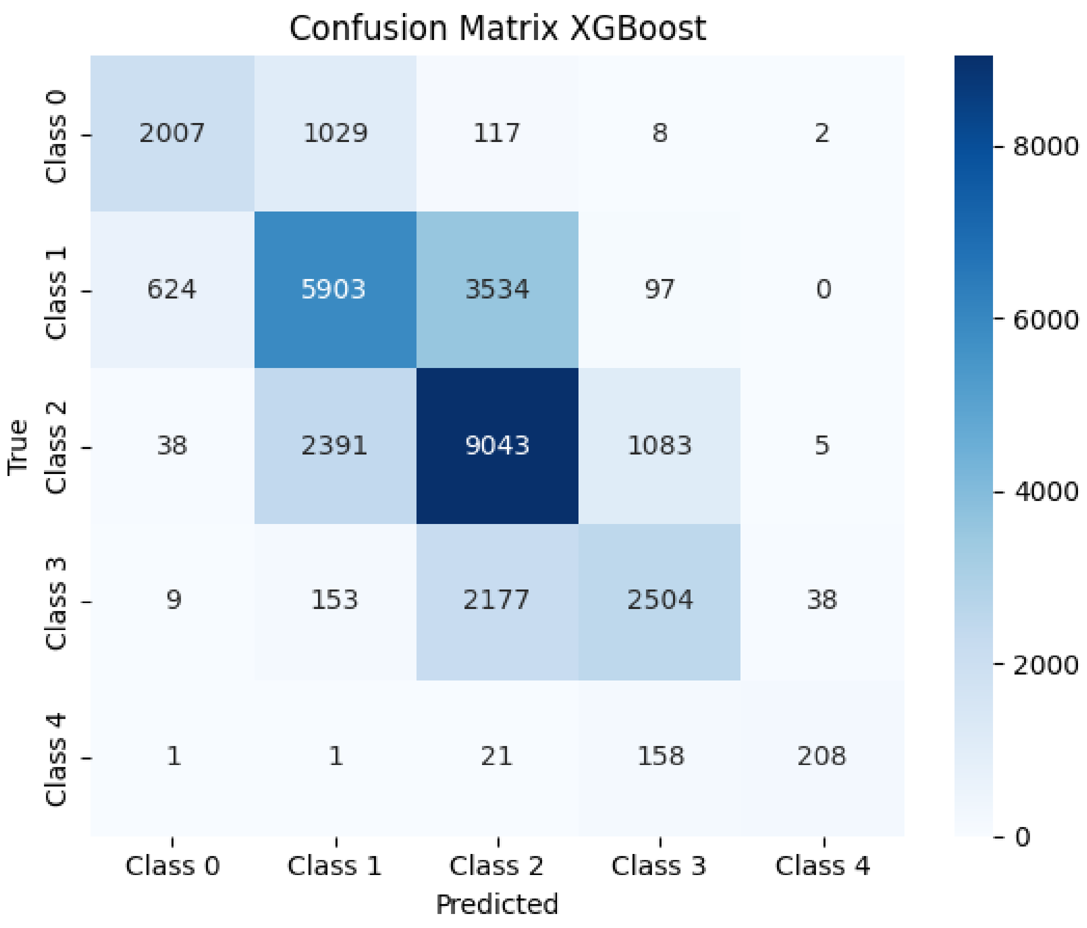


Figure 3.2: XGBoost confusion matrix

Regarding the 30 most important features obtained with the XGBoost model (figure 3.3), they are primarily past return quintiles across different frequencies—monthly and biweekly (50% importance). Following these are quarterly and weekly features. Market regime detection, VIX volatility anomalies, disparity measures, ROC and stochastic K&D, Chaikin, Momentum, and Beta also show significant importance. This suggests that past return quantiles partially predict future return quantiles, reflecting market momentum, while shorter-term reversals, such as abnormal short-term volatility over recent days, may also play a role.

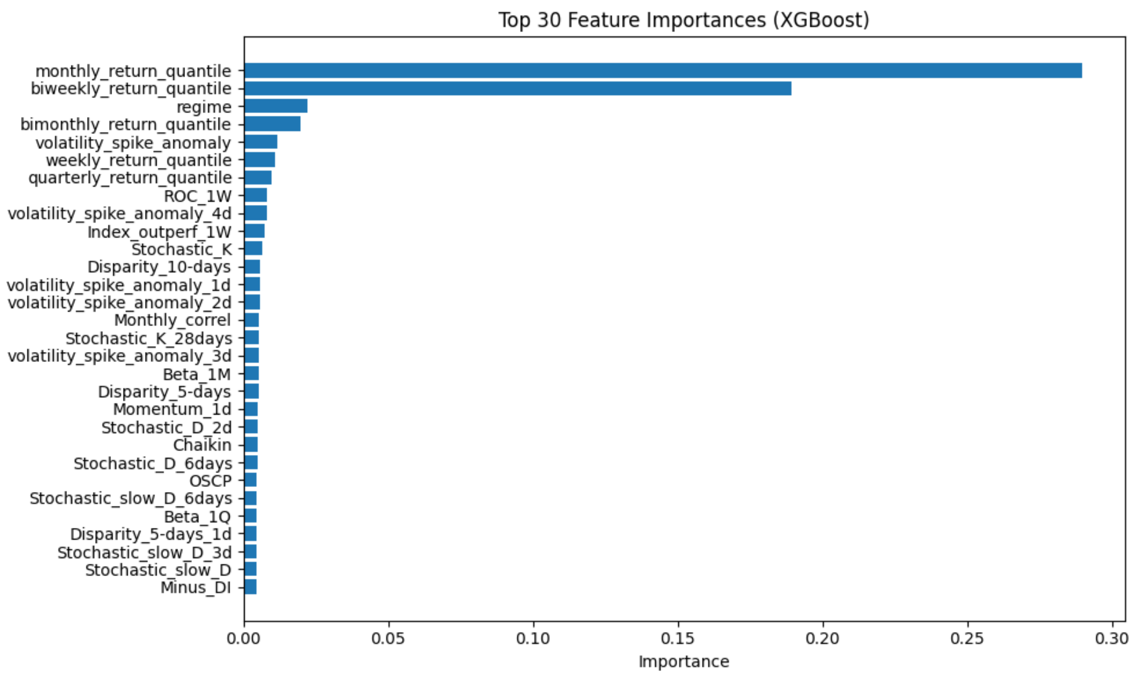


Figure 3.3: Top 30 XGBoost features

The ROC curve (figure 3.4) shows model performance by plotting the true positive rate (sensitivity) against the false positive rate at different thresholds. The AUC (Area Under Curve) summarizes this curve in a single score: the closer the AUC is to 1, the better the model's discriminative ability.

For the multiclass ROC curve (One-vs-Rest) of XGBoost—each class against all others, with the x-axis representing the false positive rate and the y-axis the true positive rate—there is clear separation between classes. The curve rises quickly toward the upper-left corner, indicating strong discriminative ability. The AUC is 0.98 for class 4 (the top performance quintile), 0.81 for class 1, 0.77 for class 2 (with a lower threshold, generating some false positives), and 0.89 for class 3, reflecting very good overall performance.

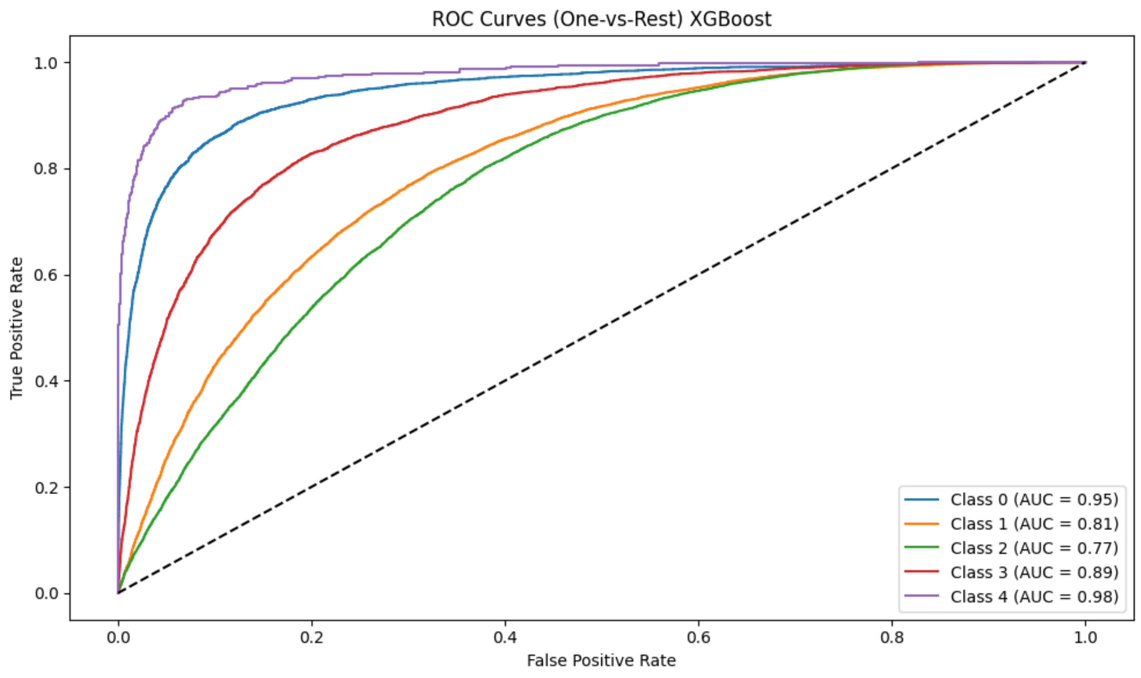


Figure 3.4: XGBoost ROC curve

3.4.5 Light GBM

Light Gradient Boosting Machine (LightGBM) [53] is a modern and highly optimized implementation of the Gradient Boosting algorithm on decision trees, proposed by Microsoft Asia Research Institute. Boosting is arguably one of the most influential machine learning ideas introduced in the past twenty years. Originally designed for classification problems, this paradigm has proven equally effective for regression tasks. The core idea of boosting relies on sequentially combining several so-called "weak" classifiers to form a strong ensemble. In this framework, boosting shares some superficial similarities with bagging and other ensemble approaches, but its foundations are fundamentally different.

The most popular boosting algorithm, proposed by Freund and Schapire (1997), is *AdaBoost* [29]. Consider a binary classification problem, where the output variable $Y \in \{-1, 1\}$. For a predictor vector X, a classifier $G(X)$ provides a prediction taking one of the two possible values. The training sample error rate is given by

$$\text{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i)),$$

and the expected error rate on new data is

$$\mathbb{E}_{X,Y}\Big[I(Y \neq G(X))\Big].$$

A weak classifier is defined as a model whose error rate is only slightly better than random guessing. The goal of boosting is to sequentially apply such a weak classifier on modified versions of the data, producing a series of models $G_m(x)$, for $m = 1, 2, \dots, M$. Their combination yields a strong classifier.

GBDT and Its Challenges

The *Gradient Boosting Decision Tree* (GBDT) is one of the most widely used algorithms due to its efficiency, accuracy, and interpretability. It has achieved state-of-the-art performance in many domains such as multi-class classification, click prediction, and learning to rank. However, the era of *big data* introduced new challenges. Modern datasets often have very large numbers of instances and features, creating a trade-off between computational efficiency and predictive accuracy.

In a classic GBDT implementation, for each feature, all instances must be scanned to estimate the information gain for every possible split point. This approach results in complexity proportional to the product of the number of features and the number of instances, making training extremely time-consuming on large datasets.

In GBDT, decision trees are learned sequentially by fitting residuals, i.e., the negative gradients of the loss function. The most computationally expensive part is finding the best split point. Popular methods include:

- The *pre-sorted algorithm*, which enumerates all possible split points. While exact, this method is memory- and time-intensive.
- The histogram-based algorithm, which groups continuous values into discrete bins and builds histograms to accelerate split search. This approach is more memory- and computation-efficient.

The overall cost can be estimated as:

$$\mathcal{O}(\#\text{data} \times \#\text{feature}) \quad \text{for histogram construction,}$$

$$\mathcal{O}(\#\text{bin} \times \#\text{feature}) \quad \text{for finding the best split.}$$

Since the number of bins is much smaller than the number of instances, histogram construction dominates the complexity.

To accelerate GBDT without sacrificing accuracy, two main ideas were introduced in *LightGBM*: *Gradient-based One-Side Sampling* (GOSS) and *Exclusive Feature Bundling* (EFB).

Gradient-based One-Side Sampling (GOSS)

In GBDT, there are no explicit weights for instances, unlike AdaBoost. However, the gradient computed for each instance provides valuable information: a high gradient indicates a poorly learned instance that contributes strongly to information gain, while a small gradient corresponds to a well-predicted instance.

GOSS keeps all instances with large gradients and randomly samples from those with small gradients. To correct the distribution bias introduced, a constant weight factor is applied to the retained small-gradient instances.

Formally, if a percentage a of the data with large gradients is retained, and b is sampled from the small gradients, then during gain calculation, the gradients of the small ones are multiplied by $\frac{1-a}{b}$.

The post-split variance is estimated by:

$$\tilde{V}_j(d) = \frac{1}{n}\left(\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2 \frac{1}{n_{j,l}(d)} + \left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2 \frac{1}{n_{j,r}(d)}\right),$$

where A denotes instances with large gradients, B denotes retained small-gradient instances, and l, r are the left and right partitions after the split.

Theoretical analysis shows this approximation is accurate for large datasets and significantly outperforms uniform random sampling.

Exclusive Feature Bundling (EFB)

In real-world applications, the number of features is often very high but sparse. Many features are nearly exclusive, i.e., they rarely take non-zero values simultaneously, such as one-hot encodings. EFB groups these mutually exclusive features into a single "bundle," reducing the effective dimensionality.

Optimal bundle construction is equivalent to a *graph coloring* problem: each feature is a vertex, and an edge is drawn if two features are not exclusive. Since this problem is NP-hard, a greedy heuristic is used to obtain a grouping with a constant approximation ratio.

During bundling, the value ranges of each feature are shifted to avoid overlap. For example, if feature A takes values in $[0, 10)$ and feature B in $[0, 20)$, we add an offset of 10 to B, making its domain $[10, 30)$. Features A and B can then be merged into a single bundled feature.

This approach reduces complexity from $\mathcal{O}(\#\text{data} \times \#\text{feature})$ to

$$\mathcal{O}(\#\text{data} \times \#\text{bundle}), \quad \text{with } \#\text{bundle} \ll \#\text{feature}.$$

Experiments show that LightGBM speeds up training compared to classic implementations while maintaining almost identical accuracy. GOSS improves generalization by focusing on difficult examples, while EFB drastically reduces computational cost by decreasing the number of features.

3.4.5.1 Selected LightGBM Hyperparameters

Here are the best hyperparameters obtained via cross-validation.

Hyperparameter	**Selected Value**	**Description**
boosting_type	gbdt	Boosting method: Gradient Boosting Decision Tree, stable and high-performing in most cases.
objective	multiclass	Objective function suitable for multi-class classification.
num_class	5	Number of classes to predict.
metric	multi-logloss	Performance metric for multi-class problems, sensitive to predicted probabilities.
num_leaves	31	Maximum number of leaves per tree. Higher = more complex model. 31 balances accuracy and generalization.
learning_rate	0.05	Learning rate: smaller values = more stable but slower training.
feature_fraction	0.9	Fraction of features used per tree. Introduces regularization.
bagging_fraction	0.8	Random subsampling of data for each tree. Reduces variance.
bagging_freq	5	Frequency of applying bagging (every 5 iterations).
lambda_l1	1	L1 regularization (sparse). Encourages removal of irrelevant features.
lambda_l2	1	L2 regularization. Stabilizes weights and improves robustness.
seed	123	Ensures reproducibility of results.

The LightGBM model achieves an overall accuracy of 64%, indicating a satisfactory classification ability on the entire test dataset. Considering that stock market data is noisy, the model performs better overall compared to XGBoost. However, the recall for class 4 is only 45%, meaning it detects only 45% of the true positive instances for this class, reflecting a difficulty in correctly capturing this category. In sensitive contexts, this value is problematic because most positive cases go unnoticed. Therefore, compared to XGBoost, the LightGBM model is considered less effective.

Class	Precision	Recall	F1-Score
0	0.75	0.60	0.66
1	0.64	0.59	0.61
2	0.61	0.73	0.67
3	0.65	0.54	0.59
4	0.73	0.45	0.56
Accuracy			0.64
Macro Avg	0.68	0.58	0.62
Weighted Avg	0.64	0.64	0.63

Table 3.3: LightGBM classification report

Regarding the 30 most important features obtained from the LightGBM model (figure 3.5), we observe that they are still mainly the quintiles of past returns over different frequencies (monthly, biweekly), with SHAP values between 2 and 2.5. In decreasing order of importance, other features include the ROC (Rate of Change) over the past week, weekly outperformance relative to the index, quarterly and biweekly returns, market regime detection, beta and correlation measures with the market, PVO, disparity, momentum measures, stochastic K&D, and also Chaikin measures of volatility and liquidity such as Kyle and Corwin-Schultz. We can deduce that previous return quantiles partially contribute to predicting future return quantiles, as also observed with the XGBoost model, which is indicative of market momentum. Additionally, volatility and liquidity measures, such as Chaikin, Kyle, and Corwin-Schultz, may also play a role.

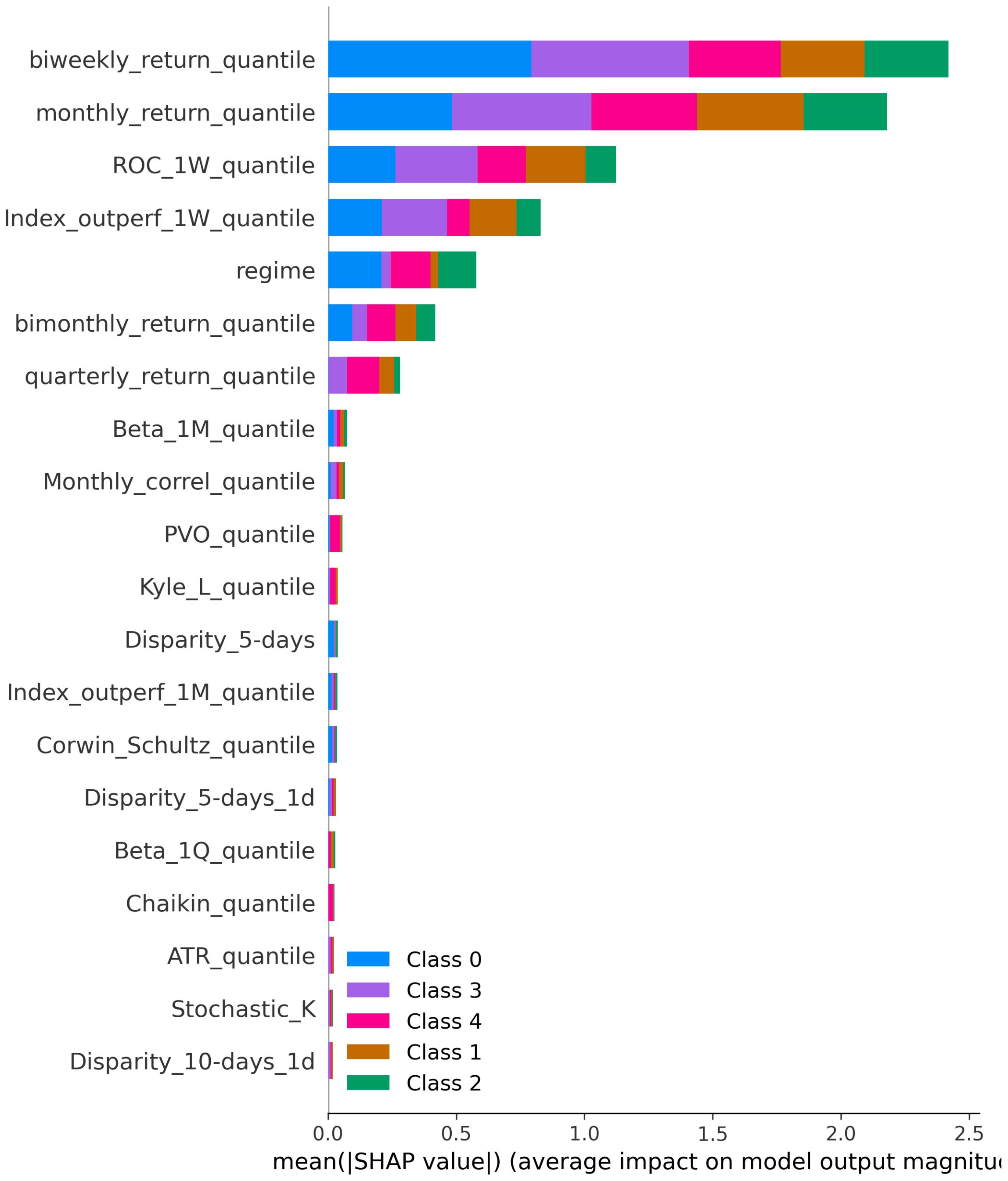


Figure 3.5: LightGBM top 30 features

3.4.6 LSTM

Classical recurrent neural networks (RNNs) are limited by numerical instability of their gradients, which tend to vanish or explode over time steps. To overcome these issues, Hochreiter and Schmidhuber introduced in 1997 the *Long Short-Term Memory* (LSTM) cells [44]. Their design relies on an internal memory with adaptive mechanisms that allow learning and retaining long-term dependencies, fundamental for sequential data processing.

LSTM uses an internal memory decomposed into two states: a short-term state $h(t)$ and a long-term state $c(t)$. The latter can be thought of as a stable memory, less prone to abrupt changes.

The operational principle relies on three gating mechanisms:

1. The forget gate (f_t) decides which information from c_{t-1} should be retained or discarded. 2. The input gate (i_t) selects the information from the current input to be incorporated into the internal memory. 3. The output gate (o_t) controls how much of the internal memory contributes to the output h_t.

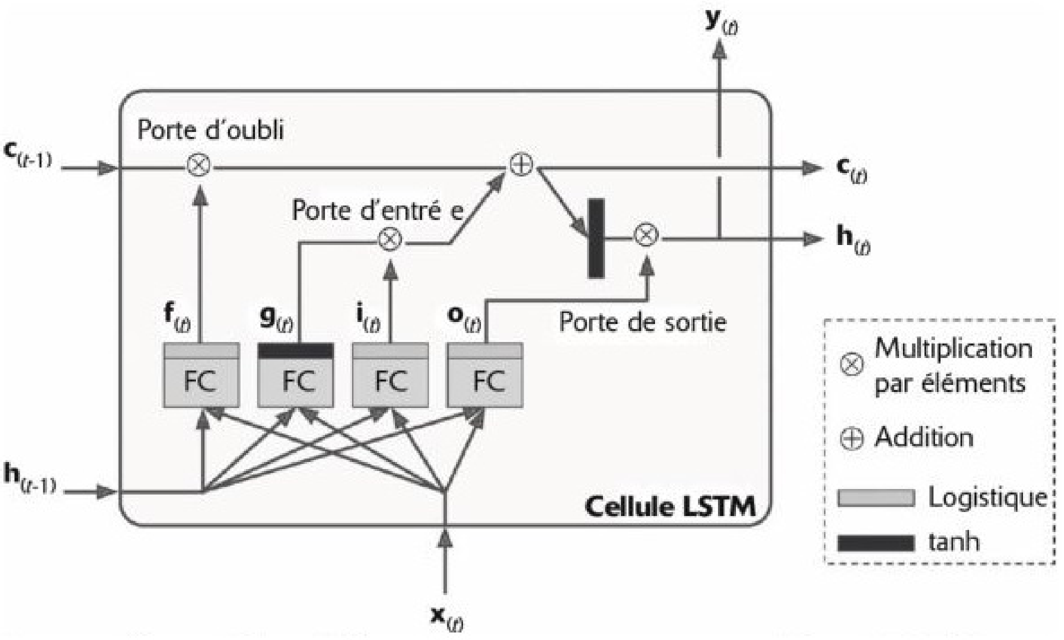


Figure 3.6: LSTM architecture [34]

Functionally, the transition from c_{t-1} to c_t is performed via partial retention (through f_t) and addition of new information filtered by i_t. Then, h_t is obtained by applying a `tanh` activation to c_t, subsequently filtered by the output gate o_t. This architecture allows the network to learn which information to retain, forget, or pass on, enabling efficient and stable learning of long-term dependencies.

A key contribution of LSTMs is addressing the *vanishing gradient* problem, where error signals diminish exponentially over long sequences. By maintaining an almost identity link (gain $= 1$) between c_{t-1} and c_t, the architecture ensures stable backpropagation, even over hundreds of steps.

LSTMs excel at processing very long sequences, whether text, audio, or time series. Their ability to retain contextual information over dozens or even hundreds of steps makes them essential building blocks for modern NLP, speech recognition, and financial predictive models. They also formed the basis of powerful models such as GPT-2 and GPT-3, despite the rise of Transformer architectures [15].

Compared to standard RNNs, LSTMs are superior for handling long-term dependencies. Modern architectures include variants such as *Bi-LSTM* (forward/backward duality) or versions with *peephole connections*, allowing greater expressiveness.

An extensive study by Greff et al. (2015) [38] tested 8 LSTM variants on speech recognition, handwriting, and polyphonic music tasks. None outperformed the standard architecture, highlighting the fundamental importance of the forget and output gates.

In 2023, a comprehensive review by Ghojogh and Ghodsi summarized RNNs, LSTMs, GRUs, bidirectional structures, and other sequential models with equations, applications, and comparative analysis [35].

3.4.6.1 Selected LSTM Hyperparameters

Here are the best hyperparameters obtained via cross-validation.

Hyperparameter	**Selected Value**	**Description**
num_lstm_layers	4	Number of stacked LSTM layers. More layers = better ability to capture complex dependencies but risk of overfitting.
hidden_size	128	Dimension of the hidden state. Controls the network's memory capacity.
dropout	0.3	Regularization rate to prevent overfitting. Applied between layers.
recurrent_dropout	0.2	Dropout applied to recurrent connections. Helps temporal robustness.
batch_size	32	Number of samples processed before updating weights. Larger = faster training, less precise.
epochs	20	Number of full iterations over the dataset. Usually combined with *early stopping*.
learning_rate	0.001	Learning rate of the optimizer (often Adam). Standard effective value.
optimizer	Adam	Gradient descent method. Adam is robust and converges well for large datasets.
loss_function	categorical_crossentropy	Loss function suitable for multi-class classification.
output_activation	softmax	Converts outputs into probabilities for 5 classes.
metrics	accuracy	Tells Keras to track accuracy during training and validation.
activation	leaky relu	Activation function used in dense layers after LSTM. Introduces nonlinearity and accelerates learning.
early_stopping_patience	5	Stops training if validation does not improve after 5 epochs.

In the LSTM architecture, two bidirectional layers are used, allowing the network to process a sequence in both directions. In this case, the dataset contains sequences of technical indicators over multiple days (1 to 5), or longer-term for fundamental indicators (e.g., one week, one month, one quarter). This allows capturing more context through sequence evolution.

Batch normalization normalizes layer activations (mean 0, variance 1) during training to accelerate convergence, stabilize learning, and reduce the risk of gradient explosion or vanishing.

With dropout, a portion of neurons (30%) is randomly deactivated in each batch. This prevents overfitting by forcing the network to learn more robust representations.

Activation Functions

ReLU (Rectified Linear Unit) is simple and efficient, avoids gradient vanishing, but can "block" neurons (dying ReLU problem).

Tanh is an activation function with values between -1 and 1, useful for centered data, but prone to gradient vanishing.

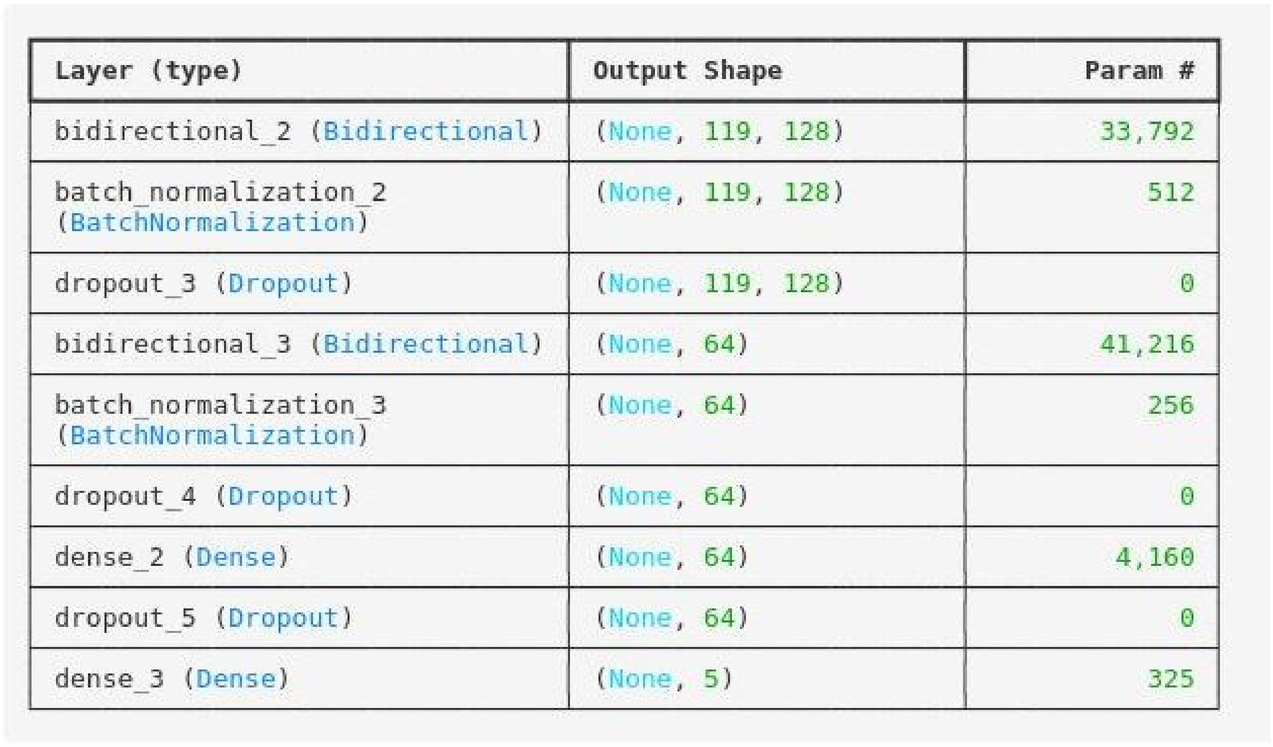
The sigmoid function has values between 0 and 1, which is good for probability calculations but saturates quickly and is also prone to gradient vanishing.

Leaky ReLU is a variant of ReLU that allows a small gradient to pass when the input is negative, thereby reducing the problem of blocked neurons.

Optimizer

The Adam optimizer combines momentum and RMSProp (Root Mean Square Propagation). It dynamically adapts the learning rate for each parameter, is robust, and widely used.

SGD with momentum is simpler and sometimes offers better generalization; RMSProp works well for sequences, while Adagrad is better for rare features, though its learning rate can decay too quickly.



Layer (type)	Output Shape	Param #
bidirectional_2 (Bidirectional)	(None, 119, 128)	33,792
batch_normalization_2 (BatchNormalization)	(None, 119, 128)	512
dropout_3 (Dropout)	(None, 119, 128)	0
bidirectional_3 (Bidirectional)	(None, 64)	41,216
batch_normalization_3 (BatchNormalization)	(None, 64)	256
dropout_4 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 64)	4,160
dropout_5 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 5)	325

Figure 3.7: LSTM Model Architecture

The LSTM model achieves an overall accuracy of 62%, indicating satisfactory classification capability on the full test dataset. Considering that stock market data is noisy, the LSTM model performs worse compared to XGBoost and LightGBM. Similar to LightGBM, the recall for class 4 is low at 47%, detecting only 47% of the true positive instances in this class, which reflects difficulty in correctly capturing this category. In sensitive contexts, this value is problematic because most positive cases go undetected. Overall, precision and recall are also lower compared to XGBoost and LightGBM.

Regarding the top 30 features obtained from the LSTM model using LIME (figure 3.8), it is surprising that the main features are VIX volatility anomalies from the past 1 to 4 days, indicating that LSTM memory cells are more sensitive to recent market volatility. The second category of indicators includes monthly, biweekly, weekly, and semiweekly return quintiles. Next, we observe the 3-day RSI and the market regime from the past week. The market regime acts as counter-evidence, suggesting it may serve as a contrarian signal, potentially predicting a rebound, for example in a bearish regime with high variance. Fundamental indicators are also

Class	Precision	Recall	F1-Score
0	0.69	0.66	0.68
1	0.61	0.57	0.59
2	0.61	0.68	0.64
3	0.61	0.55	0.57
4	0.84	0.47	0.60
Accuracy			0.62
Macro Avg	0.67	0.59	0.62
Weighted Avg	0.62	0.62	0.62

Table 3.4: LSTM Classification Report

present, such as Price/Earnings (P/E) ratios and EPS (Earnings Per Share), suggesting that financial quality or relative value may predict higher future returns. Finally, other contrarian technical indicators like KST, MACD, disparity, and EMA differences are noted, which may serve as trend reversal signals.

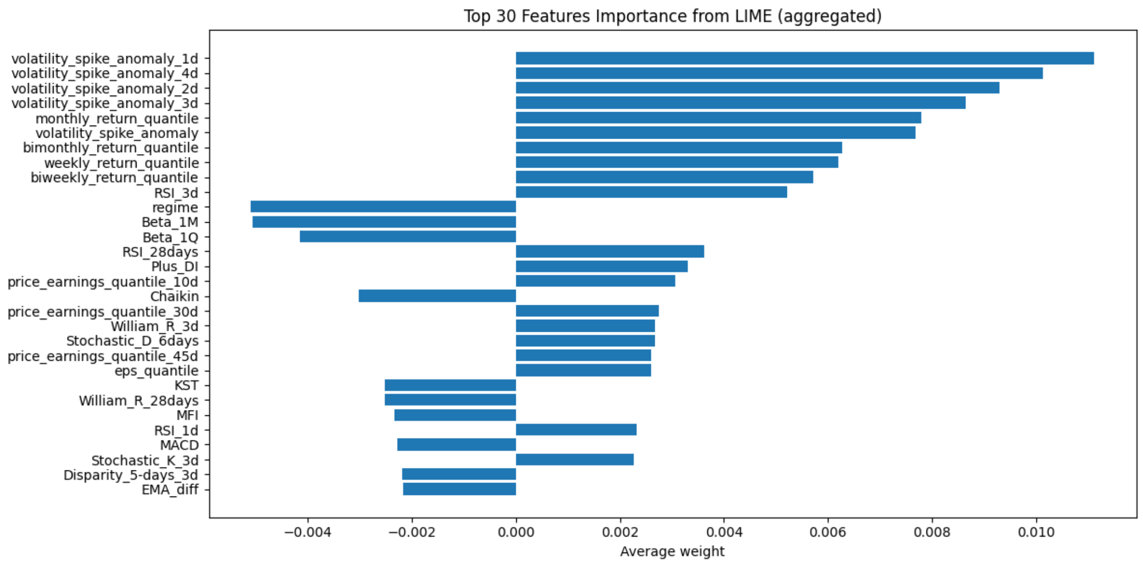


Figure 3.8: LSTM Top 30 Features

In figure 3.9, we can see the gradual increase of LSTM model accuracy over epochs, demonstrating that the model is learning patterns in the data. Validation accuracy follows a similar trend and stabilizes around 62%, which is a good sign of generalization. Training loss decreases steadily, indicating that the model is learning effectively. Validation loss also decreases and stabilizes close to the training loss curve below 0.90, indicating good generalization. There is no overfitting, as training loss does not continue decreasing while validation loss increases or stagnates, particularly since early stopping occurred around the 18th epoch.

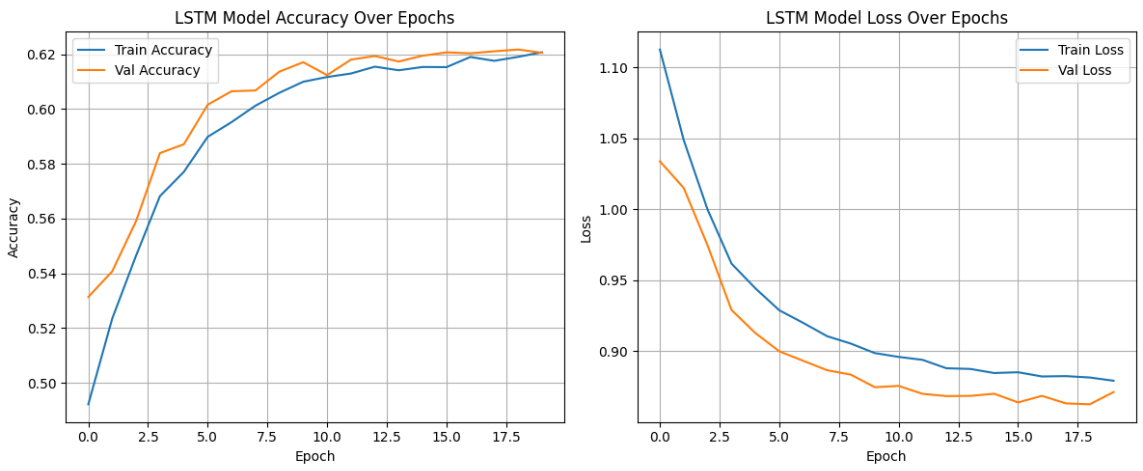


Figure 3.9: LSTM Training Accuracy and Loss

3.5 Comparison of Selected AI Models

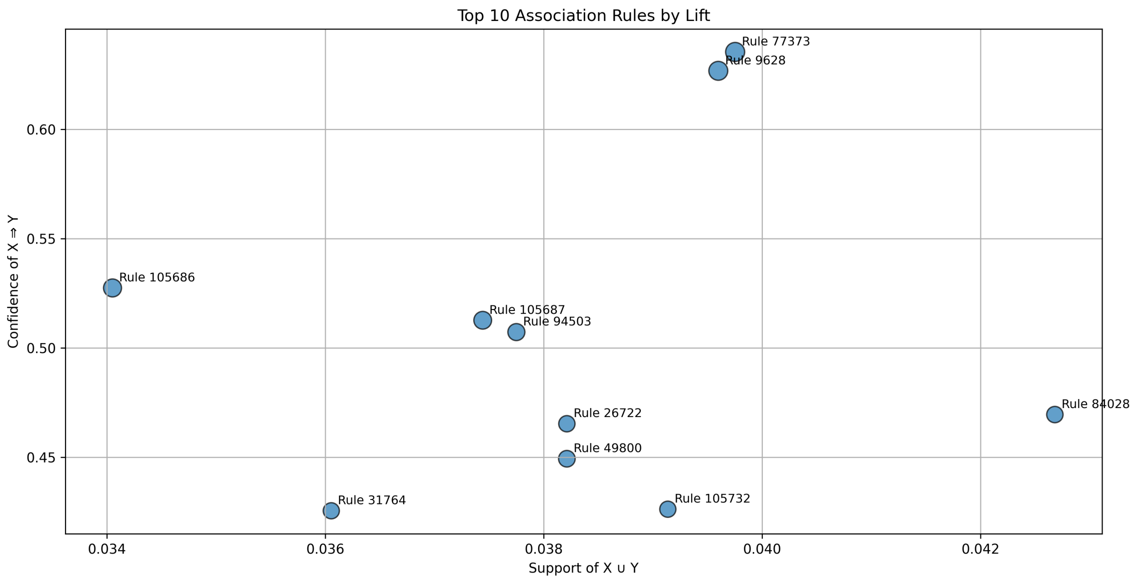
The best-performing model is XGBoost with an accuracy of 63%, average precision of 69%, recall of 60%, and F1 score of 63%. For all predicted classes, results exceed 50% compared to a random classifier. Precision above 61% for all classes indicates that the model rarely makes false positive errors. Precision is 82% for the last return quintile, which is the target class of greatest interest in the study. Observing the confusion matrix, XGBoost makes more classification errors between classes 1 to 3, where misclassification is more common for directly adjacent classes. For example, class 2 has 9043 correctly predicted instances, 3434 misclassified as class 1, and 2177 misclassified as class 3. The model has more difficulty distinguishing intermediate return classes, which is not critical here since the class of interest is the fifth quintile.

Class	Random Forest (Boruta)			LSTM			LightGBM			XGBoost		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
0	0.76	0.59	0.66	0.69	0.66	0.68	0.75	0.60	0.66	0.75	0.63	0.69
1	0.61	0.59	0.60	0.61	0.57	0.59	0.64	0.59	0.61	0.62	0.58	0.60
2	0.60	0.72	0.66	0.61	0.68	0.64	0.61	0.73	0.67	0.61	0.72	0.66
3	0.66	0.49	0.56	0.61	0.55	0.57	0.65	0.54	0.59	0.65	0.51	0.57
4	0.91	0.50	0.64	0.84	0.47	0.60	0.73	0.45	0.56	0.82	0.53	0.65
Accuracy		0.63			0.62			0.64			0.63	
Macro Avg	0.71	0.58	0.63	0.67	0.59	0.62	0.68	0.58	0.62	0.69	0.60	0.63
Weighted Avg	0.63	0.63	0.62	0.62	0.62	0.62	0.64	0.64	0.63	0.64	0.63	0.63

Table 3.5: Comparison of classification reports between Random Forest (Boruta), LSTM, LightGBM, and XGBoost

3.6 Data Mining and Association Rule Extraction for Signal Detection

The top 10 association rules are selected, where the consequent y corresponds to the best biweekly return quintile in descending order of lift, with support $X \cup Y$ greater than 0.03. This allows identification of the most interesting discrete and ordinal signals associated with the highest return class. Then, all transactions that triggered these signals over the past 10 years are selected, and trades are simulated on the exact days these signals were activated. Lift measures the importance of an association rule. The association rules, with their numeric identifiers corresponding to the combination of discrete signals in the selected rules, are shown in figure 3.10.



rule_id	x	y	support_x_y	confidence_x_y	lift_x_y
77373	39,464,209,337,471,28,478,	34	0.03975346687211094	0.6354679802955665	2.011798630301574
9628	337,209,471,172,239	34	0.03959938366718028	0.6268292682926829	1.984449732302201
105686	353,239,209,337,58,251,3	34	0.034052388289676425	0.5274463007159904	1.7959740249983096
105687	353,39,239,209,337,58,25	34	0.037442218798151	0.5126582278481012	1.7456200937744897
94503	353,471,39,12,209,337,116,34	34	0.03775038520801233	0.5072463768115942	1.6058677978084128
49800	353,39,48,464,496,209,337,343,47	34	0.03821263482280431	0.4492753623188406	1.5297991088401235
84028	353,39,173,239,48,209,337,343,217,	34	0.042681047765793526	0.4694915254237288	1.4863414634146341
26722	353,209,337,153,394,251	34	0.03821263482280431	0.4652908067542214	1.473042602846291
105732	353,39,48,209,337,218,25	34	0.03913713405238829	0.4261744966442953	1.4511398128129467
31764	209,337,470,39,251,31	34	0.03605546995377504	0.4254545454545455	1.448688352570829

Figure 3.10: Selection of the Best Association Rules

Chapter 4

Backtesting

4.1 Volatility Prediction and Protection Against Maximum Loss

The GARCH model will be used to calculate a variance interval once a stock is purchased, in order to set an expected maximum and minimum price range. If the minimum price is reached, the stock is sold at a loss, acting like a stop-loss order. Conversely, if the stock reaches the maximum price, the initial interval calculation is renewed to allow capturing optimal gains when the stock benefits from an upward movement. The triple-barrier method of Marcos López de Prado [65] is applied in this strategy, as explained below.

4.1.1 GARCH

GARCH models (*Generalized Autoregressive Conditional Heteroskedasticity*) represent a major advancement in modeling financial volatility. They allow capturing empirical phenomena such as volatility persistence, conditional heteroskedasticity, and *volatility clustering*. Initially introduced by Engle (1982) [26] in the form of the ARCH model, and later generalized by Bollerslev (1986) [12], GARCH models have become a standard reference in quantitative finance, risk management, and applied econometrics.

The ARCH model (*Autoregressive Conditional Heteroskedasticity*) proposed by Engle (1982) [26] is based on the idea that the conditional variance of a time series depends on past shocks. Formally, let r_t denote a financial return:

$$r_t = \mu + \epsilon_t, \quad \epsilon_t = \sigma_t z_t, \quad z_t \sim \mathcal{N}(0,1),$$

where the conditional variance σ_t^2 is defined as:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2, \quad \alpha_i \geq 0.$$

This model already captures conditional heteroskedasticity but has limitations when accounting for long memory.

To overcome these limitations, Bollerslev (1986) proposed the GARCH model, which adds an autoregressive component on the conditional variance:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2,$$

where:

- $\alpha_0 > 0$,
- $\alpha_i \geq 0$,
- $\beta_j \geq 0$.

The most common case is GARCH(1,1):

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2,$$

which represents a balance between the memory of past shocks (ϵ_{t-1}^2) and the persistence of variance (σ_{t-1}^2).

- α_0: long-term variance.
- α_i: immediate reaction to past shocks (impact effect).
- β_j: volatility persistence.

A sum $\alpha_1 + \beta_1$ close to 1 indicates high volatility persistence.
GARCH models adapt to the specificities of financial time series:

- *Volatility clustering.*
- Leptokurtosis (fat tails).
- Persistence of conditional variance.

Despite their success, classical GARCH models have limitations:

- Inability to model asymmetry (leverage effect).
- Inability to account for long memory.
- Sensitivity to the choice of residual distribution.

Many variants have been developed:

- **EGARCH** (Nelson, 1991) [74]: models the logarithm of conditional variance, ensuring positivity and capturing asymmetries.
- **TGARCH/GJR-GARCH** (Glosten et al., 1993) [36]: accounts for leverage effects. GJR-GARCH, proposed by Zakoian and Glosten-Jagannathan-Runkle, introduces threshold terms to differentiate the effects of positive and negative shocks.
- **FIGARCH** (Baillie et al., 1996) [7]: models long memory (Fractionally Integrated GARCH), integrating fractional memory to better represent long-term volatility persistence.

Hansen and Lunde (2005) [42] show that, despite their limitations, GARCH models provide a robust foundation for volatility forecasting. They are widely used in:

- Risk management (VaR, Expected Shortfall).
- Option pricing (Black-Scholes-GARCH type models).
- Financial market volatility forecasting.

GARCH parameters are generally estimated via Maximum Likelihood Estimation (MLE).

4.1.2 Revisited Triple Barrier Method

The triple-barrier method, proposed by Marcos López de Prado in *Advances in Financial Machine Learning* [82], is a sophisticated approach for labeling financial events in supervised learning. Unlike classical approaches that simply fix an arbitrary time horizon to determine whether an investment is profitable or not, the triple-barrier method introduces a richer and more realistic view of market behavior.

The idea is based on the fact that a financial trade can succeed or fail not only at the end of a defined period but also as soon as a critical threshold is breached. For this reason, three barriers are set around the entry price of a position:

- An **upper barrier** (take profit), corresponding to a predefined gain. If the price reaches this level, the position is considered successful (label = +1).
- A **lower barrier** (stop loss), corresponding to a maximum tolerable loss. If the price reaches this level, the position is considered a failure (label = -1).
- A **time barrier**, which corresponds to a maximum duration for the position. If neither of the other two barriers is reached before this point, the outcome is determined by the final price change: gain if the price has increased, loss if it has decreased, or neutral if the variation is negligible (label = 0).

This approach better reflects the reality of trading decisions. In financial markets, an investor does not always wait for a fixed horizon: they may exit earlier if a target or risk limit is reached. The triple-barrier method formalizes this behavior while providing more robust labeling for machine learning models.

Mathematically, let P_t denote the price at time t, and t_0 the initial time of a position. We define:

$$\begin{aligned} U &= P_{t_0} \cdot (1 + \theta_u), \\ L &= P_{t_0} \cdot (1 - \theta_l), \\ T &= t_0 + \tau, \end{aligned}$$

where U is the upper barrier (threshold θ_u), L is the lower barrier (threshold θ_l), and T is the time barrier set τ periods after t_0.

The event labeling is expressed as:

$$y = \begin{cases} +1 & \text{if } \exists t \in [t_0, T],\ P_t \geq U, \\ -1 & \text{if } \exists t \in [t_0, T],\ P_t \leq L, \\ \operatorname{sign}(P_T - P_{t_0}) & \text{if no price barrier is breached.} \end{cases}$$

This algorithm shows how to evaluate the outcome of a trade based on the defined barriers.

Algorithm 1: Triple Barrier Method

```
Input: Initial price $P_{t_0}$, thresholds $\theta_u, \theta_l$, horizon $\tau$
Output: Label $y \in \{-1, 0, +1\}$
$U = P_{t_0} \cdot (1 + \theta_u)$
$L = P_{t_0} \cdot (1 - \theta_l)$
$T = t_0 + \tau$
for $t = t_0$ to $T$ do
    if $P_t \geq U$ then
        $y = +1$ ; return $y$
    end
    if $P_t \leq L$ then
        $y = -1$ ; return $y$
    end
end
if $t \geq T$ then
    $y = \text{sign}(P_T - P_{t_0})$
else
    $y = 0$
end
return $y$
```

This method has several notable advantages. It introduces better discipline in labeling by reducing biases related to arbitrary horizons. It also accounts for asymmetric behavior: a position may close quickly if it hits a stop-loss, while another may last until the time horizon. Finally, it is flexible: thresholds can be adapted to market volatility or the investor's risk profile. The proposed modification is to recalculate the three barriers if the stock price exceeds the upper limit, in order to capture maximum profit if the stock is in a prolonged upward movement.

4.2 Results and Performance Evaluation

The following financial performance metrics are used to evaluate the return and risk-adjusted performance of the strategy in the backtest:

4.2.1 Annual Return (%)

$$R_{\text{annual}} = \left(\prod_{t=1}^{T} (1 + r_t) \right)^{\frac{252}{T}} - 1$$

where r_t is the daily return and T is the total number of days. Annualized portfolio return.

4.2.2 Cumulative Return (%)

$$R_{\text{cumulative}} = \prod_{t=1}^{T} (1 + r_t) - 1$$

Total capital growth over the period.

4.2.3 Annual Volatility (%)

$$\sigma_{\text{annual}} = \sigma_{\text{daily}} \times \sqrt{252}$$

Annualized dispersion of returns, a measure of risk.

4.3 Performance Ratios

4.3.1 Sharpe Ratio

$$\text{Sharpe} = \frac{\mathbb{E}[r_p - r_f]}{\sigma_p}$$

where r_p is the portfolio return and r_f the risk-free rate. Excess return per unit of total risk.

4.3.2 Calmar Ratio

$$\text{Calmar} = \frac{R_{\text{annual}}}{|\text{Max Drawdown}|}$$

Compares annualized return to worst loss.

4.3.3 Stability

$$\text{Stability} = R^2\Big(\text{regression of the equity curve}\Big)$$

Measures the consistency of portfolio growth.

4.4 Risk and Extreme Losses

4.4.1 Maximum Drawdown (%)

$$\text{Max Drawdown} = \max_{t\in[0,T]} \frac{\max_{s\in[0,t]} V_s - V_t}{\max_{s\in[0,t]} V_s}$$

where V_t is the portfolio value at time t. Largest relative loss from a previous peak.

4.4.2 Omega Ratio

$$\Omega = \frac{\int_L^{\infty}(1 - F(r))\,dr}{\int_{-\infty}^{L} F(r)\,dr}$$

where $F(r)$ is the return distribution and L a threshold (often 0). Compares gains above the threshold to losses below it.

4.4.3 Sortino Ratio

$$\text{Sortino} = \frac{\mathbb{E}[r_p - r_f]}{\sigma_-}$$

where σ_- is the standard deviation of negative returns only. A Sharpe ratio variant considering only downside risk.

4.4.4 Skewness

$$\text{Skew}(r) = \frac{\mathbb{E}[(r-\mu)^3]}{\sigma^3}$$

Indicates whether the return distribution is left-skewed (crash risk) or right-skewed (large upward moves).

4.4.5 Kurtosis

$$\text{Kurt}(r) = \frac{\mathbb{E}[(r-\mu)^4]}{\sigma^4}$$

Measures tail heaviness, likelihood of extreme returns.

4.4.6 Tail Ratio

$$\text{Tail Ratio} = \frac{Q_{95}(r)}{|Q_5(r)|}$$

where Q_{95} and Q_5 are the 95% and 5% quantiles. Compares magnitude of large gains vs. large losses.

4.4.7 Daily Value at Risk (%)

$$VaR_\alpha = \inf\{x \in \mathbb{R} : P(r \leq x) \geq \alpha\}$$

Maximum expected loss at a confidence level α.

4.5 Alpha and Beta

4.5.1 Alpha (Jensen's alpha)

$$\alpha = R_p - (R_f + \beta(R_m - R_f))$$

Abnormal portfolio return beyond market explanation.

4.5.2 Beta

$$\beta = \frac{\text{Cov}(r_p, r_m)}{\text{Var}(r_m)}$$

Sensitivity of the portfolio to market movements.

Here are the realized returns during the backtest of the strategy, compared to the S&P 500 benchmark. The strategy significantly outperformed the S&P 500, with cumulative returns of 503.39% versus 333.98% for the S&P 500. Average volatility is also much lower (9.785% vs 17.574%, nearly half), while the maximum drawdown reaches only -11.427% vs -30.444%, roughly one-third. Finally, the Sharpe ratio — which measures excess return per unit of risk — is 1.76 vs 0.72, more than double.

Financial Metrics	**Model**				**S&P 500**			
	Mean	**Median**	**5%**	**95%**	**Mean**	**Median**	**5%**	**95%**
Annual return (%)	18.156	18.033	11.863	24.977	11.629	11.765	1.740	21.586
Cumulative returns (%)	430.280	412.537	208.176	716.210	226.184	186.838	12.732	580.376
Annual volatility (%)	9.785	9.776	8.898	10.698	17.574	17.519	16.478	18.744
Sharpe ratio	1.760	1.760	1.230	2.280	0.720	0.720	0.220	1.250
Calmar ratio	1.770	1.710	0.850	2.980	0.430	0.380	0.020	1.020
Stability	0.970	0.980	0.920	0.990	0.740	0.840	0.110	0.970
Max drawdown (%)	-11.427	-10.939	-17.053	-7.297	-31.922	-30.444	-49.343	-19.480
Omega ratio	1.600	1.590	1.380	1.840	1.160	1.150	1.040	1.280
Sortino ratio	2.920	2.870	1.890	4.110	1.020	1.000	0.250	1.880
Skew	1.110	1.110	-0.510	2.710	-0.580	-0.570	-1.520	0.360
Kurtosis	25.120	25.500	11.420	38.660	12.870	12.820	7.600	18.210
Tail ratio	1.380	1.370	1.160	1.600	0.940	0.930	0.850	1.030
Daily value at risk (%)	-1.165	-1.162	-1.279	-1.059	-2.160	-2.160	-2.321	-2.000
Alpha	0.140	0.140	0.090	0.180	-0.020	-0.020	-0.020	-0.010
Beta	0.320	0.320	0.270	0.370	1.000	1.000	1.000	1.000

Table 4.1: Comparison of model performance vs. S&P 500 between 31/08/2015 and 19/12/2024 (115 months).

The backtest charts were generated using the Python library PyFolio from Quantopian [84]. Figure 4.1 shows the cumulative return, indicating that the strategy consistently outperforms the S&P 500 over the tested period. From the first year, the model's curve diverges clearly from the benchmark, demonstrating the ability to generate alpha consistently. Unlike the index, the strategy exhibits smoother growth, with significantly smaller drawdowns. For example, from early 2022 to late 2022, while the S&P 500 fell by about 25%, the strategy declined only 8%, reflecting reduced volatility and better risk control. By the end of the period, the cumulative return gap is significant (503% vs. 333%), confirming the robustness and superiority of the approach compared to the S&P 500.

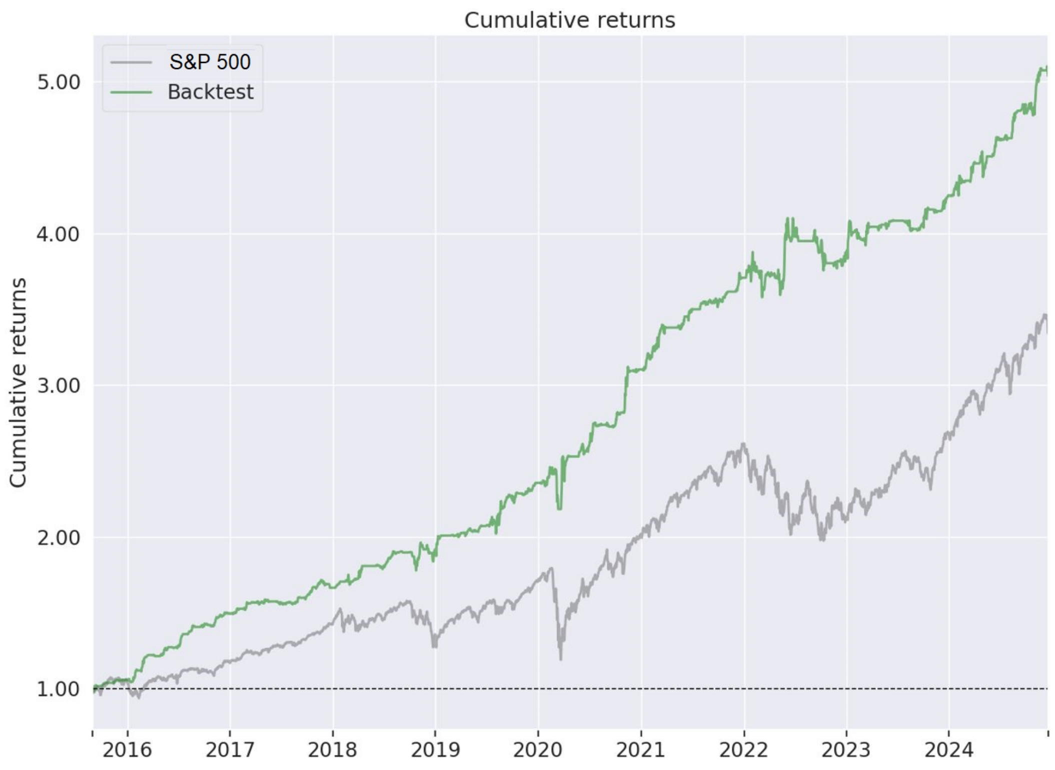


Figure 4.1: Cumulative return of 503% compared to the S&P 500 index at 333%.

The cumulative return adjusted for benchmark volatility (Figure 4.2) allows for a risk-adjusted comparison of the strategy versus the S&P 500. Even after this adjustment, the strategy continues to outperform the index, confirming that the alpha generated is not solely due to lower volatility, but reflects an intrinsic ability to capture market opportunities. In other words, the strategy delivers roughly 2.5 times higher return per unit of risk than the S&P 500, indicating enhanced efficiency.

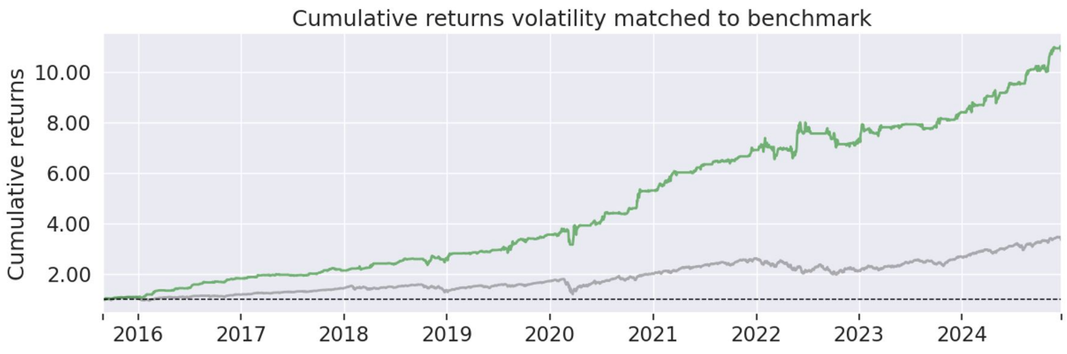


Figure 4.2: Cumulative return with volatility matched to the benchmark.

The cumulative return on a logarithmic scale (Figure 4.3) visualizes relative performance proportionally. Unlike a linear scale, it better reflects drawdown dynamics and the regularity of portfolio growth. The strategy shows an almost linear progression on a log scale, indicating consistent return generation over time. Drawdowns are also less pronounced than for the S&P 500, confirming better risk control and a steadier value creation trajectory.

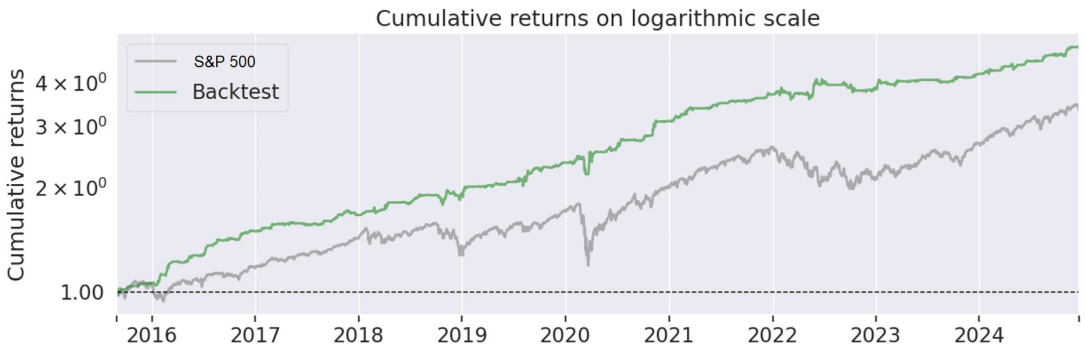


Figure 4.3: Cumulative return on a logarithmic scale.

The rolling beta (Figure 4.4), measured on a sliding window, generally stays below 1, indicating lower market sensitivity than the S&P 500. Temporary spikes toward 1 coincide with directional market phases, while declines to 0.4–0.2 suggest a more defensive exposure and better shock protection.

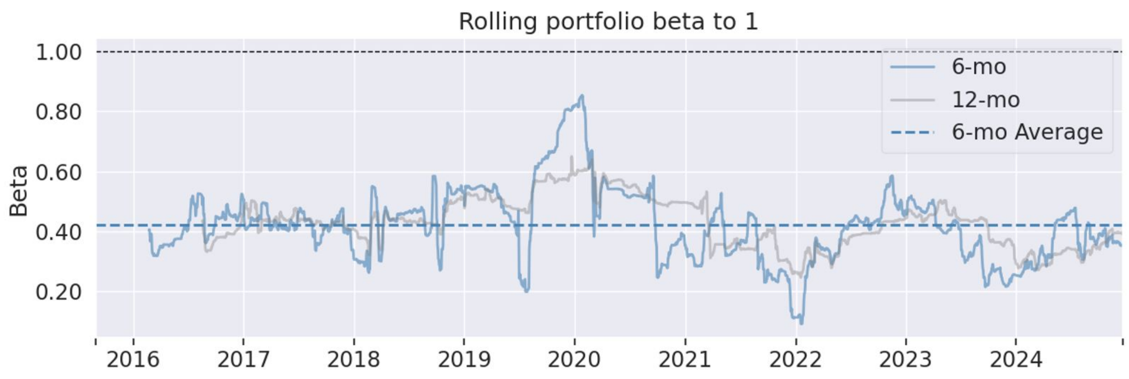


Figure 4.4: Rolling window beta

The strategy's maximum drawdowns (Figure 4.5) are significantly lower than the S&P 500, reflecting superior risk management. Drawdown periods are shorter and shallower, enhancing recovery capability after market shocks. For comparison, the S&P 500 fell about 34% from its peak on 19 February 2020 to its trough on 23 March 2020 during the COVID-19 crash, while the model lost only 12%. Similarly, the S&P 500 declined roughly 25% from 3 January 2022 to its October 2022 trough during the tech sector sell-off, whereas the model's maximum loss was 8%.

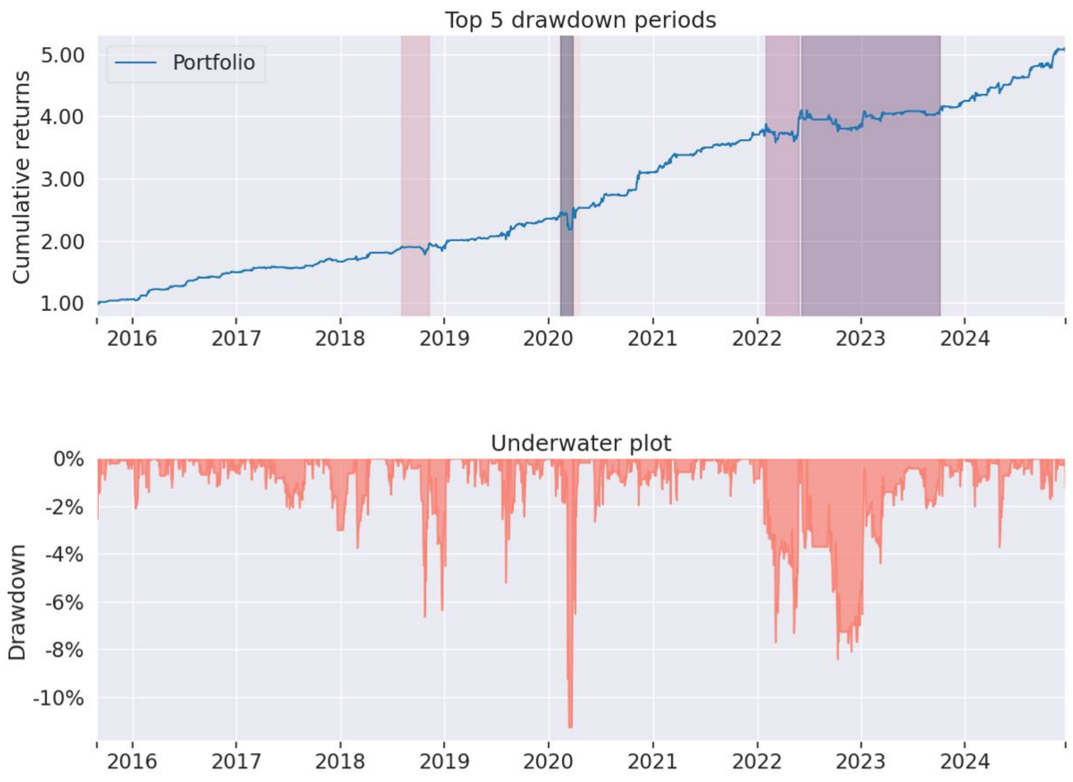


Figure 4.5: Maximum drawdown

The strategy shows positive annual performance in all years, ranging approximately from 5% to 40%, outperforming the S&P 500 in down or neutral years. This ability to limit losses during difficult years contributes strongly to long-term robustness. By comparison, the worst S&P 500 years were -19.44% in 2022 and -6.24% in 2018.

The boxplot (Figure 4.6) shows a median of daily, weekly, and monthly returns higher than the S&P 500, with lower dispersion. Negative extremes are less pronounced, reducing the risk of severe losses.

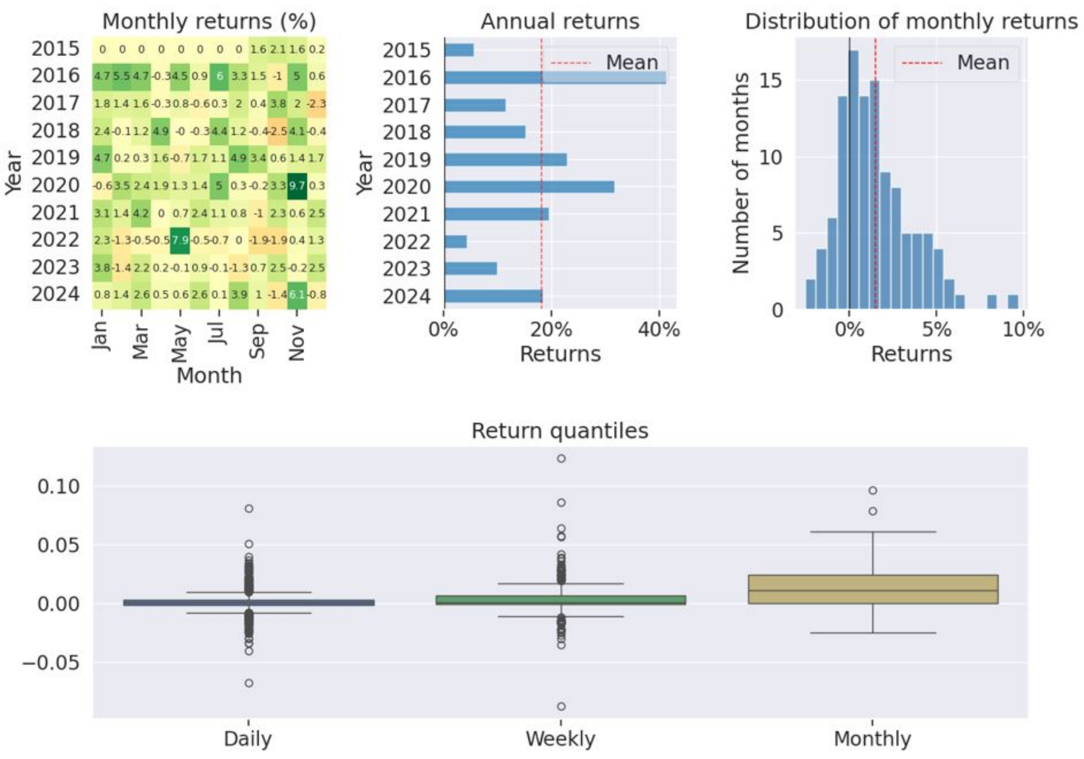


Figure 4.6: Return distribution

Each boxplot in Figure 4.7 summarizes the distribution of a financial metric from the back-test: the box represents the IQR (Q1–Q3), the central line the median, and the whiskers the range excluding outliers. A narrow box indicates a stable metric across strategies/samples, while a wide box with many outliers signals high heterogeneity and extreme behavior to investigate.

A high median with low dispersion suggests robust annual returns around 18%. High cumulative returns (median 430%) with wide dispersion reflect sensitivity to timing and reinvestment. A solid median with few outliers indicates more regular value creation, less dependent on extreme events.

Low median volatility (9.7%) with a tight box indicates homogeneous risk control. High median Sharpe (1.76) and low variability show consistently good risk-adjusted returns. Stable median Calmar (1.77) indicates efficiency relative to drawdowns. High median stability (0.97) and low IQR suggest regular equity growth. Low median drawdown (-11.4%) with few negative outliers shows resilience. High median Omega (¿1.6) with low dispersion implies consistent probability of gains exceeding losses. High median Sortino (2.92) with a tight box indicates effective downside risk control. Positive median skew (1.11) suggests favorable right tails; negative outliers warn of asymmetric crash risk. Tail ratio ¿1.38 with low dispersion shows positive extremes are more frequent/intense than negative. Median daily VaR of -1.165% with tight range indicates controlled downside risk. Positive median alpha (0.14) indicates market-independent performance generation. Median beta below 1 (0.32) with low dispersion reflects

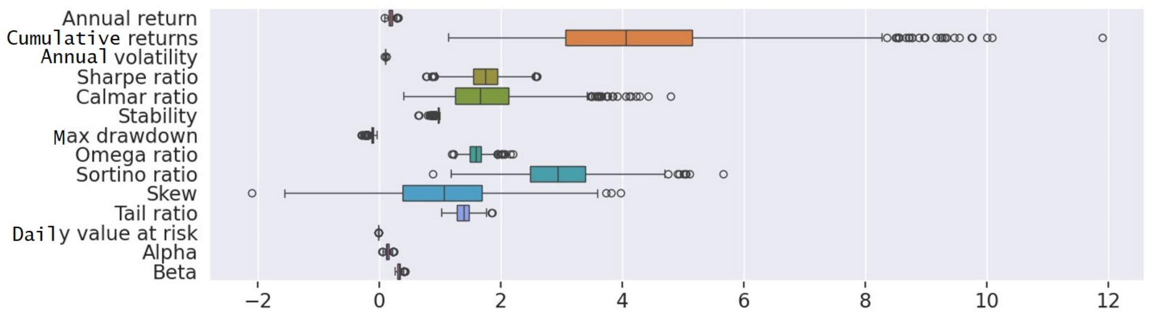


Figure 4.7: Dispersion of backtest performance metrics

moderate and stable market exposure. Overall, high and consistent Sharpe/Sortino/Calmar ratios combined with contained drawdowns support solid risk-return efficiency.

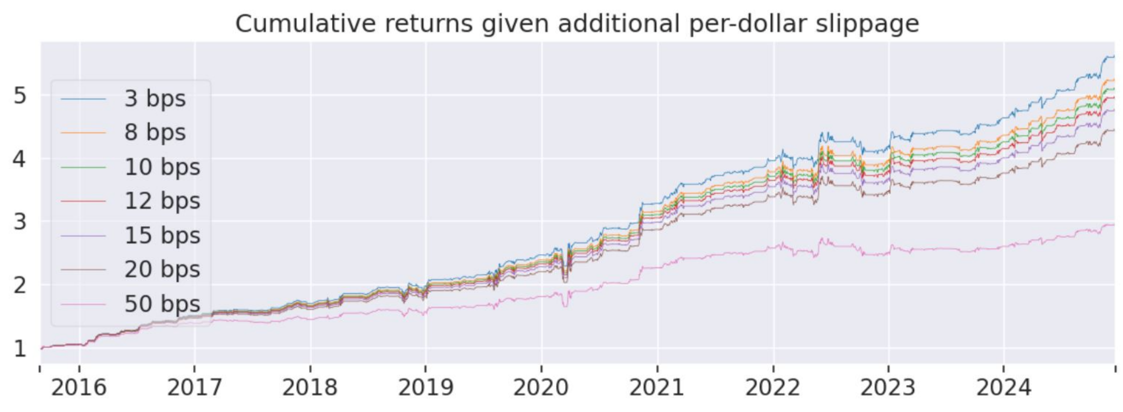


Figure 4.8: Cumulative returns with execution slippage impact

Figure 4.8 clearly shows that slippage, the execution gap, acts as a recurring cost accumulating over time, significantly reducing cumulative returns. To quantify this effect, the average annual performance loss per additional 5 bps of slippage can be estimated.

Observing the curves:

- From 3 bps to 8 bps (+5 bps), the final 2025 performance drops about 20%–40% versus the 3 bps scenario, roughly a $\approx 3\%$ annual loss.
- From 8 bps to 15 bps (+7 bps), cumulative loss reaches 40%–60% over the period, about $\approx 5\%$ per year.
- From 15 bps to 50 bps (+35 bps), performance is nearly reduced by 1.6x, implying an annual loss above 16%.

These estimates show that even a modest increase in slippage (5 bps) can significantly reduce total performance over an 8–10 year horizon. Thus, controlling transaction costs and optimizing execution are essential strategic levers to preserve long-term profitability.

Conclusion and Perspectives

Contributions

The objectives of this research were to explore the application of machine learning and data mining techniques through the extraction of association rules in the context of portfolio management, with the aim of identifying the most effective and profitable technical indicators.

1. The computation and transformation of raw price and volume data, along with fundamental metrics on U.S. companies from the S&P 500, constituted the first major phase of this work. These calculations and transformations were supported by an extensive literature review on artificial intelligence applied to investment. Signals were derived from methods such as market regime detection (low/high volatility, bullish or bearish) using Markov-Switching Regression (MSR), Hidden Markov Models (HMM), and Kaufman's Adaptive Moving Average (KAMA).

2. Following this data preparation, various machine learning models such as Random Forest, LightGBM, LSTM, and XGBoost were trained. Through feature analysis and explainability algorithms such as Boruta, SHAP, and LIME, the 30 most important features were identified for predicting the return quintile class over the following two-week period. The best-performing model was XGBoost, achieving an accuracy of 63%, which is notable given the inherent noise and complexity of predicting stock price movements in U.S. equities.

3. The next step involved constructing a dataset specifically adapted for the FGC-Stream algorithm to extract the most parsimonious and informative association rules using lift, confidence, and support metrics. These measures helped determine which signals were the most effective in predicting the future biweekly return quintile class.

4. Finally, a backtest was conducted using these extracted signals as filters across all transactions of the 500 S&P 500 constituent stocks over the past ten years. The risk management aspect was integrated through the triple-barrier method, consisting of a minimum stop loss, a maximum take-profit price, and a maximum time horizon at which the stock price would trigger an exit. This method relies on the GARCH algorithm to estimate the range of bullish and bearish price movements over a given time interval. The backtest results were illustrated using several financial performance metrics, highlighting both the returns and risk profile of the strategy. In conclusion, the strategy outperformed the S&P 500 benchmark by a significant margin, achieving a cumulative return of 503.39% compared to 333.98% for the S&P 500. The average volatility was 9.785% versus 17.574%, nearly half, while the maximum drawdown was -11.427% compared to -30.444%, approximately one-third. The Sharpe ratio—representing excess return per unit of risk—was 1.76 versus 0.72, more than double.

Perspectives

In continuation of this research work, several future directions could be explored:

1. This study highlights the significant contribution of several areas within artificial intelligence—namely, machine learning and data mining through association rule extraction—to the field of investment. Although machine learning has made remarkable progress in model explainability by numerically ranking feature importance in decreasing order, it is still often considered complex when it comes to explaining the precise rules or reasoning that led to a specific decision by portfolio managers.

2. However, data mining through association rule extraction, as implemented by the FGC-Stream algorithm, makes it possible to derive parsimonious and precise association rules that provide a high level of transparency and explainability for portfolio managers, allowing them to validate the financial logic underlying these rules.

3. Other domains of artificial intelligence could also offer valuable contributions to investment research, such as Large Language Models (LLMs), which could provide broader contextual understanding for decision-making by analyzing multiple news sources or financial statements to estimate whether a company has a high or low probability of increasing in value in the future.

Bibliography

[1] Steven B. Achelis. *Technical Analysis from A to Z*. Chicago: Probus Publishing Co., 1995, p. 331. ISBN: 9781557388162. URL: https://openlibrary.org/books/OL924413M/Technical_analysis_from_A_to_Z.

[2] Steven B. Achelis. “Technical Analysis from A to Z”. In: (2001). URL: https://www.investopedia.com/terms/m/mfi.asp.

[3] Rakesh Agrawal and Ramakrishnan Srikant. “Fast Algorithms for Mining Association Rules in Large Databases”. In: *Very Large Data Bases Conference*. 1994. URL: https://api.semanticscholar.org/CorpusID:3131928.

[4] Yakov Amihud. “Illiquidity and stock returns: cross-section and time-series effects”. In: *Journal of Financial Markets* 5.1 (2002), pp. 31–56. URL: https://doi.org/10.1016/S1386-4181(01)00024-6.

[5] Fujiang Ao et al. “Mining Maximal Frequent Itemsets in Data Streams Based on FP-Tree”. In: *Machine Learning and Data Mining in Pattern Recognition*. Ed. by Petra Perner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 479–489. ISBN: 978-3-540-73499-4.

[6] Gerald Appel. “Technical Analysis: Power Tools for Active Investors”. In: (1979). URL: https://www.investopedia.com/terms/m/macd.asp.

[7] Richard T. Baillie, Tim Bollerslev, and Hans O. Mikkelsen. “Fractionally Integrated Generalized Autoregressive Conditional Heteroskedasticity”. In: *Journal of Econometrics* 74.1 (1996), pp. 3–30. DOI: 10.1016/0304-4076(95)01749-6. URL: https://doi.org/10.1016/S0304-4076(95)01749-6.

[8] Sanjoy Basu. “Investment Performance of Common Stocks in Relation to Their Price-Earnings Ratios: A Test of the Efficient Market Hypothesis”. In: *The Journal of Finance* 32.3 (1977), pp. 663–682. URL: https://doi.org/10.2307/2326771.

[9] Leonard E. Baum and Ted Petrie. “Statistical inference for probabilistic functions of finite state Markov chains”. In: *The Annals of Mathematical Statistics* 37.6 (1966), pp. 1554–1563. URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-37/issue-6/Statistical-Inference-for-Probabilistic-Functions-of-Finite-State-Markov-Chains/10.1214/aoms/1177699147.full.

[10] Albert Bifet and Ricard Gavaldà. “Learning from Time-Changing Data with Adaptive Windowing”. In: vol. 7. Apr. 2007. DOI: 10.1137/1.9781611972771.42.

[11] William Blau. *Momentum, Direction, and Divergence: Applying the Latest Momentum Indicators for Technical Analysis*. Wiley, 1995. ISBN: 9780471027294. URL: https://www.wiley-vch.de/en/areas-interest/finance-economics-law/finance-investments-13fi/trading-13fi4/momentum-direction-and-divergence-978-0-471-02729-4.

[12] Tim Bollerslev. “Generalized autoregressive conditional heteroskedasticity”. In: *Journal of Econometrics* 31.3 (1986), pp. 307–327. URL: https://doi.org/10.1016/0304-4076(86)90063-1.

[13] John Bollinger. “Bollinger on Bollinger Bands”. In: (2001). URL: https://www.amazon.com/Bollinger-Bands-John/dp/0071373687.

[14] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32. URL: https://link.springer.com/article/10.1023/A:1010933404324.

[15] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *arXiv preprint arXiv:2005.14165* (2020). URL: https://arxiv.org/abs/2005.14165.

[16] Pedro Carmona, Francisco Climent, and Alexandre Momparler. “Predicting Failure in the U.S. Banking Sector: An Extreme Gradient Boosting Approach”. In: *International Review of Economics & Finance* 61 (2019), pp. 304–323. DOI: 10.1016/j.iref.2018.03.008. URL: https://doi.org/10.1016/j.iref.2018.03.008.

[17] Marc Chaikin. “The Chaikin Money Flow and Oscillator”. In: (1980). URL: https://www.investopedia.com/terms/c/chaikin-oscillator.asp.

[18] Tushar Chande. *The New Technical Trader: Boost Your Profit by Plugging into the Latest Indicators*. Includes discussion on the Chande Momentum Oscillator. New York: Wiley, 1994. URL: https://openlibrary.org/books/OL1425648M/The_new_technical_trader?edition=newtechnicaltrad00chan.

[19] 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*. ACM. 2016, pp. 785–794. URL: https://doi.org/10.1145/2939672.2939785.

[20] Yan Chen, Dongxu Mo, and Feipeng Zhang. “Stock market prediction using weighted inter-transaction class association rule mining and evolutionary algorithm”. In: *Economic Research-Ekonomska Istra017eivanja* 35.1 (2022), pp. 5971–5996. DOI: 10.1080/1331677X.2022.2043762. URL: https://doi.org/10.1080/1331677X.2022.2043762.

[21] Yun Chi et al. “Moment: maintaining closed frequent itemsets over a stream sliding window”. In: *Fourth IEEE International Conference on Data Mining (ICDM’04)*. 2004, pp. 59–66. DOI: 10.1109/ICDM.2004.10084.

[22] Shane A Corwin and Paul Schultz. “Estimating the bid-ask spread: Theory and evidence”. In: *The Review of Financial Studies* 25.5 (2012), pp. 1467–1509. URL: https://doi.org/10.1093/rfs/hhr030.

[23] Adrian Cretu and Mihai I. Madalina. “Vortex Indicator: A New Trend Indicator”. In: *Journal of Technical Analysis* (2010). URL: https://www.tradingview.com/wiki/Vortex_Indicator.

[24] Shubharthi Dey et al. “Forecasting to Classification: Predicting the Direction of Stock Market Price Using Xtreme Gradient Boosting”. In: *Working Paper* (2016). DOI: 10.13140/RG.2.2.15294.48968. URL: https://www.researchgate.net/publication/309492895_Forecasting_to_Classification_Predicting_the_direction_of_stock_market_price_using_Xtreme_Gradient_Boosting.

[25] Alexander Elder. *Trading for a Living*. John Wiley Sons, 1993. URL: https://www.investopedia.com/terms/f/force-index.asp.

[26] Robert F. Engle. “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation”. In: *Econometrica* 50.4 (1982), pp. 987–1007. URL: https://doi.org/10.2307/1912773.

[27] Eugene F. Fama. “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *The Journal of Finance* 25.2 (1970), pp. 383–417. DOI: 10.2307/2325486. URL: https://www.jstor.org/stable/2325486.

[28] Thomas Fischer and Christopher Krauss. “Deep learning with long short-term memory networks for financial market predictions”. In: *European Journal of Operational Research* 270.2 (2018), pp. 654–669. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.11.054. URL: https://www.sciencedirect.com/science/article/pii/S0377221717310652.

[29] Yoav Freund and Robert E Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. DOI: 10.1006/jcss.1997.1504. URL: https://doi.org/10.1006/jcss.1997.1504.

[30] Yoav Freund and Robert E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. URL: https://doi.org/10.1006/jcss.1997.1504.

[31] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “Additive logistic regression: a statistical view of boosting”. In: *The Annals of Statistics* 28.2 (2000), pp. 337–407. URL: https://doi.org/10.1214/aos/1016218223.

[32] Chuancong Gao and Jianyong Wang. “Efficient itemset generator discovery over a stream sliding window”. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM ’09. Hong Kong, China: Association for Computing Machinery, 2009, pp. 355–364. ISBN: 9781605585123. DOI: 10.1145/1645953.1646000. URL: https://doi.org/10.1145/1645953.1646000.

[33] Aurélien Géron. *Deep Learning avec Keras et TensorFlow: Mise en oeuvre et cas concrets*. O’Reilly Media, 2020. URL: https://www.oreilly.com/library/view/deep-learning-avec/9781098125943/.

[34] Aurélien Géron. *Deep Learning avec Keras et TensorFlow: Mise en œuvre et cas concrets*. 3^{e} édition. Dunod, 2024. ISBN: 978-2-10-084769-3. URL: https://www.dunod.com/sciences-techniques/deep-learning-avec-keras-et-tensorflow-mise-en-oeuvre-et-cas-concrets-0.

[35] B. Ghojogh and A. Ghodsi. “Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey”. In: *arXiv preprint arXiv:2304.11461* (2023). URL: https://arxiv.org/abs/2304.11461.

[36] Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. “On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks”. In: *Journal of Finance* 48.5 (1993), pp. 1779–1801. URL: https://EconPapers.repec.org/RePEc:bla:jfinan:v:48:y:1993:i:5:p:1779-1801.

[37] Joseph Granville. “Granville’s New Key to Stock Market Profits”. In: (1963). URL: https://www.investopedia.com/terms/o/onbalancevolume.asp.

[38] Klaus Greff et al. “LSTM: A Search Space Odyssey”. In: *arXiv preprint arXiv:1503.04069* (2015). URL: https://arxiv.org/abs/1503.04069.

[39] Massimo Guidolin. “Markov-switching models in empirical finance”. In: *Advances in Econometrics* 27A (2011), pp. 1–86. URL: https://www.emerald.com/insight/content/doi/10.1108/S0731-9053(2011)000027A006.

[40] James D. Hamilton. “A new approach to the economic analysis of nonstationary time series and the business cycle”. In: *Econometrica* 57.2 (1989), pp. 357–384. URL: https://www.jstor.org/stable/1912559.

[41] Jiawei Han, Jian Pei, and Yiwen Yin. “Mining frequent patterns without candidate generation”. In: *SIGMOD Rec.* 29.2 (May 2000), pp. 1–12. ISSN: 0163-5808. DOI: 10.1145/335191.335372. URL: https://doi.org/10.1145/335191.335372.

[42] Peter R. Hansen and Asger Lunde. “A Forecast Comparison of Volatility Models: Does Anything Beat a GARCH(1,1)?” In: *Journal of Applied Econometrics* 20.7 (2005), pp. 873–889. DOI: 10.1002/jae.800. URL: https://www.scirp.org/reference/referencespapers?referenceid=2187864.

[43] James B Heaton, Nicholas G Polson, and John H Witte. “Deep learning for finance: deep portfolios”. In: *Applied Stochastic Models in Business and Industry* 33.1 (2017), pp. 3–12. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.2203.

[44] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[45] Chih-Cheng Huang, Ming-Hsien Chen, and Chih-Hung Wang. “Learning from imbalanced data in financial risk prediction”. In: *Expert Systems with Applications* 42.5 (2015), pp. 2354–2362. DOI: 10.1016/j.eswa.2014.10.028. URL: https://doi.org/10.1016/j.eswa.2014.10.028.

[46] David Huang, Yun Sing Koh, and Gillian Dobbie. “Rare Pattern Mining on Data Streams”. In: *Data Warehousing and Knowledge Discovery*. Ed. by Alfredo Cuzzocrea and Umeshwar Dayal. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 303–314. ISBN: 978-3-642-32584-7.

[47] Ulcer Index. “Ulcer Index”. In: (). URL: https://www.investopedia.com/terms/u/ulcerindex.asp.

[48] Investopedia. *ADX: The Trend Strength Indicator*. URL: https://www.investopedia.com/articles/trading/07/adx-trend-indicator.asp.

[49] Investopedia. *Triple Exponential Average (TRIX): Overview, Calculations*. URL: https://www.investopedia.com/terms/t/trix.asp.

[50] Md. Rezaul Karim et al. “Mining maximal frequent patterns in transactional databases and dynamic data streams: A spark-based approach”. In: *Information Sciences* 432 (2018), pp. 278–300. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2017.11.064. URL: https://www.sciencedirect.com/science/article/pii/S002002551731126X.

[51] Perry J. Kaufman. *Trading Systems and Methods*. Wiley, 1995. URL: https://www.wiley.com/en-us/Trading+Systems+and+Methods%2C+5th+Edition-p-9781118043561.

[52] Jasleen Kaur and Khushdeep Dharni. “Assessing efficacy of association rules for predicting global stock indices”. In: *Decision* 49.3 (2022), pp. 329–339. DOI: 10.1007/s40622-022-00327-8. URL: https://doi.org/10.1007/s40622-022-00327-8.

[53] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:3815895.

[54] Chang-Jin Kim, Charles R. Nelson, and Richard Startz. “Testing for mean reversion in heteroskedastic data based on Gibbs-sampling-augmented randomization”. In: *Journal of Empirical Finance* 5.2 (June 1998), pp. 131–154. DOI: 10.1016/S0927-5398(97)00015-7. URL: https://doi.org/10.1016/S0927-5398(97)00015-7.

[55] Marzena Kryszkiewicz. “Concise Representations of Association Rules”. In: *Pattern Detection and Discovery*. Ed. by David J. Hand, Niall M. Adams, and Richard J. Bolton. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 92–109. ISBN: 978-3-540-45728-2.

[56] Miron B. Kursa, Aleksander Jankowski, and Witold R. Rudnicki. “Boruta – A System for Feature Selection”. In: *Fundamenta Informaticae*. 2010. URL: https://journals.sagepub.com/doi/pdf/10.3233/FI-2010-288.

[57] Albert S. Kyle. “Continuous Auctions and Insider Trading”. In: *Econometrica* 53.6 (1985), pp. 1315–1335. URL: https://doi.org/10.2307/1913210.

[58] Donald R. Lambert. “Commodity Channel Index”. In: *Commodity Traders Guide* (1980). URL: https://www.investopedia.com/terms/c/commoditychannelindex.asp.

[59] George C. Lane. “Know Sure Thing (KST) Oscillator”. In: (1994). URL: https://www.investopedia.com/terms/k/know-sure-thing-kst.asp.

[60] George C. Lane. “Stochastic Indicator”. In: *Technical Analysis of Stocks Commodities* (1950). Original concept by George Lane. URL: https://www.investopedia.com/terms/s/stochasticoscillator.asp.

[61] Hua-Fu Li and Suh-Yin Lee. “Mining frequent itemsets over data streams using efficient window sliding techniques”. In: *Expert Systems with Applications* 36.2, Part 1 (2009), pp. 1466–1477. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2007.11.061. URL: https://www.sciencedirect.com/science/article/pii/S0957417407006057.

[62] Hua-fu Li et al. “A New Algorithm for Maintaining Closed Frequent Itemsets in Data Streams by Incremental Updates”. In: *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW’06)*. 2006, pp. 672–676. DOI: 10.1109/ICDMW.2006.15.

[63] Qiubin Liang et al. “Restricted Boltzmann machine based stock market trend prediction”. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1380–1387. DOI: 10.1109/IJCNN.2017.7966014. URL: https://ieeexplore.ieee.org/document/7966014/.

[64] Yaohu Lin et al. “Improving stock trading decisions based on pattern recognition using machine learning technology”. In: *PLOS ONE* 16.8 (2021), e0255558. DOI: 10.1371/journal.pone.0255558. URL: https://doi.org/10.1371/journal.pone.0255558.

[65] Marcos López de Prado. *Advances in Financial Machine Learning*. Hoboken, NJ: John Wiley & Sons, 2018. ISBN: 9781119482086. URL: https://www.wiley.com/en-us/Advances+in+Financial+Machine+Learning-p-9781119482086.

[66] Gonzalo López Gil, Paul Duhamel-Sebline, and Andrew McCarren. “An Evaluation of Deep Learning Models for Stock Market Trend Prediction”. In: *arXiv preprint arXiv:2408.12408* (2024). DOI: 10.48550/arXiv.2408.12408. arXiv: 2408.12408. URL: https://arxiv.org/abs/2408.12408.

[67] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. “Consistent Individualized Feature Attribution for Tree Ensembles”. In: *arXiv preprint*. 2018. URL: https://arxiv.org/abs/1802.03888.

[68] Scott M. Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *NeurIPS*. 2017. URL: https://arxiv.org/abs/1705.07874.

[69] MarketVolume.com. *Directional Movement Index Rating — ADXR*. URL: https://www.marketvolume.com/technicalanalysis/adxr.asp.

[70] Tomas Martin, Guy Francoeur, and Petko Valtchev. “CICLAD: A Fast and Memory-efficient Closed Itemset Miner for Streams”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 1810–1818. ISBN: 9781450379984. DOI: 10.1145/3394486.3403232. URL: https://doi.org/10.1145/3394486.3403232.

[71] Tomas Martin, Petko Valtchev, and Louis-Romain Roux. “FGC-Stream: A novel joint miner for frequent generators and closed itemsets in data streams”. In: *2021 IEEE International Conference on Data Mining (ICDM)*. 2021, pp. 419–428. DOI: 10.1109/ICDM51629.2021.00053.

[72] Tomas Martin, Petko Valtchev, and Louis-Romain Roux. "Mining frequent generators and closures in data streams with FGC-Stream". In: *Knowledge and Information Systems* 65.8 (Aug. 2023), pp. 3295–3335. ISSN: 0219-3116. DOI: 10.1007/s10115-023-01852-3. URL: https://doi.org/10.1007/s10115-023-01852-3.

[73] Sohrab Mokhtari, Kang Yen, and Jin Liu. "Effectiveness of Artificial Intelligence in Stock Market Prediction based on Machine Learning". In: *International Journal of Computer Applications* 183 (June 2021), pp. 1–8. DOI: 10.5120/ijca2021921347.

[74] Daniel B. Nelson. "Conditional Heteroskedasticity in Asset Returns: A New Approach". In: *Econometrica* 59.2 (1991), pp. 347–370. URL: https://econpapers.repec.org/RePEc:ecm:emetrp:v:59:y:1991:i:2:p:347-70.

[75] Steve Nison. *Beyond Candlesticks: New Japanese Charting Techniques Revealed*. John Wiley & Sons, 1994. URL: https://www.wiley.com/en-us/Beyond+Candlesticks%3A+New+Japanese+Charting+Techniques+Revealed-p-9780471007205.

[76] Antonio Pagliaro. "Forecasting Significant Stock Market Price Changes Using Machine Learning: Extra Trees Classifier Leads". In: *Electronics* 12.21 (2023), p. 4551. DOI: 10.3390/electronics12214551. URL: https://www.mdpi.com/2079-9292/12/21/4551.

[77] CBOE White Paper. "CBOE Volatility Index". In: (2019). URL: https://www.sfu.ca/~poitras/419_VIX.pdf.

[78] Jian Pei et al. "PrefixSpan,: mining sequential patterns efficiently by prefix-projected pattern growth". In: *Proceedings 17th International Conference on Data Engineering*. 2001, pp. 215–224. DOI: 10.1109/ICDE.2001.914830.

[79] Stephen H. Penman. *Financial Statement Analysis and Security Valuation*. McGraw-Hill Education, 2012. URL: https://www.mheducation.com/highered/product/financial-statement-analysis-security-valuation-penman/M9780078025315.html.

[80] Piotr Pomorski. "Construction of Effective Regime-Switching Portfolios Using a Combination of Machine Learning and Traditional Approaches". Open access via UCL Discovery. Ph.D. thesis. University College London, 2024. URL: https://discovery.ucl.ac.uk/id/eprint/10192012/.

[81] Piotr Pomorski and Denise Gorse. "Improving on the Markov-Switching Regression Model by the Use of an Adaptive Moving Average". In: Mar. 2023, pp. 17–30. ISBN: 978-3-031-23843-7. DOI: 10.1007/978-3-031-23844-4_2.

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

[83] Thanadon Praphutikul and Yachai Limpiyakorn. "XGBoost-Based Multi-Factor Stock Selection Model for Rotational Trading". In: *2023 5th International Conference on Information Technology and Computer Communications (ITCC)*. 2023. DOI: 10.1145/3606843.3606862. URL: https://doi.org/10.1145/3606843.3606862.

[84] Quantopian Inc. *pyfolio: Portfolio and Risk Analytics in Python*. https://github.com/quantopian/pyfolio. Accessed: 2025-09-06. 2019.

[85] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778. arXiv: 1602.04938. URL: https://arxiv.org/abs/1602.04938.

[86] Omer Berat Sezer, M. Murat Ozbayoglu, and Erdogan Dogdu. “An Artificial Neural Network-based Stock Trading System Using Technical Analysis and Big Data Framework”. In: *Proceedings of the 2017 ACM Southeast Conference (ACMSE)*. 2017, pp. 223–226. DOI: 10.1145/3077286.3077294. URL: https://arxiv.org/abs/1712.09592.

[87] Lloyd S. Shapley. “A Value for n-Person Games”. In: *Contributions to the Theory of Games, Vol. II*. Princeton University Press, 1953. URL: https://www.rand.org/pubs/papers/P295.html.

[88] Robert J Shiller. “Do Stock Prices Move Too Much to be Justified by Subsequent Changes in Dividends?” In: *American Economic Review* 71.3 (1981), pp. 421–436. URL: https://www.aeaweb.org/articles?id=10.1257/aer.71.3.421.

[89] StockCharts.com. *Percentage Volume Oscillator (PVO)*. URL: https://chartschool.stockcharts.com/table-of-contents/technical-indicators-and-overlays/technical-indicators/percentage-volume-oscillator-pvo.

[90] StockCharts.com. *Rate of Change (ROC)*. URL: https://chartschool.stockcharts.com/table-of-contents/technical-indicators-and-overlays/technical-indicators/rate-of-change-roc.

[91] Andrew J. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 260–269. DOI: 10.1109/TIT.1967.1054010. URL: https://ieeexplore.ieee.org/document/1054010.

[92] J. Welles Wilder. *New Concepts in Technical Trading Systems*. Trend Research, 1978. URL: https://www.amazon.com/New-Concepts-Technical-Trading-Systems/dp/0934381002.

[93] Welles Wilder. “New Concepts in Technical Trading Systems”. In: (1978). URL: https://www.investopedia.com/terms/r/rsi.asp.

[94] Larry Williams. “How I Made One Million Dollars Last Year Trading Commodities”. In: (1973). URL: https://www.investopedia.com/terms/w/williamsr.asp.

[95] Larry Williams. *Ultimate Oscillator*. 1976. URL: https://chartschool.stockcharts.com/table-of-contents/technical-indicators-and-overlays/technical-indicators/ultimate-oscillator.

[96] Dominik Wolff and Fabian Echterling. “Stock picking with machine learning”. In: *Journal of Forecasting* 43 (Sept. 2023), pp. 81–102. DOI: 10.1002/for.3021. URL: https://onlinelibrary.wiley.com/doi/10.1002/for.3021.

[97] Kamran Yazdani. “Machine Learning Methods for Predicting US Recessions”. In: *Journal of Forecasting* 39.5 (2020), pp. 763–777. DOI: 10.1002/for.2671. URL: https://doi.org/10.1002/for.2671.

[98] Xianghui Yuan et al. “Integrated Long-Term Stock Selection Models Based on Feature Selection and Machine Learning Algorithms for China Stock Market”. In: *IEEE Access* 8 (2020), pp. 22672–22685. DOI: 10.1109/ACCESS.2020.2969293.