Master Thesis

Machine Learning for Dynamic Portfolio Management

to obtain the academic degree

Master of Science

at the University of Trier

Faculty IV

submitted by

Pranav Srinivas Venkatesh

Matriculation Number: 1678255

1. Supervisor: Prof. Dr. Volker Schulz

2. Supervisor: Dr. Stephan Schmidt

Trier, 7th July 2025

Statement of Authorship

I hereby declare that I have developed and written the enclosed master thesis completely by myself. I have not used sources or aids without any declaration in the text. Any thoughts, ideas, or work done by others are referenced and marked. The master thesis was not used in the same or similar version to achieve an academic grade or is being published elsewhere.

Pranav Srinivas Venkatesh

Trier, 7th July 2025

Acknowledgement

I extend my sincere gratitude to **Prof. Dr. Volker Schulz** for allowing me the privilege of pursuing my Master's Thesis under his expert supervision. His role as my thesis advisor has been immensely valuable and instrumental in shaping this work. He inspired me with the persistent encouragement and support throughout my journey.

The path through this Master's Thesis was profoundly influenced by the constant support, encouragement, and motivation from my family and friends. Their constant trust in me has been a crucial part of my success.

Furthermore, I am indebted to the entire faculty of the Data Science course for their assistance and guidance at every step of my academic endeavor. Their contributions have been crucial in my development and achievements.

Pranav Srinivas Venkatesh

Trier, 7th July 2025

Contents

List of Algorithms

List of Figures

List of Tables

List of Abbreviations

ADAM Adaptive Moment Estimation.

BL Black-Litermann Model.

BT Boosted Trees.

CAPM Capital Asset Pricing Model.

ETF Exchange-Traded Fund.

HPO Hyperparameter optimization.

LASSO Least Absolute Shrinkage and Selection Operator.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

MASE Mean Absolute Scaled Error.

ML Machine Learning.

MPT Modern Portfolio Theory.

MSE Mean Squared Error.

OLS Ordinary Least Square.

ReLU Rectified Linear Unit.

RF Random Forest.

RMSE Root mean Squared Error.

RNN Recurrent Neural Network.

SMA Simple Moving Average.

SNN Shallow Neural Networks.

SQP Sequential Quadratic Programming.

SVD Single Value Decomposition.

1 Introduction

1.1 Background Overview

A portfolio denotes a collection of investments owned by an individual, corporation, or financial entity. These types of investments may include various asset types, such as equities, fixed income securities, commodities, and real estate. Portfolio optimization seeks to maximize the expected return for a tolerable level of risk[36]. This problem lies at the heart of quantitative finance, where mathematical models guide investment decisions rather than intuition or historical bias. Portfolio optimization is a practical necessity for asset managers, institutional investors, and algorithmic trading systems. It plays a central role in wealth management, pension fund allocation, hedge fund strategies, and Exchange-Traded Fund (ETF) management. Quantitative frameworks evaluate key financial factors such as expected returns, asset correlations, and volatility. In their absence, investors risk relying on subjective judgment or overfitting to historical trends. Optimization objectives can be significantly varied across different investment horizons[20]:

- Short-term (days/weeks): Exploits transient price movements, relies on high-frequency data, and incurs greater trading costs.
- Medium-term (months): Balances tactical agility with lower transaction turnover.
- Long-term (years): Emphasis on diversification and macroeconomic trends, tolerating interim volatility.

1.2 Motivation

Classical finance assumes that prices embed all relevant information, but in practice, market frictions, behavioral biases, and data lags leave exploitable inefficiencies [51]. These deviations from theoretical efficiency create opportunities that can be strategically exploited, especially in the short- to medium-term horizon.

Short-term portfolio optimization leverages ephemeral market signals that evolve rapidly and require adaptive, data-driven models capable of quick response. Machine Learning (ML) methods are particularly well suited for this task, offering a robust framework for dynamic decision-making under uncertainty. Incorporating ML into these strategies enables

- Adaptive learning of nonlinear relationships and regime shifts.
- Frequent rebalancing to capture transient price anomalies.
- Risk-factor integration that sharpens near-term forecasts and improves resilience.

1.3 Research Objective

This study aims to identify and benchmark ML techniques that improve portfolio allocation decisions when investing through ETF. Specifically, this research aims to answer these three research questions.

1. Horizon sensitivity: Assess how weekly (short-term) and monthly (medium-term) rebalancing strategies compare to a buy-and-hold (annual, long-term) benchmark within an ETF-based portfolio optimization framework.
2. Model class effectiveness: Evaluate the performance of nonlinear models (random forests, gradient-boosted trees, Long Short-Term Memory (LSTM)) against traditional linear models (Ordinary Least Square (OLS), Least Absolute Shrinkage and Selection Operator (LASSO)) in terms of risk-adjusted returns across different time horizons.
3. Impact of Dynamic Risk Metric Integration: Investigate how incorporating dynamic risk metrics into ML-based portfolio optimization enhances portfolio adaptability and consistency across varying market regimes and investment horizons.

This research seeks to offer actionable information by addressing these concerns, influencing the selection and implementation of algorithms to construct a more resilient portfolio optimization strategy that responds to fluctuating market conditions and enhances risk-adjusted performance.

1.4 Thesis Roadmap

The structure of this thesis follows these steps.

Chapter 1: Introduces the research context, outlines the motivation behind the study, defines the core problem, and provides an overview of the thesis structure.

Chapter 2: Reviews the evolution of portfolio optimization strategies, highlights the role of machine learning models in financial forecasting, discusses existing related works, and identifies gaps and how current research is addressing them.

Chapter 3: Details the dataset used, defines the target and predictor variables, describes preprocessing, data splitting techniques, and forecasting task type, and outlines the regime-based segmentation for analysis.

Chapter 4: Explains the training methodology, discusses various ML models and their working, the hyperparameter tuning approach used, the evaluation metrics used to assess model performance and finally forecasting procedure.

Chapter 5: Defines and explains quantitative financial metrics used to evaluate forecast and portfolio risk profiles. Further discussion on composite score formulation and details the weight optimization strategy for ranking assets and portfolio allocation.

Chapter 6: Detailed discussion on the implementation of the backtesting process.

Chapter 7: Presents the empirical results of the study, including metric comparisons, model forecast behavior, composite score stability, and a comparison with classical buy-and-hold strategies.

Chapter 8: Summarizes the key findings of the research and suggests potential directions for future work in dynamic portfolio optimization and financial forecasting.

Note: From Chapters 3 to 6, the programmatic setups will also be described alongside the theoretical explanations with mathematical equations and along with their algorithms for easier interpretations.

Experimental Workflow

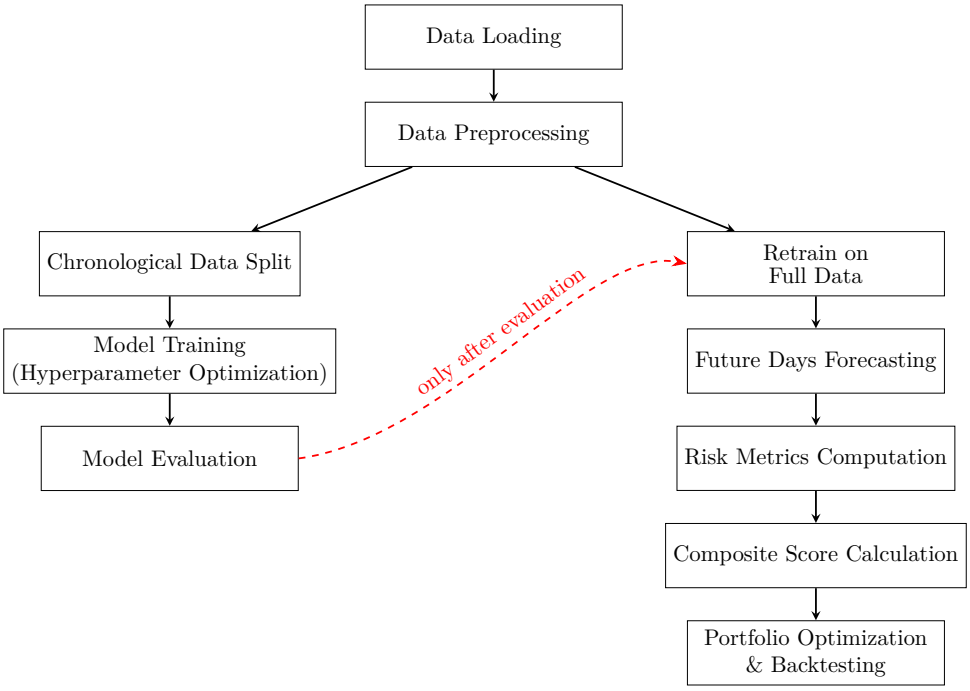

Figure 1: Workflow of the experimentation carried out

2 Literature Survey

The financial sector, which includes banks, asset management firms, credit service providers, and insurance companies, is essential for facilitating capital flow, risk assessment, and economic growth [22]. A core task for investors operating in these markets is portfolio optimization, i.e., the strategic allocation of assets like stocks, bonds, and cash to maximize return for a given level of risk. Within this ecosystem, the stock market acts as both a barometer and driver of macroeconomic performance, though its impact varies across countries based on structural and regulatory differences [37].

Modern investment practice also recognizes that portfolio objectives differ depending upon investment horizon. Although long-term strategies can focus on capital appreciation and low turnover, short- and medium-term portfolios can leverage short-term market inefficiencies and tactical opportunities.

2.1 Evolution of Portfolio Optimization

Modern Portfolio Theory (MPT)

The theoretical foundation for portfolio optimization dates back to Harry Markowitz's mean–variance framework, as mentioned in [34], which introduced the trade-off between expected return and portfolio variance as a basis for rational investment decisions. Ahmad et al. summarizes how Markowitz's MPT framework follows in [1].

$$\min_{w} \; w^\top \Sigma w \tag{1}$$

$$\text{subject to:} \quad w^\top 1 = 1, \quad w^\top \mu = \mu_p$$

where w is the vector of asset weights, Σ is the covariance matrix of asset returns, μ is the vector of expected returns, and μ_p is the desired portfolio return. However, the MPT model is sensitive to input parameters and assumes perfect markets, which often limits its real-world applicability.

Capital Asset Pricing Model (CAPM)

The above-mentioned classical approach was later extended by William Sharpe through models like the CAPM and multi-factor models, which provided closed-form solutions for expected returns based on systemic risk exposures. While these models offer interpretability and tractability, they often rely on assumptions of normally distributed returns, stable correlations, and linear dependencies; these conditions are rarely satisfied in real-world markets.

Assumption	Description
1. Mean-variance optimization	Investors maximize utility based on expected return and variance.
2. Homogeneous expectations	All investors have the same beliefs about returns, variances, and covariances.
3. Single-period horizon	All investment decisions are made for a common single period.
4. Risk-free asset	A risk-free asset exists, and investors can borrow or lend at this rate.
5. Frictionless markets	No taxes, transaction costs, or short-selling restrictions.
6. Marketable assets	All assets are publicly traded and divisible.
7. Rational investors	Investors behave logically and seek to maximize utility.

Table 1: Summary of CAPM Assumptions[47]

The CAPM framework approach can be summarized as follows [47][1]:

$$E(R_i) = r_f + \beta_i \left(E(R_m) - r_f\right) \tag{2}$$

where $E(R_i)$ is the expected return of the $asset_i$, r_f is the risk-free rate, β_i is the asset's beta, and $E(R_m)$ is the expected return of the market portfolio.

The market β is given by

$$\beta_{im} = \frac{COV(R_i, R_m)}{\sigma^2} \tag{3}$$

The beta coefficient measures an asset's exposure to systematic (market) risk and indicates how the asset moves relative to the overall market.

Black-Litermann Model (BL)

The Black-Litterman asset allocation model was developed by Fischer Black and Robert Litterman at Goldman Sachs in the early 1990s. This model addresses key problems of MPT, such as sensitiveness to the expected returns and non-incorporation of investors' views.

This model framework is a Bayesian approach to portfolio optimization that combines ideas from CAPM and the Markowitz mean-variance optimization model to provide a method for calculating optimal portfolio weights based on the inputs given. The approach of the BL method is a multistep procedure that is summarized as follows [4][24][52]:

Step 1: Start with the market equilibrium assumption for the returns

The BL model begins by assuming that market weights are optimal under equilibrium (CAPM-style world, where the expected return is proportional to the market premium scaled by risk exposure). From this assumption, compute implied expected returns (equilibrium returns):

$$\Pi = \lambda \Sigma w_{\text{mkt}} \tag{4}$$

where Π is the implied excess equilibrium return vector, Ω^{-1} the covariance matrix of returns, λ is the risk aversion coefficient, and w_{mkt} is the market capitalization weights of the assets.

Step 2: Estimate the Risk Aversion Coefficient λ

Risk-aversion coefficient λ measures the trade-off between risk and return in the market. It is estimated as follows:

$$\lambda = \frac{E(R) - r_f}{\sigma^2} \tag{5}$$

where E(R) is the expected return of the portfolio, r_f is the risk-free rate, and σ^2 is the variance of portfolio returns.

Step 3: Express Investor Views

The investor's view is represented across k linear combinations of the elements of E(R).

$$PE(R') = V + \epsilon \tag{6}$$

where P is the known k * n matrix, V is the vector of expected value of the investor views' outcomes, ϵ is an unobservable normally distributed random vector, N (0, Ω) and Ω is a denotes the uncertainty in investor views.

Step 4: Apply Bayesian Updating using the BL method.

The BL method expected return is calculated based on the vector of equilibrium returns and the vector of investor's views.

$$\mu^{BL} = \left[(\tau\Sigma)^{-1} + P^\top \Omega^{-1} P\right]^{-1} \left[(\tau\Sigma)^{-1}\Pi + P^\top \Omega^{-1} V\right] \tag{7}$$

where τ is the scalar representing confidence in the market equilibrium (smaller value implies higher trust in market equilibrium estimates, while a larger value allows investor views to have more influence on the posterior expected returns), Σ is the covariance matrix of the asset's excess returns.

Step 5: Portfolio optimization:

Perform mean variance optimization to determine portfolio weights.

$$w^* = \frac{1}{\lambda}\Sigma^{-1}\mu^{BL} \tag{8}$$

This expression is consistent with the classical MPT formulation for unconstrained mean-variance optimization, where the optimal weights are given by $w^* = \frac{1}{\lambda}\Sigma^{-1}\mu$. In the BL model, the expected return vector μ is replaced by the posterior estimate μ^{BL}, thereby incorporating both market equilibrium and investor views.

ML techniques

Machine learning (ML) models offer the flexibility to capture non-linear interactions, non-stationarity, and higher-dimensional data, which are increasingly relevant in complex and noisy financial environments. ML-driven portfolio optimization reframes the task as a data-driven prediction-and-allocation problem, where models first forecast asset-level returns or risk metrics and then use these forecasts to optimize weights under real-world constraints such as liquidity, transaction costs, or regulatory limits.

2.2 Machine learning (ML) Methodologies in Financial Forecasting

ML refers to computational techniques that allow models to recognize patterns and anticipate data outcomes without explicit programming. Regularization, ensembling, and temporal learning improve ML model performance by learning complex mapping functions from data. ML is increasingly used for dynamic market return prediction and portfolio creation. The ML models used in this study span three key methodological categories:

Linear Models

Linear models are among the most fundamental and widely used statistical modeling. The linear regression obeys the following key assumptions [9]:

- **Linearity:** The relationship between the independent variables and the dependent variable is linear.
- **Independence of Errors:** The residuals (errors) are independent of each other.
- **Homoscedasticity:** The residuals have constant variance across all levels of the independent variables. If violated, it leads to heteroscedasticity, which affects the reliability of confidence intervals.
- **Normality of Errors:** The residuals should be normally distributed, especially for small sample sizes.
- **No (Perfect) Multicollinearity:** Independent variables should not be highly correlated with each other. High multicollinearity makes it difficult to estimate individual regression coefficients accurately.

Linear models, such as the OLS and LASSO, are the most interpretable and computationally efficient approaches. OLS assumes a linear relationship between predictors and the target variable (e.g., asset return) and is optimal under conditions of homoscedasticity and no multicollinearity. LASSO extends OLS by introducing an L1 penalty, which performs both regularization and variable selection.

Tree-Based Ensemble Models

Decision Trees

A decision tree is a supervised machine learning algorithm that is used for both classification and regression tasks. Models decisions and their possible outcomes in a treelike structure. Each internal node represents a feature split, each branch a decision rule, and each leaf node a prediction.

Working of Decision Trees

1. **Splitting:** The dataset is split recursively based on feature values using a chosen metric (Mean Squared Error (MSE) for regression and Gini entropy for classification tasks).
2. **Best Split Selection:** At each node, the algorithm selects the feature and threshold that result in the greatest reduction in impurity or error.
3. **Stopping Criteria:** Splitting stops when termination criteria is met such as maximum tree depth is reached, or minimum number of samples per node is met, or further splitting does not improve the model significantly

Disadvantages of Decision Trees

- Trees can grow too deep and capture noise in the training data.

- Small changes in the data can lead to a completely different tree.
- Features with more levels tend to dominate splits.

Ensemble Methods

Ensemble learning is a strong machine learning technique that uses many models to increase forecast efficiency, robustness, and generalization. Ensemble models are especially useful in financial prediction since market data is noisy, non-linear, and frequently influenced by latent factors. They capture complex relationships and stabilize projections over time and asset classes. Tree-based ensemble approaches circumvent the limitations of decision trees by combining numerous trees to create a more powerful prediction.

Random Forest (RF) is an ensemble method based on bagging (Bootstrap Aggregating). It builds a large number of decision trees on different random subsets of the data and features and then averages their predictions to reduce variance. Boosted Trees (BT), in contrast, follow a boosting approach. Rather than training trees independently, they build it sequentially, where each tree is trained to correct the residual errors made by the previously fitted trees.

Neural Networks :

Neural networks are a class of machine learning models inspired by the structure of the human brain as depicted in Figure 2, designed to learn complex, non-linear relationships between inputs and outputs through interconnected layers of computational units known as neurons. In finance, they are particularly useful for capturing interactions and functional forms that traditional models might miss, especially in high-dimensional or noisy environments where patterns are not easily specified by domain knowledge.

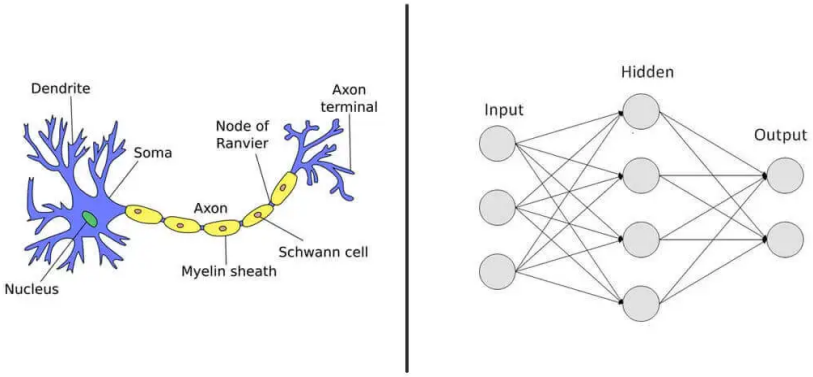

Figure 2: Neural Network similarities with human brain neuron [5]

Working of Neural Networks:

The core mechanism of neural networks involves transforming input features through affine mapping as weighted sums plus biases, followed by non-linear activation functions, enabling them to approximate virtually any continuous function given sufficient capacity, a result formalized by the universal approximation theorem. A typical architecture comprises an input layer, stacks of hidden layers, and an output layer, where each connection carries a trainable weight and each neuron adds its own bias. During a forward pass, data propagate layer by layer to produce a prediction that is evaluated by a chosen loss function such as mean-squared error or cross-entropy. The backpropagation algorithm then applies the chain rule to compute gradients of this loss with respect to every parameter, and an optimizer, often a variant of gradient descent, updates the weights using a tunable learning rate, typically after processing a mini-batch of data; a full sweep of the dataset is an epoch. Techniques such as batch normalization accelerate convergence, while regularization methods like early stopping curb overfitting for better generalization on unseen data.

The Evolution of LSTM Networks through RNN

Recurrent Neural Network (RNN) extend feed-forward nets by adding recurrent connections that feed a neuron's previous hidden state back into itself, letting the model process sequential data such as text or time series, however, standard RNNs struggle with long-term dependencies because gradients tend to vanish or explode as they are back-propagated through many time steps, making it hard to learn relationships that span more than a few steps.

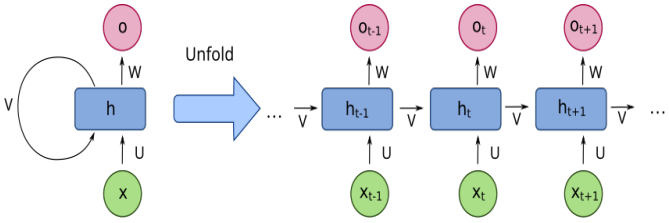

Figure 3: RNN Architecture[13]

Later LSTM networks were introduced to solve this problem by adding a memory cell and gating mechanisms: input, forget, and output gates. These gates regulate information flow, preserve gradients over long intervals, and thus enable effective learning of both short- and long-range patterns in sequences.

LSTM Memory block representation

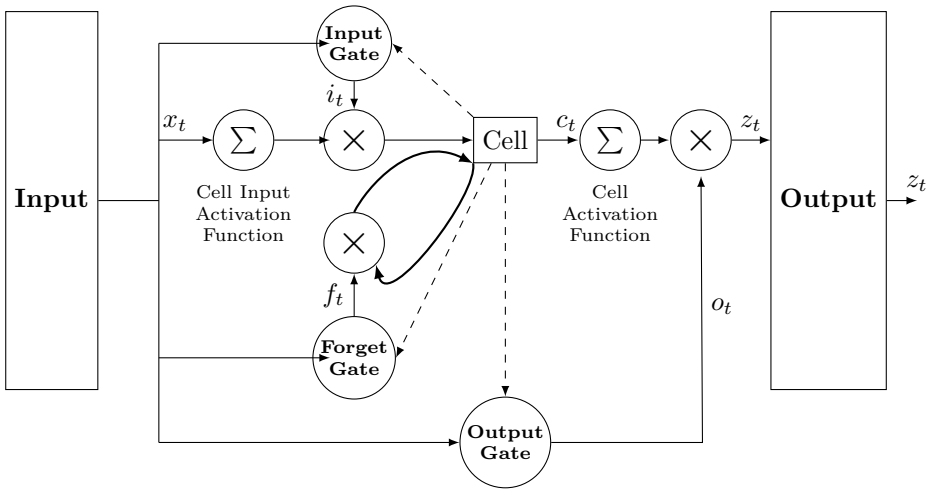

Figure 4: LSTM Block Cell [49]

Figure 4 represents the cell representation of LSTM block. The gating mechanism in the cell is represented as follows[21]:

1. **Input (x_t)**: The external input at time step t is fed into the LSTM cell.
2. **Forget Gate (f_t)**: Determines how much of the previous cell state (c_{t-1}) should be retained or discarded. It takes x_t and the previous hidden state (h_{t-1}) as inputs. The output is a value that scales the previous memory content.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{9}$$

3. **Input Gate (i_t)**: Controls how much new information should be added to the cell state. It also takes x_t and h_{t-1} as inputs and influences the new candidate memory content that updates the cell state.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{10}$$

4. **Cell State (c_t)**: The internal memory of the LSTM unit that gets updated through a combination of retaining useful past information is regulated by the forget gate while adding new relevant information is handled by the input gate.

$$\text{Candidate Cell State:} \quad \tilde{C}_t = \phi\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \quad ; \phi \text{ is the desired activation function} \tag{11}$$

$$\text{Cell State Update:} \quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{12}$$

5. **Output Gate (o_t)**: Determines how much of the internal memory should contribute to the output. It influences the hidden state passed to the next time step.

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{13}$$

6. **LSTM Layer Output (z_t)**: The final output of the LSTM block at time t, computed based on the updated cell state.

$$z_t = o_t \odot \phi(C_t) \tag{14}$$

Note: Although the output of the LSTM layer is commonly represented as h_t and is the preferred notation in most literature, it is renamed as z_t in this work to avoid confusion. This is because the symbol h is later used to represent the Hessian representation in optimization-related contexts.

2.3 Related Works

Empirical Asset Pricing with Non-Linear Machine Learning Models

Gu, Kelly, and Xiu [19] introduce a comprehensive machine learning framework for predicting cross-sectional monthly stock returns using a large-scale U.S. equity dataset. They benchmark a diverse set of models, including elastic net, principal components, partial least squares, random forests, gradient-boosted trees, and neural networks (both shallow feedforward and LSTM), to examine how effectively each captures nonlinearities and interaction effects among predictors.

Shallow Neural Networks (SNN) are neural network architectures with only one or two hidden layers, which are effective for static, cross-sectional prediction tasks but do not capture temporal dependencies in data, as they treat all input observations as independent. The dataset includes 94 firm-level characteristics, 8 macroeconomic variables for U.S. stocks. The authors emphasize prediction over portfolio optimization, evaluating model performance using out-of-sample cross-sectional R2 and Sharpe ratios of long-short decile portfolios sorted by predicted returns.

Their findings demonstrate that machine learning methods, particularly tree-based models and neural networks, substantially outperform traditional linear approaches in forecasting stock returns. These models uncover rich nonlinear relationships and interactions that linear models miss. Despite differences in architecture, the best-performing models consistently identify a shared set of predictive signals, including momentum, volatility, liquidity, and profitability factors. Furthermore, ML-based return forecasts lead to economically meaningful gains, producing portfolios with significantly higher Sharpe ratios and minimal exposure to traditional risk factors, suggesting the presence of true alpha rather than simple factor tilts.

The authors adopt a simplified portfolio construction approach using equal- and value-weighted schemes, assuming flat transaction costs and excluding turnover constraints or horizon-based considerations. While the inclusion of LSTM models hints at potential for temporal modeling, their sequential advantages are not deeply explored, and the treatment remains primarily cross-sectional. Overall, the study provides a robust predictive benchmark for asset pricing research, demonstrating the power of machine learning methods in extracting return-predictive information from high-dimensional financial data, though it stops short of presenting a fully deployable trading strategy.

Cost-Constrained and Interpretable Machine Learning for Equity Portfolios

Li, Simon, and Turkington [32] extend machine learning applications in asset management by emphasizing two underexplored but critical criteria for institutional adoption: investability and interpretability. Working within a constrained asset universe of liquid, large-cap U.S. equities (S&P 500 constituents), the authors evaluate a suite of models, ranging from linear regressions (OLS, LASSO) to nonlinear methods like random forests, gradient-boosted trees, and shallow neural networks. Central to their contribution is the introduction of explicit constraints on portfolio construction, such as turnover limits, tradability thresholds, and liquidity screens, which ensure the practical feasibility of deploying ML-based strategies at institutional scale.

The methodology centers on translating ML-predicted return signals into rule-based, long-only portfolios with monthly and annual rebalancing frequencies. Portfolios are formed using simple weighting schemes (equal-weight and volatility-scaled) that respect real-world capacity constraints and mitigate excessive trading. Their framework is deliberately designed to maintain consistency with compliance and implementation requirements. However, the study does not include sequence models such as LSTM, nor does it experiment with dynamic or multistage portfolio optimization. Instead, it prioritizes clarity, stability, and cost sensitivity over adaptability or complexity.

Perhaps the most novel aspect of the paper is the "Model Fingerprint," a diagnostic tool that dissects ML model behavior into linear effects, nonlinear transformations, and interaction effects across individual predictors. This enhances interpretability by revealing how different models arrive at their predictions, aiding users in comparing complexity versus incremental insight. The authors argue that the ultimate value of machine learning lies in producing genuinely informative predictions, ones that are not easily replicable by traditional factor models. They further advocate for integrating investor preferences and domain knowledge during model design, such as customizing loss functions or selecting return horizons, to align quantitative signals with specific portfolio objectives. This hybrid approach, combining ML rigor with human insight, demonstrates how interpretable and investable machine learning can serve as a practical bridge between data-driven predictions and institutional portfolio management.

2.4 Research Gap

This study aims to extend the previously outlined methodologies while directly addressing the associated practical limitations, thereby enhancing both the rigor and relevance of the proposed approaches. First, it evaluates model performance across multiple investment horizons, that is, weekly (short-term), monthly (medium-term), and annual (long-term), to uncover horizon-sensitive patterns in portfolio performance. Second, it compares temporal (LSTM) and non-temporal (OLS, LASSO, RF, Boosted Trees (BT)) models within a shared data environment and rolling-window setup, allowing for consistent performance benchmarking. Third, it deviates from rule-based portfolio construction by employing weight-based optimization for the objective function of the composite scores, which is formulated by inculcating different risk measures to dynamically allocate capital.

Finally, this study explicitly incorporates turnover penalties and transaction-cost drag, resulting in a practical, cost-aware, and risk-adjusted portfolio framework to replicate real-world deployment. By addressing these interlinked dimensions and model type, allocation flexibility, risk measurement, and rebalancing cost, this research offers a more comprehensive solution to data-driven portfolio optimization.

3 Data Preparation, Forecasting Framework, and Market Regime Design

This chapter presents the overall framework for preparing data, defining variables, applying preprocessing techniques. It also outlines the modeling approach, data splitting strategy, and the selection of specific market periods (2022 and 2024) for portfolio backtesting.

3.1 Dataset Overview

Dataset that is used is from Yahoo Finance data. It makes it easier to obtain historical market data, financial statements, and other stock.

Exchange-Traded Funds (ETF)

ETFs are the subject of this research study. The goal of index funds is to achieve the same as their benchmark indices. It is possible to buy and sell ETFs on an exchange throughout the day, in contrast to mutual funds[15]. For both individual and institutional investors, ETFs are a crucial part of the contemporary investment environment because they offer a flexible and affordable method of portfolio diversification. ETFs allow investors to focus on specific markets, industries, or subjects without having to deal with the complexities of researching and buying individual assets (like stocks, bonds, or commodities). Before 2008, ETFs were usually index funds. That's when the Securities and Exchange Commission of the United States gave permission for an actively managed exchange-traded fund to be made[40].

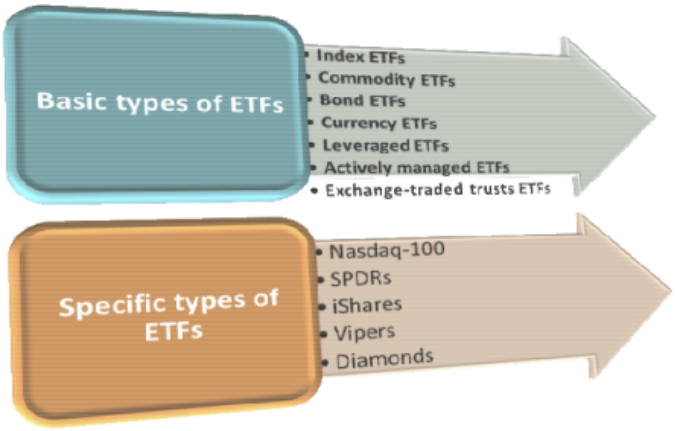

Figure 5: Classification Types of ETFs as described by Elitsa Petrova in [40]

Figure 5 shows the classification of ETFs into two main groups: basic types and specific types. Basic types are defined based on the asset class or strategy they follow. Basic types of ETFs are classified by the assets they invest in or their investment strategies. Index ETFs track market indices, while commodity, bond, and currency ETFs invest in respective asset classes like gold, government bonds, or foreign currencies. Leveraged ETFs aim to amplify returns, actively managed ETFs are overseen by fund managers, and exchange-traded trusts are structured similarly to ETFs but often focus on specific assets like precious metals. Specific types refer to well-known branded ETFs like Nasdaq-100, SPDRs, iShares, Vipers, and Diamonds. Typically, these types are subsets of the basic types, but their marketing or structure differs.

Ticker	Name	Description
SMH	VanEck Semiconductor ETF	SMH studies the 25 largest U.S.-listed semiconductor companies that power modern computers. Smartphones, calculators, and computers use semiconductor chips. These CPUs will power new devices as technology advances.
SOXX	iShares Semiconductor ETF	SOXX tracks modern computer benchmarks from semiconductor makers. Smartphones, calculators, and computers employ semiconductor chips. New devices will use these processors as technology improves. The portfolio includes 25% U.S. and 25% international stocks for regional exposure.
PSI	Invesco Semiconductors ETF	The PSI benchmark generates returns based on semiconductor firm investing needs. The fund exclusively invests in U.S. companies, providing exposure to various semiconductor makers.
XSD	SPDR S&P Semiconductor ETF	A major semiconductor manufacturer benchmark for modern computing is tracked by XSD. Computers, calculators, and smartphones require semiconductor chips. New devices will use these processors as technology improves.
IYW	iShares U.S. Technology ETF	This ETF is one of numerous ways investors might enter the volatile U.S. tech sector at little cost. IYW's sector-specific focus may be too limited for long-term buy-and-hold investors, but tactical investors seeking tech sector exposure may benefit.
XLK	Technology Select Sector SPDR Fund	XLK gives investors exposure to many IT giants under one ticker. A corporation must be in one of many technology industries to be in this ETF: IT services, wireless telecommunications, and semiconductors are examples.
VGT	Vanguard Information Technology ETF	VGT evaluates several IT software, consulting, and hardware companies. Technology leaders across market sizes are targeted by this fund. The fund invests only in U.S. companies and three stocks (25%). The portfolio has 425 stocks, 54% of which are top 10.
QQQ	The Invesco QQQ	This ETF tracks the Nasdaq-100 Index, which includes 100 of Nasdaq's largest non-financial businesses, mostly in technology. Tech-heavy diversifiers benefit from high-growth companies. Sector specialization and significant volatility have produced excellent long-term returns.
IGM	iShares Expanded Tech Sector ETF	This ETF provides low-cost exposure to the U.S. tech sector, one of many possibilities for investors wishing to enter an industry that can see sharp upward and downward movements.
IXN	iShares Global Tech ETF	For investors seeking exposure to the volatile global IT sector, this ETF is one of many options. IXN's sector-specific focus may be too narrow for long-term, buy-and-hold investors, but tactical investors seeking tech sector exposure may benefit. At an average cost, IXN has good liquidity.

Table 2: Summary of ETFs used in this research. [54]

3.2 Variable and Feature Definitions

Selection of features are crucial for forecasting model success. Relevant and high-quality predictors improve performance, while irrelevant or poorly chosen features cause complexity, overfitting, and poor generalization. For ML models, it is crucial to achieve balance in the overall number of predictors employed. Incorporating numerous variables can result in overfitting, heightened complexity, and prolonged training durations, whilst

utilising few predictors may lead to underfitting and suboptimal model efficiency. In consideration of these issues, along with practical limitations such as constrained time, a thorough feature selection approach was implemented. Notable features were selected based on statistical relevance, predictive strength, and domain expertise to maintain the model's efficiency and effectiveness.
Note: The variable consideration and feature calculation for the LSTM model differ slightly from other models. This is primarily due to LSTM's ability to capture temporal sequences through *lookback,* whereas other models lack this advantageous feature.

3.2.1 Target Variable

Log_Returns: Log returns are calculated by taking the natural logarithm of the ratio of consecutive close prices. They are widely used in quantitative finance and time series modeling due to their mathematical properties. The popular application of log transformation is to eliminate data variability, particularly in datasets containing outlier observations.

$$y_t = \ln(\frac{P_t}{P_{t-1}})$$

where:
– y_t is the logarithmic return of any ETF at time t,
– P_t, P_{t-1} are the closing price of any ETF at time t and $t-1$ respectively,

3.2.2 Predictor Variables

Lags:

In time series forecasting, lags refer to the past values of a variable, which serve as inputs to predict future values. It helps in capturing temporal dependence, as most time series data have autocorrelation, exhibiting past values influencing future values. Lags serve as independent variables (predictors) when using ML or statistical models. Incorporating lags helps models learn recurring patterns and trends.

Given a time series $\{Y_t\}$, a lag of order k is defined as:

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-k})$$

where:
– y_t is the current log returns at time t.
– y_{t-k} is the value k time steps before t.

This research considers lags calculated on *log_returns*. The model is utilizing the last 80 time steps as input features to predict the future for all traditional ML models, with a maximum latency of 80. This establishes equilibrium between capturing long-term interdependence and mitigating excessive complexity. Nonetheless, adding lags beyond this threshold may result in overfitting and computational inefficiency.

Note: For LSTM, these lags are ignored. LSTMs are able to observe entire temporal sequences of previous values due to their lookback capability.

Simple Moving Average (SMA):

The Simple Moving Average (SMA) is a commonly used statistical indicator in time series analysis and financial forecasting. It is a kind of moving average that computes the unweighted mean of a specified quantity of preceding data points. SMA helps in smoothing noisy data, making it easier to observe trends.

SMAs are computed on *Close* values for LSTM, while for other models it is computed on *log_returns*. LSTMs can learn from *Close* prices as they model temporal patterns and handle nonstationarity. Tree-based and linear models need stationary, scale-independent features, so SMA on log returns is more appropriate.

For a given time window W, the SMA at time t is computed as:

$$SMA_t = \frac{1}{W} \sum_{i=t-W+1}^{t} P_i \tag{15}$$

where:
– SMA_t is the simple moving average at time t.
– P_i is the close value at time i.
– W is the number of periods over which the average is computed.

This is the standard SMA formula and is typically used for LSTM model. Since LSTM models process temporal sequences, they use the past W data points up to time t to forecast the next value at $t+1$. However, for tree-based and linear models, this formula must be slightly modified to exclude the current time step to avoid data leakage when predicting the value at t. The modified formula is:

$$SMA_t = \frac{1}{W} \sum_{i=t-W}^{t-1} y_i$$

where:

– y_i is the log returns at time i.

This study analyzes market behavior using 5, 10, and 50-day simple moving averages (SMAs) to capture short, medium, and long-term trends. The 5-day SMA reflects short-term momentum and volatility, the 10-day SMA balances responsiveness with trend reliability, and the 50-day SMA offers a broader view of long-term market direction. For LSTM modeling, 5, 10, and 15-day windows are used to focus on recent temporal sequences that are most relevant for predicting near-future price movements. These shorter timeframes enable the LSTM to effectively learn sequential dependencies in the data without diluting important short-term patterns, making them well-suited for dynamic financial forecasting.

Seasonality Features:

Seasonality refers to repetitive patterns in time series data that occur at regular intervals as a result of external factors such as the time of day, day of the week, or particular months of the year. Incorporating seasonality is essential for enhancing forecasting models, as it enables them to recognize and anticipate cyclical fluctuations in the data.

- **Day of the week:** One of the most frequently used temporal features in finance is the day of the week, as market behavior often varies across weekdays. For instance, the stock market tends to show lower returns on Mondays, known as the "Monday effect," while Fridays often experience higher volumes and volatility as traders adjust positions before the weekend.
- **Week of the year:** The week of the year is valuable for capturing seasonal patterns tied to specific weeks rather than fixed dates. In finance, this can highlight spikes in activity during earnings season, year-end rebalancing, or tax filing deadlines. For example, hedge funds and mutual funds often restructure portfolios in the final weeks of the year, impacting asset flows and stock performance.
- **Month of the year:** The month feature helps financial models capture recurring annual patterns. Tax season in April often affects cash flows, while December typically sees increased spending and investment adjustments. January often exhibits the "January effect," where stock prices, especially small caps, tend to rise. Financial institutions use monthly trends to forecast revenues, plan risk management, and optimize asset allocation.
- **Quarter of the year:** Quarters are fundamental in the financial world due to the quarterly earnings reporting cycle. Stock prices can swing sharply based on earnings surprises or guidance provided during these reports. Investment banks and asset managers also rebalance portfolios quarterly to maintain strategy compliance or optimize performance. Including the quarter in models helps detect business cycles, fiscal strategies, and investor sentiment shifts.

Close Price:

Yfinance data indicates the asset's most recent trading price prior to the market's closure on that date. The standard closing price is employed in technical analysis, return calculations, and predictive models. The LSTM model analyzes only proximate values due to its operation with sequential input. LSTM networks are capable of learning temporal dependencies, enabling the model to capture stock price patterns over time through closing prices. Close values are crucial in identifying sequential dynamics such as momentum, reversals, and volatility patterns, as they represent an asset's daily trading price. Training the LSTM with these parameters enhances its ability to predict price movements based on historical trends.

3.3 Pre-processing Methods:

Data preprocessing is essential for any machine learning project, as the quality of the input data directly affects the accuracy and dependability of the model's predictions. This step involves transforming the raw data into a form that machine learning algorithms can comprehend and assess [7].

3.3.1 Feature Expansion with Dimensionality Reduction:

This approach is used for linear models and tree-based models to enhance their predictive power while controlling complexity and maintaining interpretability. The polynomial method is used to increase the number of features on a large scale. LSTM omits this process.

To generate only the second-degree interaction terms of the polynomial feature, transformation is configured with `degree = 2` and set `interaction_only = True` to exclude self-interaction terms. This leads to the following terms:

Original Features: Let $x = [x_1, x_2, \ldots, x_p] \in \mathbb{R}^p$ represent the feature vector of a single sample (i.e., a row in the data matrix $X \in \mathbb{R}^{n \times p}$, where n is the total datapoints and p is the number of features). The original features are directly included without modification.

$$x_j \quad \text{where} \quad j = 1, 2, \ldots, p$$

Interaction Terms:

$$x_j \cdot x_k \quad \text{where} \quad j \neq k, \quad x_j, x_k \in x, \quad j, k = 1, 2, \ldots, p \tag{16}$$

These terms allow the model to capture pairwise interactions between different features, which helps in modeling non-linear relationships in the data without increasing the complexity excessively.

Omission of Squared Terms:

Due to the polynomial transformation setting that is mentioned previously, pure squared terms of the form x_i^2 are omitted. This helps in reducing the dimensionality of the feature space and prevents overfitting by avoiding unnecessary complexity.

Dimensionality reduction by Single Value Decomposition (SVD): Though squared terms are removed, the number of interaction terms can still grow rapidly, especially with a large number of original features. This leads to a high-dimensional feature space that can cause overfitting and increase computational complexity. To address this issue, a dimensionality reduction technique is applied by making use of **SVD** [8].

$$X_k = U_k \Sigma_k V_k^T \tag{17}$$

where:

- $U_k \in \mathbb{R}^{m \times k}$ contains the top k left singular vectors.
- $\Sigma_k \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the top k singular values.
- $V_k^T \in \mathbb{R}^{k \times n}$ contains the top k right singular vectors.

This truncated representation preserves most information while reducing feature count to prevent overfitting and improve efficiency. To avoid losing original features during SVD, they are retained separately, and dimensionality reduction is applied only to interaction terms. The reduced interaction features are then combined with the original ones and scaled before modeling.

These ineraction features are kept slightly more for linear models as they require more expanded features to model nonlinearity, while tree-based models capture it inherently with fewer transformed terms.

3.3.2 Scaling:

Machine learning and deep learning require feature scaling as a preprocessing step. It guarantees that various features contribute uniformly to model training and mitigates data scaling issues. Two prevalent scaling techniques are MinMaxScaler and StandardScaler, each advantageous for certain models. This study uses MinMax Scaling for LSTM, whereas other machine learning models utilize Standard Scaling. Both input and target variables are normalized.

MinMax Scaling:

MinMax Scaler transforms your data by scaling it to a fixed range, typically between 0 and 1 or -1 and 1. Here, the default scaling range of 0 to 1 is used. The MinMax Scaler is especially advantageous for neural networks such as LSTM, as these models exhibit enhanced performance when features are normalized to a uniform scale between 0 and 1. The LSTM design employs activation functions that respond to the amount of input, and scaling data to a limited, fixed range mitigates problems such as vanishing gradients. The retention of zero values is essential when handling sequence data. The datapoints are scaled as following[7]:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}, \quad y' = \frac{y - y_{\min}}{y_{\max} - y_{\min}} \tag{18}$$

where:
- X, y : Original value of the input and target feature.
- $X_{\min}$, $y_{\min}$: Minimum value in the input and target feature.
- $X_{\max}$, $y_{\max}$: Maximum value in the input and target feature.
- X', y' : Scaled value of the input and target feature.

Note. When $X \in \mathbb{R}^{n\times p}$, the scaling is applied independently to each feature (i.e., column-wise). That is, $X_{\min}, X_{\max} \in \mathbb{R}^{1\times p}$ contain the minimum and maximum values for each column, and the operations are broadcast row-wise.

Standard Scaling:

The Standard Scaler normalizes features by eliminating the mean and adjusting to unit variance. It yields a distribution with a mean of 0 and a standard deviation of 1. This scaler exhibits greater resilience to outliers than the MinMax Scaler and is especially effective for conventional machine learning techniques. The standard scaler helps ensure adherence to the assumptions made by such methods while maintaining the structure of the normal distribution, provided the data is already normally distributed. This method scales the data as follows [7]:

$$X' = \frac{X - \mu}{\sigma}, \quad y' = \frac{y - \mu}{\sigma} \tag{19}$$

where
- X, y : Original value of the input and target feature.
- μ, σ : Mean and standard deviation of the feature considered.
- X', y' : Standardized value of feature considered.

Note. When $X \in \mathbb{R}^{n\times p}$, the standardization is applied independently to each feature (i.e., column-wise). That is, $\mu, \sigma \in \mathbb{R}^{1\times p}$ contain the mean and standard deviation of each column, and the operations are broadcast row-wise.

3.3.3 LSTM input shape:

LSTM models require input data to be in a 3-dimensional format with shape (samples, time_steps, features), where samples refers to the number of sequences in the dataset, time_steps (or look-back steps) are used for each prediction, and features represents the number of variables observed at each time step. This format allows LSTMs to capture temporal dependencies across multiple steps in sequential data. Typically, before feeding data into an LSTM, the features are flattened to 2D for scaling and then reshaped back to 3D to match the required input shape. This structure is essential for enabling LSTM layers to learn from the temporal ordering of input sequences.

3.4 Data Splitting for Model Training, Testing, and Forecasting

In time series forecasting, a chronological train-test split is more appropriate than a random split, as random splitting can disrupt the temporal structure of the data, potentially causing the model to miss important patterns and market dynamics. A chronological split preserves the natural sequence of events, preventing the risk of data leakage and ensuring more realistic model evaluation.

For all exchange-traded funds (ETFs), the preferred starting point is the year 2000. However, many ETFs were launched after 2000, resulting in differing lengths of historical data across instruments. This variation in data availability is an important consideration when aligning datasets for model training and testing.

The dataset is split chronologically, with 75% used for training and 25% for testing. This provides sufficient data for effective model training and hyperparameter optimization while ensuring that the testing phase evaluates the model on more recent, unseen data. Although this split is useful for evaluation, the model is later retrained on the full dataset up to the year 2021 for forecasting for 2022 and up to the year 2023 for forecasting for 2024 to ensure the inclusion of the most current market information.

After generating daily forecasts over the years 2022 and 2024, the results are grouped into weekly and monthly intervals, with risk metrics computed for each to assess asset performance. These metrics enable structured evaluation across temporal resolutions and provide a quantitative basis for portfolio optimization.

3.5 Task Type: Regression-Based Time Series Forecasting

Regression

A regression task is a supervised learning problem aimed at predicting a continuous outcome (dependent variable) from one or more input features (independent variables). These methods seek to formally characterize the link between inputs or independent variables and outputs, usually through parametric equations where the parameters are derived from the data. Regression is used to fit a function to the available data in order to forecast outcomes for future or holdout data points.

A regression model can be formalized as the following equation[55]:

$$y = f(X, \theta) + \varepsilon, \tag{1}$$

where
- y, X: dependent and independent variables
- θ, ε: parameter and the error term
- $f(X, \theta)$ is referred to as the regression function

Time Series Forecasting

While standard regression tasks often assume that observations are independent and identically distributed (i.i.d.), many real-world problems involve data collected sequentially over time. Time series forecasting is a specialized form of regression that explicitly models data with a temporal or chronological ordering.

In this research, by merging the strengths of regression with considerations of time series forecasting, robust models are developed that account for both the relationships among features and the sequential patterns present in time-dependent data. This stratergy enables more accurate predictions and deeper insights into how a model evolves over time.

3.6 Market Selection of the Years 2022 and 2024 for Portfolio Backtesting

Market Analysis for the Year 2022

1. **Transition to a High-Inflation Regime**

   According to the IMF's October 2022 Global Financial Stability Report:

   - The global economy experienced stubbornly high inflation, not seen in decades.
   - This marked a shift from a prolonged low-interest, low-volatility environment to one with rapid monetary tightening across advanced and emerging markets.
   - Central banks aggressively increased rates to combat inflation, causing tighter financial conditions worldwide [26].

2. **Geopolitical & Economic Shocks:**

   - The April 2022 IMF report highlights the Russia–Ukraine war as a critical event triggering supply shocks, rising commodity prices, and financial market volatility[25].
   - Financial markets, particularly in emerging economies, were severely affected by capital outflows, widening spreads, and distress in sovereign debt.

3. **Market Anticipation of Monetary Policy in 2022:**

   - In 2022, Dimensional highlighted that the market often anticipated the interest rate increases from the U.S. Federal Reserve, with prices adjusting before official announcements [28].
   - For instance, in June, the market had already priced in a rate increase days before it was formally declared, illustrating the market's forward-looking nature.

   Significance: 2022 represents the starting point of a new macro-financial regime characterized by high uncertainty, geopolitical tension, and inflation-driven policy normalization, all crucial to understanding structural shifts in global finance.

Market Analysis for the Year 2024

1. **Easing Financial Conditions Amid Vulnerabilities:**

   As per the IMF report made in October 2024 [27]:

   - While inflation had begun to moderate and monetary easing was underway in advanced economies, financial conditions remained accommodative.
   - However, this easing masked deeper systemic issues, rising global debt, non-bank leverage, and mispricing of risk.

2. **AI and Structural Change:** [27]

   - The 2024 report also uniquely focuses on AI adoption in financial markets, indicating a technological inflection point that could shape systemic risk and investment behavior in the future.

3. **International Market Overview:**

   As per reports of the Dimensional Fund Advisor, it states that[29]

   - The MSCI World USA Index posted an 11.7 percent gain, with Germany and the UK outperforming, returning 17.6 percent and 14.7 percent, respectively.

   Significance: 2024 stands as a crucial follow-up period where the consequences of 2022's shocks began to unfold, new risks emerged, and financial systems appeared stable but increasingly fragile beneath the surface.

4 Model Development and Evaluation Framework

This chapter outlines the model development and refinement process, starting with the training strategy. It covers the underlying algorithms, explaining the computational techniques and architectures used to capture complex data patterns. This chapter also details model-specific hyperparameters, their impact on performance, and the optimization methods employed to identify optimal configurations systematically. Finally, it discusses the evaluation metrics used to assess model performance and forecasting procedure.

4.1 Model Training Method

Two model training approaches can be considered. The first involves training a separate instance of each model for every ETF, enabling the capture of asset-specific dynamics. The second approach trains a single instance of each model using combined data from all ETFs to capture broader market patterns. However, since not all ETFs have sufficiently large historical datasets to support effective generalization, the first approach is adopted to maintain focused learning on each ETF's individual characteristics.

4.2 Machine Learning Training Algorithms

This study forecasts *log_returns* using five machine learning models as a foundation for portfolio optimization: a baseline OLS, regularized LASSO, ensemble methods (Random Forest and XGBoost), and an LSTM neural network to capture temporal dependencies.

Note: In this work, the input variables $\mathbf{X}$, target variable $\mathbf{y}$, and predicted outputs $\hat{\mathbf{y}}$ are scaled denoted as $\mathbf{X}'$, $\mathbf{y}'$, and $\hat{\mathbf{y}}'$ as in **sub-chapters 3.3.2**. However, to maintain notational simplicity, the prime notation is omitted throughout. This notational convention remains in effect until **sub-chapters 4.4**, where the target and predicted values are rescaled.

4.2.1 Ordinary Least Squares (OLS)

Ordinary Least Squares regression (OLS) is an established approach for determining the coefficients of linear regression equations that specify the relationship between one or more independent quantitative variables and a dependent variable. The OLS approach seeks to minimize the sum of squared deviations between actual and predicted values while addressing potential multicollinearity issues.

$$\ell(y, X^\top \beta) = \min_{\beta} \sum_{i=1}^{n} \left(y_i - X_i^\top \beta\right)^2, \quad \text{where} \quad \hat{y}_i = X_i^\top \beta \tag{20}$$

where:

- ℓ is a loss function
- y_i is the actual value of the dependent variable for the i-th observation.
- $X_i^\top \beta$ is the predicted value by the linear model, where X_i is the feature vector for the i-th observation.
- β are the model coefficients (weights for each feature/independent variable).

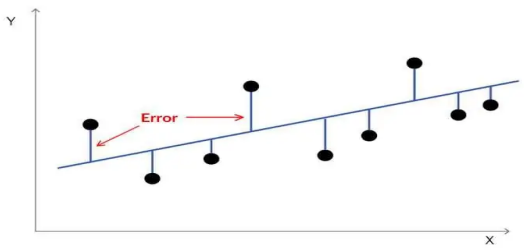

Figure 6: Ordinary Least Squares Objetive[38]

Algorithm 1 shows the working principle of the OLS model along with the hyperparameters considered.

Hyper-params for OLS:

cov_type	designates type of heteroskedasticity-consistent standard errors to be calculated. **HC0** assumes heteroskedasticity without modification; **HC1** adds a degrees-of-freedom correction akin to White's robust standard errors.
use_t	Decide whether inferential statistics (i.e. p-values and confidence intervals) rely on the $t_{\,n-k}$ distribution or the standard normal.
method	Least-squares estimate of the regression coefficients, defined by $\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|_2^2$. It is computed numerically via the chosen **Pinv** or **QR** routine below. ∗ **Pinv:** Uses the Moore–Penrose pseudo-inverse—handy when X is nearly singular. $\hat{\beta} = X^{+}y, \qquad X^{+} = (X^{\top}X)^{-1}X^{\top}.$ ∗ **QR:** Employs the numerically stable QR decomposition. $\hat{\beta} = R^{-1}Q^{\top}y, \qquad X = QR, \qquad Q^{\top}Q = I$ (Q: orthogonal matrix), $\qquad R:$ upper triangular matrix.

Algorithm 1 OLS Regression with Hyperparameter Options

Input: Training data $\{(X_i, y_i)\}_{i=1}^{n}$ with $X \in \mathbb{R}^{n\times p}$, $y \in \mathbb{R}^n$
Output: Estimated coefficients $\hat{\beta}$; optionally, $\mathrm{Cov}(\hat{\beta})$ and t-statistics if `use_t` = True
Step 1: Estimate Coefficients
if `method` = 'pinv' **then**
    $\hat{\beta} \leftarrow X^{+}y$ ▷ Pseudo-inverse
else ▷ `method` = 'qr'
    Factorize $X = QR$; $\hat{\beta} \leftarrow R^{-1}Q^{\top}y$
end if
Step 2: Residuals and Error Variance
$r \leftarrow y - X\hat{\beta}; \quad \hat{\sigma}^2 \leftarrow \dfrac{r^{\top}r}{n-p}$
Step 3: Covariance Matrix
$A \leftarrow (X^{\top}X)^{-1}X^{\top}\,\mathrm{diag}(r^2)X(X^{\top}X)^{-1}$
if `cov_type` = 'HC0' **then**
    $\mathrm{Cov}(\hat{\beta}) \leftarrow A$
else ▷ `cov_type` = 'HC1'
    $\mathrm{Cov}(\hat{\beta}) \leftarrow \dfrac{n}{n-p}A$
end if
Step 4: (Optional) t-Statistics
if `use_t` = True **then**
    for $i = 1$ **to** p **do**
        $se_i \leftarrow \sqrt{[\mathrm{Cov}(\hat{\beta})]_{ii}}; \quad t_i \leftarrow \dfrac{\hat{\beta}_i}{se_i}$ ▷ se_i: Standard error for the i-th co-efficient estimate
    end for
end if
return $\hat{\beta}$, $\mathrm{Cov}(\hat{\beta})$ (if computed), t-statistics (if requested)

Search space for OLS

```
param_space = {'cov_type': Categorical(['HC0', 'HC1']), 'use_t': Categorical([
    True, False]), 'method': Categorical(['pinv', 'qr'])}
```

4.2.2 Least Absolute Shrinkage and Selection Operator (LASSO)

The Least Absolute Shrinkage and Selection Operator (LASSO), or L1 regularization, is a popular method in statistical modeling and machine learning for predicting outcomes and selecting relevant variables. It applies shrinkage by penalizing the absolute size of coefficients, often driving some to zero, thus enabling both regularization and feature selection. Some coefficients are driven to exactly zero when α is sufficiently large. More coefficients are pushed towards zero as the strength of regularization increases with a larger α value. Conversely, if α is small, the regularization impact is reduced, which means that more variables may retain nonzero coefficients. The shape constraint of LASSO in Figure 7 indicates that the contours intersect the square, sometimes at a vertex, corresponding to a coefficient being exactly zero.

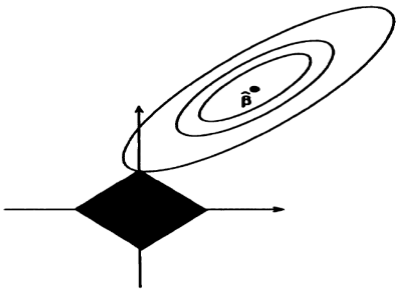

Figure 7: LASSO shape constraint [53]

LASSO Regression Objective Function
The LASSO regression minimizes the following objective function:

$$\ell(y, X^\top \beta) = \min_{\beta_0, \beta} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - X_i^\top \beta\right)^2 + \alpha \|\beta\|_1, \quad \text{where} \quad \hat{y}_i = \beta_0 + X_i^\top \beta \tag{21}$$

where:
- y_i: The response variable for the i-th observation.
- $X_i^\top \beta$ is the predicted value by the linear model, where X_i is the feature vector for the i-th observation.
- β_0 is the intercept term, and $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is the coefficients vector of the p predictors.
- n: The total number of observations.
- α: The regularization parameter that controls the strength of the L1 penalty.

Hyper-params for LASSO

alpha	Controls the strength of the L_1 penalty, shrinking large coefficients to encourage sparsity.
fit_intercept	Boolean flag deciding whether an intercept term is fitted.
pre-compute	Uses the Gramm matrix to speed up calculations when set. The Gramm matrix collects all pair-wise inner products of the feature vectors and may also be supplied explicitly.
	Gram Matrix [57]: $G = X^\top X$ (22)
max_iterations	Maximum number of iterations the algorithm is allowed to run.
tol	Convergence threshold: smaller values give higher precision but slower convergence; larger values converge faster at the possible cost of accuracy.

Algorithm 2 illustrates the working principles of LASSO with its hyperparameters considered.

Algorithm 2 Lasso Regression with Parameter Space

1: **Input:** Training set $\{(X_i, y_i)\}_{i=1}^{n}$ with $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$ where p is the number of predictors used and n is the total data points.
2: **Output:** Optimal LASSO model with coefficients $\hat{\beta}$ and intercept $\hat{\beta}_0$.
3: **Procedure:**
4: **for** each candidate hyperparameter configuration $\theta = (\alpha, \texttt{fit\_intercept}, \texttt{precompute}, \texttt{max\_iter}, \texttt{tol})$ in `search_space` **do**
5: Fit the LASSO model by solving Eq. 21 using the candidate θ.
6: Evaluate the model's performance using cross-validation.
7: **end for**
8: Select the candidate configuration that yields the best CV performance.
9: **return** The LASSO model with the optimal hyperparameters.

Hyperparameter search space for LASSO

```
param_space = {'alpha': Real(1e-7, 1e-2, prior='log-uniform'),
               'fit_intercept': Categorical([True]),
               'precompute': Categorical([True]),
               'max_iter': Integer(4000, 8000),
               'tol': Real(1e-7, 1e-4, prior='log-uniform')}
```

4.2.3 Random Forest

Random Forest is an ensemble learning system that integrates numerous decision trees, uses averaging to improve predictive accuracy, and controls overfitting to enhance the reliability and precision of predictions. This approach is also known as *Bagging*. As the number of trees in a forest grows larger, the generalization error converges to a limit[6].

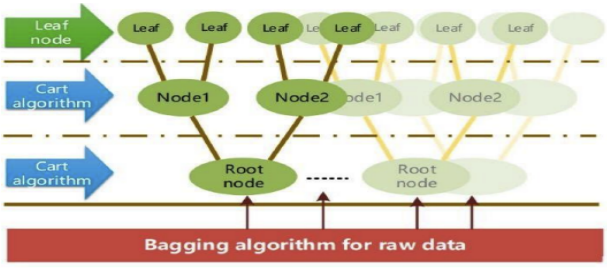

Figure 8: Bagging in random forest [44]

Random Forest Regressor:

Let $X \in \mathbb{R}^{n \times p}$ be the data matrix with n observations and p features, where each row $x_i^\top \in \mathbb{R}^p$ corresponds to the i-th observation. The target vector $y = (y_1, y_2, \ldots, y_n)^\top \in \mathbb{R}^n$.
Ensemble Prediction:

$$\hat{y} = F(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x_i), \quad \text{for } i = 1, \ldots, n, \tag{23}$$

where:

- T is the number of trees (often called as `n_estimators`),
- $f_t(x)$ is the prediction of the t-th regression tree,
- $F(x)$ is the final ensemble model (the average of all trees).

Hyperparameters for Random Forest

n_estimators	Number of trees in the forest. More trees lower variance but raise training time and memory usage.
max_depth	Maximum depth of each tree. Shallow trees train faster and curb overfitting; deeper trees capture more complexity but may overfit.
min_samples_split	Minimum number of samples required to split an internal node, controls tree growth and helps prevent overfitting.
min_samples_leaf	Minimum samples required at a leaf node. Higher values yield simpler trees that generalise better.
max_features	Number of features considered when searching for the best split. Smaller subsets add randomness and reduce overfitting risk.
max_leaf_nodes	Upper limit on the number of leaf nodes. Fewer leaves simplify the model and reduce overfitting.
bootstrap	Boolean flag: build each tree from a bootstrap sample of the data, boosting model diversity and lowering variance.
min_weight_fraction_leaf	Minimum weighted fraction of the total input required at a leaf, prevents very small splits.
ccp_alpha	Complexity parameter for minimal cost-complexity pruning. Prunes branches whose error-reduction is below this threshold thus simplifies the ensemble.

The **algorithm 3** describes the construction of a random forest. The procedure `GrowTree` outlines the recursive tree-growing algorithm building T decision trees $T_1, T_2, \ldots, T_{\texttt{n\_estimators}}$. Each tree is trained on a bootstrap sample of the data, and at each split, a random subset of predictors is considered to find the best split based on a criterion. The final model aggregates all T trees to form an ensemble for predictions, enhancing accuracy and reducing overfitting. The objective function considered is `squared_error`, which measures the difference between the true target y and the prediction $F(x)$. It is defined as

$$\ell(y, F(x)) = \frac{1}{2}(y - F(x))^2 \quad \text{(squared error loss)} \tag{24}$$

Algorithm 3 Random Forest Algorithm [6] [18]

Input: Training set $\{(X_i, y_i)\}_{i=1}^{n}$ with $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$ where p is the number of predictors used and n is the total data points.
Output: $F(x)$, the Random Forest ensemble model with its hyper-parameters.
for $t = 1, 2, \ldots, n\_estimators$ **do**
    if bootstrap = True **then**
        Draw a bootstrap sample D_t from the training set.
    else
        Let D_t be the full training set.
    end if
    Grow a decision tree f_t on D_t using GrowTree, with pruning by *ccp_alpha*.
end for
Define the ensemble function:

$$F(x) \;=\; \frac{1}{\texttt{n\_estimators}} \sum_{t=1}^{\texttt{n\_estimators}} f_t(x_i).$$

Output: $F(x)$, the Random Forest ensemble model.

Procedure: GrowTree(Node, depth)

if the number of samples in `Node` $<$ *min_samples_split* **or** $depth = max\_depth$ **or** maximum leaf nodes reached **then**
    Mark `Node` as a leaf.
else
    Randomly select a subset of features F as determined by *max_features*
    Find the best split among features in F that minimizes the total squared error loss with penalty `ccp_alpha`, i.e.,

$$\min_{\text{split}} \left[\sum_{i \in \text{Left}} \ell(y_i, F_{\text{Left}}(x_i)) + \sum_{i \in \text{Right}} \ell(y_i, F_{\text{Right}}(x_i)) \right] + \text{ccp\_alpha} \cdot \#\text{leaves}$$

$\hat{y}_{\text{Left}}$ is the average of target values in the left child node after a split.
$\hat{y}_{\text{Right}}$ is the average of target values in the right child node after a split.

    Partition `Node` into left and right child nodes.
    if either child node has fewer than *min_samples_leaf* samples **or** proportion of total weight $<$ *min_weight_fraction_leaf* **then**
        Mark `Node` as a leaf.
    else
        GrowTree(LeftChild, depth+1)
        GrowTree(RightChild, depth+1)
    end if
end if

Note: GrowTree is a modular breakdown of the tree-growing logic used within the Random Forest algorithm. This is based on decision tree construction as mentioned in **sub-chapter 2.2**. It is not a standalone program, and variables like Node and depth are local to this procedure.

Hyperparameter search spach for Random Forest

```
param_space = {
        'n_estimators': Integer(200, 500), 'max_depth': Integer(15, 22),
        'min_samples_split': Integer(2, 5), 'min_samples_leaf': Integer(1, 3),
        'max_features': Real(0.85, 1.0), 'max_leaf_nodes': Integer(250, 300),
        'bootstrap': [True], 'min_weight_fraction_leaf': Real(0.0, 0.3),
        'ccp_alpha': Real(1e-8, 1e-5, prior='log-uniform')
    }
```

- `random_state=42`: Ensures reproducibility.
- `n_jobs=-1`: Enables parallel processing by utilizing all available CPU cores for faster computation.

4.2.4 Boosted Trees

Boosted trees represent an ensemble learning method in which numerous decision trees are built in succession. In contrast to random forest, which constructs trees separately, boosting sequentially trains trees, with each consecutive tree rectifying the errors of its predecessor. This approach seeks to mitigate model bias and enhance performance, especially for difficult datasets. In boosting, each tree concentrates on the residual errors generated by the preceding trees. This procedure persists for a predetermined number of trees or until the error is sufficiently reduced. XGBoost is widely recognized for speed and efficiency, incorporating several advanced features, such as regularization, missing data management, and parallelization.

Figure 9 illustrates the XGBoost model architecture, where predictions are made by sequentially adding outputs from multiple gradient-boosted decision trees.

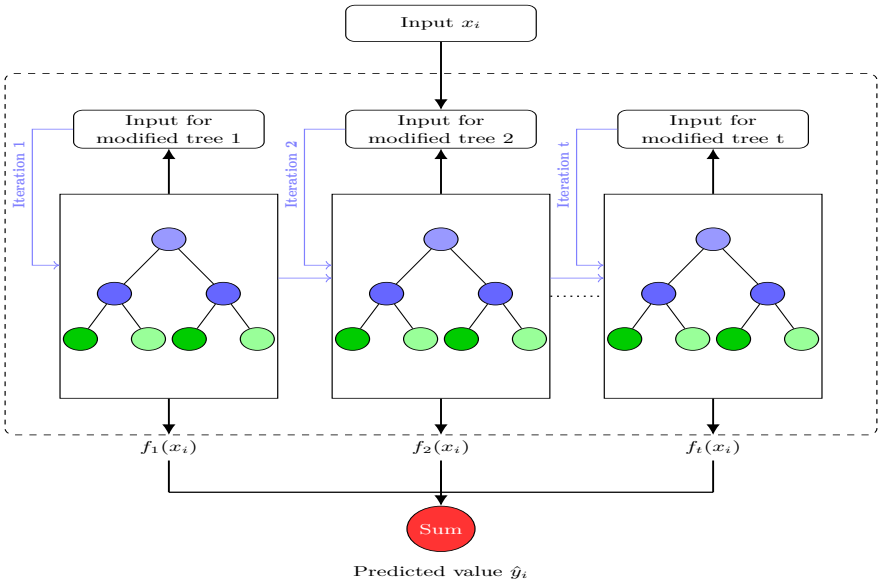

Figure 9: XGBoost Model Architecture [59]

XGBoost Regressor:

Let $X \in \mathbb{R}^{n \times p}$ be the data matrix with n observations and p features, where each row $x_i^\top \in \mathbb{R}^p$ corresponds to the i-th observation. The response vector is denoted by $y = (y_1, y_2, \ldots, y_n)^\top \in \mathbb{R}^n$.

Ensemble Prediction: [11]

$$\hat{y} = F(\mathbf{x}) \;=\; \sum_{t=1}^{T} f_t(x_i), \quad \text{for } i = 1, \ldots, n, \tag{25}$$

where T is the total number of trees, and each f_t is a regression tree.

Objective Function: [11]

$$\mathcal{O} \;=\; \sum_{i=1}^{n} \ell\big(y_i, F(x_i)\big) \;+\; \sum_{t=1}^{T} \Omega\big(f_t\big), \tag{26}$$

where:

- $\ell(\cdot)$ is a differentiable convex loss function (squared error for regression) as mentioned previously in Eq. 24. For each observation i:

$$\text{Gradient:} \quad g_i = \frac{\partial \ell\big(y_i, F(x_i)\big)}{\partial F(x_i)} \qquad \text{Hessian:} \quad h_i = \frac{\partial^2 \ell\big(y_i, F(x_i)\big)}{\partial \big(F(x_i)\big)^2}$$

- $\Omega(\cdot)$ is the regularization term that penalizes model complexity.

Regularization Term with leaf penalty γ, α (L1) and λ (L2): [11]

$$\Omega(f_t) \;=\; \gamma\, T_{f_t} \;+\; \frac{1}{2}\lambda \sum_{j=1}^{T_{f_t}} w_j^2 \;+\; \alpha \sum_{j=1}^{T_{f_t}} |w_j|,$$

where:

- T_{f_t} is the number of leaves in the t-th regression tree f_t,
- w_j is the weight (score) for the j-th leaf and γ is a complexity penalty for each leaf,
- λ, α: L_2 and L_1 regularization terms on leaf weights,

By combining the loss function $l(\cdot)$ with the regularization term $\Omega(\cdot)$, XGBoost balances model fit and complexity, reducing overfitting and enhancing generalization. **Algorithm 4** illustrates this process.

Hyperparameters for Boosted Trees

n_estimators	Total number of boosted trees. Training time and memory grow roughly linearly with this count.
max_depth	Maximum depth of each tree. Deeper trees capture complex patterns but risk overfitting; shallow trees train faster and are more robust.
learning_rate	Shrinks the contribution of each new tree Smaller values improve generalisation but need more trees, while larger values speed training but may miss the optimum.
sub_sample	Fraction of training instances sampled for each tree. Lower values inject randomness and curb overfitting; higher values use more data per tree.
colsample_bytree	Fraction of features sampled for each tree. Lower values add randomness; higher values improve fit but increase complexity.
colsample_bylevel	Fraction of features sampled at each tree level, adding diversitycross levels and aiding gegeneralization
colsample_bynode	Fraction of features sampled at each split (node). Prevents the model from relying on a few dominant features and improves robustness.
reg_alpha	L_1 regularisation on leaf weights. Larger values encourage sparsity and reduce overfitting; smaller values allow more flexibility.
reg_lambda	L_2 regularisation on leaf weights. Penalises large weights to control overfitting.
gamma	Minimum loss reduction required to make a split. Higher values limit splits and simplify the model; lower values allow more splits and complexity.
min_child_weight	Minimum sum of instance weights (Hessian) needed in a child node. Larger values restrict small splits and reduce overfitting; smaller values allow more detailed trees.

Algorithm 4 XGBoost Algorithm [11]

Input: Training set $\{(X_i, y_i)\}_{i=1}^{n}$ with $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$ where p is the number of predictors used and n is the total data points.
Parameters: Hyperparameters listed above for the XGBoost
Initialize: $F^{(0)}(x) \leftarrow \overline{y}$ (the mean of $\{y_i\}$)
for $t = 1 \rightarrow$ `n_estimators` **do**
  for $i = 1 \rightarrow n$ **do**
    $g_i^{(t)} \leftarrow \dfrac{\partial \ell\big(y_i, F^{(t-1)}(x_i)\big)}{\partial F^{(t-1)}(x_i)}$
    $h_i^{(t)} \leftarrow \dfrac{\partial^2 \ell\big(y_i, F^{(t-1)}(x_i)\big)}{\partial \big(F^{(t-1)}(x_i)\big)^2}$
  end for
  Fit a regression tree f_t using procedure `FitTree`($\{(x_i, g_i^{(t)}, h_i^{(t)})\}$ hyperparameters).
  Update model:

$$F^{(t)}(x) \leftarrow F^{(t-1)}(x) + \texttt{learning\_rate} \times f_t(x).$$

end for
$F_{\mathrm{XGB}}(x) \leftarrow F^{(n\_estimators)}(x)$
Output: $F_{\mathrm{XGB}}(x)$, the final boosted model.

Procedure: FitTree ($\{(x_i, g_i, h_i)\}$, Hyperparameters of XGBoost)

Input: Data $\{(x_i, g_i, h_i)\}$ (optionally subsampled), where $i \in I$, and I is the set of data points at the current node; and the parameters.
Feature Sampling: Sample features based on: `colsample_bytree` for the entire tree, `colsample_bylevel` at each level of the tree, `colsample_bynode` at each split node.
if stopping criteria met (i.e., *depth* = `max_depth` or insufficient gain or $\sum_i h_i <$ `min_child_weight`) **then**
    Mark node as a leaf and assign the optimal weight using soft-thresholding:

$$w^* = -\frac{\operatorname{sgn}\left(\sum_{i\in I} g_i\right) \cdot \max\left(\left|\sum_{i\in I} g_i\right| - \alpha,\, 0\right)}{\sum_{i\in I} h_i + \lambda}$$

    Limit the weight change using `max_delta_step`:

$$w^* = \operatorname{clip}\left(w^*, -\texttt{max\_delta\_step}, \texttt{max\_delta\_step}\right)$$

else
    Evaluate potential splits on the sampled features using:

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

    Choose the split that maximizes Gain.
    Partition the node into left child I_L and right child I_R.
    Recursively call FitTree on each child with $depth + 1$.
end if

Notation and Definitions:

- **I**: The set of indices corresponding to the data points in the current node.
  - I_L, I_R: The subset of indices for data points assigned to the left child node and right child node.

Procedure `FitTree` defines how XGBoost grows trees by recursively selecting splits that maximize regularized gain, assigning optimal leaf weights, and applying constraints like subsampling, column sampling, and step limits.

Python Implementation for the XGBoostRegressor

```

param_space = {
        'n_estimators': Integer(600, 700), 'max_depth': Integer(3, 6),
        'learning_rate': Real(0.01, 0.1, prior='log-uniform'),
        'subsample': Real(0.5, 0.8), 'colsample_bytree': Real(0.6, 0.8),
        'colsample_bylevel': Real(0.6, 0.8),
        'colsample_bynode': Real(0.6, 0.8),
        'gamma': Real(0.01, 5.0, prior='log-uniform'),
        'reg_alpha': Real(0.00001, 0.1, prior='log-uniform'),
        'reg_lambda': Real(0.01, 10.0, prior='log-uniform'),
        'min_child_weight': Integer(10, 50), 'max_delta_step': Integer(1, 5),
    }
```

4.2.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of RNN designed to overcome the limitations of traditional RNNs, particularly the vanishing and exploding gradient problems that occur during training. These issues make it challenging for standard RNNs to learn and retain dependencies over long sequences. LSTMs address this by introducing a more sophisticated architecture. This unique capability makes LSTMs highly effective in comprehending and predicting patterns in sequential data types such as time series, text, and speech[23].

LSTM model architecture

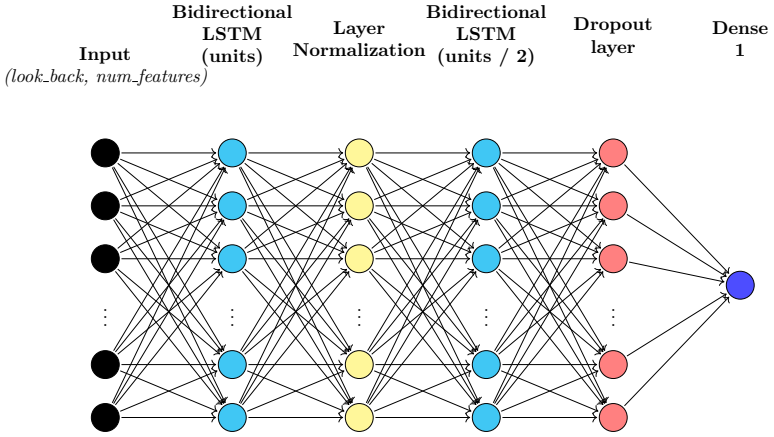

Figure 10: LSTM model architecture [35].

The architecture shown in Figure 10 is inspired by the work of Gustavo Martins et al.[35] and the layer structure is modified accordingly to this research.

1. **Input layer**

   - This layer serves as a placeholder to specify the shape of the data entering the model. It defines the structure of the model's input tensor with `look_back` and `num_features` that denote the number of variables or features at each time step.

2. **First Bidirectional-LSTM layer**

   - This layer processes input sequences bidirectionally, capturing dependencies from both past and future time steps with `units` neurons and the `ReLU` activation.
   - Bidirectional traversal enables the model to access both previous and upcoming context, yielding richer hidden representations for learning complex temporal dependencies [48].
   - It consists of two LSTM layers where one processes the sequence from start to end (forward) while the other processes it from end to start (backward) [21]. At each timestep t outputs of both are combined. They both have independent parameters but share an input vector list. The output of this layer will be a modified version of the LSTM output as mentioned in Eq. 14

   $$z_t = \overrightarrow{z}_t \oplus \overleftarrow{z}_t \tag{27}$$

   where $\overrightarrow{z}\_t$ and $\overleftarrow{z}\_t$ are the forward and backward LSTM outputs at time t, respectively, and $\oplus$ denotes concatenation.
   - Since sequences need to be preserved for the further layers, sequences are preserved in the model setting by making `return_sequences=True`.

3. **LayerNormalization layer:**

   - Layer Normalization is applied after the first BiLSTM layer, where it normalizes the activations across the feature (hidden) dimension for each individual time step of each sample. It operates on the intermediate layer outputs and includes learnable parameters for adaptive rescaling.
   - This is a batch-independent operation as it computes statistics across features within each sample, so it works consistently for any batch size and fits well for sequential models.
   - Its computes mean and variance across all features for a single sample

   $$\mu = \frac{1}{\mathtt{d}} \sum_{j=1}^{\mathtt{d}} z_j, \quad \sigma^2 = \frac{1}{\mathtt{d}} \sum_{j=1}^{\mathtt{d}} (z_j - \mu)^2$$

   where z_j is output (activation) from the previous layer for feature j, d is number of features (hidden size)
   - Then normalize the input

   $$z'_j = \frac{z_j - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

   - Finally, scale and shift using learnable parameters γ and β respectively

   $$\tilde{z}_j = \gamma_j z'_j + \beta_j, \quad \text{where } \tilde{z}_j \text{ is the output passed to the next layer}$$

4. **Second Bi-directional Layer with Reduced Units**

   - This layer follows with half of the neurons, reducing the model's complexity while still capturing temporal patterns from the previous layer's output.
   - This step-down in neuron count helps prevent overfitting and ensures the model generalizes well to unseen data.
   - Unlike first Bi-LSTM layer, here sequences are not returned to following layers as sucessive layers are Dropout and Dense layers, which requires only final timestep's output vector z_t.

5. **Dropout layer**

   - A dropout network tries to combine an exponentially large number of different neural network designs by randomly turning off a subset of hidden units during each training iteration. It makes sure that all of these subnetworks have the same weights [41].
   - This stochastic omission aids in the prevention of overfitting by compelling the network to acquire more resilient and generalizable properties that are not dependent on specific neurons.

6. **Fully connected Dense layer**

   - The final layer of the model is a `dense` (fully connected) output layer, containing a single neuron (`Dense(1)`).
   - Since this is a regression problem, the output layer does not use an activation function, meaning it performs a linear transformation of the last LSTM output.

Model Configurations

Model Configurations outlines the key components and hyperparameters used to design and train the LSTM model, including the lookback(timestep), activation functions, loss function, and optimization strategy.

Activation Function

The fundamental purpose of an activation function is to incorporate nonlinearity into the neural network, allowing it to learn intricate patterns and generate predictions that exceed mere linear combinations. In the absence of nonlinearity, the network would merely function as a linear model, constrained in its capacity to address intricate issues. Every node in a neural network obtains input from other nodes, which are assigned weights. The activation function subsequently processes the aggregate of these weighted inputs and converts them into an output.

Rectified Linear Unit (ReLU): In this research, the activation function used in this LSTM architecture is ReLU. Compared to traditional activation functions like sigmoid or tanh, ReLU avoids the vanishing gradient problem, ensuring better gradient flow during backpropagation, which significantly accelerates training. Additionally, ReLU is computationally efficient as it requires only a simple thresholding operation at zero. Another advantage is that it promotes sparsity in neural networks, as some neurons output zero, helping to reduce overfitting and improve model generalization[2]. Equation 28 represents the ReLU function in its piecewise form for its inputs.

$$ReLU(x) = \quad max(0, x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{28}$$

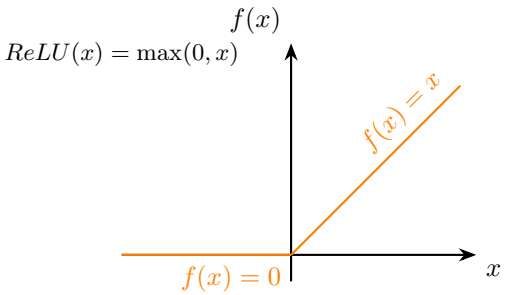

Figure 11: Plot of ReLU function for its input[43].

ReLU Working steps: If the input x is positive ($x > 0$), the output is the same as the input. If the input x is zero or negative ($x \leq 0$), the output is zero. When employing the ReLU activation function, values that are exceedingly insignificant will not be obtained.

Loss Function

A loss function is a mathematical function that quantifies the difference between the predicted output of a model and the actual values. It guides the training of models by penalizing inaccurate predictions and helps optimize the model's parameters. Mean Squared Error is used as a loss function here.

Mean Squared Error (MSE): MSE is a commonly used loss function for regression problems. It measures the average of the squares of the errors as illustrated in Eq. 24. The squaring gives more weight to larger errors.

Optimizer

Optimizers aim to find the set of model parameters that result in the lowest possible loss (or error) on the training data. The following steps show the general working flow of an optimizer.

1. **Forward Pass:** The model makes predictions on the training data based on its current parameters.
2. **Loss Calculation:** The loss function evaluates the difference between the model's predictions and the actual target values.

3. **Backpropagation:** The gradient of the loss function with respect to the model's parameters is calculated using backpropagation.

4. **Parameter Update:** The optimizer uses the calculated gradient and the learning rate to update the model's parameters.

5. **Iteration:** This process is repeated iteratively until the loss function reaches a minimum or a specified number of iterations is reached.

Adaptive Moment Estimation (ADAM) Optimizer: This research uses the ADAM optimizer, which is known for its adaptability and efficiency. By combining momentum and RMSProp, ADAM adapts learning rates per parameter, handles noisy and sparse data well, and improves convergence stability with minimal hyperparameter tuning, making it superior to standard SGD. **Algorithm 5** illustrates the working principle of the ADAM optimizer tp update weights and biases.

Algorithm 5 Adam Optimizer for Updating Weights and Biases [30]

1: **Input:** Initial weights w_0 and b_0, learning rate lr; decay rates $\rho_1, \rho_2 \in [0, 1)$; small constant ϵ;.

2: **Initialize:**

$$m_{w,0} \leftarrow 0, \quad v_{w,0} \leftarrow 0, \quad m_{b,0} \leftarrow 0, \quad v_{b,0} \leftarrow 0.$$

3: **for** $t = 1$ **to** T iterations **do**

4: **Compute Gradients (First-Order Derivatives):**

$$g_{w,t} \leftarrow \nabla_w \, \ell(w_{t-1}, b_{t-1}), \quad g_{b,t} \leftarrow \nabla_b \, \ell(w_{t-1}, b_{t-1})$$

5: **Update Moment Estimates:**

- First order moments: (mean of the gradients)

$$m_{w,t} \leftarrow \rho_1 \, m_{w,t-1} + (1 - \rho_1) \, g_{w,t}, \quad m_{b,t} \leftarrow \rho_1 \, m_{b,t-1} + (1 - \rho_1) \, g_{b,t}$$

- Second order moments: (uncentered variance of the gradients)

$$v_{w,t} \leftarrow \rho_2 \, v_{w,t-1} + (1 - \rho_2) \, g_{w,t}^2, \quad v_{b,t} \leftarrow \rho_2 \, v_{b,t-1} + (1 - \rho_2) \, g_{b,t}^2$$

6: **Compute Unbiased Moment Estimates:**

$$\hat{m}_{w,t} \leftarrow \frac{m_{w,t}}{1 - \rho_1^t}, \qquad \hat{v}_{w,t} \leftarrow \frac{v_{w,t}}{1 - \rho_2^t}$$
$$\hat{m}_{b,t} \leftarrow \frac{m_{b,t}}{1 - \rho_1^t}, \qquad \hat{v}_{b,t} \leftarrow \frac{v_{b,t}}{1 - \rho_2^t}$$

7: **Update Model Parameters:**

$$w_t \leftarrow w_{t-1} - lr \cdot \frac{\hat{m}_{w,t}}{\sqrt{\hat{v}_{w,t}} + \epsilon}, \qquad b_t \leftarrow b_{t-1} - lr \cdot \frac{\hat{m}_{b,t}}{\sqrt{\hat{v}_{b,t}} + \epsilon}$$

8: **end for**

9: **return** (w_T, b_T)

Lookback:

LSTM model also incorporates `lookback` that refers to the number of past time steps used as input to predict future values. It defines the length of the input sequence fed into the LSTM model, helping it learn temporal dependencies in time-series data. In some context it is also referred as timesteps. A larger `lookback` window allows the model to capture long-term trends but may increase computational complexity and risk overfitting, while a smaller lookback window focuses on short-term patterns, which might miss

crucial long-range dependencies. In this research, `lookback` is set to 10, where it can see into temporal sequences of past 10 days $t, t-1, \ldots, t-9$ to predict next sequence t+1. This combination of bidirectional LSTMs, dropout regularization, and optimized training with `Adam` makes this model powerful for sequential data tasks, capable of learning intricate patterns while avoiding overfitting.

Hyper-params for LSTM:

units	Number of memory cells in each LSTM layer. Fewer units simplify the model and speed training but limit pattern capture; more units boost expressiveness at the cost of higher overfitting risk and computation.
dropout	Fraction of neurons randomly deactivated during training to limit overfitting. Lower values mean weaker regularization; higher values provide stronger regularization.
learning_rate	Step size for optimizer weight updates. Lower rates yield smoother, more precise convergence but slower training; higher rates speed training yet may overshoot or fail to converge.

Algorithm 6 explains how the LSTM Regressor model works with the considered hyperparameters and other configurations.

Algorithm 6 LSTMRegressor Training and Prediction Process

Input: Training set $\{(X, y)\}$, where $X \in \mathbb{R}^{n \times \texttt{look\_back} \times \texttt{num\_features}}$, $y \in \mathbb{R}^n$ and hyperparameters: `units`, `dropout`, `learning_rate`, `epochs`, `batch_size`, `look_back`, `num_features`
Output: Trained model $F(x)$
Step 1: Model Construction Build the LSTM architecture as mentioned in Figure 10
Step 2: Model Training
for *epoch* $\leftarrow 1$ to `epochs` **do**
  for $(X_{\text{batch}}, y_{\text{batch}})$ in mini-batches of size `batch_size` **do**
    Forward Pass: Compute predictions $\hat{y} = M(X_{\text{batch}})$
    Loss Computation: Compute loss

$$\ell = \frac{1}{N} \sum \left(y_{\text{batch}} - \hat{y}\right)^2, N = \textit{number of samples in the batch}$$

    Backward Pass: Compute gradients $\nabla \ell$ w.r.t. model parameters (weights and biases).
    Parameter Update: Update model using Adam optimizer and `learning_rate`
  end for
end for
return Trained model $F(x) = M(x)$

Hyperparameter search space for LSTM

```
param_space = {'units': Integer(32, 128), 'dropout': Real(0.3, 0.7),
'learning_rate': Real(0.001, 0.05, prior='log-uniform')}
```

4.3 Hyperparameter optimization methods

Hyperparameter optimization (HPO) is a crucial aspect of developing machine learning models, as it involves adjusting the parameters that govern the learning process. Models can achieve enhanced performance by optimizing these hyperparameters, which is crucial in various domains, including finance and image classification. The selected optimization approach can substantially influence the efficiency and efficacy of the model. Hyper-parameter tuning is crucial in different scenarios, such as when a model with a complicated architecture necessitates the adjustment of numerous hyper-parameters. Also, a meticulously developed model requires each hyper-parameter to be fine-tuned within a precise range to replicate accuracy[58].

In HPO, the optimal configuration for a specific model is determined by systematic exploration of hyperparameter spaces. Numerous deep learning hyperparameters possess unique optimal values for various datasets or contexts. Consequently, it enhances the reproducibility of models and research[12]. The objectives include superior model performance, efficient resource utilization, and dependable generalization to novel data. Each model used in this research has its set of hyperparameters to be tuned for improving its efficiency.

4.3.1 Bayesian Optimization

The Bayesian optimization method is applied to all models to maintain uniformity. It effectively navigates the hyperparameter space, balancing exploration with exploitation. It constructs a probabilistic model of the objective function and selects hyperparameters informed by prior information, rendering it especially advantageous for models with intricate, high-dimensional search spaces. This informed, iterative process allows for efficient tuning of hyperparameters while reducing the computational burden typically associated with exhaustive search methods. Bayesian optimization is a more efficient approach that builds a probabilistic model of the objective function and selects hyperparameters based on an acquisition function. Instead of exhaustively searching, it prioritizes promising regions in the search space.

4.3.2 Visualization of the Bayesian Workflow

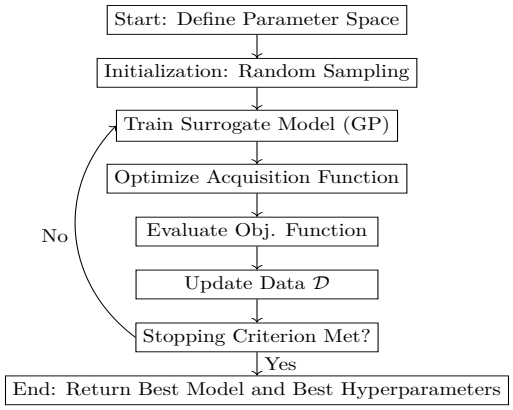

Figure 12: Bayesian Optimization Workflow

```
from skopt import BayesSearchCV
bayes_search = BayesSearchCV(
    estimator=ESTIMATOR, search_spaces=SEARCH_SPACES,
    n_iter=NUM_ITER, cv=CV_FOLDS,
    scoring='SCORE_METRIC', random_state=RANDOM_STATE,
    fit_params={        # (if applicable)
        "early_stopping_rounds": EARLY_STOPPING, "verbose": False
    })
```

- `estimator`: model to train
- `search_spaces`: hyperparameter bounds
- `n_iters`: number of combinations to test
- `cv`: cross-validation folds
- `scoring`: `neg_MSE` for LSTM, `neg_MAE` for others
- `fit_params`: extra fit args, e.g. `early_stopping_rounds` for XGBoost

Note: `BayesianSearchCV` from `skopt` uses Gaussian Process as a surrogate model and `gp_hedge` as an acquisition function as default arguments. `gp_hedge` probabilistically chooses one acquisition function among

Expected Improvement (EI), Probability of Improvement (PI), and Lower Confidence Bound (LCB) using the softmax function.
Algorithm 7 defines the works of the Bayesian optimization [50][17][46].

Algorithm 7 Bayesian Optimization for Hyperparameter Tuning

Inputs and Notation:

$\mathcal{P}$	Hyperparameter search space
$f(p)$	Negative cross-validated score of model with parameters p
T	Total number of iterations
k	Number of initial random evaluations
$\mathcal{D}_t$	Dataset of evaluated points: $\{(p^{(i)}, f(p^{(i)}))\}_{i=1}^{t}$
$\mu_{t-1}(p), \sigma_{t-1}(p)$	GP predictive mean and standard dev. for parameter p
$\Phi(\cdot), \varphi(\cdot)$	Standard normal CDF and PDF
$\Delta_{t-1}(p)$	$f^*_{t-1} - \mu_{t-1}(p)$, where $f^*_{t-1} = \min\{f(p^{(i)})\}_{i=1}^{t-1}$

Initialization:
for $i = 1$ to k **do**
  Sample $p^{(i)} \sim \text{Uniform}(\mathcal{P})$
  Evaluate $f(p^{(i)})$
end for
Set $\mathcal{D}_k = \{(p^{(i)}, f(p^{(i)}))\}_{i=1}^{k}$
Initialize gain vector $g_{\text{EI}} = g_{\text{PI}} = g_{\text{LCB}} = 0$
for $t = k+1$ to T **do**
  Fit Gaussian Process surrogate model using $\mathcal{D}_{t-1}$

$$f(p) \sim \mathcal{GP}\left(\mu_{t-1}(p), \sigma^2_{t-1}(p)\right)$$

  Compute:

$$\text{EI}(p) = \Delta_{t-1}(p) \cdot \Phi\left(\frac{\Delta_{t-1}(p)}{\sigma_{t-1}(p)}\right) + \sigma_{t-1}(p) \cdot \varphi\left(\frac{\Delta_{t-1}(p)}{\sigma_{t-1}(p)}\right);$$

$$\text{PI}(p) = \Phi\left(\frac{\Delta_{t-1}(p)}{\sigma_{t-1}(p)}\right)$$

$$\text{LCB}(p) = \mu_{t-1}(p) - \kappa \cdot \sigma_{t-1}(p); \quad \kappa = \textit{tunable exploitation vs exploration parameter, default value = 1.96}$$

  Each acquisition function proposes a candidate:

$$p^{(t)}_{\text{EI}} = \arg\max_{p \in \mathcal{P}} \text{EI}(p); \qquad p^{(t)}_{\text{PI}} = \arg\max_{p \in \mathcal{P}} \text{PI}(p); \qquad p^{(t)}_{\text{LCB}} = \arg\min_{p \in \mathcal{P}} \text{LCB}(p)$$

  Compute selection probabilities:

$$\pi_i = \frac{\exp(\eta g_i)}{\sum_j \exp(\eta g_j)} \quad \text{where } i \in \{\text{EI, PI, LCB}\} = \texttt{softmax}(\eta * g\_i)$$

  (η is a fixed softmax parameter in ***gp_hedge****, not the model's learning rate, default value is 1.0)*
  Select acquisition function $a^* \sim \text{Categorical}(\pi_{\text{EI}}, \pi_{\text{PI}}, \pi_{\text{LCB}})$
  Select $p^{(t)} = p^{(t)}_{a^*}$
  Evaluate $f(p^{(t)})$, and update:

$$\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(p^{(t)}, f(p^{(t)}))\}$$

  Update gain:

$$g_{a^*} \leftarrow g_{a^*} - \mu_t(p)$$

end for
Return: $p^* = \arg\min_{(p, f(p)) \in \mathcal{D}_T} f(p)$

4.4 Model Evaluation Metrics

Models are evaluated by using three different metrics: Mean Absolute Error (MAE), Root mean Squared Error (RMSE) and Mean Absolute Scaled Error (MASE). All three metrics are widely used to evaluate the performance of regression models. Both measure the difference between actual and predicted values, but do so in different ways, leading to different interpretations and use cases.

Note: The prime notation omission made (in **subchapter 4.2**)for notational simplicity ends here. The target values and forecasted values are rescaled back to their original scale. The prime notation is reinstated to clearly distinguish between scaled and original values. Specifically, the prime notation (e.g., y', $\hat{y}'$) refers to *scaled* values, while unprimed symbols (e.g., y, $\hat{y}$) denote the *original (re-scaled)* values.

4.4.1 Mean Absolute Error (MAE):

MAE is a value that is calculated by comparing the actual values to the projected values. This provides a straightforward measurement of the degree to which the model's predictions depart from actual values, without taking into consideration the direction in which mistakes occur. MAE is less sensitive to outliers and provides a fair depiction of overall prediction accuracy. This is because it handles all errors equivalently. This makes it a viable option for situations in which all deviations, regardless of how large or tiny they are, should be weighted equally. The changes in MAE are linear and, therefore intuitive[45].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{29}$$

where:

- y_i is the actual target value,
- $\hat{y}_i$ is the predicted target value,
- n is the total number of observations.

4.4.2 Root Mean Squared Error (RMSE):

RMSE begins by squaring the differences before averaging them. After that, it takes the square root, which magnifies the impact of more significant errors. RMSE penalizes greater deviations more than the MAE, which makes it beneficial in situations where significant mistakes have severe implications. The existence of outliers and their probability of occurrence is well described by the normal distribution underlying the use of the RMSE[10]. This sensitivity to high mistakes is responsible for this.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{30}$$

4.4.3 Mean Absolute Scaled Error (MASE):

MASE is a scale-independent error metric commonly used in time-series forecasting. It improves upon metrics like MAE by making it comparable across different datasets or naive forecasts on the same dataset.

MASE is computed by using below formula [16]:

$$\text{MASE} = \frac{\sum_{t=1}^{n}|y_t - \hat{y}_t|}{n} \Big/ \frac{\sum_{t=2}^{n}|y_t - y_{t-1}|}{n-1} \tag{31}$$

where

- y_t, y_{t-1} : Actual target value at time t and previous time step
- $\hat{y}_t$: Predicted target value value at time t.
- n : Total number of observations.

Interpretation: MASE < 1 implies the model performs better than the naive model, and MASE > 1 implies it performs worse.

4.5 Forecasting Procedure

Following model evaluation to ensure it has an achieving acceptable error metrics on the validation set, each model is retrained on the full dataset up to the most recent data point using the best hyperparameter configurations. This final retraining step prepares the models for forecasting, as outlined in **sub-chapter 3.4**.

Approaches to Forecasting

Both approaches aim to forecast future points but differ fundamentally in how they model time: one through static features, the other through learned temporal dynamics.

- **Linear and Tree-based models:** Forecasts are generated using lagged features and summary statistics like Simple Moving Averages (SMA) calculated up to the previous day. Here, the input for a given time point t includes data from $t-1, t-2, \ldots, t-n$, and the model is trained to predict the value at t. This is commonly referred to as *point forecasting* or *direct forecasting*, where each prediction is made using a fixed-size feature vector based on historical data.
- **LSTM model:** In contrast, Long Short-Term Memory (LSTM) models operate on sequences rather than individual feature vectors. Instead of hand-crafting lagged features, LSTMs take a window of past time steps (*lookback*) say from time t to $t + \text{look\_back} - 1$ and learn to predict the next value at $t + \text{look\_back}$. This approach is called *sequence-to-one forecasting* and is well-suited for capturing complex temporal dependencies in the data. The input to the model is a 3D array shaped as $(\text{samples}, \text{time\_steps}, \text{features})$, allowing the LSTM to internally learn how patterns evolve, rather than relying on explicitly defined lags. Though less interpreted, LSTM models can outperform traditional approaches when nonlinear or long-range dependencies exist in the data.

Preprocessing steps performed while forecasting

- **For Linear and Tree-based models:**

  For the initial forecast, all necessary input features are derived from historical data. When forecasting for the years 2024 and 2022, the historical data is used up to December 31, 2024, and December 31, 2021, respectively. The maximum number of lags and SMA (Simple Moving Average) windows are computed to match those used during model training, ensuring consistency in feature generation. The next step involves applying polynomial feature expansion and SVD transformation. The same transformation parameters fitted during training are applied here, maintaining consistent interaction features and dimensionality reduction of the same order.

  All input features are then scaled using the same scalers fitted during training. The model predicts the *log_returns*, which is subsequently re-scaled using the inverse transform of the target scaler used during training. The forecasted log return is stored in two separate `dataframes`; one where it is appended as the next point in the historical dataset, and another that holds only the forecasted values. After each prediction, SMA features are recalculated dynamically for the newly forecasted day, which are then used for forecasting the following day. This process continues recursively until the target end date, December 31, 2024, or December 31, 2022.

- **For LSTM:**

  The LSTM model uses the same approach as other models, but without feature expansion and dimension reduction, as they weren't used in training. Features are maintained consistently as per training. The same scalers are used for transformation and reverse transformation at the necessary steps. The only additional part is that the input is in 3D shape, while scaling it is converted to 2D scaling, and before predicting, input features are converted back to 3D again. Lookback sequences and other input features are computed dynamically as the forecasting continues and used for the following step.

5 Comprehensive Framework for Risk Analysis and Score-Based Weight Optimization

This chapter outlines a structured approach for evaluating risk and allocating portfolio weights. It covers key risk metrics (Rachev, Sharpe, Sortino), analyzes volatility clustering, computes composite scores, and optimizes weights using SQP. A smoothing process ensures stable final allocations. **Note:** The notation r_b consistently serves as a benchmark criterion across all risk metrics but its interpretation varies.

5.1 Quantitative Risk Measures

All quantitative risk measures are applied to grouped daily forecasted log return values over monthly and weekly horizons.

- Let $\hat{y}_t$ denote the forecasted log return for a single business day indexed by $t \in \mathcal{T}$, where $\mathcal{T}$ is the set of all forecasted daily time steps.
- Let $\mathbb{H} = \{1, \ldots, H\}$ denote the set of forecast horizons, with $H = 12$ (monthly) or $H = 52$ (weekly).
- Each horizon index $h \in \mathbb{H}$ corresponds to a subset of business days:

$$\mathcal{T}_h \subset \mathcal{T}, \quad \text{with } \mathcal{T}_h = \{t_1, t_2, \ldots, t_{n_h}\}; \qquad n_h \text{ is the total days in horizon } h$$

- The grouped forecasts within horizon period h for ETF p are denoted by:

$$\hat{\mathbf{r}}_{p,h} = \{\hat{y}_t\}_{t \in \mathcal{T}_h}$$

- The complete collection of daily forecasts grouped over all horizon periods is:

$$\hat{\mathbf{r}}_{p,\mathbb{H}} = \{\hat{\mathbf{r}}_{p,h}\}_{h \in \mathbb{H}} = \{\{\hat{y}_t\}_{t \in \mathcal{T}_h}\}_{h=1}^{H}$$

5.1.1 Rachev Ratio:

The Rachev Ratio (Tail Risk Ratio) is a risk-adjusted performance metric that focuses on the asymmetry of return distributions. It is particularly useful for evaluating financial assets with non-normal distributions, emphasizing the downside risk relative to the upside potential.

The Rachev Ratio is defined as the ratio of *Expected Tail Gain (ETG)*, the average gain in the best $\epsilon_1\%$ of outcomes (i.e., the upper tail beyond the $(1-\epsilon_1)$-quantile) to the *Expected Tail Loss (ETL)*, the average loss in the worst $\epsilon_2\%$ of outcomes (i.e., the lower tail below the ϵ_2-quantile). The Rachev ratio for ETF p at horizon h is calculated as [42, pp. 332–333] :

$$\mathrm{RaR}_{\epsilon_1,\epsilon_2,h,p} = \frac{\mathrm{AVaR}_{\epsilon_1}(r_b - \hat{\mathbf{r}}_{p,h})}{\mathrm{AVaR}_{\epsilon_2}(\hat{\mathbf{r}}_{p,h} - r_b)} \tag{32}$$

where:
- $\hat{\mathbf{r}}_{p,h}$, r_b are list of returns of the ETF at horizon h being assessed and returns of the benchmark portfolio, respectively.
- ϵ_1, ϵ_2 are tail probabilities for the representing gains and losses, respectively.
- AVaR(Average Value at Risk), which measures the expected shortfall beyond a specified quantile level.

Experimental settings for Rachev Ratio

In this research, the following important considerations are made to the formula for easier interpretability:

1. **Exclusion of a Benchmark Portfolio ($r_b = 0$):**
   - It simplifies interpretation as it measures *absolute tail risk* rather than relative performance, as it directly conveys the *expected gain per unit of expected loss* in the tails of the return distribution, making it easier to interpret and apply.

- Since risk measures, such as *VaR* and *ETL*, focus on the *absolute risks* of the distribution without comparing to a benchmark, assuming $r_b = 0$ is consistent with these measures, allowing a pure assessment of tail risks, both for potential gains and losses.
- This setting avoids benchmark selection bias, ensuring that the Rachev Ratio purely reflects the portfolio's intrinsic risk characteristics.

2. **Selection of 5% and 95% Quantiles for Tail Risk Assessment:**

- It balances sensitivity to outliers by capturing significant and practically relevant tail risks. Also, using extreme quantiles like 1% and 99% might make the measure overly sensitive to outliers or rare events.
- The 5% and 95% quantiles are widely accepted in financial risk management as a standardized basis for calculating VaR and ETL. This consistency enhances the practical applicability and interpretability of the Rachev Ratio in professional risk management contexts.
- A broader tail region (5% instead of 1%) includes more data points, making the computed averages statistically more robust and less prone to noise. This ensures that the Rachev Ratio reflects a genuine and reliable risk assessment of the portfolio's performance in extreme conditions.

Note: After incorporating these considerations, the Rachev Ratio is designed to strictly focus on the risk and return characteristics of the ETF itself, without the influence of an external benchmark. This approach ensures that the metric captures the pure tail risks associated with the ETF's performance, providing a clear and unbiased assessment of its potential gains and losses in extreme market conditions.

5.1.2 Sharpe Ratio:

The Sharpe Ratio is a widely used financial metric that measures the risk-adjusted return of an investment. It was introduced by William F. Sharpe in 1966 and is particularly useful for comparing the performance of different investments or portfolios. In the intricate realm of investment, relying solely on returns does not provide a comprehensive understanding. A higher return might look attractive, but if it comes with significantly higher risk, the investment might not be as profitable as it seems.

The Sharpe Ratio for ETF p at horizon h is calculated as [42, pp. 337–338]:

$$SR_{h,p} = \frac{\mathbb{E}[\hat{\mathbf{r}}_{p,h} - r_b]}{\sigma(\hat{\mathbf{r}}_{p,h} - r_b)} \tag{33}$$

– r_b is the constant daily benchmark return (e.g., risk-free rate).

– $\mathbb{E}[\hat{\mathbf{r}}_{p,h} - r_b]$ is the expected excess return over the benchmark within horizon h.

– $\sigma(\hat{\mathbf{r}}_{p,h} - r_b)$ is the standard deviation of excess returns within horizon h.

Experimental settings for Sharpe Ratio

Just like the Rachev Ratio, there are certain necessary considerations that are also made in the Sharpe Ratio.

1. **Benchmark Return r_b is considered as a constant**

- This simplifies comparison across ETFs and isolates the ETF's performance relative to a stable, known benchmark. It also avoids introducing additional volatility that could complicate the interpretation of the Sharpe Ratio.
- Considered annual risk free rate as 10% to demonstrate strong risk-adjusted performance.

2. **Setting the weights $w = 1$ for each ETF**

- This calculation is made on a single ETF each, aligning with the approach of evaluating ETFs individually. With $w = 1$, the weighted simplifies to portfolio's mean return μ.

The Sharpe ratio (SR) with constant benchmark and weight = 1 modifies the Eq.33 as following[42]:

$$SR_{h,p} = \frac{\mu_p - r_b}{\sigma_{r_p}}, \quad \text{where } r_b \text{ is now the constant risk-free rate.} \tag{34}$$

5.1.3 Sortino Ratio:

The Sortino Ratio is a risk-adjusted performance metric that measures the return of an investment relative to the downside risk it carries. Unlike the Sharpe Ratio, which penalizes both upside and downside volatility, the Sortino Ratio focuses only on downside volatility (i.e., returns that fall below a certain target or threshold). The Sortino ratio is given by [42, pp. 329–330]:

$$\text{Sortino}_{h,p} = \frac{\bar{\hat{r}}_{p,h} - r_b}{\hat{\sigma}^-_{p,h}(r_b)} \tag{35}$$

- $\bar{\hat{r}}_{p,h}$: Average forecasted return for ETF p over horizon h,
- r_b: Target benchmark portfolio return.
- $\hat{\sigma}^-_{p,h}(r_b)$: Downside deviation of forecasted returns below the benchmark over horizon h, defined as:

$$\hat{\sigma}^-_{p,h}(r_b) = \sqrt{\frac{1}{n_h} \sum_{t \in \mathcal{T}_h} \max(r_b - \hat{y}_t, 0)^2}$$

- $\hat{y}_t$: Forecasted log return for portfolio p on business day t.

The downside deviation considers only returns below the target, ignoring upside volatility. A higher Sortino ratio indicates a better risk-adjusted return for a given level of downside risk. A lower Sortino ratio suggests poor risk-adjusted performance with significant downside risk.

Experimental setting for Sortino Ratio

1. **Setting constant benchmark or target log-return**
   - In real-world investing, a 4% annual return is often considered a conservative target for portfolios with moderate risk.
   - By setting a target log-return of 0.04, the Sortino Ratio is aligned to compare performance above a risk-free or low-risk benchmark. Using 0.04 as a target provides a balanced perspective on risk-adjusted performance, avoiding overly optimistic targets that may exaggerate the downside.
   - This makes 0.04 a well-justified and practical target for evaluating risk-adjusted returns in real-life scenarios.

5.1.4 Volatility Clustering:

Volatility clustering is a phenomenon in financial markets where large changes in asset prices are often followed by large changes, and small changes are followed by small changes. This implies that volatility (the degree of variation in asset prices) tends to persist over time. The concept is grounded in the idea that volatility is not constant but tends to cluster; periods of high volatility are followed by more high volatility, and periods of low volatility are followed by more low volatility.

The volatility clustering can be calculated for ETF p at the horizon h and is calculated as follows[14]:

$$C^{(p)}_{|\hat{y}|,h}(\tau) = \text{corr}\left(|\hat{y}^{(p)}_t|,\ |\hat{y}^{(p)}_{t+\tau}|\right), \quad t, t+\tau \in \mathcal{T}_h \tag{36}$$

where:

- $C^{(p)}_{|\hat{y}|,h}(\tau)$: Volatility clustering measure for ETF p in horizon h at lag τ.
- $\hat{y}^{(p)}_t$: Forecasted log return for asset p on day $t \in \mathcal{T}_h$.
- τ: Lag .

This correlation is formally calculated using the Pearson correlation coefficient.

Pearson correlation coefficient

The Pearson correlation coefficient is a statistical measure that quantifies the strength and direction of the linear relationship between two variables. Its value ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating no linear correlation, defined as[56]:

$$\text{corr}(A, B) = \frac{\text{Cov}(A, B)}{\sigma_A \sigma_B} \tag{37}$$

where:

- A and B are list of aribitrary values.
- $\text{Cov}(A, B) = \mathbb{E}[(A - \mathbb{E}[A])(B - \mathbb{E}[B])]$ is the covariance of A and B.
- $\sigma_A = \sqrt{\mathbb{E}[(A - \mathbb{E}[A])^2]}$ is the standard deviation of A.
- $\sigma_B = \sqrt{\mathbb{E}[(B - \mathbb{E}[B])^2]}$ is the standard deviation of B.

Applying Pearson Correlation to Volatility Clustering
Substituting $A = |\hat{y}_t^{(p)}|$ and $B = |\hat{y}_{t+\tau}^{(p)}|$ in horizon h:

$$C_{|\hat{y}|,h}^{(p)}(\tau) = \frac{\mathbb{E}\left[\left(|\hat{y}_t^{(p)}| - \mathbb{E}[|\hat{y}_t^{(p)}|]\right)\left(|\hat{y}_{t+\tau}^{(p)}| - \mathbb{E}[|\hat{y}_{t+\tau}^{(p)}|]\right)\right]}{\sigma_{|\hat{y}_t^{(p)}|} \cdot \sigma_{|\hat{y}_{t+\tau}^{(p)}|}} \tag{38}$$

- $\mathbb{E}[|\hat{y}_t^{(p)}|]$, $\mathbb{E}[|\hat{y}_{t+\tau}^{(p)}|]$: Expected values of the absolute forecasted returns for portfolio p at times t and $t + \tau$.
- $\sigma_{|\hat{y}_t^{(p)}|}$, $\sigma_{|\hat{y}_{t+\tau}^{(p)}|}$: Standard deviations of the absolute forecasted returns at times t and $t+\tau$, respectively.

Experimental setting for Volatility Clustering

1. **Consideration of only Lag 1**

- Lag 1 captures the most recent and impactful volatility clustering, making it directly relevant for risk assessment.
- Using lag 1 avoids unnecessary complexity and prevents data sparsity, especially for weekly or monthly returns, thus maximizing valid observations while retaining meaningful clustering insights.

5.2 Composite Scores computation

Composite scores are computed by leveraging various risk-adjusted performance ratios derived from the forecasted returns. As a next step, a composite score formula is constructed, which in turn is used as an objective function for the Sequential Quadratic Programming (SQP) method, where this technique maximizes the score by optimizing weights.

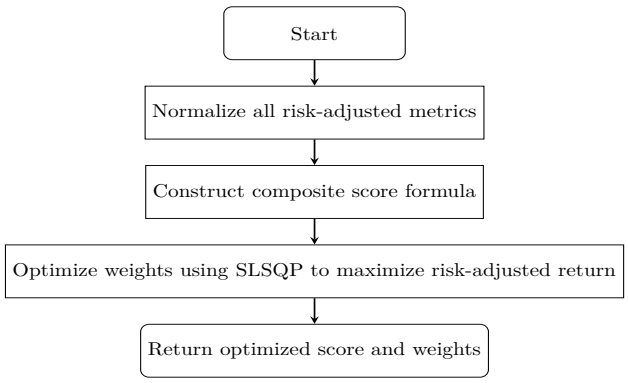

Figure 13: Flowchart of Risk-Adjusted Optimization Process

Throughout this work, all risk-adjusted performance metrics, normalization procedures, and composite score calculations are computed separately for each forecasting horizon and each portfolio or ETF. Each raw risk metric is denoted by $R_{i,h,p}$, and its normalized form by $R'_{i,h,p}$. The final composite score is represented as $S_{h,p}$. For clarity and brevity, the subscripts h (horizon) and p (portfolio) are omitted in the formulas that follow. Unless otherwise specified, all expressions are implicitly defined for each horizon and each portfolio.

1. **Normalization of the risk-adjusted metrics:**

   - Risk-adjusted metrics calculated (Sharpe Ratio, Sortino Ratio, Rachev Ratio, and Volatility Clustering Index), exist on different scales; direct comparison or aggregation can lead to distorted results.
   - To ensure a fair and balanced weighting, each ratio is normalized using standard z-score normalization. Additionally, some ETFs might have extremely high values for certain ratios due to the calculation of the ratios within shorter time frames.
   - Unnormalized ratios could exert an excessive influence, skewing the final score. Hence the ratios are normalized and transformed into the same scale while preserving essential statistical properties.

   $$R'_i = \frac{R_i - \mu_i}{\sigma_i}$$

   where:

   – R_i, R'_i: Original and standardized values of the i-th risk or performance ratio.

   – μ_i, σ_i: Mean and standard deviation of the i-th ratio across all portfolios and horizons.

   The above z-score normalization is applied to each of the four risk-adjusted metrics—Rachev, Sharpe, Sortino, and Volatility Clustering—yielding normalized values R'_1, R'_2, R'_3, R'_4, where $i = 1, 2, 3, 4$ corresponds to these metrics in order.

2. **Composite Score Formulation:** Composite score S at horizon h for each ETF p is formulated as weighted sum of normalized ratios over a stadard deviation of these weighted metrics.

   $$S = f(w_i) = -\frac{\sum_{i=1}^{4} w_i R'_i}{\sigma_{\text{weighted}} + \epsilon} = -\frac{w_1 R'_1 + w_2 R'_2 + w_3 R'_3 - w_4 R'_4}{\sigma(w_1 R'_1 + w_2 R'_2 + w_3 R'_3 - w_4 R'_4) + \epsilon} \tag{39}$$

   ϵ a small noise to to prevent division by zero

   - The weighted standard deviation penalizes volatility, ensuring the composite score remains stable and reliable, unlike a simple weighted sum which ignores risk.
   - It prevents highly volatile instruments from receiving inflated scores, balancing performance with stability.
   - This approach values both magnitude and consistency, penalizing high-performing yet volatile metrics to reduce excessive risk exposure.
   - As a result, weights favor instruments with strong and predictable financial characteristics.
   - Without this adjustment, weights may favor high-ratio but high-risk assets; incorporating risk promotes balanced allocation and avoids dominance by any single volatile instrument.
   - Volatility cluster is negative, to penalize high volatility.

5.3 Weight Optimization using SQP method:

- Weight optimization aims to allocate weights to assets or metrics to maximize outcomes under constraints. SQP efficiently solves such constrained problems, balancing performance and risk while ensuring feasible solutions.
- SQP is a gradient-based method for nonlinear, constrained optimization. It approximates the original problem through quadratic subproblems, making it effective for portfolio optimization, hyperparameter tuning, and multi-objective decision models.

The goal of this optimization problem is to maximize the function $f(w_i)$ mentioned in Step 2 which serves as the objective function

Since it is a minimization problem, the objective function is can be formulated as:

$$\min_{w} -\frac{\sum_{i=1}^{4} w_i R_i'}{\sigma_{\text{weighted}} + \epsilon}$$

Constraints:

The optimization is subject to the following constraints to ensure feasibility and meaningful allocation:

Sum-to-One Constraint (Equality Constraint):

$$c(w) = \sum_{i=1}^{4} w_i - 1 = 0$$

- Ensures that the total allocation is **fully distributed** across financial metrics.

Bound Constraints (Inequality Constraints):

$$b(w) = 0 \leq w_i \leq 1, \quad \forall i \in \{1, 2, 3, 4\}$$

– Ensures that each weight remains non-negative and prevents any single metric from dominating the allocation.

Sequential Least Squares Quadratic Programming (SLSQP)

In this research, the SLSQP method is used to solve a nonlinear optimization problem by iteratively approximating it with quadratic programming (QP) subproblems. At each step, it uses first- and second-order derivative information to update the weights efficiently.

Checkpoint: In order to implement SQP, the objective function and constraints need to be twice continously differentiable. To check whether:

(a) First derivatives (Jacobian or Gradient) exist and are continuous.

(b) Second derivatives (Hessian) exist and are continuous.

Lets analyse this theory into this research and how SLSQP helps to avoid violating the above the conditions.

Objective Function Analysis:

$$f(w) = -\frac{\sum w_i R_i}{\sigma_{\text{weighted}} + \epsilon}$$

First Derivative (Gradient)

$$\nabla f(w) = \frac{\sigma_{\text{weighted}} \cdot \nabla(\sum w_i R_i) - (\sum w_i R_i) \cdot \nabla \sigma_{\text{weighted}}}{(\sigma_{\text{weighted}} + \epsilon)^2}$$

where:

- $\nabla_w \left(\sum w_i R_i\right) = R$, where $R = [R_1, R_2, R_3, R_4]^T$ is the vector of normalized ratios (linear and differentiable).
- $\nabla\sigma_{\text{weighted}}$ is differentiable almost everywhere, except when variance is exactly zero.

Second Derivative (Hessian)

$$\nabla^2 f(w) = -\frac{2R \cdot \nabla\sigma_{\text{weighted}}(w)^T + (w^T R) \cdot \nabla^2\sigma_{\text{weighted}}(w)}{\left(\sigma_{\text{weighted}}(w) + \epsilon\right)^3}$$

- The numerator is differentiable except when $\sigma_{\text{weighted}} \to 0$ and the denominator is always positive due to ϵ.
- The function remains second-order differentiable except at zero variance.

Note: Variance will be zero only when the returns of all the ETFs becomes identitical over a certain time-frame and their risk performances metrics becomes identitical. Hence it can be safely conclude that variance being 0 is very highly unlikely. Hence objective function is twice continously differentiable.

Equality constraint Analysis:

$$c(w) = \sum w_i - 1 = 0$$

- **First derivative (Jacobian):**

$$\nabla c(w) = [1, 1, 1, 1]^T$$

- **Second derivative (Hessian):**

$$\nabla^2 c(w) = 0$$

In-equality Constraints Analysis:

$$b(w) = 0 \leq w_i \leq 1$$

- **Inside the feasible region** $(0 < w_i < 1)$:
  - The constraint is **inactive** $\Rightarrow$ No effect on differentiability.
- **At the boundary** $w_i = 0$ or $w_i = 1$:
  - The feasible direction for gradient descent is constrained, which may cause non-smooth behavior in projected optimization steps.
  - Not twice continuously differentiable at the boundary due to non-smooth transitions introduced by active inequality constraints.

The optimization problem is solved using the SLSQP method, which handles regions with non-twice continuously differentiable constraints. **Algorithm 8** summarizes its core steps based on the work of Kraft [31] and Ma et al. [33], with the backtracking line search adapted from Bertsekas [3].

Algorithm 8 Sequential Quadratic Programming (SQP) Optimization Algorithm [33] [31]

Input: Initial weight vector w, tolerance ϵ, step size α
Initialize: Hessian approximation H_k, iteration counter $k = 0$, and a small positive threshold δ for safeguarding the BFGS update (e.g., $\delta = 10^{-8}$)
procedure SQP Optimization
    Step 1: Initialize Weights and Identify Active and Inactive Weights
        Set initial weights: $w_1 = w_2 = w_3 = w_4 = 0.25$
        Classify each weight w_i as:
        Active if $w_i = 0$ or $w_i = 1$
        Inactive if $0 < w_i < 1$
    Step 2: Compute First-Order Condition (Gradient Calculation)
        Compute gradient of the objective function:

$$\nabla f(w) = \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \ldots, \frac{\partial f}{\partial w_n}\right]^T$$

    Step 3: Compute Lagrange Multipliers for Active Weights
        Introduce multipliers λ_i and μ_j for equality and bound constraints.

$$\nabla f(w) + \sum \lambda_i \nabla c_i(w) + \sum \mu_j \nabla b_j(w) = 0$$

    Step 4: Solve the Quadratic Programming (QP) Subproblem for Inactive Weights
        Solve the QP subproblem:

$$\min_{\Delta w} \quad \frac{1}{2}\Delta w^T H_k \Delta w + \nabla f(w_k)^T \Delta w$$

        Subject to:

$$c(w_k) + \nabla c(w_k)^T \Delta w = 0, \quad b(w_k) + \nabla b(w_k)^T \Delta w \leq 0$$

        (If the LSQ/QP subproblem is infeasible or yields no descent, fallback to full QP solver is triggered as a restoration mechanism)
    Step 5: Update the Hessian approx. for Inactive Weights
        Use a BFGS-like Quasi-Newton update[39, p. 140]:

$$H_{k+1} = H_k + \frac{yy^T}{y^T s} - \frac{H_k s s^T H_k}{s^T H_k s}$$

        where $s = w_{k+1} - w_k$ and $y = \nabla f(w_{k+1}) - \nabla f(w_k)$
        (Update is applied only if $y^\top s > \delta$; otherwise, the Hessian approximation is left unchanged)
    Step 6: Perform Backtracking Line Search and Check Convergence using Armijo rule[3]
        Update weights:

$$w_{k+1} = w_k + \Delta w_k$$

        Step is accepted if:

$$f(w_k + \alpha \Delta w_k) \leq f(w_k) + c\alpha \nabla f(w_k)^T \Delta w_k \qquad ; \text{where } \alpha \text{ is the step size.}$$

        (If $w_k + \alpha \Delta w_k$ violates constraints, reduce α until feasibility is restored)
    Note: Here, $c \in (0, 1)$ is the Armijo rule constant and is not related to the constraint function $c(w)$.
        If $\|\nabla f(w)\| < \epsilon$, $|c(w)| < \epsilon$, and $b(w) \leq \epsilon$, terminate; otherwise, increment k and repeat
end procedure

5.4 Smoothing Process:

Smoothing is a technique used to reduce noise and highlight trends in a time series or set of data points. It works by averaging or adjusting values over a window of time so that short-term fluctuations are minimized while the overall pattern or direction becomes clearer. Final portfolio optimization will be carried out on both composite scores and smoothed scores in order to check their stability over time horizons.

Reasons to include smoothing process:

1. **Reduce Sensitivity to Short-Term Noise:** Raw composite scores may include high-frequency fluctuations due to volatile model predictions, sudden feature changes, or market microstructure noise.
2. **Enable More Robust Decision-Making:** Without smoothing, even minor variations in score can trigger frequent changes in asset selection, leading to overtrading or whipsaw effects. Smoothed scores provide a **more stable basis for ranking and portfolio allocation**.

Exponential Smoothing:

In this research, exponential smoothing is carried out. Exponential smoothing is a technique used to smooth time series data by applying exponentially decreasing weights to past observations. That means recent values have more influence on the smoothed result than older ones.

$$S_h^{\text{smoothed}} = \theta S_h + (1-\theta) S_{h-1}^{\text{smoothed}}$$

where:

- S_h is the unsmoothed score at horizon h for any ETF p,
- S_h^{smoothed} is the smoothed score at horizon h for any ETF p,
- $\theta \in [0, 1]$ is the smoothing factor (higher values give more weight to recent observations).

The smoothing factor θ is set as 0.1 to strongly dampen short-term fluctuations and emphasize longer-term trends in the composite score. This choice reflects a deliberate bias toward stability, which is particularly important in financial time series. Significance of choosing θ as 0.1:

- **Low sensitivity to noise:** A small α allocates less weight to the latest score S_t, minimizing the impact of erratic or one-off changes.
- **Greater inertia:** The smoothed score responds slowly, reducing the chance of over-adjusting portfolio positions based on volatile inputs.
- **More consistent ranking:** This helps maintain ranking stability across time steps, avoiding frequent asset turnover driven by spurious signals.

6 Portfolio Optimization Backtesting process

This chapter outlines the backtesting framework used to evaluate portfolio optimization strategy. It covers the setup of the backtest environment and the key steps involved in managing the portfolio over time.

6.1 Initialization

After computing composite scores for monthly and weekly windows, ETFs are ranked to identify the top 2 performers for investment decisions. As all ETFs are U.S.-listed, prices and values are in USD. Investments are made using the closing price on the first trading day of each period.

For the first week and the first month, the two top-ranked ETFs receive an equal capital allocation of 50,000 units. Using this price, the total number of shares that can be acquired is precisely calculated, ensuring an accurate and systematic allocation. These shares are then securely placed into the portfolio bucket, establishing a structured and disciplined approach to portfolio management.

6.2 Management

Scenario 1: No Change in ETFs:

If the top two ETFs in the next period remain the same (irrespective of the order) as those already in the portfolio, no transactions take place. The portfolio remains intact, avoiding unnecessary trading costs.

Scenario 2: Full Replacement of ETFs:

If both ETFs in the portfolio are different from the newly ranked top two ETFs, a complete rebalancing is performed:

- The existing ETFs in the portfolio are sold at the closing price of the first trading day of the new period.
- The proceeds from these transactions are reinvested into the newly ranked ETFs at their respective closing prices on the same day.
- Each transaction (buy and sell) is subject to a transaction fee factor of 0.9975, accounting for trading costs.
- Order of rank matters here. The i^{th}-ranked ETF in the current portfolio) (in any horizon h) will be replaced by the i^{th} ETF of the following week or month (horizon h+1). Similarly follows for the $(i+1)^{\text{th}}$ ranked ETF.

For example, if the portfolio currently holds ETF_A and ETF_B, the new top ETFs are ETF_C and ETF_D, then:

Sell ETF_A, use proceeds to buy ETF_C.	Both ETF_A and ETF_C first ranked ETFs.
Sell ETF_B, use proceeds to buy ETF_D.	Both ETF_B and ETF_D first ranked ETFs.

Scenario 3: Partial Replacement of ETFs

If only one of the ETFs in the current portfolio is present in the new ranking, a partial reallocation occurs:

- The ETF that is in the portfolio bucket, and also in the next horizon's top two-ranked ETF, then it is retained.
- The ETF that is not in the top two will be sold, and used to purchase the highest-ranked ETF in the following horizon.

For example:

$$\text{If the portfolio holds } (\text{ETF}_A, \text{ETF}_B), \text{ but the next period's top ETFs are } (\text{ETF}_A, \text{ETF}_C), \\ \text{then sell } \text{ETF}_B \text{ and buy } \text{ETF}_C.$$

Similarly,

$$\text{If the portfolio holds } (\text{ETF}_A, \text{ETF}_B), \text{ but the new top ETFs are } (\text{ETF}_B, \text{ETF}_C), \\ \text{then sell } \text{ETF}_A \text{ and buy } \text{ETF}_C.$$

Transaction Formulas

The transaction formulas apply at the individual ETF level and are used for each buy/sell action triggered by portfolio changes. They are executed per ETF involved in the reallocation.

Initial ETF Purchase

For an initial ETF purchase, the number of shares bought is calculated as:

$$Shares_{p,h} = \frac{I \times 0.9975}{P_{p,h}} \tag{40}$$

where I is the investment amount, and $P_{p,h}$ is the closing price of ETF p on the first trading day of the horizon h.

Selling an ETF

For selling an ETF in the next month, the selling value is calculated as:

$$V_{p,h+1} = Shares_{p,h} \times P_{p,h+1} \times 0.9975 \tag{41}$$

where $P_{p,h+1}$ is the ETF's closing price at the first day of the next period $h+1$.

Buying an ETF with Proceeds

When purchasing a new ETF using the proceeds from selling another ETF, the number of shares acquired is:

$$Shares_{new-p,h+1} = \frac{V_{p,h+1} \times 0.9975}{P_{new-p,h+1}} \tag{42}$$

where $V_{p,h+1}$ is the selling value of ETF p, and $P_{new-p,h+1}$ is the closing price of the newly purchased ETF on the first day of the next period $h+1$.

Final Portfolio Valuation

This systematic portfolio reallocation process continues until the end of the year. At year-end, the final portfolio value is computed using:

$$V_{\text{final}} = \sum_{i=1}^{N} Shares_{final-p,H} \cdot P_{final-p,H+1}, \tag{43}$$

where:

- $Shares_{final-p,H}$ is the number of shares held of any ETF in the portfolio at the final period T.
- $P_{final-p,H+1}$ is the closing price on the first trading day of the following year.

This final valuation represents the total capital achieved through the strategy, providing a comprehensive measure of portfolio performance. By adhering to this structured investment approach, the strategy ensures a disciplined and dynamic portfolio adjustment mechanism.

7 Numerical Results

This chapter presents the key numerical findings. It includes a comparison of model evaluation metrics, examination of portfolio stability, and a performance comparison between model-based strategies and the buy-and-hold approach.

7.1 Comparitive analysis of model evaluation metrics

Evaluation measures are used to evaluate model behavior, including overfitting, underfitting, general stability, and training efficiency. They do not necessarily indicate whether a model is significantly superior to others; rather, they serve as essential standards for guiding model development and refinement, ensuring alignment with data characteristics and practical constraints.

Note: The reported metrics for each model represent the average performance across all ETFs, enabling more straightforward comparison and interpretation.

Metric	OLS	LASSO	Boosted Trees	Random Forest	LSTM
Training MAE	0.0116	0.0115	0.0111	0.0116	0.0121
Test MAE	0.0143	0.0141	0.0140	0.0139	0.0132
Training RMSE	0.0163	0.0164	0.0156	0.0167	0.0173
Test RMSE	0.0197	0.0195	0.0195	0.0193	0.0185
Training MASE	0.69	0.68	0.66	0.69	0.70
Test MASE	0.70	0.69	0.69	0.68	0.69
Total Time (s)	106.27	35.62	222.63	668.25	2404.51
Avg. Time per ETF (s)	10.63	3.56	22.26	66.83	240.45

Table 3: Model Evaluation Metrics for 2024

Metric	OLS	LASSO	Boosted Trees	Random Forest	LSTM
Training MAE	0.0119	0.0118	0.0114	0.0119	0.0123
Test MAE	0.0121	0.0119	0.0118	0.0117	0.0112
Training RMSE	0.0167	0.0167	0.0160	0.0171	0.0176
Test RMSE	0.0176	0.0174	0.0174	0.0172	0.0165
Training MASE	0.69	0.68	0.66	0.69	0.69
Test MASE	0.69	0.68	0.67	0.67	0.67
Total Time (s)	107.08	35.62	221.52	518.91	2191.22
Avg. Time per ETF (s)	10.71	3.56	22.15	51.89	219.122

Table 4: Model Evaluation Metrics for 2022

1. **Training & Test Metric Gaps Are Relatively Small:** All models show small differences between training and test evaluation metrics. This minimal gap indicates adequate generalization behavior. The observed consistency suggests the models are learning meaningful patterns without becoming overfitted to the training data.

2. **No Evidence of Underfitting in Any Model:** Despite some variation in absolute error levels across models, the gap between training and test performance remains narrow. This rules out possibility of underfitting, which would present as high and similar errors on both sets due to insufficient model capacity or poor representation.

3. **Performance Stability Across Volatile Test Conditions:** All models performed better when forecasting 2022 than 2024, due to lower volatility and more stable temporal patterns in the 2022 data. Sequence-sensitive models like LSTM especially benefited from this stability. The performance gap

reflects increased complexity and structural shifts in 2024, though all models maintained a favorable bias-variance tradeoff.

7.2 Stability Analysis of Model-Based Portfolio Selections

This sub-chapter analyzes the stability of monthly final portfolio selections for 2024 and 2022, as generated by the backtesting framework for systematic ETF portfolio construction. The goal is to assess temporal consistency across models and examine how smoothing reduces turnover while improving signal robustness. Each model selects the top 2 ETFs per month based on composite scores, which integrate forecasted returns with risk-adjusted performance metrics varying by model architecture and scoring logic. For clarity, the analysis focuses on month-wise selections in 2024 and 2022, similar trends are observed at the weekly level.

The objective of this comparative analysis is two-fold:

- **Model Stability Assessment:** By tracking portfolio composition changes across months, assess the stability and consistency of each model. More volatile shifts suggest sensitivity to noise, whereas recurrent ETF selections will indicate model robustness.
- **Transaction Cost Implications:** Frequent changes in ETF constituents translate to higher turnover rates, which can incur significant transaction costs in practice. Thus, models that exhibit smoother allocation dynamics may be more desirable from an implementation standpoint.

Final Portfolio Selection per Month without Smoothing of Scores

Linear models like OLS and LASSO exhibit the highest volatility, with frequent full portfolio changes, often replacing both ETFs each month. Tree-based models (XGBoost, Random Forest) show better consistency, with reduced full portfolio rebalancing every month than linear based, suggesting more robust internal signal processing. LSTM, shows the higher degree of stability, with partial or no ETF changes between months. Very rarely full replacement of ETFs happens. This variance highlights how different model classes respond to short-term score fluctuations, with linear models being most reactive and sequence models inherently more stable.

Month	OLS	LASSO	XGBoost	Random Forest	LSTM
1	[PSI, SMH]	[PSI, IGM]	[XLK, VGT]	[VGT, SMH]	[PSI, IXN]
2	[PSI, SOXX]	[SMH, VGT]	[IGM, SMH]	[XLK, PSI]	[IXN, PSI]
3	[IGM, PSI]	[SMH, VGT]	[IXN, QQQ]	[PSI, IGM]	[VGT, PSI]
4	[SMH, QQQ]	[SMH, VGT]	[VGT, QQQ]	[XLK, SMH]	[PSI, IXN]
5	[SMH, VGT]	[IYW, VGT]	[VGT, QQQ]	[IGM, PSI]	[PSI, VGT]
6	[PSI, XSD]	[IXN, XSD]	[PSI, QQQ]	[QQQ, XSD]	[PSI, VGT]
7	[QQQ, XLK]	[QQQ, SOXX]	[QQQ, SOXX]	[XLK, VGT]	[VGT, QQQ]
8	[SMH, QQQ]	[XLK, SMH]	[IYW, VGT]	[IYW, VGT]	[VGT, PSI]
9	[XSD, PSI]	[SOXX, XSD]	[PSI, QQQ]	[XSD, VGT]	[VGT, PSI]
10	[SOXX, IGM]	[SOXX, VGT]	[PSI, QQQ]	[QQQ, XLK]	[VGT, QQQ]
11	[SOXX, IXN]	[SMH, SOXX]	[QQQ, XLK]	[XSD, XLK]	[VGT, PSI]
12	[XLK, IYW]	[IGM, SOXX]	[VGT, XLK]	[VGT, QQQ]	[VGT, PSI]

Table 5: Final Portfolio Selection per Month (Without Smoothing) for 2024

Month	OLS	LASSO	XGBoost	Random Forest	LSTM
1	[PSI, XSD]	[PSI, VGT]	[IYW, VGT]	[PSI, VGT]	[VGT, PSI]
2	[PSI, SOXX]	[IXN, PSI]	[IYW, PSI]	[QQQ, IXN]	[VGT, PSI]
3	[PSI, QQQ]	[SMH, PSI]	[XLK, PSI]	[PSI, VGT]	[VGT, PSI]
4	[SMH, PSI]	[XLK, VGT]	[XSD, SMH]	[SOXX, IYW]	[VGT, PSI]
5	[SMH, VGT]	[SMH, VGT]	[QQQ, SMH]	[SOXX, IGM]	[PSI, VGT]
6	[SMH, PSI]	[IXN, SMH]	[IYW, SOXX]	[XLK, IYW]	[PSI, VGT]
7	[PSI, QQQ]	[SOXX, VGT]	[SMH, PSI]	[XSD, PSI]	[VGT, PSI]
8	[VGT, SMH]	[VGT, IXN]	[VGT, SOXX]	[PSI, XSD]	[PSI, VGT]
9	[XSD, VGT]	[IXN, XSD]	[XSD, VGT]	[VGT, SMH]	[PSI, SOXX]
10	[IGM, XSD]	[IGM, PSI]	[IGM, XSD]	[XSD, IGM]	[PSI, XSD]
11	[SOXX, PSI]	[IGM, VGT]	[XSD, QQQ]	[IYW, SMH]	[PSI, SOXX]
12	[PSI, XSD]	[XSD, SOXX]	[XSD, IGM]	[XSD, IGM]	[PSI, SOXX]

Table 6: Final Portfolio Selection per Month (Without Smoothing) for 2022

Final Portfolio Selection per Month with smoothing of scores

Smoothing significantly enhances selection stability by reducing the frequency of month-to-month changes in portfolio composition. For most models, this results in persistent ETF selections sustained over multiple months, effectively dampening reactions to short-term noise. This leads to fewer transactions, thereby minimizing turnover and associated costs in practical deployment. The effect is most pronounced in linear models, where previously unstable selections become consistent and predictable. Tree-based models also benefit, further stabilizing their output with even fewer changes per month. In contrast, LSTM shows minimal benefit from smoothing, as its sequence-aware structure already captures temporal continuity, yielding inherently stable outputs with very few or no changes even without post-processing.

Month	OLS	LASSO	XGBoost	Random Forest	LSTM
1	[PSI, SMH]	[PSI, IGM]	[XLK, VGT]	[VGT, SMH]	[PSI, IXN]
2	[PSI, SMH]	[PSI, IGM]	[XLK, VGT]	[VGT, SMH]	[PSI, IXN]
3	[PSI, SMH]	[VGT, PSI]	[XLK, VGT]	[VGT, SMH]	[PSI, VGT]
4	[PSI, SMH]	[VGT, PSI]	[XLK, VGT]	[VGT, SMH]	[PSI, VGT]
5	[PSI, SMH]	[VGT, PSI]	[XLK, VGT]	[VGT, SMH]	[PSI, VGT]
6	[PSI, SMH]	[VGT, PSI]	[XLK, VGT]	[VGT, SMH]	[PSI, VGT]
7	[PSI, SMH]	[VGT, PSI]	[XLK, VGT]	[VGT, XLK]	[PSI, VGT]
8	[SMH, PSI]	[VGT, PSI]	[XLK, VGT]	[VGT, XLK]	[PSI, VGT]
9	[PSI, SMH]	[VGT, PSI]	[XLK, VGT]	[VGT, XLK]	[VGT, PSI]
10	[SMH, PSI]	[VGT, PSI]	[XLK, VGT]	[VGT, XLK]	[VGT, PSI]
11	[SMH, PSI]	[VGT, PSI]	[XLK, VGT]	[VGT, XLK]	[VGT, PSI]
12	[SMH, PSI]	[VGT, PSI]	[XLK, VGT]	[VGT, XLK]	[VGT, PSI]

Table 7: Final Portfolio Selection per Month (With Smoothing) for 2024

Month	OLS	LASSO	XGBoost	Random Forest	LSTM
1	[PSI, XSD]	[PSI, VGT]	[IYW, VGT]	[PSI, VGT]	[VGT, PSI]
2	[PSI, XSD]	[PSI, VGT]	[IYW, XSD]	[PSI, VGT]	[VGT, PSI]
3	[PSI, QQQ]	[PSI, VGT]	[IYW, PSI]	[PSI, VGT]	[VGT, PSI]
4	[PSI, QQQ]	[PSI, VGT]	[IYW, XSD]	[PSI, VGT]	[VGT, PSI]
5	[PSI, XSD]	[VGT, PSI]	[IYW, XSD]	[PSI, VGT]	[VGT, PSI]
6	[PSI, XSD]	[VGT, PSI]	[IYW, XSD]	[PSI, VGT]	[VGT, PSI]
7	[PSI, XSD]	[VGT, PSI]	[IYW, XSD]	[PSI, XSD]	[VGT, PSI]
8	[PSI, SMH]	[VGT, PSI]	[XSD, PSI]	[PSI, XSD]	[VGT, PSI]
9	[PSI, VGT]	[VGT, PSI]	[XSD, VGT]	[PSI, XSD]	[VGT, PSI]
10	[PSI, XSD]	[VGT, PSI]	[XSD, SMH]	[PSI, XSD]	[PSI, VGT]
11	[PSI, XSD]	[VGT, PSI]	[XSD, SMH]	[PSI, XSD]	[PSI, VGT]
12	[PSI, XSD]	[VGT, PSI]	[XSD, SMH]	[XSD, PSI]	[PSI, VGT]

Table 8: Final Portfolio Selection per Month (With Smoothing) for 2022

Without smoothing, portfolio selections change frequently across months, reflecting instability in model outputs in the tradiotional ML methods. These monthly results illustrate clear patterns of model-specific stability, with smoothing consistently reducing portfolio turnover and enhancing temporal consistency across architectures. The same methodological framework and behavioral trends are expected to persist in the weekly selection horizon, where smoothing will similarly aid in dampening high-frequency noise and promoting more stable, cost-effective portfolio decisions. This weekly selection of ETFs is mentioned later in the Appendix.

7.3 Comparative Analysis of Portfolio values and Buy-and-Hold Strategy

This sub-chapter presents a comparative analysis of the total portfolio values accumulated over the course of a year using different forecasting models from the backtesting. The performance of each model-driven strategy is evaluated in contrast to a buy-and-hold approach, which serves as a benchmark. For this comparison, the ETF SPY is selected, it is purchased at the beginning of the year and held without any transactions throughout the period. This allows to assess how dynamic, forecast-driven strategies perform relative to a passive investment held over the same horizon.

Reason for choosing SPY as a Buy and Hold benchmark

1. **Market Representation:** SPY is an ETF that tracks the S&P 500, representing a broad cross-section of the U.S. equity market. Its performance is widely considered a benchmark for overall market health, making it an ideal reference point for comparing model-driven strategies against a standard, passive investment.

2. **High Liquidity and Low Slippage:** SPY is one of the most liquid ETFs in the world, with high daily trading volume and tight bid-ask spreads. This ensures that transaction-related distortions such as slippage and liquidity constraints are minimal, allowing for cleaner comparisons across trading strategies.

3. **Stability and Historical Depth:** SPY has a long and stable historical price record, covering various market cycles, including booms and downturns. This makes it a reliable instrument for evaluating strategy robustness across different time frames and economic conditions.

ETF	**Year**	**Buy and Hold value**
SPY	2024	125275.33
SPY	2022	81150.22

Table 9: Buy and Hold results for SPY across years

Table 10 shows backtested portfolio values at the end of 2024. In 2024, LSTM emerged as the most stable and consistent model, showing minimal dependence on smoothing and limited sensitivity to forecasting frequency. Its portfolio returns remained robust across all setups, demonstrating resilience to short-term volatility and strong generalization over time.

Random Forest, while achieving competitive smoothed returns, exhibited substantial variance between smoothed and unsmoothed results, particularly in weekly forecasts, indicating that its raw predictions are volatile. Boosted Trees performed relatively steadily, especially in month-wise configurations, but still showed more sensitivity to week-wise settings.

In contrast, OLS and LASSO showed significant improvements only when smoothing was applied. Their performance dropped sharply without smoothing, especially in week-wise forecasts, highlighting their limited robustness and high sensitivity to temporal noise. OLS fared slightly better than LASSO in the monthly horizon.

Model	**Year**	**Month-wise W/O Smoothing**	**Month-wise With Smoothing**	**Week-wise W/O Smoothing**	**Week-wise With Smoothing**
OLS	2024	128140.21	133961.93	106412.49	128756.17
LASSO	2024	117395.06	131003.69	104276.29	128616.26
Boosted Trees	2024	127159.29	128327.92	104262.91	119211.14
Random Forest	2024	125943.98	144462.14	109982.59	128623.15
LSTM	2024	135547.20	127757.95	126466.39	127567.69

Table 10: Portfolio values for models in 2024

Table 11 summarizes backtested portfolio values at the end of 2022. In 2022, a year marked by higher market volatility, LSTM remained the most stable and reliable model among all, showing minimal variation between month-wise and week-wise results and less variation between smoothed and unsmoothed values across horizons. This consistent performance highlights LSTM's strong adaptability and low sensitivity to temporal noise, making it well-suited for turbulent market conditions without relying on post-processing techniques like smoothing.

Random Forest and Boosted Trees followed closely, showing solid performance with notable gains from smoothing, especially in the week-wise setting. Their substantial gap between raw and smoothed returns

indicates high volatility in unsmoothed forecasts, but also their ability to stabilize when temporal aggregation is applied. OLS and LASSO, on the other hand, were significantly more sensitive to both forecast frequency and lack of smoothing.

Despite LSTM being the most consistent model, its return trailed SPY, which itself did not outperform the initial investment over the year, highlighting overall market decline.

Model	**Year**	**Month-wise W/O Smoothing**	**Month-wise With Smoothing**	**Week-wise W/O Smoothing**	**Week-wise With Smoothing**
OLS	2022	59628.05	63242.78	54936.79	59501.38
LASSO	2022	61113.46	66361.22	56389.75	63446.43
Boosted Trees	2022	67512.79	68198.43	58823.51	66094.36
Random Forest	2022	68154.62	71879.04	54484.94	65272.32
LSTM	2022	67617.80	66361.22	67549.86	66361.22

Table 11: Portfolio values for models in 2022

Overall, compared to traditional machine learning models, LSTM consistently delivered more stable forecasts and superior risk-adjusted performance across both weekly and monthly horizons. Its reduced sensitivity to short-term volatility underscores its robustness across evaluation intervals and results in more reliable yield outcomes.

8 Conclusion

This chapter summarizes the main findings of the study and outlines potential directions for future research.

8.1 Summary

Firstly, feature selection served as a foundational pillar in this research. The inclusion of seasonality-based features enabled the models to internalize recurring patterns across time, thus enhancing their ability to respond to cyclical market behaviors. Moreover, the integration of Simple Moving Averages (SMA) across varying window lengths acted as a smoothing mechanism, dampening high-frequency noise and allowing models to focus on medium-term trends rather than reacting to momentary fluctuations. From the preprocessing perspective, the use of polynomial feature expansion facilitated the modeling of nonlinear dependencies and feature interactions, giving both linear models and tree-based ensembles a much-needed boost in capturing curvature and conditional relationships within the input space. Despite these sophisticated engineering steps, one fundamental limitation that persists in linear and ensemble-based models is learning temporal dynamics. This is where LSTM models excel. LSTM's ability to retain long-range dependencies and sequence-aware context allowed it to capture subtle temporal lags and trend shifts, offering it a significant edge, especially in financial domains where time dependency is paramount.

Importantly, from a model evaluation perspective, all algorithms were optimized to avoid classical pitfalls, such as overfitting or underfitting, while maintaining a statistical balance in the bias-variance trade-off. While the performance gaps between training and test sets varied across 2022 and 2024, these slight changes largely reflect the increased volatility and non-stationarity of market conditions recently. However, such drifts can be mitigated to a certain extent by retraining, where each model was retrained with the most up-to-date data up to the forecast start point, improving robustness to market regime shifts. The use of risk performance metrics also proved to be effective as benchmarks for the risk assessment of assets and played a key role in computing composite scores for informed decision-making.

Among all models, LSTM emerged as the superior choice for forecasting. It not only delivered the highest cumulative portfolio values in most configurations but also demonstrated the most stable and consistent returns, minimizing sharp drawdowns and erratic trading behavior. While some models, such as XGBoost and OLS, occasionally performed close to LSTM in certain settings, their performance was not consistent across different markets and time horizons. This highlights that selecting a model based on isolated conditions may present an incomplete picture, as reliability across varying scenarios is essential for practical deployment. In 2024, LSTM managed to outperform even the buy-and-hold strategy, a notable achievement considering SPY's strong market performance.

However, the same did not hold true in 2022, a year marred by global macroeconomic shocks and widespread equity sell-offs. In fact, the downturn was so pervasive that even the buy-and-hold strategy could not generate positive returns, despite SPY being one of the most stable large-cap ETFs. Nevertheless, LSTM's relative performance remained the most resilient, reaffirming the idea that forecasting stability is more valuable than absolute yield in turbulent markets. However, its edge comes at a computational cost, where this model required significantly more training time per ETF than any other model in the study.

The integration of risk performance metrics such as the Rachev Ratio, Sharpe Ratio, Sortino Ratio, and Volatility Clustering provided a robust framework for evaluating and validating the quality of forecasts. These metrics offered deeper insights into the risk-adjusted performance of predictions, ensuring that forecast models not only aimed for accuracy but also aligned with risk management principles. Overall, this combination covers a majority of risk scenarios such as including downside deviation, tail risks, and time-dependent volatility, making the evaluation process both comprehensive and practically relevant.

Another notable technical contribution is the use of Sequential Quadratic Programming (SQP) for weight optimization during composite score calculation. SQP offered an elegant solution that was not only computationally efficient and convergent but also adaptively balanced multiple risk-performance criteria

in the scoring function. This facilitated the construction of well-ranked and diversified portfolios, making it both a theoretically sound and practically efficient choice for model comparison in financial forecasting.

Overall, LSTM outperformed other models in terms of stability and reliability. Its ability to model temporal dependencies led to fewer changes in top ETF rankings, reducing the transaction frequency and associated costs. This showcase LSTM also achieved stronger risk-adjusted performance. In contrast, tree-based and linear models performed well over the monthly horizon; however, they struggled to maintain yield and exhibited weaker risk profiles on a weekly basis, which indicates that their forecasts are sensitive to shorter terms due to high volatility. While smoothing significantly improved yield for other models by reducing volatility in forecasts, its effect on LSTM is observed to be marginal, serving a great role as a stability evaluation benchmark.

8.2 Future Prospects

1. **Unified Multi-Asset Modeling Concepts**
    - The current technique trains different models for each asset to capture each ETF's unique temporal characteristics and behavioral patterns. This granularity presents ETF-specific information; however, a multi-asset forecasting model trained on all ETFs may be constructed in the future.
    - The model needs a much larger and more diverse historical dataset to generalize across asset classes without compromising precision. Sequence-based transformer architectures can be implemented on large data sets.
2. **Hybrid Forecasting Models: Combining Temporal and Structural Learning**
    - Future research could explore hybrid model frameworks that combine LSTM with traditional machine learning models such as Random Forest or XGBoost to enhance generalization and improve interpretability.
    - In such architectures, the LSTM component captures temporal dependencies, while the traditional model learns from engineered static features or residual patterns. This combination may lead to greater forecasting robustness, especially under volatile market conditions.
3. **Exploring different Risk Metrics Frameworks and Advanced Optimization for Composite Scoring**
    - The current risk framework can be extended with metrics like the Calmar and Omega Ratios to improve evaluation granularity across diverse financial scenarios.
    - Composite score weights are optimized using SQP for efficiency and performance; future work may explore unconstrained methods to enhance flexibility in high-dimensional or multi-objective scenarios.

References

[1] Ahmad Abdur, Ahmad Abdur Rahman Khan, and Lokesh Tanwani. "Portfolio Optimization: Theory, Methods, and Applications". In: (June 2024). [Online; accessed 16-June-2025]. URL: https://www.researchgate.net/publication/381458899_Portfolio_Optimization_Theory_Methods_and_Applications.

[2] Yuhan Bai. "RELU-Function and Derived Function Review". In: *SHS Web of Conferences* 144 (Aug. 2022), p. 02006. DOI: 10.1051/shsconf/202214402006.

[3] Dimitri P Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334. DOI: https://doi.org/10.1057/palgrave.jors.2600425.

[4] Fischer Black and Robert Litterman. "Global Portfolio Optimization". In: *Financial Analysts Journal* 48.5 (Sept. 1992), p. 28. DOI: 10.2469/faj.v48.n5.28. URL: https://doi.org/10.2469/faj.v48.n5.28.

[5] Blog by Jacob Joseph on CleverTap. *A Gentle Introduction to Neural Networks*. URL: https://clevertap.com/blog/neural-networks/.

[6] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[7] Arya Brijith. "Data Preprocessing for Machine Learning". In: (Oct. 2023). [Online; accessed 17-June-2025], p. 2023. URL: https://www.researchgate.net/publication/375003512_Data_Preprocessing_for_Machine_Learning.

[8] Steven Brunton and J. Kutz. "Singular Value Decomposition (SVD)". In: Feb. 2019, pp. 3–46. ISBN: 9781108422093. DOI: 10.1017/9781108380690.002.

[9] Alexander Burton. "OLS (Linear) Regression". In: Aug. 2021, pp. 509–514. ISBN: 9781119110729. DOI: 10.1002/9781119111931.ch104.

[10] Tianfeng Chai, Roland R Draxler, et al. "Root mean square error (RMSE) or mean absolute error (MAE)". In: *Geoscientific model development discussions* 7.1 (2014), pp. 1525–1534. DOI: https://doi.org/10.5194/gmd-7-1247-2014.

[11] 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*. KDD '16. ACM, Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[12] Anjir Chowdhury et al. "A Comparative Study of Hyperparameter Optimization Techniques for Deep Learning". In: (Jan. 2022), pp. 509–521. DOI: 10.1007/978-981-19-0332-8_38.

[13] Wikimedia Commons. *File:Recurrent neural network unfold.svg — Wikimedia Commons, the free media repository*. [Online; accessed 14-June-2025]. 2024. URL: https://commons.wikimedia.org/w/index.php?title=File:Recurrent_neural_network_unfold.svg&oldid=854998628.

[14] Rama Cont. "Volatility Clustering in Financial Markets: Empirical Facts and Agent-Based Models". In: *Long Memory in Economics*. Ed. by Gilles Teyssière and Alan P. Kirman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 289–309. ISBN: 978-3-540-34625-8. DOI: 10.1007/978-3-540-34625-8_10. URL: https://doi.org/10.1007/978-3-540-34625-8_10.

[15] Laurent Deville. "Exchange Traded Funds: History, Trading, and Research". In: *Handbook of Financial Engineering*. Ed. by Constantin Zopounidis, Michael Doumpos, and Panos M. Pardalos. Boston, MA: Springer US, 2008, pp. 67–98. ISBN: 978-0-387-76682-9. DOI: 10.1007/978-0-387-76682-9_4. URL: https://doi.org/10.1007/978-0-387-76682-9_4.

[16] Philip Hans Franses. "A note on the Mean Absolute Scaled Error". In: *International Journal of Forecasting* 32.1 (2016), pp. 20–22. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2015.03.008. URL: https://www.sciencedirect.com/science/article/pii/S0169207015000448.

[17] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML]. URL: https://arxiv.org/abs/1807.02811.

[18] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". In: *Machine learning* 63 (2006), pp. 3–42. DOI: https://doi.org/10.1007/s10994-006-6226-1.

[19] Shihao Gu, Bryan Kelly, and Dacheng Xiu. "Empirical Asset Pricing via Machine Learning". In: *The Review of Financial Studies* 33.5 (Feb. 2020), pp. 2223–2273. ISSN: 0893-9454. DOI: 10.1093/rfs/hhaa009. eprint: https://academic.oup.com/rfs/article-pdf/33/5/2223/33209812/hhaa009.pdf. URL: https://doi.org/10.1093/rfs/hhaa009.

[20] Joko Haryanto. "Short-Term Versus Long-Term Portfolio Management Strategies and the Selection of Securities". In: *Advances in Management & Financial Reporting* 2 (Jan. 2024). DOI: 10.60079/amfr.v2i1.247.

[21] Bo He et al. "Bi-directional LSTM-GRU Based Time Series Forecasting Approach". In: *International Journal of Computer Science and Information Technology* 3 (July 2024), pp. 222–231. DOI: 10.62051/ijcsit.v3n2.26.

[22] Richard J Herring and Anthony M Santomero. *The role of the financial sector in economic performance*. [Online; accessed 26-June-2025]. Wharton School, University of Pennsylvania, 1995. URL: https://www.researchgate.net/publication/2595907_The_Role_of_the_Financial_Sector_in_Economic_Performance.

[23] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. "A Review on the Long Short-Term Memory Model". In: (2020). DOI: https://doi.org/10.1007/s10462-020-09838-1.

[24] Thomas M. Idzorek. *A Step-by-Step Guide to the Black-Litterman Model: Incorporating User-Specified Confidence Levels*. White Paper. Available at https://people.duke.edu/~charvey/Teaching/BA453_2006/Idzorek_onBL.pdf. Ibbotson Associates, 2004.

[25] International Monetary Fund. *Global Financial Stability Report, April 2022: Shockwaves from the War in Ukraine Test the Financial System's Resilience*. [Online; Last accessed 17-June-2025]. URL: https://www.imf.org/en/Publications/GFSR/Issues/2022/04/19/global-financial-stability-report-april-2022.

[26] International Monetary Fund. *Global Financial Stability Report, October 2022: Navigating the High-Inflation Environment*. [Online; Last accessed 18-June-2025]. URL: https://www.imf.org/en/Publications/GFSR/Issues/2022/10/11/global-financial-stability-report-october-2022.

[27] International Monetary Fund. *Global Financial Stability Report, October 2024 - Steadying the Course: Uncertainty, Artificial Intelligence, and Financial Stability*. [Online; Last accessed 18-June-2025]. URL: https://www.imf.org/en/Publications/GFSR/Issues/2024/10/22/global-financial-stability-report-october-2024.

[28] Issued by Dimensional Fund Advisors Ltd. (Dimensional UK), 20 Triton Street, Regent's Place, London, NW1 3BF. Dimensional UK is authorised and regulated by the Financial Conduct Authority (FCA) - Firm Reference No. 150100. *Dimensional's Takes on the Biggest Financial Topics of 2022*. [Online; Last accessed 17-June-2025]. URL: https://www.dimensional.com/fi-en/insights/dimensionals-takes-on-the-biggest-financial-topics-of-2022.

[29] Issued by Dimensional Fund Advisors Ltd. (Dimensional UK), 20 Triton Street, Regent's Place, London, NW1 3BF. Dimensional UK is authorised and regulated by the Financial Conduct Authority (FCA) - Firm Reference No. 150100. *Market Review 2024: Stocks Overcome Uncertainty to Notch Another Strong Year*. [Online; Last accessed 17-June-2025]. URL: https://www.dimensional.com/fi-en/insights/market-review-2024-stocks-overcome-uncertainty-to-notch-another-strong-year.

[30] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[31] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Technical Report DFVLR-FB 88-28. [Online; accessed 28-June-2025]. Koln, Germany: DLR German Aerospace Center — Institute for Flight Mechanics, 1988. URL: https://degenerateconic.com/uploads/2018/03/DFVLR_FB_88_28.pdf.

[32] Yimou Li, Zachary Simon, and David Turkington. "Investable and interpretable machine learning for equities". In: *The Journal of Financial Data Science* 4.1 (2022), pp. 54–74. DOI: DOI:10.3905/jfds.2021.1.084.

[33] Yingjie Ma et al. "Improved SQP and SLSQP Algorithms for Feasible Path-based Process Optimisation". In: (Feb. 2024). DOI: 10.48550/arXiv.2402.10396.

[34] Harry Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (1952). [Online; accessed 16-June-2025], pp. 77–91. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2975974 (visited on 05/04/2025).

[35] Gustavo Martins et al. "Design and analysis of recurrent neural networks for ultrafast optical pulse nonlinear propagation". In: *Optics Letters* 47 (Oct. 2022). DOI: 10.1364/OL.472267.

[36] Danilo Milhomem and Maria Dantas. "Analysis of new approaches used in portfolio optimization: a systematic literature review". In: *Production* 30 (Jan. 2020). DOI: 10.1590/0103-6513.20190144.

[37] Rju Mohan. "Stock Markets: An Overview and A Literature Review". In: (Apr. 2019). DOI: 10.13140/RG.2.2.22639.05289.

[38] Neri Van Otten. *Understand Ordinary Least Squares: How To Beginner's Guide [Tutorials In Python, R & Excell]*. [Online; Last accessed 22-June-2025]. URL: https://spotintelligence.com/2024/08/29/ordinary-least-squares-ols/.

[39] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Second Edition. Springer Series in Operations Research and Financial Engineering. ISSN: 1431-8598. Springer New York, NY, 2006. ISBN: 978-0-387-40065-5. DOI: https://doi.org/10.1007/978-0-387-40065-5.

[40] Elitsa Petrova. "A brief overview of the types of ETFs". In: *Annals of Spiru Haret University Economic Series* 15 (Sept. 2015), p. 39. DOI: 10.26458/1534.

[41] Alvin Poernomo and Dae-Ki Kang. "Biased Dropout and Crossmap Dropout: Learning towards effective Dropout regularization in convolutional neural network". In: *Neural Networks* 104 (2018), pp. 60–67. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2018.03.016. URL: https://www.sciencedirect.com/science/article/pii/S0893608018301096.

[42] Svetlozar T. Rachev, Stoyan V. Stoyanov, and Frank J. Fabozzi. *Advanced Stochastic Models, Risk Assessment, and Portfolio Optimization*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008. ISBN: 978-0-470-05316-4.

[43] Fatima Es-sabery et al. "Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier". In: *IEEE Access* PP (Jan. 2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3053917.

[44] Hasan Salman, Ali Kalakech, and Amani Steiti. "Random Forest Algorithm Overview". In: *Babylonian Journal of Machine Learning* 2024 (June 2024), pp. 69–79. DOI: 10.58496/BJML/2024/007.

[45] Patrick Schneider and Fatos Xhafa. "Chapter 3 - Anomaly detection: Concepts and methods". In: *Anomaly Detection and Complex Event Processing over IoT Data Streams*. Ed. by Patrick Schneider and Fatos Xhafa. Academic Press, 2022, pp. 49–66. ISBN: 978-0-12-823818-9. DOI: https://doi.org/10.1016/B978-0-12-823818-9.00013-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128238189000134.

[46] scikit-optimize developers. *Scikit-Optimize Documentation (Development Version)*. Last Accessed: 2025-06-25. 2021. URL: https://scikit-optimize.github.io/dev//_downloads/scikit-optimize-docs.pdf.

[47] William F. Sharpe. "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk". In: *The Journal of Finance* 19.3 (1964), pp. 425–442. ISSN: 00221082, 15406261. DOI: https://doi.org/10.2307/2977928. URL: http://www.jstor.org/stable/2977928.

[48] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "The Performance of LSTM and BiLSTM in Forecasting Time Series". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 3285–3292. DOI: 10.1109/BigData47090.2019.9005997.

[49] Urminder Singh et al. "Ensemble of deep long short term memory networks for labelling origin of replication sequences". In: Oct. 2015, pp. 1–7. DOI: 10.1109/DSAA.2015.7344871.

[50] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. DOI: https://doi.org/10.48550/arXiv.1206.2944. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[51] Lin Song. "Research on Choices of Investment Strategies". In: Jan. 2020. DOI: 10.2991/assehr.k.200316.292.

[52] Tamara Teplova et al. "Black-Litterman Model with Copula-Based Views in Mean-CVaR Portfolio Optimization Framework with Weight Constraints". In: *Economic Change and Restructuring* 56.1 (2023), pp. 515–535. DOI: 10.1007/s10644-022-09435-y. URL: https://doi.org/10.1007/s10644-022-09435-y.

[53] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996). Accessed 19 June. 2025, pp. 267–288. URL: http://www.jstor.org/stable/2346178.

[54] VettaFi. *ETF Database by*. [Online; Last accessed 18-June-2025]. URL: https://etfdb.com/.

[55] Wenmin Wang. "Regression Task". In: *Principles of Machine Learning: The Three Perspectives*. Singapore: Springer Nature Singapore, 2025, pp. 421–447. ISBN: 978-981-97-5333-8. DOI: 10.1007/978-981-97-5333-8_13. URL: https://doi.org/10.1007/978-981-97-5333-8_13.

[56] Eric W. Weisstein. *Statistical Correlation*. From MathWorld–A Wolfram Web Resource. [Online; Last accessed 17-June-2025]. URL: https://mathworld.wolfram.com/StatisticalCorrelation.html.

[57] Wenkai Xiang et al. "Gram matrix: an efficient representation of molecular conformation and learning objective for molecular pretraining". In: *Briefings in Bioinformatics* 25.4 (July 2024), bbae340. ISSN: 1477-4054. DOI: 10.1093/bib/bbae340. eprint: https://academic.oup.com/bib/article-pdf/25/4/bbae340/58507303/bbae340.pdf. URL: https://doi.org/10.1093/bib/bbae340.

[58] Tong Yu and Hong Zhu. "Hyper-Parameter Optimization: A Review of Algorithms and Applications". In: (2020). arXiv: 2003.05689 [cs.LG].

[59] Miao Zou et al. "Optimized XGBoost Model with Small Dataset for Predicting Relative Density of Ti-6Al-4V Parts Manufactured by Selective Laser Melting". In: *Materials* 15.15 (2022). ISSN: 1996-1944. DOI: 10.3390/ma15155298. URL: https://www.mdpi.com/1996-1944/15/15/5298.

Appendix

Tools and Libraries

Library/Tool	Description / Components
Python 3.11.6	Core language for scripting and automation.
Pandas, NumPy, SciPy	Data manipulation and numerical computing.
yfinance	Download market data from Yahoo Finance.
statsmodels	OLS statistical modeling.
scikit-learn	ML models and preprocessing: MinMaxScaler, PolynomialFeatures, TruncatedSVD Lasso, RandomForestRegressor mean_squared_error, mean_absolute_error
skopt	Bayesian hyperparameter optimization: BayesSearchCV, Integer, Real, Categorical
xgboost	Scalable gradient boosting implementation.
tensorflow.keras	LSTM-based deep learning: Sequential, LSTM, Dropout, Adam, EarlyStopping
matplotlib	Visualization of forecasts and performance metrics.
datetime, statistics	Time manipulation and statistical calculations.

Key Tools and Libraries Used

Code Implementation Walkthrough

Data Acquisition and Caching Process

Historical financial data was programmatically acquired using the `yfinance` Python package, which interfaces with the Yahoo Finance API. A custom function was implemented to automate this process, enabling the download of adjusted closing prices for selected tickers from January 1, 2000, to January 10, 2025. To minimize redundant downloads and reduce API calls, the function checks for a locally cached `.csv` file before fetching new data. If the file does not exist, it proceeds to download the data, removes unnecessary columns (High, Low, Open, Volume), and saves the cleaned dataset to a designated directory (`data/etfs/`) for further analysis.

The complete data acquisition and caching workflow is implemented in the Jupyter notebook `Data_Download_and_Caching.ipynb`, which programmatically handles historical data retrieval, local storage validation, and file-based persistence for reproducible experiments.

Save and load model artifacts for reproducibility

Model artifacts are organized hierarchically under `Model_artifacts/`, with separate folders for each model type (i.e., `OLS`, `LASSO`, `RF`, `BT`, `LSTM`). Within each model folder, artifacts are further divided by year (i.e., `2022`, `2024`). Each year-specific directory contains subfolders for key components such as `best_hyper_params` (optimized parameters per ETF), `feature_cols`, and various scalers (`feature_scalers`, `target_scalers`, `polynomial_scalers`, `svd_scalers`) used during preprocessing. The LSTM model will consist only of `feature_scalers` and `target_scalers`. This structured layout ensures clarity, reproducibility, and ease of access for model training and evaluation.

Reproducibilty guide for loading and inferencing from the model

The codebase implements two distinct workflows: one for training models from scratch and saving the resulting artifacts, and another for loading the saved artifacts for inference and evaluation. It is recommended to use the load-only workflow for reproducibility. This approach ensures consistency in results and is explicitly guided in the accompanying Jupyter notebooks.

Other Tables

W	Without Smoothing					With Smoothing				
	OLS	LASSO	RF	BT	LSTM	OLS	LASSO	RF	BT	LSTM
1	IGM, PSI	SMH, PSI	VGT, PSI	VGT, XSD	PSI, VGT	IGM, PSI	SMH, PSI	VGT, PSI	VGT, XSD	PSI, VGT
2	IGM, QQQ	XSD, VGT	IYW, SMH	XLK, IGM	PSI, IXN	IGM, PSI	SMH, PSI	VGT, SMH	XSD, VGT	PSI, VGT
3	VGT, XLK	SMH, PSI	IXN, XLK	QQQ, IYW	PSI, VGT	IGM, PSI	SMH, PSI	VGT, SMH	XSD, IYW	PSI, VGT
4	IGM, SOXX	IYW, XSD	PSI, QQQ	PSI, SOXX	PSI, VGT	IGM, PSI	SMH, PSI	VGT, SMH	XSD, XLK	PSI, VGT
5	PSI, XSD	SMH, PSI	SMH, PSI	XSD, SOXX	PSI, VGT	IGM, PSI	SMH, PSI	VGT, SMH	XSD, SOXX	PSI, VGT
6	IGM, PSI	IYW, VGT	XSD, SOXX	PSI, QQQ	PSI, VGT	IGM, PSI	SMH, PSI	VGT, SMH	XSD, SOXX	PSI, VGT
7	SMH, IGM	SMH, QQQ	XLK, IYW	XLK, PSI	PSI, VGT	IGM, PSI	SMH, PSI	VGT, SMH	XSD, XLK	PSI, VGT
8	SMH, PSI	QQQ, SMH	XLK, QQQ	PSI, IGM	PSI, VGT	IGM, PSI	SMH, PSI	SMH, VGT	XSD, PSI	PSI, VGT
9	PSI, SOXX	VGT, PSI	PSI, IYW	QQQ, PSI	PSI, VGT	IGM, PSI	SMH, PSI	VGT, PSI	XSD, PSI	PSI, VGT
10	SOXX, XLK	PSI, XSD	IYW, SOXX	IGM, SOXX	PSI, VGT	PSI, IGM	SMH, PSI	PSI, VGT	XSD, PSI	PSI, VGT
11	IXN, PSI	SOXX, IYW	IXN, IYW	XLK, QQQ	PSI, VGT	PSI, SOXX	SMH, PSI	PSI, VGT	XLK, XSD	PSI, VGT
12	SOXX, IYW	XLK, PSI	PSI, XLK	XLK, QQQ	PSI, VGT	PSI, SOXX	PSI, SMH	PSI, VGT	XLK, PSI	PSI, VGT
13	XSD, XLK	PSI, QQQ	PSI, SOXX	SOXX, PSI	PSI, VGT	PSI, SOXX	PSI, SMH	PSI, VGT	PSI, XLK	PSI, VGT
14	SMH, QQQ	SMH, XLK	VGT, IXN	IXN, SOXX	VGT, PSI	PSI, SOXX	SMH, PSI	PSI, VGT	PSI, XLK	PSI, VGT
15	QQQ, IXN	SMH, IXN	PSI, IGM	QQQ, VGT	VGT, PSI	PSI, SOXX	SMH, PSI	PSI, VGT	PSI, QQQ	PSI, VGT
16	SOXX, IGM	SMH, VGT	QQQ, SOXX	QQQ, IYW	PSI, VGT	SOXX, PSI	SMH, PSI	PSI, VGT	QQQ, PSI	PSI, VGT
17	VGT, XLK	PSI, IYW	XLK, IGM	QQQ, PSI	PSI, VGT	SOXX, PSI	SMH, PSI	PSI, SMH	QQQ, PSI	PSI, VGT
18	IYW, PSI	VGT, SOXX	XSD, SOXX	IXN, QQQ	PSI, VGT	PSI, SOXX	SMH, PSI	PSI, SMH	QQQ, PSI	PSI, VGT
19	IGM, QQQ	SMH, SOXX	IXN, SOXX	PSI, VGT	PSI, VGT	SOXX, PSI	SMH, PSI	PSI, SMH	QQQ, PSI	PSI, VGT
20	SOXX, XSD	VGT, PSI	XSD, IGM	XSD, PSI	PSI, VGT	SOXX, PSI	SMH, PSI	PSI, VGT	QQQ, PSI	PSI, VGT
21	PSI, SOXX	IYW, XLK	IGM, XSD	IYW, PSI	PSI, VGT	SOXX, PSI	SMH, VGT	PSI, XSD	QQQ, PSI	PSI, VGT
22	SMH, VGT	VGT, XSD	PSI, IYW	QQQ, IGM	PSI, VGT	SOXX, PSI	SMH, VGT	PSI, XSD	QQQ, PSI	PSI, VGT
23	IYW, SOXX	IXN, VGT	VGT, XSD	IGM, QQQ	PSI, XLK	SOXX, SMH	SMH, VGT	PSI, XSD	QQQ, PSI	PSI, VGT
24	VGT, SOXX	XSD, VGT	SOXX, XSD	XLK, XSD	PSI, XLK	SOXX, VGT	SMH, VGT	PSI, XSD	QQQ, PSI	PSI, VGT
25	IXN, XSD	IXN, XLK	XSD, IXN	PSI, QQQ	PSI, VGT	SOXX, VGT	SMH, VGT	XSD, PSI	QQQ, PSI	PSI, VGT
26	XSD, VGT	IYW, IXN	IXN, PSI	QQQ, IGM	PSI, VGT	VGT, XSD	SMH, VGT	PSI, IXN	QQQ, PSI	PSI, VGT
27	IXN, SOXX	XSD, SMH	VGT, QQQ	QQQ, IYW	PSI, VGT	XSD, SOXX	SMH, VGT	XSD, PSI	QQQ, PSI	PSI, VGT
28	SOXX, IXN	IYW, SOXX	XSD, IGM	SOXX, QQQ	PSI, VGT	SOXX, IXN	SMH, VGT	XSD, VGT	QQQ, PSI	PSI, VGT
29	XLK, IXN	SOXX, QQQ	IGM, IYW	QQQ, PSI	PSI, VGT	IXN, SOXX	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
30	XSD, VGT	SOXX, QQQ	QQQ, IXN	QQQ, VGT	PSI, VGT	XSD, SOXX	SOXX, SMH	XSD, SOXX	QQQ, PSI	PSI, VGT
31	IGM, VGT	XSD, IYW	VGT, XLK	VGT, QQQ	PSI, VGT	SOXX, VGT	SOXX, SMH	XSD, VGT	QQQ, VGT	PSI, VGT
32	PSI, VGT	IXN, SMH	SMH, XLK	IGM, QQQ	VGT, PSI	VGT, IXN	SOXX, SMH	XSD, VGT	QQQ, PSI	PSI, VGT
33	XSD, SOXX	XSD, IYW	XSD, IGM	QQQ, IGM	PSI, VGT	VGT, IXN	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
34	IYW, IXN	QQQ, PSI	XSD, VGT	QQQ, IYW	PSI, VGT	VGT, IXN	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
35	QQQ, XSD	PSI, SOXX	IYW, XSD	PSI, XSD	PSI, VGT	VGT, XSD	SOXX, SMH	XSD, VGT	QQQ, PSI	PSI, VGT
36	SOXX, VGT	SMH, SOXX	IXN, SOXX	PSI, QQQ	VGT, PSI	VGT, SOXX	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
37	SMH, XLK	IYW, IXN	IGM, SMH	VGT, PSI	PSI, VGT	VGT, XSD	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
38	SMH, VGT	IYW, VGT	SMH, XLK	QQQ, IXN	PSI, VGT	VGT, XSD	SMH, SOXX	XSD, SMH	QQQ, PSI	PSI, VGT
39	PSI, VGT	VGT, XSD	SMH, VGT	IYW, PSI	PSI, VGT	VGT, XSD	SMH, SOXX	XSD, SMH	QQQ, PSI	PSI, VGT
40	SMH, QQQ	SMH, XSD	PSI, VGT	SMH, IGM	VGT, PSI	VGT, XLK	SMH, XSD	XSD, SMH	QQQ, PSI	PSI, VGT
41	VGT, SMH	SMH, IXN	XSD, IXN	SMH, PSI	PSI, VGT	VGT, XLK	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
42	IGM, SOXX	XSD, SOXX	XSD, PSI	XLK, SMH	PSI, VGT	VGT, SOXX	SMH, XSD	XSD, PSI	QQQ, PSI	PSI, VGT
43	PSI, IGM	SOXX, IGM	XLK, PSI	PSI, QQQ	PSI, VGT	VGT, SOXX	SOXX, SMH	XSD, PSI	QQQ, PSI	PSI, VGT
44	XSD, IGM	IYW, QQQ	IXN, XLK	XSD, IXN	PSI, VGT	VGT, PSI	SOXX, SMH	XSD, PSI	QQQ, PSI	PSI, VGT
45	XLK, XSD	SMH, IYW	VGT, XLK	IXN, QQQ	PSI, VGT	VGT, XSD	SMH, SOXX	XSD, VGT	QQQ, PSI	PSI, VGT
46	VGT, XSD	XSD, IXN	XSD, QQQ	QQQ, SOXX	PSI, VGT	VGT, XSD	SMH, XSD	XSD, XLK	QQQ, PSI	PSI, VGT
47	XLK, SMH	SMH, SOXX	XLK, XSD	QQQ, SOXX	PSI, VGT	VGT, XSD	SMH, SOXX	XSD, XLK	QQQ, PSI	PSI, VGT
48	IYW, SMH	QQQ, SMH	XLK, XSD	VGT, SOXX	PSI, VGT	VGT, XLK	SMH, XSD	XSD, XLK	QQQ, PSI	PSI, VGT
49	SMH, XSD	IYW, IGM	XSD, VGT	SOXX, XLK	PSI, VGT	XSD, VGT	SMH, XSD	XSD, XLK	QQQ, SOXX	PSI, VGT
50	PSI, IYW	XLK, IGM	VGT, XLK	VGT, PSI	PSI, VGT	XSD, PSI	SMH, XSD	XSD, XLK	QQQ, PSI	PSI, VGT
51	PSI, IGM	QQQ, SMH	XSD, QQQ	SOXX, IYW	PSI, VGT	PSI, XSD	SMH, XSD	XSD, XLK	QQQ, SOXX	PSI, VGT
52	PSI, QQQ	QQQ, PSI	XSD, XLK	QQQ, SOXX	PSI, VGT	PSI, XSD	SMH, XSD	XSD, XLK	QQQ, SOXX	PSI, VGT

Final Portfolio Selection per Week for 2024: With and Without Smoothing

W	Without Smoothing					With Smoothing				
	OLS	LASSO	RF	BT	LSTM	OLS	LASSO	RF	BT	LSTM
1	SOXX, VGT	VGT, SOXX	PSI, IGM	QQQ, SOXX	VGT, PSI	SOXX, VGT	VGT, SOXX	PSI, IGM	QQQ, SOXX	VGT, PSI
2	SOXX, XSD	SMH, PSI	IGM, SOXX	XLK, VGT	VGT, PSI	SOXX, VGT	VGT, SOXX	PSI, IGM	QQQ, SOXX	VGT, PSI
3	XSD, SMH	XLK, IGM	VGT, IYW	XSD, PSI	VGT, SOXX	SOXX, VGT	VGT, SOXX	PSI, IGM	QQQ, SOXX	VGT, PSI
4	PSI, QQQ	QQQ, IXN	PSI, QQQ	IYW, SOXX	VGT, SOXX	SOXX, VGT	VGT, SOXX	PSI, IGM	QQQ, SOXX	VGT, PSI
5	SMH, PSI	SMH, IGM	IXN, IYW	IGM, PSI	VGT, SOXX	SOXX, VGT	VGT, SMH	PSI, IGM	QQQ, SOXX	VGT, PSI
6	IGM, IXN	QQQ, IXN	PSI, IGM	PSI, IGM	VGT, SOXX	SOXX, VGT	VGT, SMH	PSI, IGM	QQQ, SOXX	VGT, PSI
7	QQQ, SOXX	IYW, QQQ	IXN, VGT	PSI, IYW	VGT, SOXX	SOXX, VGT	VGT, SMH	PSI, IGM	QQQ, SOXX	VGT, PSI
8	XSD, PSI	PSI, IXN	SMH, IXN	PSI, XSD	VGT, IGM	SOXX, VGT	VGT, SMH	PSI, IGM	QQQ, SOXX	VGT, PSI
9	IXN, QQQ	IXN, VGT	XSD, XLK	PSI, XLK	VGT, IGM	SOXX, VGT	VGT, SMH	PSI, IGM	SOXX, SMH	VGT, PSI
10	IYW, SOXX	XSD, IXN	IYW, QQQ	VGT, PSI	VGT, IGM	SOXX, VGT	VGT, SMH	PSI, IGM	SOXX, QQQ	VGT, PSI
11	IYW, PSI	XLK, SMH	IGM, XSD	QQQ, PSI	VGT, SOXX	SOXX, PSI	VGT, SMH	PSI, IGM	SMH, QQQ	VGT, PSI
12	XLK, IYW	IYW, IGM	IXN, XSD	IGM, SMH	VGT, IGM	SOXX, PSI	SMH, VGT	PSI, XSD	SMH, QQQ	VGT, PSI
13	IGM, SMH	IYW, PSI	VGT, IXN	IGM, VGT	VGT, IGM	SOXX, PSI	VGT, SMH	PSI, IXN	SMH, SOXX	VGT, PSI
14	XSD, QQQ	QQQ, XSD	IXN, IYW	XSD, SMH	VGT, SOXX	SOXX, PSI	SMH, VGT	PSI, IYW	SMH, SOXX	VGT, PSI
15	SMH, IYW	IGM, SOXX	IYW, QQQ	XSD, SMH	VGT, SOXX	SOXX, PSI	VGT, SMH	PSI, IYW	SMH, SOXX	VGT, PSI
16	SOXX, XSD	VGT, SMH	XSD, VGT	QQQ, SMH	VGT, PSI	SOXX, PSI	SMH, VGT	PSI, VGT	SMH, PSI	VGT, PSI
17	SOXX, IYW	VGT, XLK	VGT, PSI	XSD, PSI	VGT, PSI	SOXX, IYW	VGT, SMH	PSI, IYW	SMH, PSI	VGT, PSI
18	SOXX, IYW	PSI, SMH	PSI, XLK	SMH, PSI	PSI, VGT	SOXX, IYW	SMH, VGT	PSI, IYW	SMH, PSI	VGT, PSI
19	IYW, XLK	SMH, XSD	PSI, XSD	IYW, SOXX	PSI, VGT	SOXX, IYW	VGT, SMH	PSI, XSD	SMH, SOXX	VGT, PSI
20	SOXX, XSD	XSD, PSI	XSD, PSI	SOXX, IXN	VGT, PSI	SOXX, IYW	VGT, SMH	PSI, XSD	SOXX, SMH	VGT, PSI
21	VGT, XLK	VGT, IGM	PSI, SOXX	SOXX, XSD	VGT, PSI	SOXX, IYW	VGT, SMH	PSI, XSD	SMH, SOXX	VGT, PSI
22	SMH, VGT	PSI, XSD	QQQ, PSI	SMH, IYW	VGT, PSI	SOXX, IYW	SMH, VGT	PSI, XSD	SOXX, SMH	VGT, PSI
23	SOXX, XSD	PSI, XSD	PSI, XSD	XLK, IYW	VGT, PSI	SOXX, IYW	SMH, VGT	PSI, XSD	SMH, SOXX	VGT, PSI
24	VGT, XSD	PSI, IYW	IYW, QQQ	IXN, IYW	PSI, VGT	SOXX, IYW	VGT, SMH	PSI, XSD	SOXX, SMH	VGT, PSI
25	SOXX, XLK	IXN, IYW	XLK, XSD	PSI, SOXX	VGT, PSI	SOXX, IYW	SMH, VGT	PSI, XSD	SOXX, SMH	VGT, PSI
26	SOXX, VGT	IXN, IYW	PSI, SOXX	SOXX, PSI	VGT, PSI	SOXX, IYW	PSI, SMH	PSI, XSD	SOXX, SMH	VGT, PSI
27	IYW, XLK	IXN, SMH	XSD, IXN	SMH, IYW	VGT, PSI	SOXX, IYW	PSI, XSD	PSI, XSD	SMH, SOXX	VGT, PSI
28	SOXX, QQQ	IGM, IXN	IXN, SMH	VGT, SMH	VGT, PSI	SOXX, IYW	SMH, VGT	PSI, XSD	SMH, SOXX	VGT, PSI
29	PSI, SMH	SOXX, IGM	PSI, XSD	QQQ, IYW	VGT, PSI	SOXX, PSI	PSI, SMH	XSD, PSI	SOXX, SMH	VGT, PSI
30	PSI, IGM	SOXX, XLK	XSD, IYW	SOXX, XSD	VGT, IGM	SOXX, PSI	PSI, XSD	XSD, PSI	SOXX, SMH	VGT, PSI
31	PSI, IGM	IXN, IYW	XSD, PSI	IGM, VGT	VGT, IGM	PSI, SOXX	PSI, IYW	XSD, PSI	SOXX, SMH	VGT, PSI
32	PSI, XSD	IXN, XLK	QQQ, XSD	XLK, SMH	VGT, IGM	SOXX, PSI	PSI, XSD	XSD, PSI	SOXX, SMH	VGT, PSI
33	VGT, SOXX	XSD, IXN	SMH, XSD	SOXX, IXN	VGT, PSI	PSI, SOXX	PSI, IYW	XSD, PSI	SOXX, SMH	VGT, PSI
34	IYW, SOXX	IYW, SMH	XLK, PSI	SOXX, IXN	VGT, IGM	SOXX, PSI	PSI, XSD	XSD, PSI	SOXX, SMH	VGT, PSI
35	IYW, SOXX	IYW, PSI	SMH, QQQ	XSD, SMH	VGT, IGM	SOXX, IYW	PSI, SOXX	XSD, PSI	SMH, SOXX	VGT, PSI
36	XSD, IYW	XLK, XSD	IXN, XSD	XLK, SMH	VGT, IGM	SOXX, IYW	SOXX, PSI	XSD, PSI	SOXX, SMH	VGT, PSI
37	XSD, PSI	QQQ, XSD	VGT, PSI	XLK, QQQ	VGT, IGM	IYW, SOXX	PSI, SOXX	PSI, XSD	SOXX, XSD	VGT, PSI
38	XSD, IGM	XSD, IYW	XLK, PSI	IGM, SOXX	VGT, PSI	IYW, XSD	SOXX, PSI	PSI, XSD	SOXX, XSD	VGT, PSI
39	XSD, XLK	IXN, VGT	VGT, IXN	VGT, IGM	VGT, IGM	XSD, IYW	IXN, SOXX	PSI, XSD	SOXX, XSD	VGT, PSI
40	SOXX, SMH	SMH, IYW	IXN, VGT	PSI, SOXX	VGT, IGM	XSD, PSI	IXN, SOXX	PSI, XSD	SOXX, XSD	VGT, PSI
41	SMH, XSD	SMH, VGT	PSI, XSD	IGM, QQQ	PSI, IYW	XSD, SOXX	IXN, SOXX	PSI, XSD	SOXX, XSD	VGT, PSI
42	SOXX, IGM	IGM, SOXX	SMH, IYW	VGT, SOXX	IGM, PSI	XSD, SOXX	IXN, IYW	PSI, XSD	XSD, SOXX	VGT, PSI
43	SOXX, VGT	IGM, SOXX	IGM, IYW	XSD, IGM	VGT, PSI	SOXX, XSD	IYW, IXN	PSI, XSD	XSD, SOXX	VGT, PSI
44	IGM, SOXX	XSD, IXN	XSD, XLK	XSD, SOXX	VGT, PSI	SOXX, IGM	IXN, IYW	PSI, XSD	XSD, SOXX	VGT, PSI
45	XSD, SOXX	IGM, QQQ	IGM, QQQ	SMH, SOXX	VGT, PSI	SOXX, XSD	XSD, IYW	PSI, XSD	XSD, SOXX	VGT, PSI
46	SOXX, IYW	PSI, QQQ	XSD, XLK	VGT, QQQ	VGT, PSI	SOXX, IGM	XSD, IYW	XSD, PSI	XSD, SOXX	VGT, PSI
47	SOXX, XLK	IGM, QQQ	XSD, VGT	XLK, XSD	VGT, PSI	SOXX, IGM	IYW, IXN	PSI, XSD	XSD, SOXX	VGT, PSI
48	IYW, PSI	SOXX, IGM	PSI, IGM	QQQ, SMH	VGT, PSI	SOXX, IGM	IYW, PSI	PSI, XSD	XSD, SOXX	VGT, PSI
49	PSI, XSD	IYW, XSD	VGT, IXN	IXN, QQQ	VGT, PSI	SOXX, XSD	IXN, IYW	PSI, XSD	XSD, SOXX	VGT, PSI
50	PSI, XSD	XSD, IGM	QQQ, IGM	XSD, SMH	VGT, PSI	SOXX, XSD	IYW, IXN	XSD, PSI	XSD, IXN	VGT, PSI
51	PSI, IYW	IYW, XSD	XSD, IGM	IXN, XSD	VGT, PSI	SOXX, PSI	IXN, IYW	XSD, PSI	XSD, IXN	VGT, PSI

Final Portfolio Selection per Week for 2022: With and Without Smoothing