Trading Strategy for Equity Market and F&O

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Devadharshini.T

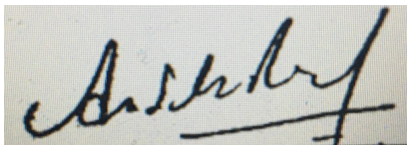
Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2025

Supervisor: Dr.Aniruddha Pant

Certificate

This is to certify that this dissertation entitled **Trading Strategy for Equity Market and F&O** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by **Devadharshini.T** at Indian Institute of Science Education and Research under the supervision of **Dr.Aniruddha Pant**, CEO, Algoanalytics Private Limited , during the academic year 2024-2025.



Dr.Aniruddha Pant

Committee:

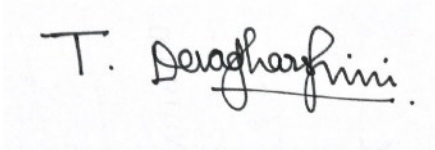
Dr.Aniruddha Pant

Prof.Anindya Goswami

This thesis is dedicated to my family and friends

Declaration

I hereby declare that the matter embodied in the report entitled **Trading Strategy for Equity Market and F&O**, are the results of the work carried out by me at the AlgoAnalytics Private Limited, Pune, under the supervision of Dr.Aniruddha Pant, and the same has not been submitted elsewhere for any other degree.



Devadharshini.T

Acknowledgments

I would like to express my sincere gratitude to my project supervisor, Dr.Aniruddha Pant for giving me the opportunity to work under his guidance. His insightful suggestions, encouragement, and constant support throughout the course of this project have been immensely valuable.

I am also deeply thankful to my project expert Dr.Anindya Goswami for his expert insights and valuable feedback, which helped me refine my work and develop a better understanding of the subject.

A special thanks to my mentor, Mr.Garv Anand, for his consistent guidance and support during our weekly meetings. His clarity of thought and constructive suggestions were instrumental in helping me stay focused and motivated throughout the project.

I would also like to thank my fellow interns for their support and teamwork, which made this experience more meaningful and enjoyable. Their friendship and cooperation played an important role in overcoming challenges and exchanging ideas. Finally, I am deeply grateful to my friends and family for their unwavering moral support and encouragement during the entire project duration. Their belief in me kept me going, even during the more difficult phases of the work.

Abstract

Financial markets are complex, dynamic systems influenced by numerous macroeconomic and microeconomic factors. The ability to predict stock price movements and design effective trading strategies is a long-standing challenge for traders, investors, and researchers. With the advent of machine learning (ML) and deep learning (DL) methodologies, data-driven approaches have gained prominence in forecasting stock trends and optimizing trading strategies.

This thesis investigates the application of 1-Dimensional Convolutional Neural Networks (1-D CNNs) and eXtreme Gradient Boosting (XGBoost) in trading strategies within the equity market and the futures & options (F&O) segment. The study leverages a rolling window approach, analyzing 22 key financial features from both Indian and Foreign Market indices to improve predictive performance. The research explores various hyperparameter configurations, filter sizes, activation functions, and rolling window sizes to optimize model performance.

Our findings indicate that while 1-D CNNs are effective in capturing sequential dependencies in financial time series data, they often struggle with generalization due to overfitting on training data. On the other hand, XGBoost demonstrates robustness by effectively handling structured financial datasets, although hyperparameter tuning remains a key challenge. Additionally, the Fama-French Three-Factor Model (FF3) is explored as a potential sector rotation strategy, offering insights into factor-based investing in the Indian market. The results suggest that combining deep learning models with ensemble learning techniques could enhance stock market prediction accuracy. Future research can focus on integrating hybrid architectures, cost-sensitive loss functions, and alternative feature engineering strategies to further refine predictive models. This work contributes to the growing literature on AI-driven trading strategies and provides a foundation for deploying automated financial models in real-world market scenarios.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Financial markets are complex, dynamic systems shaped by different macroeconomic and microeconomic factors, such as interest rates, inflation, global economic trends, and geopolitical events. Investors and traders have long sought reliable methods to predict stock price movements and develop profitable trading strategies. Traditional financial models, such as fundamental and technical analysis, have been used extensively to understand market behavior. However, these conventional approaches often struggle to capture the nonlinear and stochastic nature of stock price movements, leading to inconsistent predictive performance.

Financial market analysis has been transformed by the rise of artificial intelligence (AI) and machine learning (ML). With the help of historical market data, machine learning algorithms may find hidden patterns, identify irregularities, and produce predictive insights with little human intervention.Deep learning models,such as 1-Dimensional Convolutional Neural Networks (1-D CNNs), have become increasingly popular for time-series forecasting because they effectively identify patterns and trends in stock price movements over time. Similarly, ensemble learning models such as eXtreme Gradient Boosting (XGBoost) have shown strong predictive capabilities in structured financial datasets.
Although significant progress has been made, several challenges still exist:high-dimensional financial data, excessive noise, overfitting in ML models, and the lack of adaptability to evolving market conditions. Addressing these issues is essential for developing a robust and

reliable trading strategy that performs well across diverse market environments. This thesis explores the potential of 1-D CNNs, XGBoost, and the Fama-French Three-Factor Model (FF3) in building an effective trading strategy for the Equity and F&O (Futures & Options) markets.

1.2 Problem Statement

Although machine learning has significantly improved financial forecasting, designing an optimal trading strategy remains challenging due to:

- Market Complexity : Stock prices are influenced by multiple economic, political, and psychological factors, making predictions inherently uncertain.
- Noisy and High-Dimensional Data: Market data consists of numerous indicators, many of which may be irrelevant or redundant, leading to overfitting in ML models.
- Hyperparameter Optimization: Finding the right combination of parameters for deep learning models like CNNs and ensemble methods like XGBoost requires extensive tuning.
- Generalization Issue: While many ML models perform well on training data ,they often fail to generalize effectively to unseen market conditions.

This thesis explores these issues by addressing the following research questions:

- Can deep learning models such as **1-D CNNs** improve financial market predictions compared to traditional models?
- How does **XGBoost**, an ensemble-based technique, compare with deep learning in trading strategy development?
- How can factor-based investing, such as the **Fama-French Three-Factor Model**,help improve market predictions?

By evaluating and comparing these models, the study aims to develop a trading framework that balances predictive accuracy, computational efficiency, and adaptability to changing market dynamics.

1.3 Models and Tools Overview:

Over the past decade, researchers have explored deep learning and machine learning in financial forecasting. Various models have been studied, each offering unique advantages and challenges.

1.3.1 Machine Learning Models in Stock Market Prediction

- **1-D Convolutional Neural Networks (1-D CNNs):** CNNs are primarily designed for image processing; however, their ability to extract hierarchical features has led to their adaptation for time-series applications. One-dimensional CNNs provide computational advantages over LSTMs and efficiently detect localized trends in stock price movements.
- **XGBoost and Tree-Based Models:** Decision trees, Random Forest, and XGBoost have been widely used for structured financial data. XGBoost is particularly powerful due to its ability to handle missing values, avoid overfitting through regularization, and efficiently process large datasets.
- **Fama-French Three-Factor Model (FF3)**: The FF3 model expands the Capital Asset Pricing Model (CAPM) by incorporating size and value factors in addition to market risk. The model has been widely used in empirical asset pricing and portfolio management strategies.

1.3.2 Challenges in Developing Robust Financial ML Models

- **Overfitting and Poor Generalization:**Many machine learning algorithms perform well on historical data but fall short in evolving market situations.

- **Feature Selection Complexity:** Identifying the most relevant financial indicators from a large pool of technical, fundamental, and macroeconomic features is an ongoing challenge.

- **Hyperparameter Optimization:** Selecting the right number of filters, kernel sizes, and learning rates for CNNs and the optimal depth, learning rate, and boosting rounds for XGBoost requires systematic tuning.

This thesis investigates a hybrid methodology that integrates factor-based models, ensemble learning, and deep learning to develop a robust trading strategy in response to these challenges.

1.4 Methodology and Scope

This research investigates the predictive capabilities of the Fama-French Three-Factor Model, XGBoost, and 1-D CNNs in financial markets. The methodology is structured as follows:

1.4.1 Data Collection and Preprocessing

The input dataset consists of 22 financial features, which includes:

- **Indian Market Indicators:** Nifty futures return, volume, volatility, open interest change, high regime probability, etc.

- **Foreign Market Indicators:** S& P 500, DAX, FTSE 100, KOSPI, NYSE Composite, etc.

1.4.2 Model Training and Validation

- **Rolling Window Approach:** Models are trained using a 50-day rolling window, where past 50 days' data is used to predict the next day's price movement.

- **Hyperparameter Tuning:** Optimization of filter sizes, batch sizes, dropout rates (for CNNs), and depth, learning rate, and boosting rounds (for XGBoost).

1.4.3 Performance Evaluation

- Models are evaluated based on:

- Accuracy and Loss Values: To measure predictive performance.
- Robustness and Generalization: Performance consistency across different time periods.

The study also explores the Fama-French Model for sector rotation strategies, analyzing how factor-based investing influences stock market prediction.

1.5 Contribution and Significance

This research contributes to the intersection of AI and finance by:

1. Developing an AI-driven trading strategy: Applying 1-D CNNs and XGBoost to real-world stock market data.

2. Comparing deep learning and ensemble learning: Identifying the strengths and weaknesses of each approach.

3. Enhancing factor-based investing strategies: Evaluating the effectiveness of Fama-French Three-Factor Model in the Indian market.

4. Providing practical insights: Offering recommendations for automated trading systems and portfolio optimization.

The findings of this study will aid traders, financial analysts, and researchers in designing more reliable, data-driven trading models. The integration of deep learning, ensemble learning, and factor investing could set the foundation for future advancements in algorithmic trading.

Chapter 2

Preliminaries

This section outlines the theoretical and conceptual foundations of the approaches and issues addressed in this thesis. By drawing upon established knowledge in machine learning and related fields, it delves into essential concepts necessary for a comprehensive understanding of the study, incorporating insights from IBM's overview of machine learning.

2.1 Overview of ML and DL

Machine Learning (ML) is a subset of artificial intelligence that enables systems to learn from data and improve performance without explicit programming. In financial applications, ML is widely used for predictive analytics, especially for forecasting stock prices and identifying trading opportunities.

A core advancement within ML is **Deep Learning (DL)**, which employs deep neural networks to model complex, non-linear patterns. DL has demonstrated high accuracy in tasks involving large-scale and sequential data, such as financial time-series.

Within stock market forecasting, ML models primarily serve two objectives: **classification** (e.g., predicting if a stock will rise or fall) and **regression** (e.g., forecasting price levels). Supervised learning techniques dominate, utilizing historical stock data to train models that recognize trends and patterns.

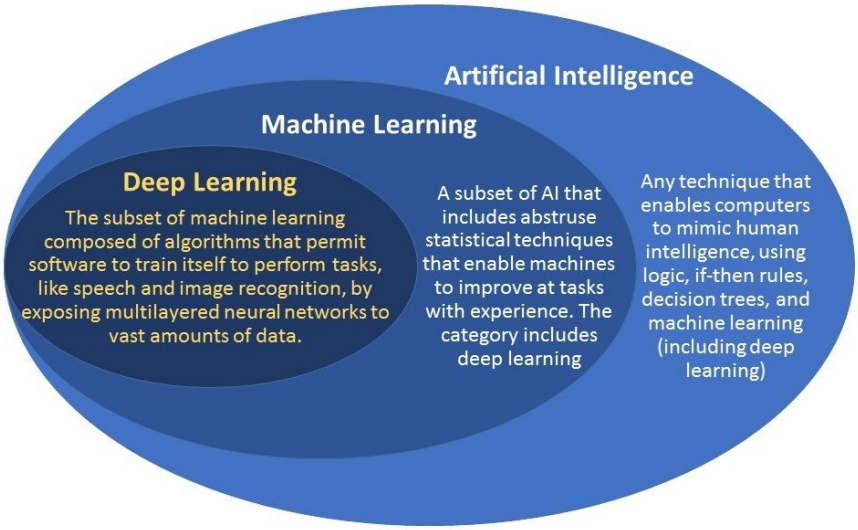


Figure 2.1: Relationship between AI,Machine Learning, and Deep Learning

Common ML algorithms for binary classification include:

- **Logistic Regression** – a statistical model used for binary outcomes;
- **Support Vector Machines (SVM)** – effective for high-dimensional data;
- **XGBoost** – an ensemble tree-based model well-suited for structured data;
- **1D-CNN** – adapted from image processing, capable of capturing sequential patterns in time-series data.

In this study, **XGBoost** and **1D-CNN** were selected for their effectiveness in modeling non-linear dependencies in stock market data. XGBoost provides fast, accurate predictions with strong regularization to prevent overfitting. The 1D-CNN architecture, by convolving over time steps, extracts spatially local but temporally meaningful features directly from price signals.

ML model performance is typically evaluated using classification metrics such as **accuracy**, **precision**, **recall**, and **F1-score**. These indicators help assess the model's ability to distinguish between profitable and non-profitable trade signals, aligning directly with the objectives of short-term trading strategies.

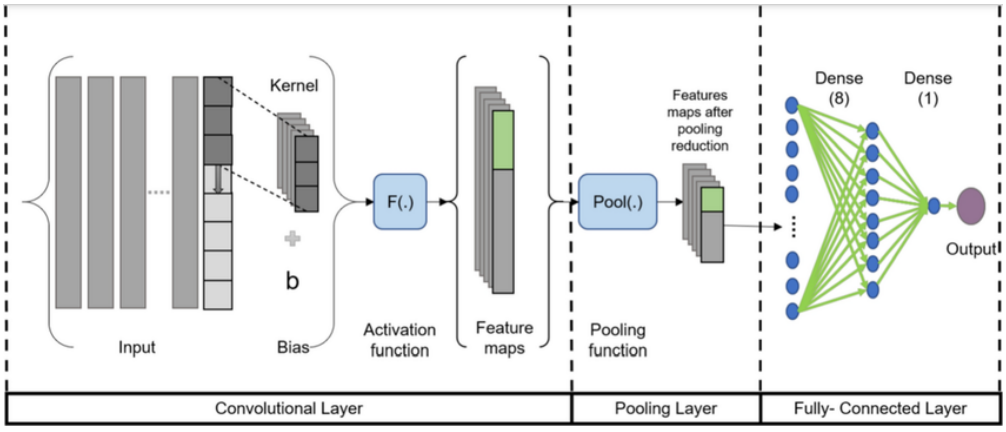


Figure 2.2: Architecture of 1-D CNN

By focusing on relevant models and metrics, this section lays the groundwork for applying ML to binary classification in financial forecasting.

2.1.1 1-Dimensional Convolutional Neural Networks(1-D CNNs):

One-dimensional Convolutional Neural Networks (1-D CNNs) are a specialized type of neural network designed for processing sequential data, including time-series signals. These networks utilize convolutional filters to traverse input sequences, enabling the automatic and adaptive learning of spatial feature hierarchies. This capability makes 1D CNNs particularly effective for applications such as signal processing and natural language processing.

2.1.1.1 Architecture of a 1-D CNN

A standard one-dimensional Convolutional Neural Network (1D CNN) architecture typically comprises multiple layers, each serving a distinct function:

- **Input Layer:** Receives sequential input data, generally formatted as (samples, timesteps, features).
- **Convolutional Layer (Conv1D):** Utilizes convolutional filters to identify local patterns within the input sequence. The filters systematically traverse the sequence, generating feature maps.

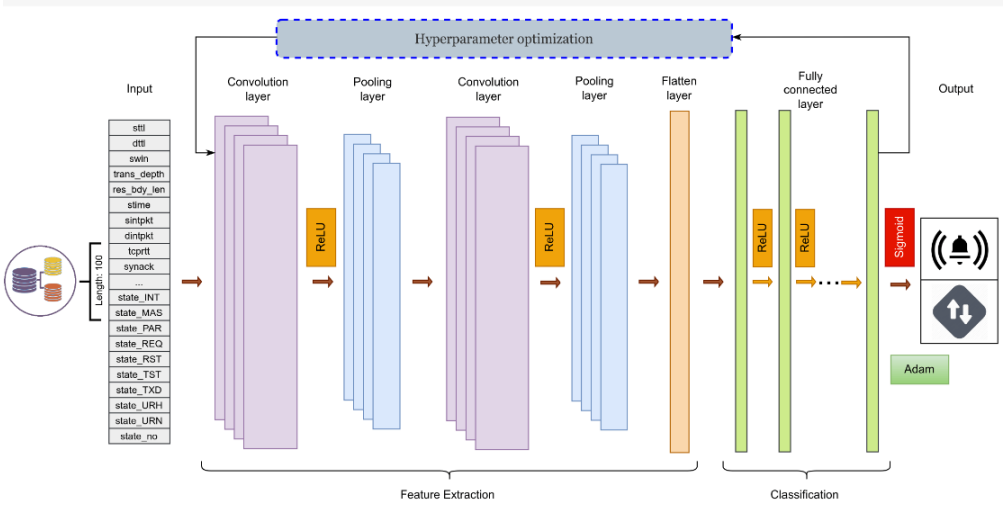


Figure 2.3: 1-D CNN Hyperparameters

- **Activation Function:** Introduces non-linearity, allowing the model to learn intricate relationships within the data. Frequently used activation functions include ReLU, sigmoid, and tanh.

- **Pooling Layer (e.g., MaxPooling1D):** Reduces the dimensionality of feature representations, preserving essential information while minimizing computational complexity.

- **Flattening Layer:** Transforms the reduced feature maps into a one-dimensional vector to facilitate input to subsequent layers.

- **Fully Connected (Dense) Layer:** Executes high-level decision-making and produces the final output predictions.

2.1.1.2 Key Hyperparameters in 1-D CNNs

The essential hyperparameters in a one-dimensional Convolutional Neural Network (1D CNN) include the following:

- **Number of Filters:** Specifies the count of feature maps generated by the convolutional layer.

- **Kernel Size (Filter Size):** Defines the width of the convolutional filters applied to the input sequence.
- **Stride:** Indicates the step size with which the filter moves along the sequence during convolution.
- **Padding:** Determines how the network processes boundary elements, influencing the output dimensions.
- **Activation Function:** Introduces non-linearity into the model to enhance its learning capacity.
- **Pooling Size:** Defines the dimensions of the pooling layer, which helps in downsampling feature maps.
- **Neurons in Fully Connected Layers:** Governs the model's ability to learn complex representations by adjusting the number of neurons in dense layers.
- **Learning Rate:** Controls the magnitude of weight updates during optimization.
- **Optimizer:** Specifies the technique used to modify network parameters for loss minimization.
- **Loss Function:** Quantifies the error between predicted and actual values, guiding model adjustments.
- **Dropout Rate:** A regularization strategy that randomly deactivates a fraction of neurons during training to mitigate overfitting.

2.1.1.3 Adam Optimizer

The Adam optimizer (Adaptive Moment Estimation) is a widely used optimization technique that assigns adaptive learning rates to individual parameters. It integrates the benefits of two established stochastic gradient descent (SGD) extensions: the *momentum method* and *RMSProp*. By leveraging estimates of the first and second moments of gradients, Adam computes parameter-specific learning rates, improving convergence efficiency and stability.

The governing equations of the Adam optimization algorithm are as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.1}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.2}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.3}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.4}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.5}$$

where:

- m_t and v_t denote the first and second moment estimates at iteration t.
- β_1 and β_2 are decay rate parameters that regulate the exponential moving averages of moment estimates.
- g_t represents the gradient of the loss function with respect to the model parameters at time step t.
- α signifies the learning rate.
- ϵ is a small positive constant introduced to prevent numerical instability due to division by zero.

2.1.1.4 Activation Functions

Activation functions play a crucial role in neural networks by introducing non-linearity, which allows the model to capture complex relationships within data. Several commonly used activation functions include:

- **Sigmoid:** The sigmoid function transforms input values into a range between 0 and 1, making it suitable for probability estimation. However, it is prone to the vanishing gradient problem. It is mathematically represented as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

- **Hyperbolic Tangent (Tanh):** Unlike the sigmoid function, tanh maps inputs to a range of -1 to 1, which helps in centering the data and often facilitates faster convergence during training. It is given by:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.7}$$

- **Rectified Linear Unit (ReLU):** ReLU is computationally efficient and helps mitigate the vanishing gradient issue by directly outputting positive inputs, while negative values are set to zero. However, it may lead to inactive neurons, a phenomenon known as the "dying ReLU" problem. It is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{2.8}$$

- **Leaky ReLU:** To address the dying ReLU issue, Leaky ReLU allows a small non-zero gradient for negative inputs, ensuring that neurons remain active throughout training.

- **Parametric ReLU (PReLU):** PReLU extends Leaky ReLU by making the negative slope learnable, enabling the model to determine the optimal level of activation for negative inputs.

- **Exponential Linear Units (ELU):** ELU differs from ReLU by outputting negative values for negative inputs, helping to bring activation means closer to zero, which can enhance learning efficiency and overall performance.

The choice of activation function depends on the neural network architecture and the specific requirements of the given problem.

2.1.1.5 Loss Functions

Loss functions quantify the difference between predicted values and actual target values, thereby guiding the optimization process during model training. Several commonly used loss functions include:

- **Mean Squared Error (MSE):** MSE computes the mean of the squared differences

between predicted and actual values. It is widely applied in regression tasks due to its sensitivity to large errors.

- **Mean Absolute Error (MAE):** MAE determines the mean of the absolute deviations between predictions and actual values. Unlike MSE, it is less sensitive to outliers, making it a robust alternative for certain regression problems.

- **Binary Cross-Entropy:** Binary cross-entropy evaluates the divergence between predicted probabilities and true binary labels. It is commonly used in binary classification scenarios.

  - **Objective:** Measures the accuracy of predicted probabilities against actual binary class labels.
  - **Predicted Output:** The probability estimate ($\hat{y}$) for the positive class.
  - **True Label:** The ground truth value (y), which can be either 0 or 1.
  - **Mathematical Representation:**

$$\text{Loss}(y, \hat{y}) = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right] \tag{2.9}$$

  - **Value Range:** Loss values vary between 0 (perfect predictions) and ∞ (poor predictions).
  - **Effect:** Assigns higher penalties for incorrect predictions, ensuring precise probability estimations.
  - **Application:** Frequently employed in neural networks, particularly when using the sigmoid activation function in the output layer.

- **Categorical Cross-Entropy:** This loss function generalizes binary cross-entropy for multi-class classification problems by comparing predicted probability distributions against actual class labels.

- **Huber Loss:** Huber loss merges the benefits of MSE and MAE by behaving quadratically for minor errors and linearly for significant errors. This characteristic makes it both sensitive to small deviations and resistant to outliers.

2.1.2 Tree-based Models

Tree-based algorithms, including decision trees, divide the feature space into distinct subsets according to feature values. While these models are highly interpretable and easy to understand, they are susceptible to overfitting. To mitigate this issue and enhance predictive accuracy, ensemble techniques such as Random Forests integrate multiple decision trees, thereby improving generalization performance.

2.1.3 Introduction to XGBoost

XGBoost (eXtreme Gradient Boosting) is an advanced and efficient implementation of the gradient boosting framework, extensively utilized in both machine learning competitions and practical applications. It operates as an ensemble learning technique, integrating multiple decision trees to enhance predictive performance. By employing gradient boosting, the model iteratively refines its predictions by minimizing errors from prior iterations. XGBoost is distinguished by its computational efficiency, predictive accuracy, and scalability, making it a preferred choice for various machine learning tasks.Figure 2.4 shows how the XGBoost model is developed from a simple decision tree model.

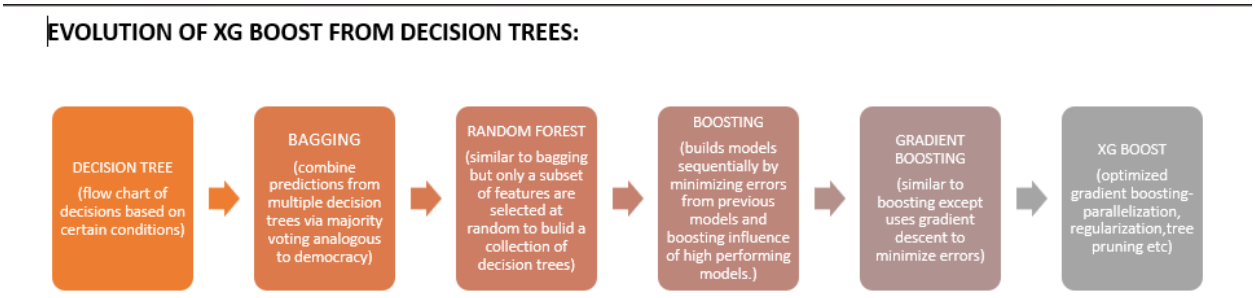


Figure 2.4: Transition from Decision Trees to XGBoost

2.2 Decision Trees vs. Ensemble Methods

Machine learning models make predictions based on patterns found in data. One common approach is using decision trees, which follow a series of logical rules to arrive at a deci-

sion.However, a single decision tree may not always provide reliable results, as it is prone to overfitting the training data. This overfitting can lead to high accuracy on previously seen data while compromising the model's generalization ability to unseen scenarios.

2.2.1 Decision Trees

A decision tree partitions data into smaller subsets by applying specific conditions. The left side of Figure 2.5 presents a basic decision tree model used to determine house purchases based on location and property type. Nevertheless, relying on a single decision tree may result in limited generalization to unseen data.

2.2.2 Random Forest and Ensemble Learning

To improve accuracy and robustness, multiple decision trees can be combined in a method called ensemble learning. The right side of Figure 2.5 illustrates a Random Forest, which is an ensemble learning technique that integrates multiple decision trees to reach a final decision through a voting mechanism. By aggregating the outputs of several trees rather than depending on a single one, this approach minimizes errors and enhances predictive accuracy.

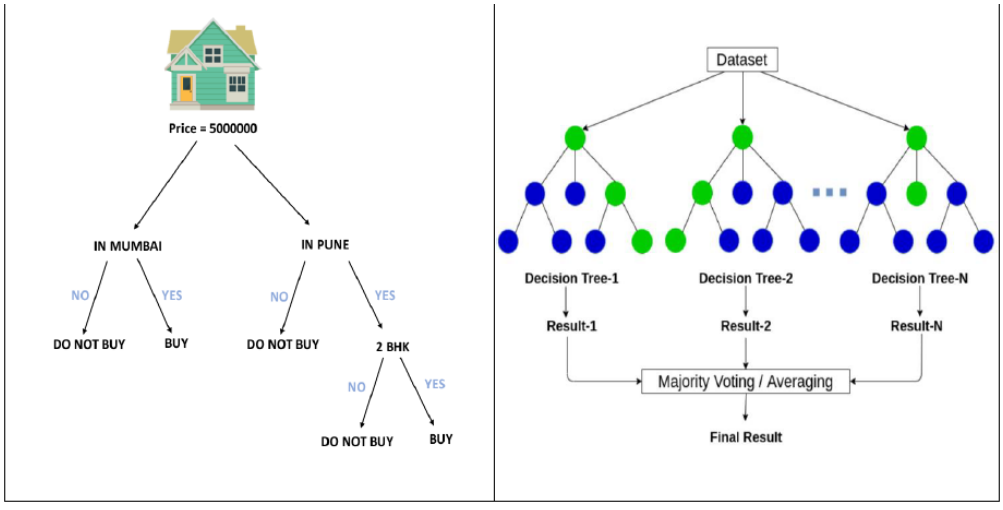


Figure 2.5: A comparison between a single decision tree (left) and an ensemble of decision trees (right).

2.2.3 Ensemble Classifiers

Beyond decision trees, ensemble learning applies to other types of models. As shown in Figure 2.6, multiple classifiers process data and combine their predictions.Rather than depending on a single predictive model, ensemble techniques integrate multiple diverse models to enhance overall accuracy.

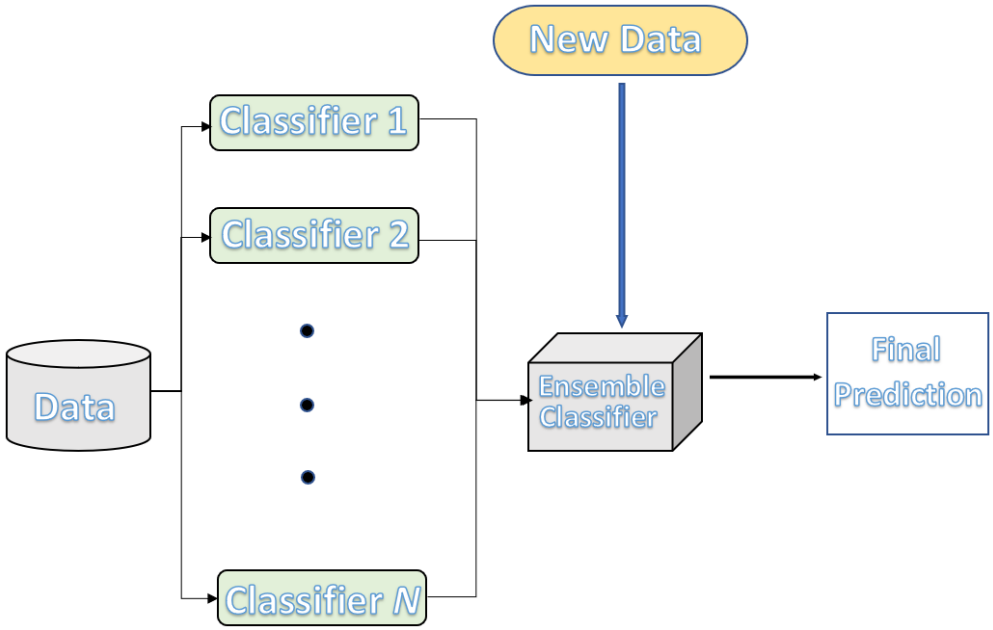


Figure 2.6: An ensemble classifier combining multiple models for better accuracy.

2.2.3.1 How Does XGBoost Work?

- **Boosting Technique:**
  - Boosting builds an ensemble of weak learners and combines them to form a strong predictive model.
  - The final prediction is an aggregate of the predictions from all the trees.
- **Gradient Boosting**
  - XGBoost uses a technique called gradient boosting, where new models (typically decision trees) are added sequentially.

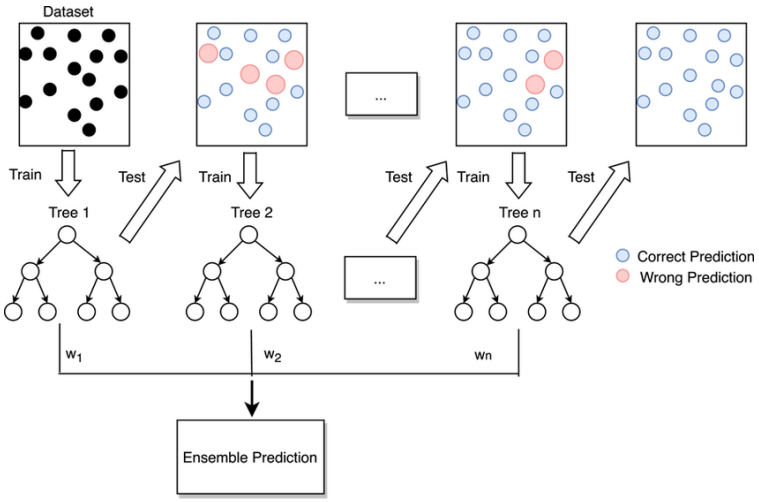


Figure 2.7: The flowchart depicts gradient boosting, where an ensemble of weak classifiers iteratively corrects errors by emphasizing misclassified instances. The final prediction results from a weighted combination of all classifiers' outputs.

- Each new model tries to correct the errors made by the previous model by minimizing a loss function.

2.2.3.2 Optimization and Objective Function

1. **Objective Function**

- The **objective function** comprises two main components:
  - **Loss Function:** This component evaluates the model's effectiveness in fitting the training data. Some widely used loss functions include:

$$\textit{Squared Loss:} L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad \text{(for regression)} \tag{2.10}$$

$$\textit{Logistic Loss:} L(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad \text{(for classification)} \tag{2.11}$$

  - **Regularization Term:** This component penalizes model complexity to mitigate overfitting. It is often achieved through L_1 (Lasso) and L_2 (Ridge) regularization methods:

* L_1 Regularization:

$$R_1(\theta) = \lambda \sum_{j=1}^{p} |\theta_j|$$

* L_2 Regularization:

$$R_2(\theta) = \frac{\lambda}{2} \sum_{j=1}^{p} \theta_j^2$$

2. **Gradient Descent**

- XGBoost minimizes the loss function using **gradient descent**, where parameter updates follow:

$$\theta \leftarrow \theta - \eta \nabla L$$

  Here, η represents the learning rate, while ∇L denotes the gradient of the loss function.

- Additional trees are iteratively incorporated to predict residual errors from prior trees, leveraging first-order (gradients) and second-order (Hessian) derivatives.

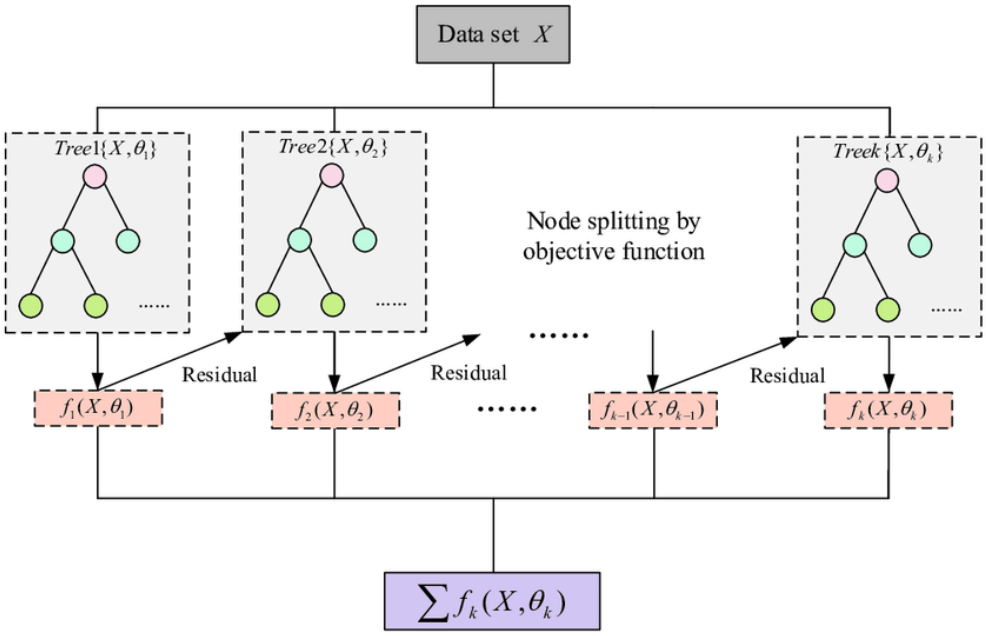


Figure 2.8: The visual representation of XGBoost algorithm as a sequence of decision trees, where each successive tree is constructed to rectify the errors made by its predecessor. The final prediction is obtained by computing a weighted sum of the outputs from all individual trees.

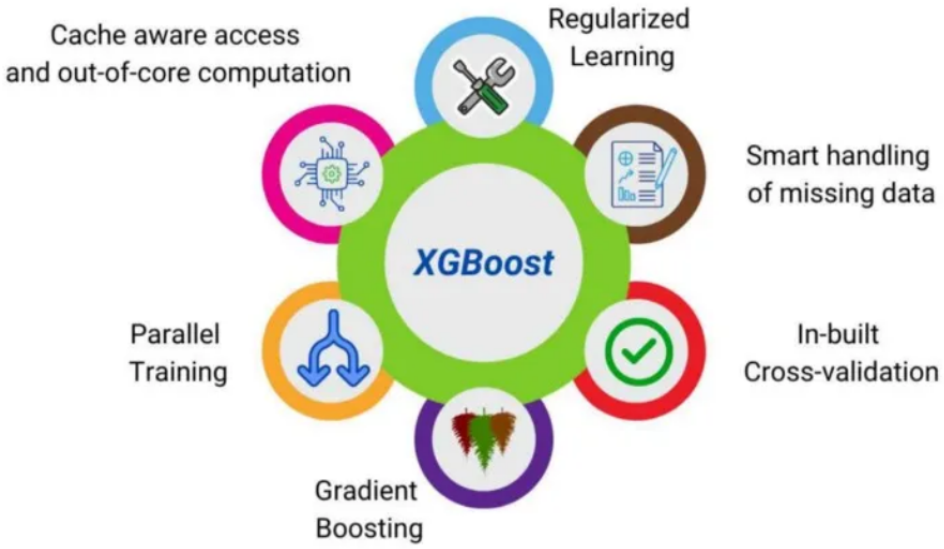


Figure 2.9: Key features of XGBoost

2.3 Fundamentals of XGBoost

2.3.1 XGBoost: Key Features and Overfitting Prevention

XGBoost, short for eXtreme Gradient Boosting, is a highly efficient and scalable machine learning algorithm designed for structured data problems. It incorporates multiple optimization techniques to improve performance, prevent overfitting, and ensure efficient training on large datasets.

2.3.1.1 Key Features of XGBoost

- **Regularization:**
  - Uses both L_1 (Lasso) and L_2 (Ridge) regularization terms in its objective function to control model complexity, enhance generalization, and reduce overfitting.
- **Learning Rate (Shrinkage):**
  - Introduces a learning rate parameter (η) that scales the contribution of each new

tree, ensuring a more conservative and robust model.

- **Level-Wise Tree Growth:**
  - Constructs trees level-by-level (breadth-first), adding nodes simultaneously across all features before progressing to deeper levels. This technique improves computational efficiency while maintaining model complexity control.
- **Parallel and Distributed Computing:**
  - Supports parallel execution, enabling faster training and efficient scalability for large datasets.

2.3.1.2 Overfitting Prevention Techniques in XGBoost

To enhance generalization, XGBoost integrates multiple mechanisms to mitigate overfitting:

- **Regularization:**
  - Both L_1 and L_2 penalties discourage overly complex models, improving predictive stability.
- **Shrinkage (Learning Rate):**
  - A lower learning rate (η) ensures gradual updates, reducing the risk of overfitting by preventing overly aggressive updates from individual trees.
- **Early Stopping:**
  - Automatically halts training if performance on a validation set ceases to improve, ensuring the model does not fit excessively to training data.
- **Pruning:**
  - Trims unimportant branches during tree construction to restrict unnecessary tree depth, thereby reducing model complexity.

2.3.1.3 Tree Structure and Split Evaluation in XGBoost

XGBoost employs a specialized tree-building approach to optimize performance:

- **Level-Wise Growth Strategy:**
  - Builds trees in a breadth-first manner, evaluating and adding splits simultaneously across all features before proceeding deeper. This approach enhances computational efficiency and prevents overgrown trees.
- **Efficient Split Evaluation:**
  - Selects the best splits at each level based on their contribution to minimizing the objective function, ensuring that the most relevant splits are incorporated early.
- **Optimized Feature Utilization:**
  - Avoids redundant re-evaluation of features at the same tree depth, reducing computational overhead while preserving feature interactions.

2.3.2 Hyperparameters in XGBoost

Machine learning models contain two primary categories of parameters: **trainable parameters** and **hyperparameters**. Trainable parameters, such as the weights in neural networks, are adjusted during the training process based on the data. In contrast, hyperparameters are predefined before training and influence the learning process. Selecting appropriate hyperparameters plays a crucial role in optimizing model performance, as it ensures a balance between model complexity and generalization, thereby reducing the risk of overfitting. Moreover, effective hyperparameter tuning enhances computational efficiency by minimizing unnecessary operations and accelerating convergence, ultimately shortening the overall training time.

XGBoost (Extreme Gradient Boosting) is a robust ensemble learning algorithm that constructs decision trees sequentially, refining previous predictions at each step. The effectiveness of XGBoost is highly contingent on well-tuned hyperparameters, as they help

Parameter	Description
Learning Rate (eta)	Modifies how much each tree contributes to the final prediction. Smaller values often lead to more accurate models but require more trees.
max_depth	Controls the depth of each tree to prevent overfitting, essential for managing model complexity
gamma	Determines when to split a node based on the decrease in loss. Higher values make the algorithm more conservative, reducing unnecessary splits.
Subsample	Manages the percentage of data sampled to grow each tree. A low value may lead to underfitting.
Colsample_bytree	Sets the percentage of features sampled for each tree
n_estimators	Specifies the number of boosting rounds (trees).
Lambda (L2) and Alpha (L1)	Control the strength of regularization; higher values lead to stronger regularization.
min_child_weight	Influences tree structure by controlling the minimum data required for a new node.
scale_pos _weight	Adjusts the balance of positive and negative weights in imbalanced class scenarios.

Figure 2.10: Key Hyperparameters of XGBoost

maintain a balance between bias, variance, and computational cost, ensuring optimal model performance. Figure 2.10 illustrates the key hyperparameters in XGBoost:

The optimal selection of hyperparameters can be achieved through techniques such as grid search, random search, and Bayesian optimization. Proper fine-tuning of these parameters enhances the efficiency and effectiveness of XGBoost in addressing real-world challenges.

2.4 Statistical Concepts

This section discusses some important statistical concepts that are used for making financial market decisions.

2.4.1 Regression Analysis

Regression analysis is a foundational technique for modeling relationships between variables, frequently applied in financial research for tasks such as forecasting returns or evaluating economic indicators. Classical methods like linear, multiple, polynomial, and logistic regression have traditionally served as benchmarks for prediction and classification problems.

However, in this thesis, deep learning methods—specifically 1D-CNNs—are employed for binary classification tasks in time series forecasting. Hence, while regression techniques provide useful context, they are not directly implemented in the experimental framework.

2.4.1.1 Application in Financial Markets

Regression analysis plays a crucial role in financial decision-making by identifying relationships between economic indicators and market behavior. Some common applications include:

- **Stock Price Prediction:** Regression models are employed by analysts to predict future stock prices by leveraging historical data, corporate earnings, and macroeconomic indicators.
- **Risk Management:** Financial institutions employ regression models to assess portfolio risks by evaluating factors such as volatility, interest rate changes, and economic shocks.
- **Algorithmic Trading:** Machine learning models in trading strategies often incorporate regression analysis to identify profitable trading signals.

2.4.2 Time Series Analysis

- **Definition and Significance:** Time series analysis focuses on examining data points recorded at consistent time intervals to detect underlying patterns, trends, and periodic variations. This analytical approach is particularly valuable across various domains, such as finance, where interpreting temporal data plays a crucial role in forecasting and strategic decision-making.

- **Key Components of Time Series Data:** A comprehensive understanding of the core components of time series data enhances both its interpretation and modeling. The ability to distinguish and analyze these elements is essential for accurate forecasting. The primary components include:

    1. **Trend:** Indicates the overall direction of data movement over an extended period.
    2. **Seasonality:** Represents recurring patterns that occur within fixed time intervals, typically less than a year.
    3. **Cyclical Patterns:** Encompasses fluctuations that occur over irregular or extended periods, usually exceeding one year.
    4. **Noise:** Denotes random fluctuations or unpredictable variations within the data.

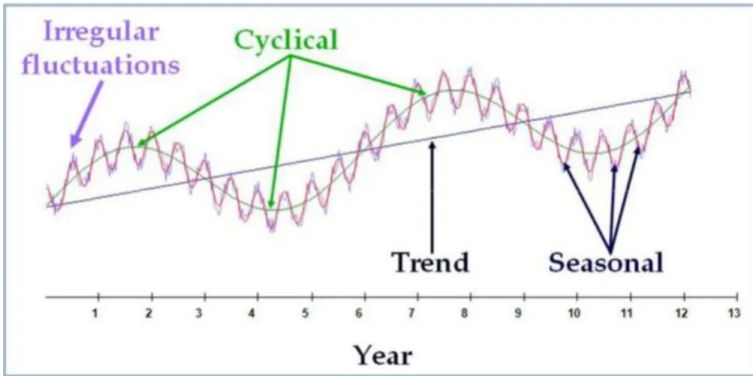


Figure 2.11: Components of Time Series Data

- **Application of Time Series Analysis in Financial Markets:** Financial instruments such as stock prices, indices (e.g., NIFTY), and futures contracts are naturally sequential, making time series analysis vital for identifying market trends and cycles, predicting future price movements based on historical data.

- **Benefits of Time Series Analysis in Finance:** The application of time series analysis in financial markets offers several advantages:

    – **Pattern Recognition:** Machine learning models can capture trends and seasonalities in stock market behavior.

    – **Forecasting Power:** Time series models help predict future stock movements based on past performance, which is crucial in finance.

- **Advancements in Time Series Forecasting with Machine Learning:**

  Traditional time series models, such as the Autoregressive Integrated Moving Average (ARIMA), may be inadequate for handling complex and high-dimensional datasets like NIFTY Futures. In contrast, advanced machine learning and deep learning approaches, including one-dimensional Convolutional Neural Networks (1D CNNs) and Boosting algorithms, have demonstrated enhanced predictive accuracy in financial time series forecasting.

2.5 Fama-French Model

2.5.1 Define the Model and its Factors

The Fama-French Three-Factor Model expands on the Capital Asset Pricing Model (CAPM) by adding size and value factors to the market risk factor. The three factors are:

- **Market Risk (Beta):** Captures the sensitivity of the asset's returns to the overall market returns.

- **Size (SMB - Small Minus Big):** Accounts for the outperformance of small-cap stocks over large-cap stocks.

- **Value (HML - High Minus Low):** Reflects the tendency of value stocks (high book-to-market ratio) to outperform growth stocks (low book-to-market ratio).

2.5.2 Applicability in the Indian Market

Applying the Fama-French model to the Indian market requires consideration of local market conditions, data availability, and investor behavior. Adjustments may be necessary to accurately capture the unique characteristics of the Indian equity market.

2.5.3 Stock Price Forecasting and Trading Strategies

Stock prices are influenced by a complex interplay of market dynamics, economic indicators, and investor sentiment. Traditional methods such as fundamental and technical analysis have long guided investment decisions. However, with increased data availability and computing power, quantitative strategies—especially those leveraging machine learning—have gained prominence.

In this context, **alpha**, defined as the excess return over a benchmark, becomes a key metric. Modern predictive models aim to generate consistent alpha by identifying directional movements in asset prices. Among them, binary classification models, which forecast upward or downward price changes, form the backbone of many algorithmic trading strategies.

Chapter 3

Literature Review

3.1 1-D CNN:

Stock market forecasting has been a critical area of research for decades, attracting scholars and practitioners aiming to develop reliable models for predicting market trends. Traditional models have relied on statistical techniques, while more recent approaches incorporate machine learning and deep learning. This section reviews key literature relevant to stock market forecasting and machine learning applications.

3.1.1 1D-CNN for Financial Forecasting

Kim et al. (1) presented a detailed guide on implementing hybrid models combining 1D-CNN and BiLSTM for time series prediction, focusing on forecasting peak electricity demand and system marginal prices. Their findings suggest that such deep learning architectures are well-suited to capture temporal dependencies in data, offering promising applications in financial forecasting scenarios.

In a comprehensive review, Kiranyaz et al. (2) explored the structure, functionality, and advantages of 1D convolutional neural networks (1D-CNNs), particularly in processing sequential data such as time series. Their work positions 1D-CNNs as efficient alternatives to traditional recurrent architectures, particularly in tasks involving financial data streams.

As deep learning technologies evolve, new methods tailored for time-dependent data have emerged. Although 2D CNNs have proven highly effective in visual recognition tasks, their reliance on large datasets and computational intensity limits their use in one-dimensional contexts. Addressing this limitation, 1D-CNNs offer a streamlined solution optimized for sequential data analysis.

Kiranyaz et al. (2) elaborate on the internal composition of 1D-CNNs, which typically include convolution, activation, and pooling layers integrated with fully connected units. Their design emphasizes computational efficiency, enabling deployment in real-time and low-resource environments. The core strengths of these models lie in their ability to:

- unify feature extraction and classification in an end-to-end framework;
- maintain high performance even with limited labeled data; and
- perform consistently in time-sensitive applications.

These attributes have been effectively demonstrated in multiple domains:

- **ECG monitoring:** facilitating real-time, patient-specific arrhythmia detection;
- **Structural health monitoring:** through vibration-based damage localization;
- **Industrial fault detection:** particularly in motors and power electronics;
- **Speech recognition:** leveraging MFSC or raw signal inputs for acoustic modeling.

On the practical front, Kim et al. (1) offer a real-world implementation of a hybrid 1D-CNN and BiLSTM model. Applied to the task of forecasting electricity demand and marginal pricing for Jeju Island, their framework outlines a clear methodological pipeline:

- **Data Collection:** Involves gathering features like historical consumption, meteorological data, calendar effects, and tourism statistics;
- **Preprocessing:** Encompasses imputation, creation of temporal features (e.g., lags), and categorical transformations;

- **Model Development:** Employs a 1D-CNN layer for spatial extraction followed by a BiLSTM for temporal modeling, finalized with fully connected layers;
- **Evaluation:** Uses performance metrics such as RMSE, MASE, and WRMSSE to validate prediction accuracy.

The hybrid structure exploits CNNs' proficiency in local pattern recognition and BiLSTM's ability to learn long-range dependencies. The model achieved strong results for consistent patterns like electricity demand but showed relatively lower accuracy for more volatile targets such as marginal price.

Together, these contributions affirm the utility of 1D-CNNs in modeling real-world time series. Kiranyaz et al. (2) further point to novel frameworks such as Generalized Operational Perceptrons (GOPs) and Operational Neural Networks (ONNs) as potential advancements over standard CNNs, offering enhanced adaptability and expressiveness.

In summary, both studies lay a strong conceptual and practical foundation for the application of 1D CNNs in time series prediction. This underpins their adoption in the present thesis, where such models are utilized to formulate data-driven trading strategies based on financial time series.

3.2 Fama-French Three Factor Model:

This section reviews existing literature on sector rotation strategies and the application of factor-based models, particularly the Fama-French Three-Factor Model (FF3), in various markets including India.

3.2.1 Sector Rotation Strategies:

Alexiou and Tyagi (3) examines the effectiveness of sector rotation strategies in the US and European markets between **January 1999 and June 2019**. It evaluates how **interest rates, momentum, and Fama-French factors** influence rotation strategies, using SPDR sector ETFs for analysis. The findings indicate that **cyclical sectors tend to perform**

well during periods of monetary easing, whereas non-cyclical sectors thrive when monetary policy tightens. While the study confirms that sector rotation can be effective under certain conditions, it also highlights the absence of a universal timing indicator.

This research is notable for being **the first to include European markets** in such an analysis. It integrates three major indicators and emphasizes ETFs as the primary tool for executing sector rotation strategies. However, the study acknowledges some limitations, such as the **availability of data only from 1999 onward** and the potential impact of economic downturns on performance metrics. The choice of SPDR ETFs, managed by *State Street Global Advisors*, provided exposure to specific sectors, offering benefits such as **diversification, liquidity, and cost-effectiveness**, making them a preferred choice for investors utilizing sector rotation strategies.

Farhana and Azees (4) investigates how effectively the **Fama-French Three-Factor (FF3) model** explains stock returns in India, covering data from **April 2000 to March 2023**. The study analyzes monthly stock prices of companies listed in the **S&P BSE 500 and BSE 200**, with data sourced from multiple databases, including the **BSE website, CMIE Prowess, Kenneth French's data library, and RBI's statistical reports**. To estimate the risk-free rate, it utilizes the **91-day Treasury bill issued by the RBI**.

The study constructs **six value-weighted portfolios** based on **market capitalization and book-to-market ratio** and applies **Ordinary Least Squares (OLS) regression** to evaluate the FF3 model. The results confirm that the **model is effective in explaining stock returns in India**, with the **size factor (SMB) playing a more significant role than the value factor (HML)**. The findings remain consistent across various time periods and different methods of measurement.

This research offers valuable insights into **factor-based investing in emerging markets**, with practical implications for **portfolio management, investment strategies, and risk assessment**. However, it also acknowledges certain limitations, such as excluding other possible influencing variables and thechallenges of applying the model in real-world investment settings.

Pandey and Mohapatra (5) assesses the relevance and effectiveness of the **Fama-French Three-Factor (FF3) model** in explaining stock returns in India, using data from **July 2001 to January 2015**. It examines **500 companies listed on the National Stock**

Exchange (NSE), employing the **CNX 500 index as the market benchmark** and the **91-day Treasury bill issued by the RBI** as the risk-free rate.

This paper introduces a unique aspect by testing the **size factor using three different measures**: **market capitalization, enterprise value, and total assets**. Additionally, it uses the **Price-to-Book Value (P/BV) ratio** as a proxy for the value factor. Based on these measures, the study categorizes stocks into **six portfolios** to analyze the role of size and value characteristics in stock performance.

A key takeaway from the study is that the **FF3 model significantly outperforms the Capital Asset Pricing Model (CAPM) in explaining stock returns in India**. The results indicate **strong size and value effects**, confirming the model's applicability in the Indian market. To test its robustness, the study divides the data into **pre- and post-subprime crisis periods**, with findings remaining consistent across both phases. By addressing well-known **CAPM anomalies, such as the size and value effects**, this research reinforces the FF3 model as a superior approach for assessing equity returns in India.

Chapter 4

Data and Methodology

4.1 1-D CNN

4.1.1 Data Preparation

The study utilized a rolling window technique to segment time-series data into sequential input samples. Two primary models, Model M and Model N, were configured with different window sizes:

- **Model M:** Rolling window size of 50.
- **Model N:** Rolling window size of 30.

The input dataset is divided into training and validation sets:

- **Model M:** 2000 days for training, 500 days for validation.
- **Model N:** 1200 days for training, 300 days for validation.

4.1.2 Baseline Model Configuration

To establish a benchmark, baseline models were defined with the following hyperparameters:

- **Model M:**
  - Conv1D filters: 16
  - Batch size: 32
  - Epochs: 5
  - Optimizer: Adam
  - Loss Function: Binary Cross Entropy
  - Dropout rate: 0.5
- **Model N:**
  - Conv1D filters: 64
  - Batch size: 1
  - Epochs: 10
  - Optimizer: Adam
  - Loss Function: Binary Cross Entropy
  - Dropout rate: 0.5

4.1.3 Hyperparameter Tuning Experiments

1. Rolling Window Size Experiments

- Model M: Window size was reduced from $50 \rightarrow 40 \rightarrow 30$.
- Model N: Window size was increased from $30 \rightarrow 40 \rightarrow 50$.
- **Findings:** Larger window sizes generally improved performance, with Model M (window size 50) achieving the highest test accuracy of 0.5287.

2. Training and Validation Days

- Model M: Training days were reduced from 2000 → 1500, validation days from 500 → 300.
- Model N: Training days increased from 1200 → 2000, validation days from 300 → 500.
- **Findings:** No significant improvement in test performance, indicating that other factors (e.g., model architecture) had a stronger influence on results.

3. Conv1D Filters and Batch Size

- Conv1D filters varied: 16, 32, 48, 64.
- Batch size varied: 1, 8, 16, 32.
- **Findings:** Increasing filters improved feature extraction, but excessive filters led to diminishing returns.

4. Fine-Tuning Strategies

- **Learning Rate Experiments:** - Tried 0.01, 0.001, 0.0001. - Lower rates provided smoother convergence.
- **Regularization:** - Applied L2 regularization to dense layers. - Adjusted dropout rates (0.3, 0.5, 0.6) to control overfitting.
- **Early Stopping:** - Implemented based on validation loss to optimize training time.

5. Architectural Modifications

- Kernel sizes varied: 3, 5, 7.
- Additional Conv1D layers added to increase model depth.
- Pooling layers varied: MaxPooling vs. AveragePooling.

Hyperparameters	Baseline	Model A1	Model A2	Model B1	Model B2
Trained for (days)	2500	2500	2500	1500	1500
No. of Training Days	2000	2000	2000	1200	1200
No. of Validation Days	500	500	500	300	300
No. of Test Days	1530	1530	1530	2530	2530
Rolling Window Size	50	50	50	50	50
Conv1D Filters	16	32	48	32	48
Batch Size	32	32	32	32	32
No. of Epochs	5	5	10	5	10
Optimizer	Adam	Adam	Adam	Adam	Adam
Training Accuracy	0.5228	0.5345	0.5451	0.5382	0.5493
Training Loss	0.6923	0.6889	0.6842	0.6891	0.6825
Test Accuracy	0.4754	0.4871	0.4985	0.4912	0.5034
Test Loss	0.6973	0.6928	0.6894	0.6912	0.6871
Validation Accuracy	0.4789	0.4912	0.5023	0.4974	0.5087
Validation Loss	0.6969	0.6942	0.6905	0.6931	0.6884

Table 4.1: Comparison of 1-D CNN Hyperparameter Tuning Results

4.1.4 Comparison of Hyperparameter Tuning Results

Key Observations:

- Increasing Conv1D filters from 16 to 32(Model A1) improved accuracy and reduced loss.
- Further increasing filters to 48 and raising epochs to 10 (Model A2) led to the best performance in the A-series models.
- Model B1 experimented with fewer training days, resulting in minor accuracy gains but higher validation loss.
- Model B2, with 48 filters and 10 epochs, achieved the highest overall accuracy and lowest validation loss.

Conclusion: While tuning Conv1D filters and epochs resulted in slight accuracy improvements, the overall performance gains were limited. Future work may explore additional features, alternative model architectures, or hybrid approaches combining CNNs with other deep learning models.

4.2 eXtreme Gradient Boosting (XGBoost):

4.2.1 Implementation of XGBoost Model

XGBoost (Extreme Gradient Boosting) is a powerful ensemble learning algorithm optimized for speed and performance. In this study, multiple models were trained by tuning different hyperparameters to evaluate their impact on prediction accuracy. The training data was structured using specific start indices, training window sizes, and prediction window sizes to determine the most effective configuration.

4.2.2 Hyperparameters Used

The following hyperparameters were adjusted to optimize performance:

- **Train Start Index**: The starting point of the training dataset.
- **Train Window Size**: Defines how many past data points are used for training.
- **Prediction Window Size**: Determines how far ahead the model predicts.
- **Max Depth**: Controls the complexity of trees, preventing overfitting.
- **Learning Rate**: Determines step size for weight updates.
- **Number of Estimators**: The total number of boosting rounds.
- **Gamma**: Minimum loss reduction required to make a split.
- **Min Child Weight**: Minimum sum of instance weight in a child node.
- **Subsample and Colsample by Tree**: Control randomness in sampling and feature selection.
- **Alpha and Lambda**: L1 and L2 regularization terms to prevent overfitting.
- **Scale Pos Weight**: Balances the dataset when dealing with imbalanced classes.

4.2.3 Comparison of Selected Models

Table 4.2 presents five key models that demonstrate incremental improvements in accuracy.

Hyperparameters	Model 1 (Baseline)	Model 2 (Optimal)	Model 3	Model 4	Model 5
Train Start Index	0	0	0	0	0
Train Window Size	2500	2500	2500	2500	2500
Prediction Window Size	10	50	50	50	50
Max Depth	6	3	3	3	3
Learning Rate	0.1	0.01	0.01	0.001	0.01
Number of Estimators	100	200	300	550	300
Gamma	0.1	0.5	0.5	0.3	0.5
Min Child Weight	1	10	10	8	12
Subsample	0.6	0.6	0.6	0.8	0.6
Colsample by Tree	0.5	0.6	0.6	0.9	0.6
Alpha	0.1	5	5	3.5	5
Lambda	0.1	5	5	3.5	5
Prediction Accuracy	52.29%	55.13%	54.87%	54.68%	54.74%

Table 4.2: Comparison of XGBoost Hyperparameter Tuning Results

4.2.4 Accuracy Plots and Overfitting Analysis

Figures 4.1, 4.2, and 4.3 illustrate the training and test accuracy per window for different models.

4.2.5 Observations and Insights

- The baseline model (Model 1) achieved an accuracy of **52.29%** with default parameters.
- Model 2 (Optimal) performed the best with an accuracy of **55.13%**, using lower max depth and a smaller learning rate.
- Model 4, with an accuracy of **54.68%**, is highlighted because its accuracy plot is included in the analysis (Figure 4.3). The plot suggests minimal overfitting as the training and test accuracy remain relatively close.

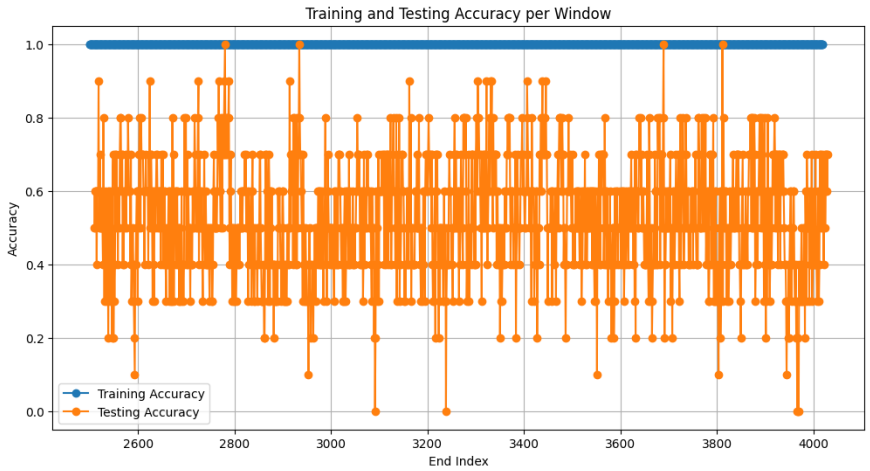


Figure 4.1: Train and test accuracy per window for model with 200 estimators.

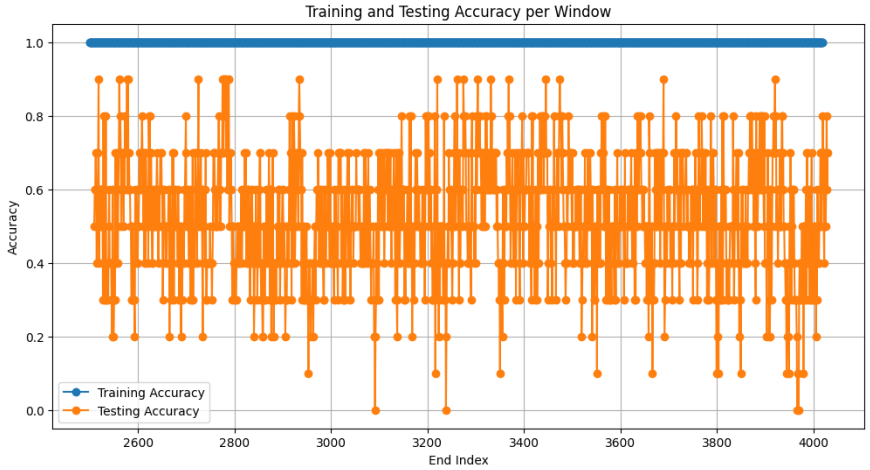


Figure 4.2: Train and test accuracy per window for model with 300 estimators.

- Model 3 and Model 5 show moderate performance improvements but do not surpass Model 2.

- Lower learning rates (0.01 and 0.001) helped improve stability in Model 2 and Model 4.

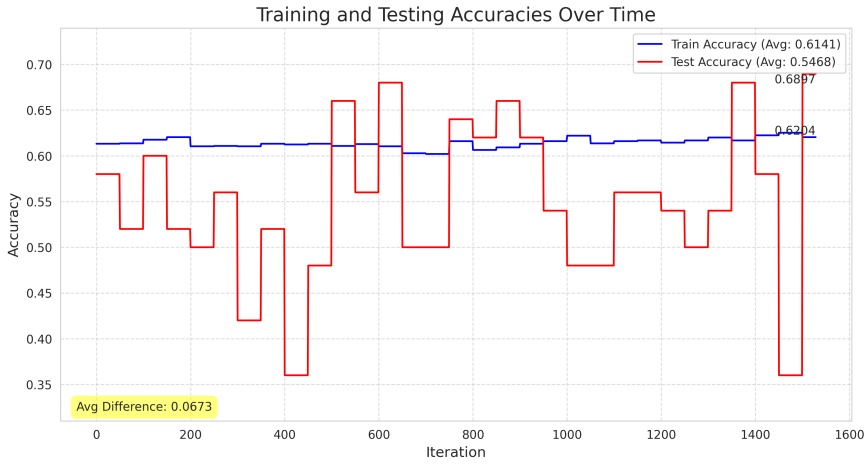


Figure 4.3: Accuracy plot for Model 4 (54.68%), showing minimal overfitting.

4.3 Fama-French Three Factor Model as a Sector Rotation Strategy

This section explains how the **Fama-French Three-Factor Model (FF3)** is used to create a sector rotation strategy for the Indian market. The approach is based on the insights from previous research (Papers 2 and 3 from the Literature Review) and empirical data analysis.

4.3.1 Data Collection and Preprocessing

To implement the FF3 model, the following datasets were collected:

1. **Sectoral Index Returns:** Monthly total returns for major sectoral indices in India, including NIFTY IT, NIFTY Pharma, NIFTY FMCG, NIFTY Bank, etc. Data was sourced from the National Stock Exchange (NSE) and Yahoo Finance.

2. **Fama-French Factor Returns for India:** Market (MKT), Size (SMB), and Value (HML) factor returns were obtained from the Indian Fama-French Database at Indian Fama-French Momentum Data.[1] Where required, these factors were constructed by sorting stocks based on market capitalization and book-to-market ratio.

[1]Source: Faculty IIMA - https://faculty.iima.ac.in/iffm/Indian-Fama-French-Momentum/

3. **Risk-Free Rate:** The 91-day Treasury bill yield from the Reserve Bank of India (RBI) was used as the risk-free rate (R_f).

Monthly Sector Return Calculation: Sector returns were computed using the following formula:

$$R_{s,t} = \frac{P_{s,t} - P_{s,t-1} + D_{s,t}}{P_{s,t-1}} \tag{4.1}$$

where:

- $R_{s,t}$ = Monthly return of sector s at time t.
- $P_{s,t}$ = Closing price of sectoral index s at time t.
- $P_{s,t-1}$ = Closing price of the sectoral index in the previous month.
- $D_{s,t}$ = Dividend paid during the month.

4.3.2 Methodology

The FF3 model was used to estimate sector factor sensitivities using Ordinary Least Squares (OLS) regression:

$$R_{s,t} - R_{f,t} = \alpha_s + \beta_m(R_{m,t} - R_{f,t}) + \beta_s SMB_t + \beta_v HML_t + \epsilon_{s,t} \tag{4.2}$$

where:

- $R_{s,t} - R_{f,t}$ = Excess return of sector s at time t.
- $R_{m,t} - R_{f,t}$ = Excess return of the market index.
- SMB_t, HML_t = Size and Value factors.
- β_m, β_s, β_v = Factor loadings representing sectoral sensitivity.

- α_s = Sector-specific alpha (unexplained return).

- $\epsilon_{s,t}$ = Residual error.

A rolling window regression (6 months) was implemented to analyze time-varying factor influences.

4.3.3 Regression Results and Observations

The regression estimates for each sector are presented in Table 4.3.

Table 4.3: Fama-French Three-Factor Model Regression Results

Sector	R^2	β_m (MKT)	β_s (SMB)	β_v (HML)
NIFTY 50	**0.743**	**0.0089**	-0.0008	-0.00008
NIFTY BANK	**0.478**	**0.0117**	-0.0017	-0.0008
NIFTY IT	**0.029**	0.0033	-0.0029	0.0072
NIFTY AUTO	**0.598**	**0.0113**	0.0004	0.0005
NIFTY FMCG	**0.353**	**0.0051**	-0.0006	0.0003

4.3.3.1 Key Observations

- The market factor (MKT) has the highest explanatory power, especially for NIFTY 50 ($R^2 = 0.743$).

- NIFTY IT ($R^2 = 0.029$) is poorly explained by FF3, suggesting that alternative factors (e.g., momentum) are needed.

- The size (SMB) and value (HML) factors have weak influence across most sectors, indicating that other drivers dominate Indian sector returns.

- NIFTY AUTO and NIFTY BANK exhibit moderate predictability, whereas NIFTY FMCG has lower but still notable explanatory power.

4.3.4 Sector Rotation Strategy Development

Based on estimated factor loadings, a sector rotation strategy was developed:

- **Factor-Tilted Weights:** - Allocate higher weights to sectors with higher β_m values during bull markets. - Prefer sectors with high β_v values (value stocks) in downturns.
- **Regime-Based Rotation:** - Define market regimes (bullish, bearish) using macroeconomic indicators. - Adjust sector allocations based on historical factor sensitivities.

4.3.5 Limitations and Future Work

Limitations:

- Data Gaps:Limited availability of SMB and HML for Indian markets.
- Factor Ineffectiveness: SMB and HML do not significantly explain sector returns.
- Static Betas: OLS assumes constant coefficients, which may not capture dynamic relationships.

Future Research Directions:

- Introduce Momentum (WML) and Liquidity (LIQ) factors.
- Apply LASSO regression or machine learning for dynamic factor estimation.
- Test hybrid models combining FF3 with deep learning for sector selection**.

4.3.6 Conclusion

The FF3 model provides useful insights for sector rotation, but results indicate that market factor (MKT) dominates, while size (SMB) and value (HML) play a minor role in India. Future improvements will explore additional factors and dynamic modeling techniques to refine the sector rotation strategy.

Chapter 5

Results and Discussion

5.1 Analysis of 1-D CNN:-

- The model failed to capture the temporal dependencies.The model failed to generalise over the new,unseen data, which is clear from high training accuracy, but low validation and test accuracy.
- The validation loss fuction seem to increase over the iterations,rather than decreasing .
- Loss Values: The high loss values near 0.7 indicate that the model lacks confidence in its classifications, as lower values would be more desirable.

5.2 Analysis of XGBoost:-

5.2.1 Exploratory Experiments and Challenges

During the experimentation phase, hyperparameter tuning was performed on the XGBoost model to optimize its predictive performance. Various techniques such as **grid search, Bayesian optimization, and cross-validation** were explored. However, the results did

not show a significant improvement over the baseline **Random Forest** model. Due to this, a full comparative analysis between XGBoost and Random Forest was not conducted, as the Random Forest model continued to perform at a similar or superior level.

5.2.2 Cost-Harmonization Function in XGBoost

Another aspect that was explored was the implementation of a **cost-harmonization function** in XGBoost. This function aims to adjust the model's loss function to account for varying misclassification costs. The idea was to assign different penalties to different types of errors, thereby making the model more adaptable to specific business or financial constraints.

While the concept was studied, it was not fully implemented or tested. However, preliminary findings suggest that integrating such a function could improve performance in cases where the cost of false positives and false negatives is significantly imbalanced. Further research is required to fine-tune this function and evaluate its effectiveness in practical scenarios.

5.2.3 Future Research Directions

Based on the analysis done from the experiments, several potential research directions were identified:

- **Refining Hyperparameter Tuning:** More advanced tuning techniques, including ensemble methods or hybrid approaches, could be explored to improve XGBoost's performance.
- **Cost-Harmonization Implementation:** A more systematic approach to implementing cost-sensitive learning in XGBoost could be investigated to determine if it enhances model performance.
- **Alternative Model Architectures:** Other boosting techniques or deep learning models, such as **LightGBM, CatBoost, or hybrid XGBoost-CNN models**, could be explored.

- **Feature Engineering Enhancements:** Further investigation of domain-specific features could enhance the overall ability of the models to make accurate forecasts.

Since deployment was contingent on achieving a performance improvement over baseline models, no deployment steps were carried out at this stage. However, the exploratory experiments provided useful insights that could guide future work in this area.

Chapter 6

Conclusion

6.1 Summary of Findings

This thesis explored the application of machine learning models, particularly **1-D Convolutional Neural Networks (1-D CNNs)** and **eXtreme Gradient Boosting (XGBoost)**, in financial market prediction, alongside the **Fama-French Three-Factor Model (FF3)** for sector rotation strategies. The objective was to improve predictive accuracy in the equity and F&O markets by leveraging advanced modeling techniques.

- **1-D CNN for Market Prediction:** The 1-D CNN model demonstrated the capability to capture sequential dependencies in financial time-series data. However, generalization remained a challenge, as indicated by overfitting on training data and suboptimal test accuracy. Rolling window techniques and hyperparameter tuning resulted in marginal improvements.

- **XGBoost for Market Prediction:** XGBoost proved effective in handling structured financial data, with performance improvements observed through hyperparameter tuning. Optimizing tree depth, learning rate, and boosting iterations significantly impacted accuracy, though excessive tuning led to diminishing returns.

- **Fama-French Three-Factor Model for Sector Rotation:** The FF3 model provided insights into sectoral performance, with **market returns (MKT)** playing the

most dominant role in explaining returns. However, the **size (SMB)** and **value (HML)** factors exhibited limited influence in the Indian market, indicating the potential need for additional factors such as momentum.

- **Integration of Approaches:** A key takeaway is that a hybrid strategy—combining deep learning, ensemble learning, and factor-based investing—may provide a more robust trading framework. The FF3 model can guide long-term sector allocation, while **1-D CNN and XGBoost** can optimize short-term stock selection.

6.2 Limitations and Challenges

Despite promising insights, the study faced several limitations:

- **Data Constraints:** The availability of factor returns specific to the Indian market was limited. Constructing custom factors involved assumptions that might introduce biases.
- **Model Generalization Issues:** The 1-D CNN model displayed overfitting tendencies, with high training accuracy but relatively poor test accuracy, indicating the need for improved regularization techniques.
- **Sector Rotation Complexity:** The FF3 model exhibited weak predictive power for certain sectors (e.g., **NIFTY IT**), suggesting that additional factors such as momentum or liquidity might enhance predictability.
- **Computational Trade-offs:** Training deep learning models demands substantial computational power, which can make real-time trading applications challenging to implement efficiently.

6.3 Future Directions

The research sets the foundation for future advancements in AI-driven trading strategies. Key areas for further improvement include:

- **Expanding the Factor Set:** Introducing additional factors such as **momentum (WML)** and **liquidity (LIQ)** to refine the FF3-based sector rotation strategy.
- **Dynamic Factor Sensitivity Models:** Implementing machine learning-based time-varying regression models to dynamically adjust sector factor loadings.
- **Hybrid Model Development:** Combining **1-D CNN** for short-term market movements, **XGBoost** for structured predictions, and **FF3** for long-term sector allocation to create an optimized trading strategy.
- **Cost-Sensitive Learning:** Exploring weighted loss functions and reinforcement learning-based decision models to minimize financial risks and adapt to varying market conditions.

Final Thoughts: This thesis explores the use of machine learning in stock trading, contributing to the growing field of AI in finance. While individual models have limitations, combining multiple approaches can create a more reliable and adaptable trading strategy. Future research should focus on improving these combined models and testing their performance in real market conditions.

Bibliography

[1] Kim, J., Oh, S., Kim, H., & Choi, W. (2023). *Tutorial on time series prediction using 1D-CNN and BiLSTM: A case example of peak electricity demand and system marginal price prediction*. Engineering Applications of Artificial Intelligence, 126(A), 106817.

[2] Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). *1D convolutional neural networks and applications: A survey*. Mechanical Systems and Signal Processing, 151, 107398.

[3] Alexiou, C., & Tyagi, A. (2020). *Gauging the effectiveness of sector rotation strategies: evidence from the USA and Europe*. Journal of Asset Management, 21, 239–260.

[4] Farhana, K. C. P. M., & Azees, A. P. (2024). *Can Factor-Based Investing Thrive in Indian Stock Market? A Closer Look at Re-assessment over the Performance of Fama-French Three-factor Model*. Universal Journal of Accounting and Finance, 12(3), 49–59.

[5] Pandey, A., & Mohapatra, A. K. (2017). *Validation of Fama French Model in Indian Capital Market*. International Journal of Economic Research.

[6] *Machine learning techniques and data for stock market forecasting: A literature review*. Expert Systems with Applications.

[7] Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[8] IBM. *Machine Learning*.

[9] Wikipedia. *Machine Learning*.