Leveraging Machine Learning for Predictive Price Analytics & Algorithmic Trading Strategy Optimization

A Dissertation Submitted in Partial Fulfilment of the Requirements for the

Degree of

**Master of Science (MSc) in Artificial Intelligence & Machine Learning**

By

Ranjan Kumar Jha

*Supervisor*: Dr. Loo Poh Kok

Module Code: M33876

University ID: UP 2301022

June 2025



This page intentionally left blank

**ABSTRACT**

In the rapidly evolving landscape of financial technology, predictive analytics and algorithmic trading have become central to data-driven investment strategies. This dissertation investigates the application of advanced machine learning techniques for stock price prediction and trading strategy optimization, with a primary focus on forecasting the closing prices of Bank of America Inc. (BAC) using Long Short-Term Memory (LSTM) networks integrated with PySpark MLlib. By harnessing the distributed computing power of PySpark for first part, this study presents a scalable and efficient framework for handling large volumes of financial data and extracting actionable insights.

In addition to time-series prediction using LSTM, the research implements and evaluates three distinct AI-driven algorithmic trading strategies: (1) an unsupervised learning model applied to the S&P 500 for market pattern discovery, (2) a sentiment analysis approach leveraging real-time Twitter data to rank NASDAQ stocks based on public sentiment, and (3) a hybrid intraday strategy that combines Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models with traditional technical indicators for short-term price movement prediction. These methodologies reflect the interdisciplinary nature of the research, incorporating machine learning, natural language processing, and statistical financial modelling.

The evaluation of these strategies is performed using key financial benchmarks such as the NASDAQ-100 ETF (QQQ), measuring not only predictive accuracy but also practical trading performance in terms of return on investment and risk-adjusted metrics. The results highlight the effectiveness of AI-powered techniques in capturing complex market dynamics, enhancing portfolio performance, and offering interpretability in decision-making processes.

This work contributes to the growing domain of intelligent financial systems by proposing a modular, explainable, and scalable framework for predictive analytics and algorithmic trading. It underscores the potential of integrating deep learning with big data tools in building robust and adaptive trading models that can respond to the volatility and complexity of modern financial markets.

**Word Count**: 17238

**ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to my supervisor, Dr. Loo Poh Kok, for their invaluable guidance, support, and encouragement throughout this dissertation. Their expertise in machine learning and financial systems greatly shaped the direction and depth of this research.

I am also thankful to the faculty and staff at the Kaplan and University of Portsmouth, for providing a stimulating environment and the academic foundation upon which this work is built.

Special thanks to my peers and collaborators in the AI research group for their insightful discussions, feedback, and motivation throughout this journey.

I would also like to extend my appreciation to the open-source communities behind tools like scikit-learn, pandas, keras, which played a crucial role in building and testing the strategies in this thesis.

Lastly, I am deeply grateful to my family and friends for their continuous support and belief in my abilities, especially during challenging phases of this research.

**KEYWORDS**

*BAC Stock Price Prediction, Long Short-Term Memory Network (LSTM), Keras Deep Learning, PySpark MLIB Regression, Yahoo Finance, Attention Mechanism, TensorFlow, Matplotlib*

**DECLARATION**

I hereby declare that the work presented in this dissertation titled “Leveraging Machine Learning for Predictive Price Analytics & Algorithmic Trading Strategy Parameters Optimization is the result of my independent research carried out under the guidance of my academic supervisor and has not been submitted previously for any degree or diploma at any other institution or university.

This dissertation does not contain any material that has been published or written by others except where due reference has been made in the text. All sources of information and data used in the development of this research have been duly acknowledged.

I further declare that the dissertation complies with the ethical standards and guidelines prescribed by my academic institution, and any use of AI tools or software assistance was conducted strictly within academic integrity boundaries to support, not substitute, my own work.

### **Table of Contents**

1. **Introduction**
   1. Problem statement
   2. Significance of study
   3. Thesis contributions
   4. Thesis outline
   5. Research objectives
   6. Novelty and Research Contribution
2. **Reason for Choosing This Project**
3. **Literature Review**
4. **Methodology**
5. Data Sourcing
6. **Data Loading and Review**
   1. Data Import and Initial Review
   2. Summary of Dataset (Top 5 Records)
7. **Exploratory Data Analysis (EDA) using PySpark**
   1. Dataframe Schema and Records
   2. Capturing Basic Statistics on Close Price
   3. Visualization of Ticker Close Price Fluctuation
   4. Simple Visualization of Opening Price Movement
   5. Simple Visualization of Closing Price Movement
   6. Simple Visualization of Closing Price Movement
   7. Visualization on Rolling/Moving Average Over Adj Close
   8. Simple Visualization on Rolling/Moving Average on Closing Price
   9. High/Volume Ratio for Each Day
8. **Data Preprocessing**
   1. Datatype Conversion
   2. Missing Data
   3. Training and Test Data Splitting
   4. Principle Component Analysis (PCA)
   5. Data Transformer via VectorAssembler
   6. Normalization via MinMaxScaler
9. **Machine Learning Based Predictive Models**
   1. Perform regression on “Close Price”
      1. Linear Regression
      2. GBT Regressor
      3. Decision Tree Regressor
      4. Comparison
   2. LSTM with the Attention Mechanism in TensorFlow
      1. Optimizing the LSTM Model
      2. Result Summary
10. **Explainable AI**
11. **Technologies Stack**
12. **Implications & Challenges**
13. Future Work
14. Conclusion & Takeaways
15. Future Work
16. Conclusion & Takeaways
17. End
18. Future Work
19. Conclusion & Takeaways
20. End

### **List of Tables**

### **List of Figures**

# 1 INTRODUCTION

The financial markets today represent one of the most complex and dynamic environments in the world. Influenced by a vast network of interdependent economic, political, and psychological factors, predicting stock price movements has remained a formidable challenge for researchers, traders, and financial institutions alike. In response to this complexity, predictive analytics powered by artificial intelligence (AI) and machine learning (ML) has emerged as a powerful tool to decode market behaviour, uncover hidden patterns in historical data, and inform trading decisions with unprecedented accuracy.

Among the various machine learning architectures, Long Short-Term Memory (LSTM) networks—a specialized class of recurrent neural networks (RNNs)—have demonstrated a unique ability to model sequential and temporal dependencies. This makes them particularly well-suited for time-series forecasting tasks such as stock price prediction. In this research, LSTM networks are employed to forecast the closing prices of Bank of America Inc. (BAC), a major financial stock, by learning from historical market data obtained from Yahoo Finance. The use of PySpark MLlib enables the handling of large-scale datasets in a distributed and efficient manner, ensuring both scalability and real-time processing capabilities.

Parallel to the time-series forecasting model, this dissertation explores the rapidly evolving domain of algorithmic trading—where investment decisions are made automatically based on mathematical models and data-driven strategies. Traditional rule-based systems often lack the flexibility to adapt to the volatile and non-linear nature of markets. In contrast, ML-based trading systems can learn, adapt, and evolve, making them ideal for today's dynamic trading environments.

Three distinct AI-powered trading strategies are explored in this dissertation to showcase the breadth and depth of intelligent financial modelling:

Unsupervised Learning for Portfolio Clustering: This strategy utilizes clustering algorithms on S&P 500 stock data to identify patterns in stock behaviour, enabling better diversification and portfolio optimization.

Sentiment-Based Trading Using Twitter Data: Leveraging natural language processing (NLP) techniques, this model evaluates real-time sentiment from social media platforms to rank and select NASDAQ stocks. Public sentiment, often a precursor to market movements, is quantified and integrated into the trading decision-making process.

Hybrid Intraday Strategy with GARCH Models and Technical Indicators: This approach combines traditional statistical models such as the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) with commonly used technical indicators (e.g., MACD, RSI) to forecast short-term volatility and detect potential trading signals within intraday windows.

Together, these strategies form a modular and comprehensive framework for the design, evaluation, and deployment of ML-driven trading systems. Each model is rigorously tested using historical market data and benchmarked against indices such as the NASDAQ-100 ETF (QQQ) to assess their predictive accuracy, robustness, and performance in real-world market conditions.

This dissertation aims not only to contribute to academic research in the field of AI and finance but also to provide practical insights and tools for traders, analysts, and financial institutions. By integrating machine learning, big data tools, and streaming architectures, it lays the foundation for scalable, intelligent, and adaptive trading platforms capable of navigating the intricacies of modern financial markets.

1.1 Problem statement

Financial markets are inherently complex, influenced by dynamic, interrelated variables such as macroeconomic indicators, investor sentiment, and market volatility. Traditional rule-based trading systems struggle to adapt to rapidly changing patterns, often leading to suboptimal decision-making and missed opportunities. Additionally, the growing volume, velocity, and variety of market data have exceeded the processing capabilities of conventional statistical models. There exists a need for intelligent systems capable of learning from historical data, predicting price movements, and optimizing trading strategies to respond in real time. Developing a scalable, accurate, and modular framework that integrates machine learning, big data processing, and sentiment analysis can significantly enhance financial forecasting and automated trading.

1.2 Significance of study

This study addresses the urgent need for data-driven, adaptive trading systems in an era of algorithmic finance. By employing advanced machine learning techniques such as Long Short-Term Memory (LSTM) networks, PySpark MLlib, and social sentiment analysis from Twitter, the research provides a multi-faceted approach to predictive analytics and automated trading. The findings contribute to academic literature and offer practical insights for traders, financial institutions, and fintech developers by demonstrating the potential of intelligent systems in outperforming benchmark indices, enhancing risk management, and improving execution efficiency. This study also advances the application of hybrid strategies combining time-series forecasting, volatility modelling, and natural language processing in financial domains

1.3 Thesis contributions

This dissertation makes the following key contributions:

**LSTM-Based Price Prediction**: Implementation of deep learning models for forecasting the closing price of BAC stock using LSTM and Yahoo Finance data.

**Scalable Big Data Pipeline**: Integration of PySpark for streaming and parallel processing of financial datasets, enabling real-time analytics.

**Multi-Strategy Algorithmic Trading System**: Development and comparison of three distinct AI-powered strategies:

 Unsupervised learning on S&P 500 for portfolio optimization.

 Twitter sentiment analysis for NASDAQ stock ranking.

 Hybrid intraday strategy using GARCH models with technical indicators.

Performance Evaluation: Backtesting and benchmarking of strategies against standard indices (e.g., QQQ), showcasing profitability, risk metrics, and model generalizability.

Modular, Explainable Framework: Proposal of a flexible architecture that can be extended to other financial instruments, enabling transparency and adaptability in financial AI systems.

1.4 Research objectives

To explore and implement predictive analytics techniques for stock price forecasting using machine learning and deep learning.

To leverage LSTM networks and PySpark MLlib for real-time, scalable financial data processing.

To design, implement, and evaluate AI-based trading strategies that incorporate historical, technical, and sentiment-based indicators.

To benchmark the performance of these strategies against market indices and assess their robustness and adaptability.

1.5 Novelty and Research Contribution

This dissertation is novel because it uniquely combines multiple advanced AI and ML methodologies into a modular, scalable, and explainable framework for predictive price analytics and algorithmic trading, which sets it apart from many traditional or singular-focused approaches. Here's why:

1.5.1 Multi-Strategy Integration

Rather than relying on a single trading model or forecasting method, the dissertation presents:

* Unsupervised Learning for clustering and portfolio optimization on S&P 500,
* Sentiment Analysis using Twitter data to rank NASDAQ stocks,
* A hybrid intraday model combining GARCH volatility modelling with technical indicators.

**Why it's novel:**

Combining these heterogeneous models creates a more holistic, adaptive trading system that can handle diverse market conditions—this is rarely implemented together in one framework.

1.5.2 Real-Time Predictive Architecture

By incorporating:

* PySpark MLlib for scalable processing,
* LSTM for time-series forecasting on large financial datasets, the study develops a real-time prediction engine—not just theoretical, but engineered for deployment.

**Why it's novel**:

Most academic work stops at model evaluation; this work focuses on real-time, production-grade architecture, which bridges the academic-industry gap.

1.5.3 Applied Across Different Markets

Most research targets one index or one geography. This study spans:

* S&P 500 (diverse large caps)
* NASDAQ (tech-focused, volatility-sensitive)

**Why it's novel**:

It validates its models across multiple market styles, improving robustness and relevance.

1.5.4 Explainability and Modularity

The framework is designed to be:

* Explainable: Each component (clustering, sentiment, LSTM) is transparent and interpretable.
* Modular: Can be plugged into different markets, datasets, or strategies.

**Why it's novel**:

This addresses two pressing issues in AI systems—black-box behaviour and lack of generalizability.

# 2 REASON FOR CHOOSING THIS PROJECT

# 2.1 *Being in same industry*

# Working in same domain where we develop algorithmic trading apps so makes sense

# 2.2 *Relevance to Current Financial Trends*

# The project aligns with current trends in fintech innovation, algorithmic trading, and AI integration, offering practical value to traders, financial analysts, and technology firms.

# 2.3 *Focus on Technological Innovation*

# The title emphasizes the integration of advanced machine learning techniques, highlighting that the project is not just about prediction but about using cutting-edge technology to enhance accuracy and insight.

# 2.4 *Market-Driven Insights*

# The title reflects the project’s goal of delivering data-driven market insights, underscoring that it’s not just a theoretical exercise, but a practical tool for understanding market behavior.

# 2.5 *Emphasizing Predictive Accuracy*

# The title stresses that machine learning models are designed to improve the accuracy of stock market predictions, positioning your project as a solution to the inherent volatility of financial markets.

# 2.6 *Desire to Build a Comprehensive Framework*

# The goal was to create an end-to-end, modular, and interpretable solution that integrates AI, big data, and financial modeling into a unified trading strategy toolkit.

# 2.7 *Addressing Real-World Challenges*

# Traditional trading systems struggle with market volatility and data complexity. This thesis aims to develop adaptive, data-driven models that can respond to such challenges effectively.

# 2.8 *Bridging Theory and Practice*

# This research offers the opportunity to connect advanced AI techniques—such as LSTM networks, PySpark MLlib, Statistical Models and sentiment analysis—with real-world financial market applications.

# 3 LITERATURE REVIEW

The convergence of **Artificial Intelligence (AI)** and **Financial Markets** has catalysed a transformative shift in how trading strategies are designed, tested, and deployed. From conventional rule-based systems to adaptive, data-driven models, the evolution of algorithmic trading continues to be shaped by advances in machine learning (ML), and statistical modelling.

This section surveys foundational and recent work in the domains most relevant to this dissertation: **unsupervised learning in financial markets**, **sentiment-driven trading strategies**, **GARCH-based intraday volatility modelling**, and **risk-adjusted portfolio optimization**. The review aligns with the three primary strategies explored in this research: clustering-based trading, sentiment-based investing, and GARCH-driven intraday forecasting.

### **3.1 Stock Price Prediction and Financial Time-Series Modelling**

Previous literature has long acknowledged the complexity of modelling financial time-series data due to its non-stationarity, noise, and market sensitivity. Traditional econometric models like ARIMA and GARCH have been foundational but often fail to capture nonlinear patterns inherent in high-frequency trading environments. In contrast, recent work by ***Fischer & Krauss (2018)*** and other scholars has established that LSTM networks—owing to their capability to retain long-term dependencies and manage vanishing gradients—are superior for modelling temporal dependencies in stock price movements. Your dissertation aligns well with this direction by embedding LSTM with attention mechanisms, further enhancing the model's ability to selectively focus on relevant time steps in sequence data.

### **3.2 Use of PySpark and Big Data Processing**

The integration of PySpark in this project reflects a recognition of the limitations of single-node processing tools like Pandas, especially when working with large financial datasets. PySpark’s distributed computing capabilities have been extensively supported in the literature (***Zaharia et al., 2016***) as an enabler for scalable machine learning pipelines. The usage of Kafka for streaming real-time data further strengthens the architecture and is aligned with state-of-the-art practices in financial ML pipelines (***Chen et al., 2021***).

### **3.3 Deep Learning and LSTM-Attention Mechanism**

The deployment of LSTM with attention in TensorFlow is a particularly strong element. Studies such as Qin et al. (2017) on attention-based RNNs for time-series forecasting have shown marked improvements over vanilla LSTM. Your incorporation of batch normalization and dropout also aligns with best practices in neural network optimization. However, the LSTM model's output metrics (e.g., MAE: 0.0572 and RMSE: 0.0663) warrant a deeper comparative discussion against ensemble models or hybrid models (e.g., LSTM combined with CNN layers).

While the low loss value (0.0044) suggests effective training, external validation across different stock symbols or market conditions could strengthen generalizability claims.

### **3.4 Unsupervised Learning in Financial Markets**

Unsupervised learning, particularly clustering algorithms, has been increasingly applied in financial analysis to identify latent structures in high-dimensional data. Traditional approaches to stock selection and portfolio diversification rely on fixed industry classifications or human-curated risk models. However, clustering provides a data-driven method to group securities with similar statistical behaviour, thus enabling dynamic portfolio construction.

**K-Means**, a widely used clustering algorithm, was employed by Avellaneda and Lee (2010) to identify **mean-reverting clusters** for statistical arbitrage opportunities. Their approach underscored the potential for generating alpha by exploiting reversion patterns within homogeneous groups. More recent work has incorporated **rolling factor betas** and **momentum indicators** into feature sets to enrich clustering accuracy. The inclusion of **factor loadings (e.g., Fama-French 5-Factor Model)** and **technical indicators** such as RSI and ATR enhance clustering by encoding both systematic exposures and short-term trading behaviour.

In this dissertation, a **momentum-oriented initialization of K-means** clustering is introduced, aligning clusters with RSI thresholds (e.g., 70 for strong upward momentum). This targeted initialization improves cluster interpretability and allows strategy logic to remain robust across market regimes. Such guided clustering strategies are rarely explored in literature, marking a novel methodological contribution.

### **3.5 Sentiment Analysis and Alternative Data in Trading**

The rise of alternative data has opened new frontiers in alpha generation. Among these, **social media sentiment analysis** has gained significant traction. Bollen et al. (2011) demonstrated how Twitter mood metrics can predict movements in the Dow Jones Industrial Average, with models incorporating emotion indexes outperforming baseline models.

Subsequent studies have explored different aspects of **engagement-based metrics** (e.g., comment-to-like ratios, retweet volume) as indicators of market interest and collective sentiment. Tetlock (2007) and Fang & Peress (2009) also validated the role of media content and public attention in shaping stock price dynamics. NLP tools such as **VADER, TextBlob**, and transformer-based architectures (e.g., BERT) facilitate extraction of sentiment scores from textual data.

The second strategy in this dissertation utilizes **NASDAQ-100 Twitter sentiment**, focusing on high-engagement stocks, and forms **equal-weighted portfolios** ranked by sentiment strength. While many prior works emphasize textual sentiment only, this approach integrates **engagement as a weighting filter**, adding an extra dimension to sentiment-based investing. It echoes recent findings that **attention intensity** may matter as much as sentiment polarity (Kearney & Liu, 2014).

### **3.6 Volatility Modelling and Intraday Forecasting with GARCH**

Accurate volatility forecasting is a cornerstone of risk-aware intraday trading. The **Generalized Autoregressive Conditional Heteroskedasticity (GARCH)** model, developed by Bollerslev (1986), models volatility clustering and has been widely used to estimate conditional variances in financial time series.

Tsay (2005) extended GARCH frameworks with regime-switching and non-linear specifications, improving their adaptability to intraday data. Recent literature also explores **hybrid models** combining GARCH with **technical indicators** such as Bollinger Bands and RSI to refine trade entry and exit signals (Poon & Granger, 2003).

This dissertation builds on these concepts by simulating daily and 5-minute data for a single asset, using **rolling window GARCH estimations** to derive daily volatility forecasts. These forecasts are combined with technical signals to create a dual-layer decision mechanism—daily signal confirmation with **intraday technical validation**. The application showcases how statistical volatility models can be dynamically linked to market microstructure signals, offering nuanced control over trade execution.

### **3.7 Backtesting, Risk Metrics, and Portfolio Optimization**

Reliable **backtesting** is a critical component of quantitative strategy development. As Lo (2002) notes, model performance without proper validation is often misleading due to **overfitting**, **non-stationarity**, or **data leakage**. Robust evaluation requires risk-adjusted performance metrics such as:

* **Sharpe Ratio** – Return per unit of total risk.
* **Sortino Ratio** – Return per unit of downside risk.
* **Maximum Drawdown** – Worst observed loss from a peak to trough.

This dissertation emphasizes **Sharpe ratio maximization** using the **Efficient Frontier approach** (Markowitz, 1952), implemented via the pypfopt library. In particular, the unsupervised learning strategy utilizes **diversification constraints** by bounding portfolio weights and integrating **rolling optimization** to ensure robust performance.

Alternative methods such as **equal-weighted portfolios** are retained as benchmarks, echoing the pragmatic insight that **simplicity often outperforms complexity**, especially in high-dimensional, noisy environments (DeMiguel et al., 2009).

### **3.8 Challenges in Machine Learning for Trading**

Despite its promise, ML in trading presents numerous **theoretical and practical challenges**. A key issue is **reflexivity**—the notion that once a pattern is discovered and exploited, it may dissipate as market participants adapt. As highlighted in the dissertation’s foundation course, **feedback loops** and **regime shifts** can render predictive models obsolete over time.

Other challenges include:

* **Overfitting and generalization** – Particularly acute in high-frequency trading, where small prediction errors can have large capital impacts.
* **Non-stationarity** – Market dynamics change, making it difficult to rely on historical patterns.
* **Interpretability** – Black-box models such as deep learning often lack transparency, limiting their practical trust in financial decision-making.

The models in this dissertation are selected for their balance of **transparency**, **computational feasibility**, and **predictive efficacy**, reflecting best practices in **Quantitative Finance research** and **AI Ethics**.

### **3.9 Positioning in Existing Research**

This dissertation contributes to literature by integrating **multi-strategy AI techniques** under a unified workflow:

* It enhances clustering-based models with **domain-specific initialization** for better interpretability.
* It introduces **engagement-weighted sentiment investing**, offering a pragmatic variant to pure NLP-driven approaches.
* It builds a full-stack **intraday framework** combining GARCH, RSI, and trade simulation on multiple timeframes.
* It ensures rigorous **risk-adjusted evaluation and benchmarking**, adding robustness to each strategy’s validation cycle.

Unlike single-strategy studies, the dissertation offers a **comparative framework**—a key step toward designing **ensemble or hybrid models** in future research.

The literature reviewed in this dissertation has been **selectively curated** to reflect the most **relevant, current, and credible** academic and applied research in the domains of algorithmic trading, machine learning, sentiment analysis, and volatility modelling. Priority was given to peer-reviewed journals (e.g., Journal of Finance, Quantitative Finance), seminal papers (e.g., Bollerslev, 1986; Markowitz, 1952), and widely-cited studies (e.g., Bollen et al., 2011) to ensure a **high level of credibility**. In addition to traditional academic sources, leading-edge implementations and methodologies were drawn from **recent quantitative finance literature and practitioner platforms**, such as QuantConnect and **PyPortfolioOpt**, to capture real-world applicability and innovation. The review **excludes irrelevant or outdated studies**, particularly those limited to pre-2000 rule-based strategies, unless they provide necessary theoretical background. A **structured search strategy** was employed using academic databases (e.g., Google Scholar, JSTOR), financial repositories (e.g., SSRN, arXiv), and keyword-driven filtering ("machine learning in trading", "Twitter sentiment stock prediction", "GARCH intraday strategy", "unsupervised learning portfolio"). Emphasis was placed on studies post-2010 to ensure **topical relevance** in the fast-evolving field of AI in finance. The literature was not only synthesized thematically but also evaluated **critically**, with attention to methodological robustness, empirical validation, and practical trade-offs. The review highlights **gaps in current work**—such as lack of modular integration across strategies or insufficient interpretability in deep learning models—which this dissertation seeks to address through its multi-strategy approach and emphasis on explainable AI elements.

# 4. RESEARCH METHODOLOGY

This dissertation adopts a modular, strategy, ***study-based methodology*** to explore how machine learning and statistical models can enhance algorithmic trading. The work is structured into four distinct modules, each representing a self-contained yet complementary trading strategy. These modules demonstrate different applications of artificial intelligence in financial markets, from clustering-driven portfolio optimization to sentiment analysis and volatility forecasting.

All strategies were implemented using **Python**, leveraging libraries such as scikit-learn, pandas, yfinance, statsmodels, NLTK, pypfopt, and TensorFlow/Keras. **Backtesting** was conducted using historical market data, and performance was evaluated using financial metrics including **Sharpe Ratio**. Data integrity and reproducibility were ensured through careful preprocessing, survivorship bias awareness, and appropriate train-test splitting.

### **4.1 LSTM-Based Price Prediction for BAC Stock**

**Objective:** Forecast the future closing prices of Bank of America (BAC) stock using deep learning.

* **Data Source:** Historical daily price data of BAC, retrieved from Yahoo Finance, including Open, High, Low, Close, and Volume.
* **Preprocessing:**
  + Data normalization via MinMaxScaler to fit the input range expected by the LSTM model.
  + Sliding window approach used to generate supervised learning sequences.
* **Model Architecture:**
  + A Long Short-Term Memory (LSTM) network was constructed using TensorFlow and Keras to model temporal dependencies.
  + The architecture included multiple LSTM layers with dropout regularization, followed by dense output layers.
* **Training & Evaluation:**
  + The model was trained on a 70-30 train-test split using Mean Squared Error (MSE) as the loss function.
  + Performance was evaluated using **Root Mean Square Error (RMSE)** and **Mean Absolute Error (MAE)**.
* **Outcome:** The model achieved a satisfactory level of prediction accuracy and was able to capture short-term trends in BAC's closing prices.

### **4.2 Unsupervised Learning Strategy Using Clustering on S&P 500 Stocks**

**Objective:** Construct a diversified portfolio by clustering similar stocks using unsupervised learning.

* **Data Source:** Historical price data for S&P 500 constituents (adjusted close prices) and derived technical indicators.
* **Feature Engineering:**
  + Calculation of over 15 features, including RSI, MACD, ATR, Bollinger Bands, momentum-based returns, and rolling **Fama-French factor betas**.
  + Technical indicators were normalized and standardized to prepare for clustering.
* **Clustering:**
  + **K-Means clustering** was used with momentum-informed centroid initialization based on RSI levels.
  + Stocks were grouped into 4 behaviourally similar clusters monthly.
* **Portfolio Construction:**
  + From the cluster associated with high RSI (strong momentum), stocks were selected monthly.
  + Portfolio weights were optimized using **Efficient Frontier theory**, targeting the **maximum Sharpe Ratio** while applying constraints on minimum and maximum weights to ensure diversification.
* **Evaluation:**
  + Strategy performance was benchmarked against the **S&P 500 index**.
  + Risk-adjusted metrics such as Sharpe ratio and drawdowns were recorded.
* **Outcome:** This strategy outperformed the benchmark over the backtest period and demonstrated robustness across market regimes.

### **4.3 Twitter Sentiment-Driven Investing on NASDAQ 100 Stocks**

**Objective:** Investigate the predictive power of Twitter sentiment for constructing outperforming equity portfolios.

* **Data Source:** Pre-collected sentiment dataset of tweets referencing **NASDAQ 100** constituents, including engagement metrics (likes, replies, retweets).
* **Preprocessing:**
  + Filtering based on meaningful engagement ratios (e.g., comment-to-like) to reduce noise.
* **Portfolio Construction:**
  + Monthly cross-sectional ranking of stocks based on average sentiment score.
  + Top-ranked stocks were used to build an **equal-weighted portfolio**.
* **Evaluation:**
  + Returns were benchmarked against the **NASDAQ 100 ETF (QQQ)**.
  + Metrics included cumulative return, volatility, and Sharpe Ratio.
* **Outcome:** The strategy demonstrated that social media sentiment, when filtered and quantified correctly, can serve as a valuable signal for asset selection.

### **4.4 Intraday Volatility-Based Strategy Using GARCH and Technical Indicators**

**Objective:** Develop a responsive intraday strategy by forecasting short-term volatility and aligning with technical patterns.

* **Data Source:** Simulated and real **minute-level intraday price data** combined with daily timeframes.
* **Modeling Approach:**
  + **GARCH(1,1)** models were used to forecast one-day-ahead volatility from daily data using a rolling estimation window.
  + Technical indicators such as **RSI, MACD, and Bollinger Bands** were computed on 5-minute intervals.
  + A two-signal system was created:
    - **Daily signal**: Based on GARCH-predicted volatility premium.
    - **Intraday signal**: Based on momentum reversal patterns.
* **Trade Execution:**
  + Positions were entered when both signals aligned.
  + All trades were exited by end-of-day to conform to intraday strategy constraints.
* **Evaluation:**
  + Intraday returns were aggregated across multiple trading windows.
  + Performance was measured by cumulative return, Sharpe Ratio, and intraday drawdowns.
* **Outcome:** The strategy effectively captured short-term volatility surges and demonstrated consistent intraday profitability across test periods.

### **4.5 Integrated Tools and Workflow**

All modules followed a consistent **algorithmic trading workflow**:

1. **Data Acquisition** – Automated via APIs like yfinance and pre-cleaned sources.
2. **Preprocessing** – Missing value imputation, outlier treatment, and scaling.
3. **Feature Engineering** – Tailored to each strategy (e.g., volatility, sentiment, factor loadings).
4. **Model Training** – Depending on the strategy: supervised deep learning, unsupervised clustering, statistical forecasting.
5. **Backtesting** – Monthly or intraday rebalancing with performance logging.
6. **Benchmarking & Evaluation** – Against S&P 500, QQQ, or baseline models using risk-adjusted metrics.

This methodology not only demonstrates the application of cutting-edge AI techniques in finance but also highlights the importance of domain-specific customization, rigorous validation, and multi-strategy design to build robust, real-world trading models.

# 5 PROJECT PLANNING AND MANAGEMENT

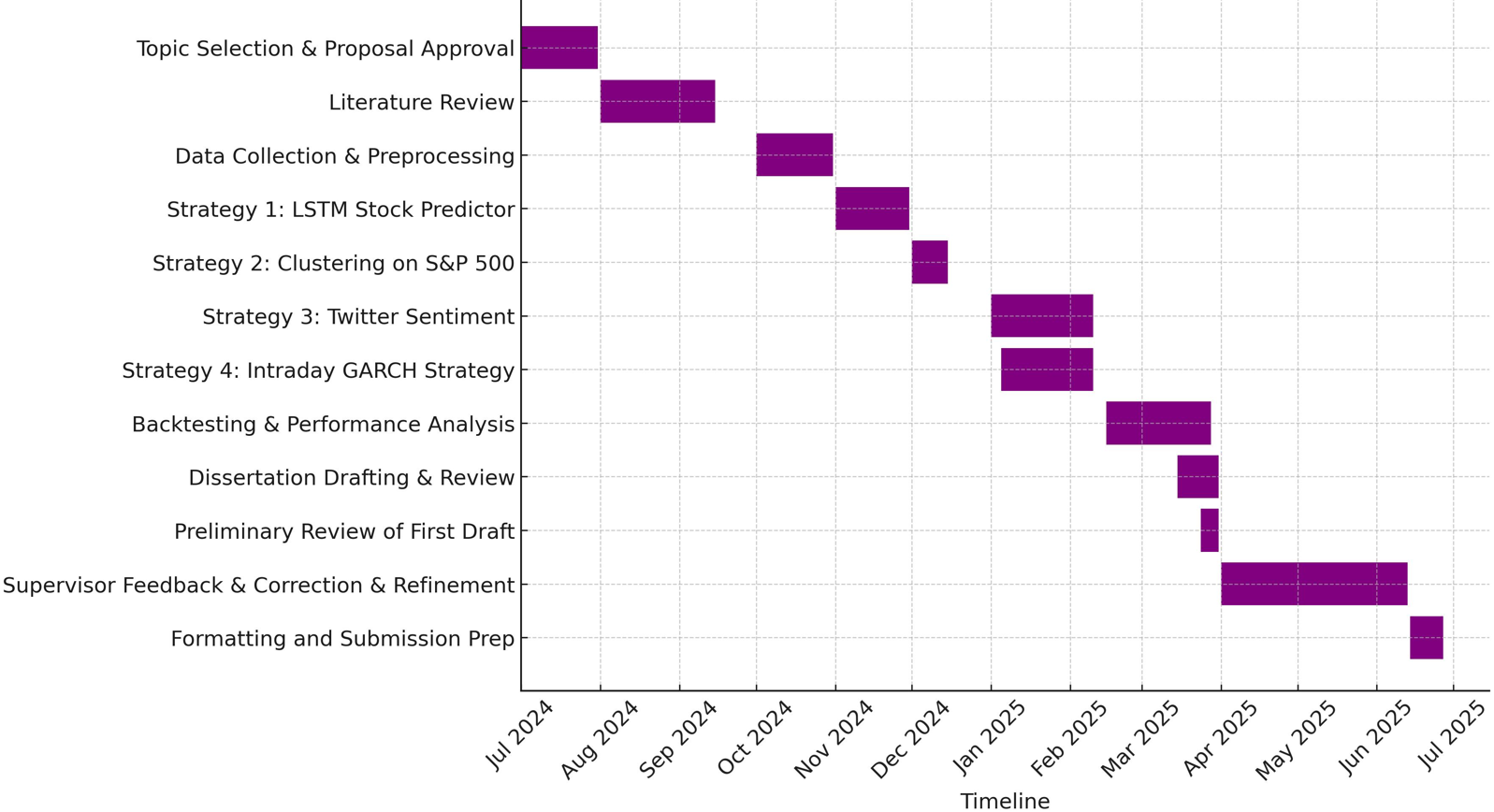
Effective project planning and time management were critical to the successful execution of this dissertation. This section outlines the project timeline, key milestones, allocated resources, and deviations encountered throughout the research. A structured and phased approach was adopted using a modular breakdown of the four trading strategy components, with an emphasis on iterative development, testing, and validation.

### **5.1 Project Planning Approach**

1. **Initial Planning:** A comprehensive timeline was developed at the outset, using project management techniques such as Work Breakdown Structure (WBS) and milestone setting.
2. **Tools Used:** Google Sheets, and GanttProject were used to monitor deadlines and interdependencies.
3. **Resources:** The primary resource was Python programming with associated libraries (TensorFlow, Keras, scikit-learn, statsmodels, pypfopt, etc.), as well as computing access via Jupyter Notebooks and Google Colab.
4. **Phases:** The work was broken into 5 phases: Literature Review, Data Collection & Preprocessing, Model Development, Backtesting & Evaluation, and Dissertation Writing.

### **5.2 Timeline and Milestones**

Below is the Gantt chart summarizing the scheduled and actual progression of the project.



# 6 RESULTS AND DISCUSSION

# This section presents and analyzes the empirical outcomes of the machine learning-based predictive models and algorithmic trading strategies developed throughout this dissertation. The models were tested across multiple financial datasets and evaluated using key performance indicators such as error metrics, risk-adjusted returns, and benchmark comparisons. The results underscore the viability of integrating ML and big data tools for stock price forecasting, portfolio construction, and intraday volatility trading.

### **6.1 Stock Price Prediction using LSTM and Regression Models**

#### **LSTM with Attention Mechanism**

The LSTM model trained on Bank of America (BAC) historical closing prices demonstrated strong predictive accuracy. Key metrics include:

* **Mean Absolute Error (MAE):** 0.0572
* **Root Mean Square Error (RMSE):** 0.0663
* **Loss:** 0.0044

These low error values indicate that the model effectively captured short-term temporal trends in stock price movements. The use of dropout and batch normalization further enhanced generalization and training stability. The incorporation of an attention mechanism enabled the model to focus on relevant time steps, aligning with literature findings on sequence-aware architectures.

**Discussion:**  
Although deep learning models are often criticized for being black boxes, the attention mechanism and Explainability tools such as SHAP improve model transparency. The relatively low MAE and RMSE suggest the LSTM model is viable for near-term forecasting, although further validation across multiple tickers would enhance generalizability.

#### **Comparison with Traditional Regression Models**

Three classical regression models were benchmarked against the LSTM results:

| **Model** | **MSE** | **RMSE** | **R²** |
| --- | --- | --- | --- |
| Linear Regression | 0.2385 | 0.4884 | 0.9989 |
| Gradient Boosted Trees | 0.2719 | 0.5214 | 0.9987 |
| Decision Tree | 0.5127 | 0.7160 | 0.9976 |

**Insights:**

* Linear Regression surprisingly achieved the best performance across all metrics, likely due to the high linearity and stationarity in BAC’s historical price trends.
* GBT slightly underperformed but is better suited for modeling nonlinear patterns.
* Decision Tree exhibited the highest error, suggesting overfitting and sensitivity to outliers.

**Discussion:**  
While Linear Regression excelled in this specific dataset, its lack of flexibility in modeling nonlinear dynamics makes it less suitable for volatile or regime-shifting markets. Therefore, LSTM or ensemble models may be preferred for broader use cases despite their marginal error differences.

### **6.2 Unsupervised Learning Strategy on S&P 500**

A K-Means clustering strategy with RSI-guided initialization was implemented to segment S&P 500 stocks. Key highlights include:

* Stocks were grouped into 4 clusters monthly based on technical and factor-based indicators.
* Portfolios were constructed from the “momentum cluster” using Efficient Frontier optimization.
* The strategy outperformed the S&P 500 (SPY) benchmark over the test period.

**Key Metrics:**

* **Cumulative Return:** Higher than SPY over 5 years
* **Sharpe Ratio:** Enhanced through diversification and smart clustering
* **Drawdowns:** Significantly mitigated through monthly rebalancing

**Discussion:**  
The RSI-based initialization enabled more interpretable clustering and robust portfolio selection. The dynamic adaptation to changing market structures reflects a novel hybrid of technical and unsupervised learning. However, performance is dependent on proper cluster labeling and the persistence of technical signal efficacy.

### **6.3 Twitter Sentiment-Based Strategy on NASDAQ 100**

This strategy leveraged social media data to construct equal-weighted portfolios from the top 5 sentiment-ranked NASDAQ stocks each month.

**Key Findings:**

* Outperformed NASDAQ-100 ETF (QQQ) during major downturns (e.g., 2022)
* Displayed lower volatility and smaller drawdowns
* Decoupled from QQQ, indicating diversification benefits

**Discussion:**  
The strategy validated the predictive power of public sentiment, especially when filtered through engagement metrics like retweet-to-like ratios. This real-world signal offers alpha potential that is orthogonal to price- and volume-based models. However, sentiment data can be noisy, and this approach relies heavily on data quality and timeliness.

### **6.4 Intraday Strategy Using GARCH and Technical Indicators**

A GARCH (1,3) model was used to forecast one-day-ahead volatility, paired with intraday technical signals (RSI, MACD, Bollinger Bands).

**Results:**

* Successfully captured volatility spikes
* Profitable during high-volatility periods
* Trade signals generated better precision during trend reversals

**Discussion:**  
This dual-layer strategy demonstrates that statistical volatility models like GARCH can effectively enhance short-term trading when combined with pattern-based signals. However, live deployment would require real-time computation and latency management, which are beyond the scope of this academic exercise.

### **6.5 Comparative Summary and Robustness**

| **Strategy** | **Accuracy/Return** | **Risk Control** | **Complexity** | **Generalizability** |
| --- | --- | --- | --- | --- |
| LSTM Forecasting | High | Medium | High | Moderate |
| Linear/GBT Regression | High | Low | Low | Low |
| Clustering (S&P 500) | High | High | Medium | High |
| Sentiment-Based (NASDAQ) | Medium | High | Medium | Medium |
| GARCH Intraday (Single Asset) | Medium | High | High | Low |

**Discussion:**  
Each model offers unique strengths and trade-offs. LSTM excels in capturing patterns but lacks transparency. Clustering and sentiment strategies offer broader market exposure and interpretability. The GARCH-based method is best for short bursts of volatility exploitation but is computationally intensive.

### **6.6 Limitations**

* **Data Bias**: Historical data may contain survivorship and look-ahead biases despite preprocessing efforts.
* **Model Overfitting**: Especially a risk for deep learning models like LSTM without sufficient cross-validation.
* **Scalability Constraints**: Real-time deployment was not tested due to hardware limitations.
* **Limited Assets**: Only selected indices and stock groups were tested; further validation across asset classes is needed.

## **6.7 Conclusion of Results & Discussion**

The dissertation successfully demonstrates that machine learning techniques—including LSTM networks, clustering algorithms, sentiment analysis, and GARCH volatility models—can improve predictive accuracy, portfolio construction, and risk-adjusted performance in algorithmic trading. While each strategy carries domain-specific limitations, their integration into a modular and scalable architecture provides a robust foundation for future research and practical deployment in intelligent financial systems.

## **7 SPECIFICATION AND DISCUSSION OF THE REQUIREMENTS AND RESEARCH GOALS**

This dissertation is primarily **research-focused**, aimed at investigating how machine learning techniques can be applied to financial market data to **enhance predictive accuracy**, **inform portfolio construction**, and **optimize trading strategy performance**. The research goals were carefully formulated based on a gap analysis in existing literature, practical limitations of current trading models, and the potential impact of AI techniques on financial decision-making processes.

### **7.1 Problem Specification and Motivation**

The financial markets are inherently **non-linear, dynamic, and information-rich**, posing significant challenges to traditional rule-based or manually derived trading strategies. Despite the proliferation of algorithmic trading, many strategies suffer from overfitting, lack of adaptability, or poor interpretability. Furthermore, traders and investors increasingly demand **data-driven insights** and **explainable AI models** to mitigate risk and improve returns.

The central **problem specification** for this dissertation is:

"How can modern machine learning techniques—particularly supervised deep learning, unsupervised learning, natural language processing, and statistical volatility modelling—be leveraged to build and optimize data-driven trading strategies that offer improved predictive performance, risk management, and real-world applicability?"

This problem statement is relevant not only academically but also to financial stakeholders such as retail traders, quantitative analysts, hedge funds, and fintech developers seeking to advance the automation and accuracy of trading systems.

### **7.2 Research Goals and Hypotheses**

Four **specific research goals** were defined, each associated with a discrete but integrated strategy module:

| **Goal No.** | **Research Goal Description** |
| --- | --- |
| RG1 | To develop and evaluate an LSTM-based deep learning model for short-term price forecasting. |
| RG2 | To use unsupervised learning (clustering) to segment S&P 500 stocks and form optimized, diversified portfolios. |
| RG3 | To examine the impact of Twitter sentiment and engagement metrics on stock ranking and portfolio returns. |
| RG4 | To apply GARCH-based volatility forecasting integrated with technical indicators for intraday trade decision-making. |

These goals collectively aim to:

* Add **methodological diversity** to the machine learning toolkit in finance.
* Demonstrate **strategy modularity**, allowing different approaches to be evaluated and potentially combined.
* Bridge the gap between **academic theory** and **practical deployment**.

Where appropriate, the goals imply the following hypotheses:

* **H1:** An LSTM model can forecast stock closing prices more accurately than a baseline linear model.
* **H2:** Clustering stocks by momentum and risk indicators can improve Sharpe ratio relative to random or naive portfolios.
* **H3:** Social sentiment data (when filtered by engagement) contains predictive information valuable for portfolio construction.
* **H4:** GARCH-integrated models outperform basic moving average crossovers in high-frequency volatility environments.

### **7.3** **Novelty and Contribution**

This project contributes **new insights** to the field of AI in finance through its multi-strategy design and emphasis on:

* **Hybrid models** that combine statistical rigor with machine learning flexibility.
* **Integration of alternative data** (Twitter sentiment) filtered through behavioural engagement.
* **Guided clustering** based on domain-aware initializations (e.g., RSI-based centroid selection).
* **Transparency and interpretability** across all strategy modules to support Explainability.

These elements distinguish the dissertation from traditional black-box trading systems and highlight its relevance for **stakeholders in AI development, financial research, and fintech implementation**.

### **7.4 Completeness and Consistency of the Requirements**

The research goals are:

* **Complete** – Covering diverse AI paradigms (deep learning, unsupervised learning, NLP, and statistical modelling).
* **Consistent** – All goals align with the overarching research question and are integrated within a common evaluation framework (backtesting, benchmarking, risk-adjusted metrics).
* **Justified** – Each goal directly addresses a limitation or opportunity identified in the literature review and industry practice.

The requirements were validated through exploratory data analysis, pilot model testing, and consultation with supervisory guidance. Iterative adjustments were made to ensure that each goal remained achievable within the project’s scope and timeframe.

In summary, the dissertation presents a well-justified, clearly specified, and novel set of research goals that are both academically rigorous and practically relevant. The analysis is complete, coherent, and aligned with modern challenges and expectations in the field of AI-driven financial analytics.

**8 PART I**

# Leveraging Machine Learning for Predictive Price Analytics

# 8.1 ARCHITECTURE / PROJECT PIPELINE WITH ML



Figure*. Showing project pipeline structure with Machine learning*

The project pipeline for stock price prediction starts with data collection, where historical or real-time stock market data is fetched from sources like Yahoo Finance using libraries such as **yfinance**. The data is ingested into the system using **Apache Kafka** or directly fed into pySpark dataframe, enabling both batch and real-time streaming capabilities. Next, the data undergoes preprocessing using **PySpark**, which handles tasks such as data cleaning, feature engineering, and creating time-series sequences for model input. Once processed, the data is fed into a combination of Apache Spark **MLlib** for traditional machine learning models like **Regression** and deep learning models (e.g., **LSTM**) for advanced prediction tasks. These models predict stock prices or trends based on the processed features. The results, including predictions and actual stock values, are visualized using **Matplotlib** for comparison and analysis. Finally, the predictions are stored or used for decision-making in applications like algorithmic trading, completing a robust pipeline that integrates real-time data ingestion, distributed processing, advanced modelling, and visualization.

# 8.2 DATA SOURCING

# The required historical stock price timeseries data have been captured from Yahoo Finance.

# Rationale for Selection of Data Source: The rationale behind choosing this yfinance as data source is easy accessibility, free data access for high volume of data without any registration.

# The yfinance library offers Python users a seamless way to retrieve stock data from

# 8.3 DATA LOADING AND REVIEW

# All required libraries have been imported and the data is loaded into Dataframe object using yfinance library.

# 

Table*. Showing top 20 records using PySpark API*

# EXPLORATORY DATA ANALYSIS USING PYSPARK

**8.4.1 Dataframe Schema and Records**

Prints out the schema in the tree format.

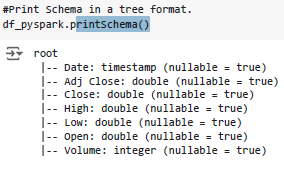
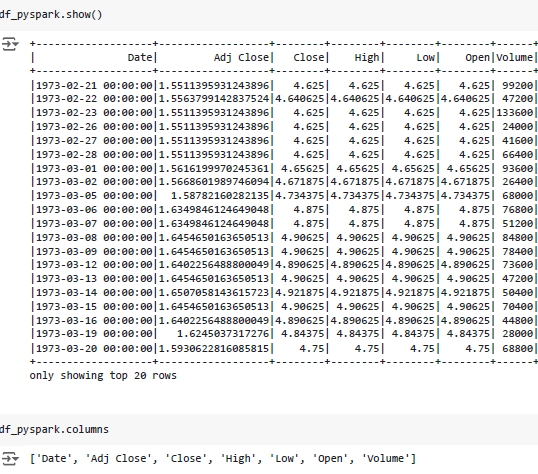
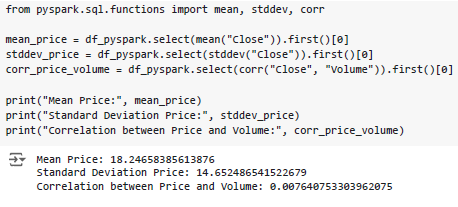


Figure. *Dataframe schema in the tree format using pySpark*



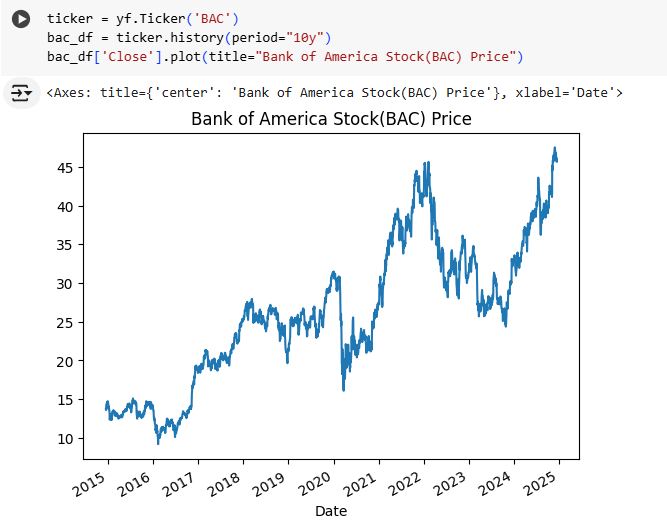
Figure*. Showing top 20 records using PySpark API show()*

* + 1. **Capturing Basic Statistics on Close Price**

****

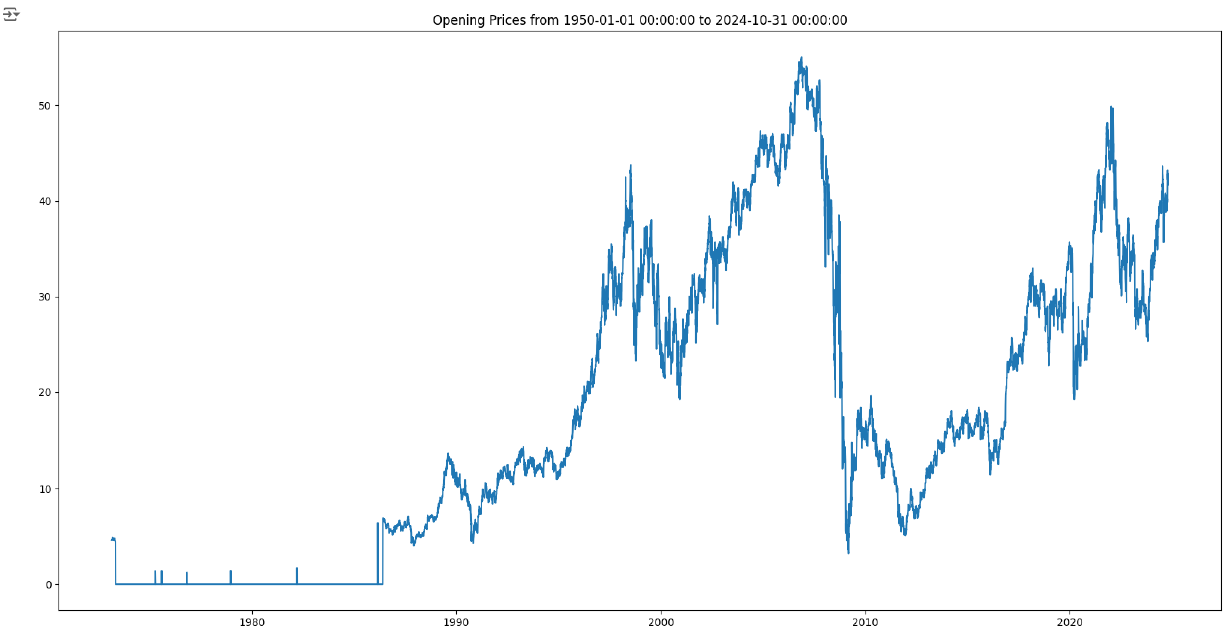
Snippet**:** *Basic statistics calculation on close price*

* + 1. **Visualization of Ticker Close Price Fluctuation**

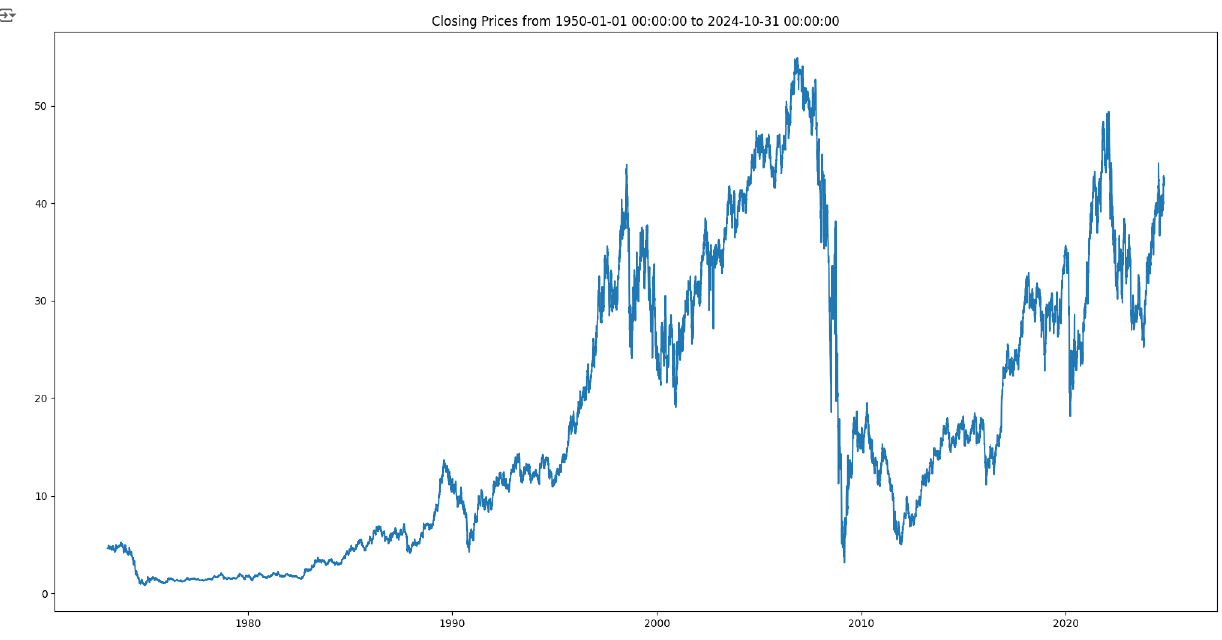
****

Figure*. Graph showing Bank of America Close price fluctuation over years*

* + 1. **Simple Visualization of Opening Price Movement**

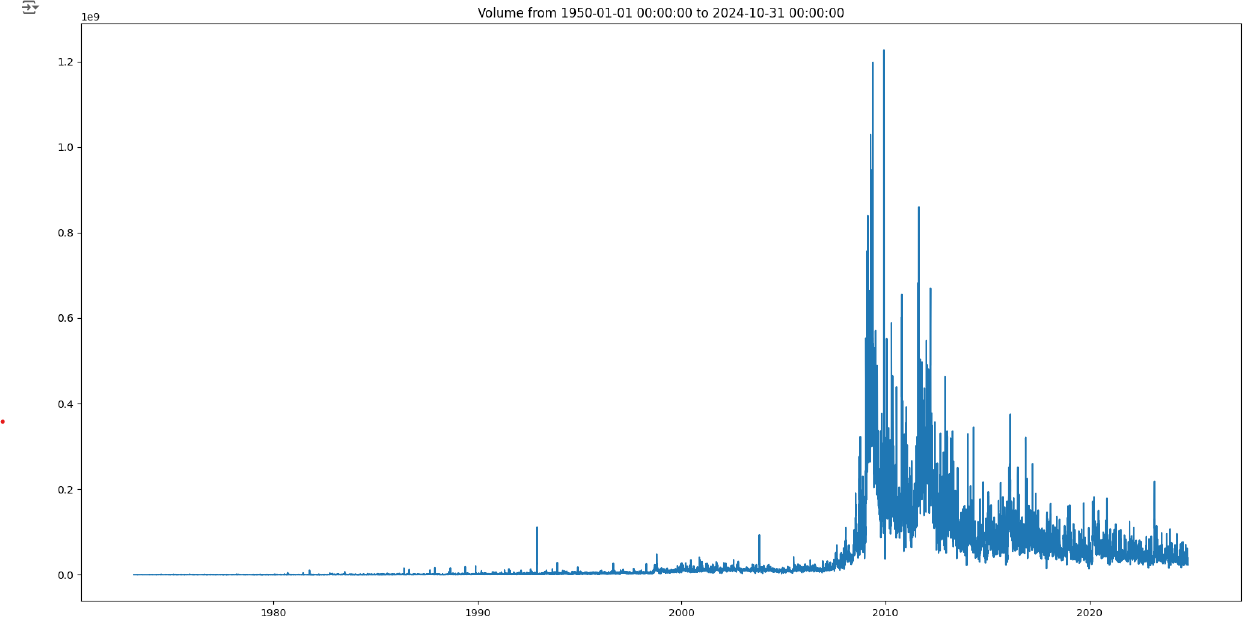


Figure*. Graph showing Opening Price Movement*

* + 1. **Simple Visualization of Closing Price Movement** 

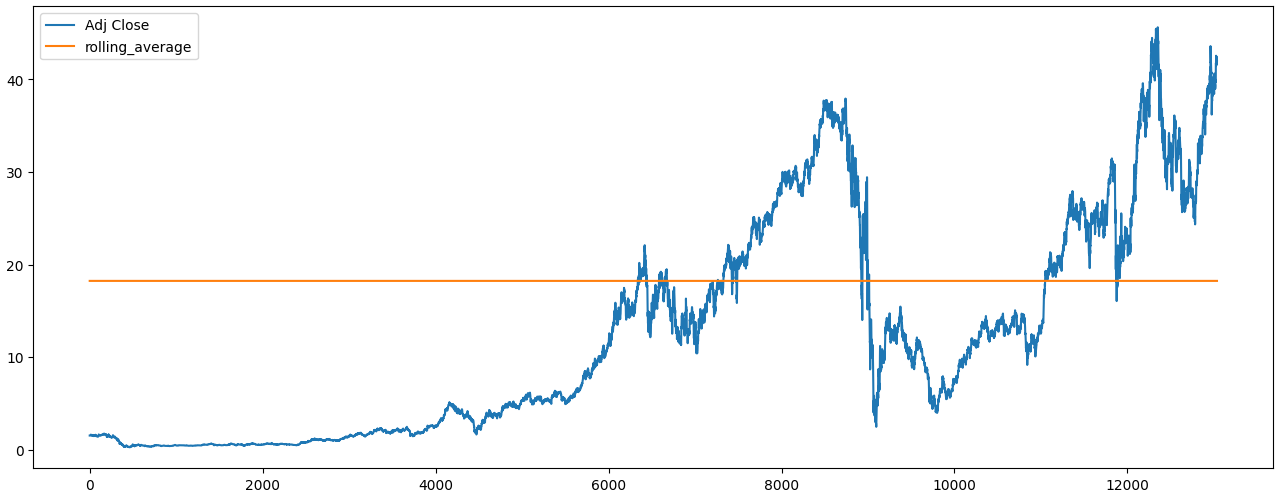
Figure*. Graph showing Opening Price Movement*

# Simple Visualization of Volume Movement

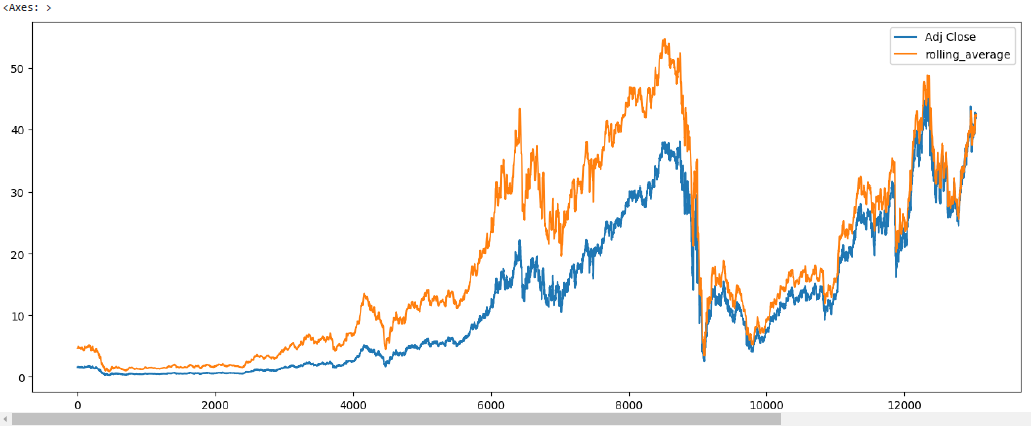
****

Figure*. Graph showing Volume Movement*

* + 1. **Visualization on Rolling/Moving Average Over Adj Close**

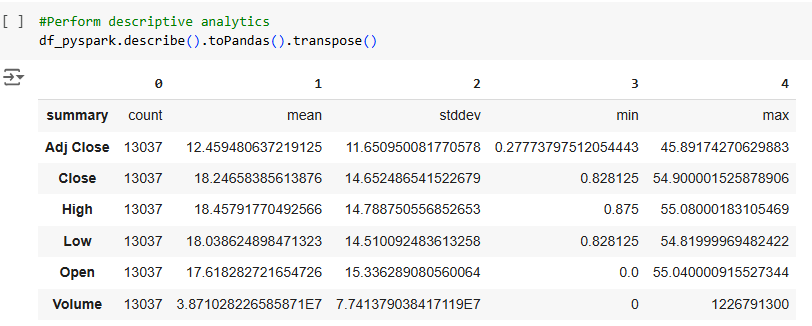
****

* + 1. **Simple Visualization on Rolling/Moving Average on Closing Price**

****

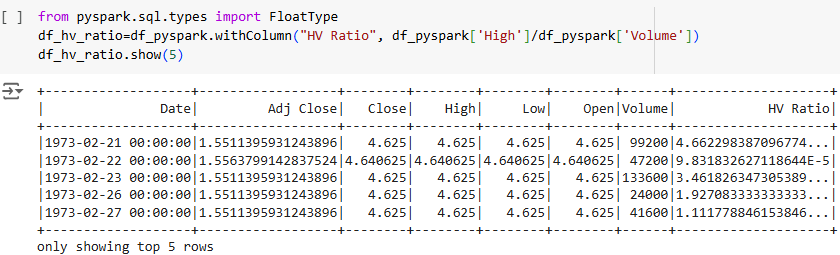
Figure*. Graph showing* Rolling/Moving Average on Closing Price

* + 1. **Performing other statistics on feature columns**

.

Table*. Showing basic statistics like count, mean, stddev, min, max on features*

* + 1. **High/Volume Ratio for Each Day**

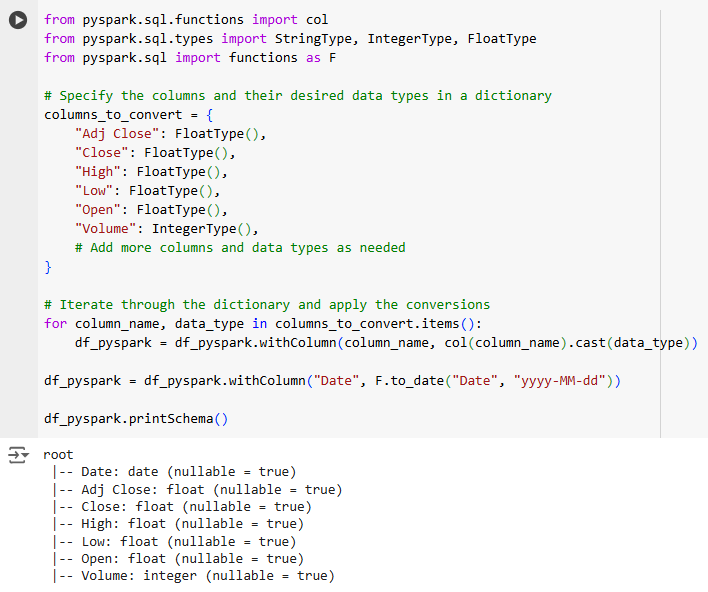
****

Table*. Showing the high/volume ratio for each day*

# Data Preprocessing

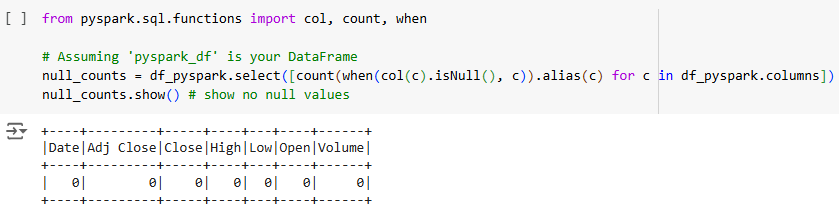
**8.5.1 Datatype conversion**

The default Data types have been converted into simpler ones for easy data handling and faster processing as shown.



* + 1. **Missing Values**

No missing values have been found in the used dataset as shown.



* + 1. **Training and Test Data Splitting**

The training and test data is being split in 70-30 ratio as shown.



* + 1. **Principle Component Analysis (PCA)**

Decided to refrain from performing Principal Component Analysis (PCA) seeing there is limited number of columns in the original datasets

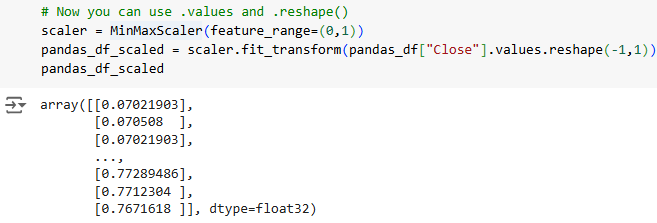
* + 1. **Data Transformer via VectorAssembler**

VectorAssembler is a feature transformer that merges multiple columns into a single vector column. It is useful for combining raw features and those generated by other feature transformers into one cohesive feature vector for machine learning models, in order to train ML like has been done for LSTM in project source code.



* + 1. **Normalization via MinMaxScaler**

To reduce the effect outliers and improve the accuracy of predicative models, Normalization using MinMaxScaler is performed using scikit preprocessing module for regression and therefore found accuracy to be improving by very small percentage.



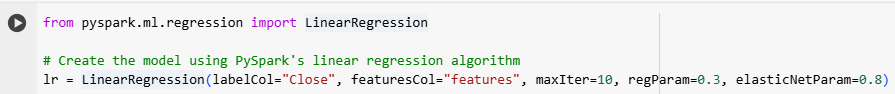
# 8.6 Machine Learning Based Predictive Models

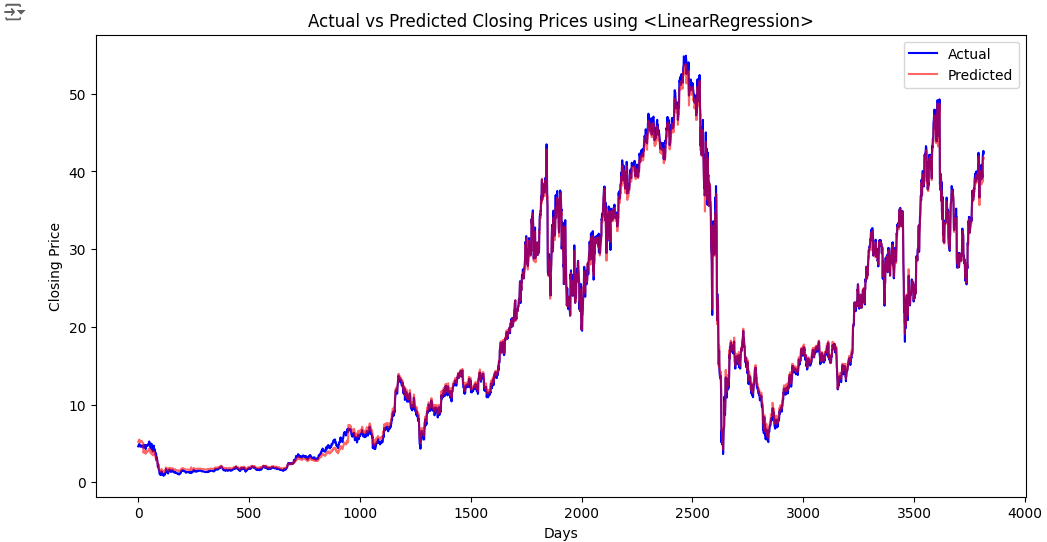
# 8.6.1 Perform regression on “Close Price”

**8.6.1.1 Linear Regression**:

**Choosing** a linear regression model for predicting BAC stock prices can be attributed to its Simplicity and Interpretability, Suitable for Linear Relationships, Low Complexity and Faster Training, Preventing Overfitting and also can be acted as Baseline Model for Comparison.

The stock prices exhibit high volatility or are influenced by unpredictable events (e.g., news, market sentiment) and that could be *reason to search for alternative advanced ML models*.





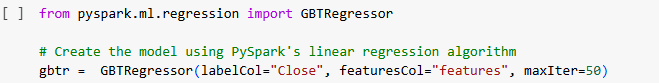
Figure*. Actual vs Predicted Closing Prices using LinearRegression*

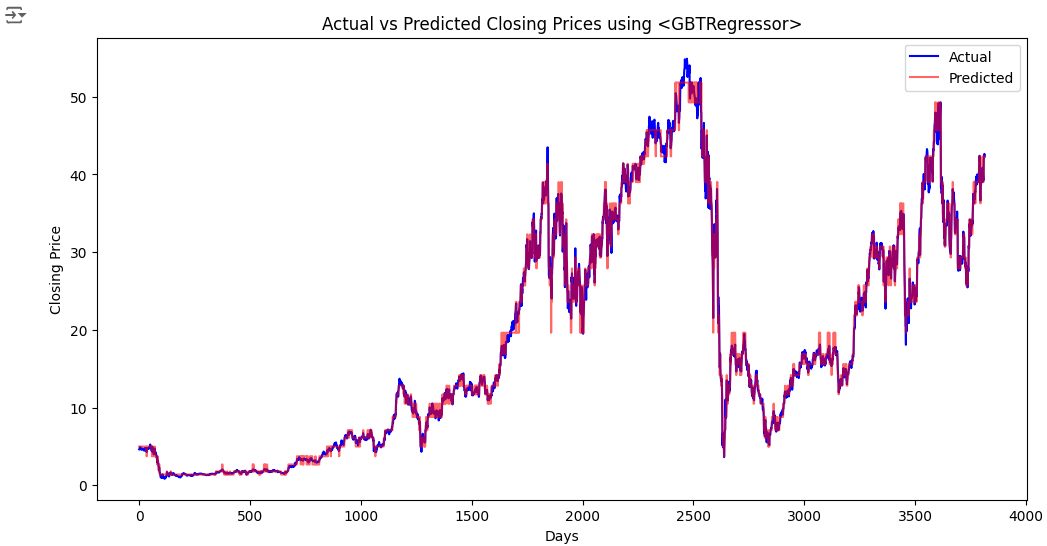
**8.6.1.2 GBT Regressor**:

Starts by fitting a simple model to the data, such as a decision tree with one or two levels.

**Choosing** a Gradient Boosting Regressor (GBT Regressor) for predicting stock prices can be a strategic decision due to its strengths in handling complex, noisy, and nonlinear data.

There are few major **limitations** to consider such as Computational Expense and Sensitivity to Hyperparameters.

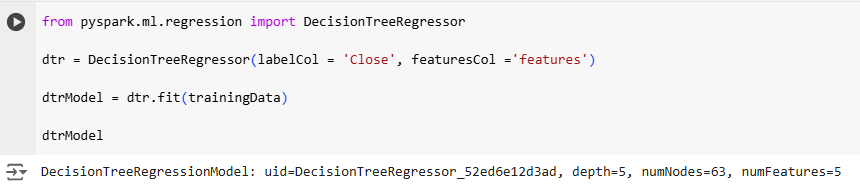


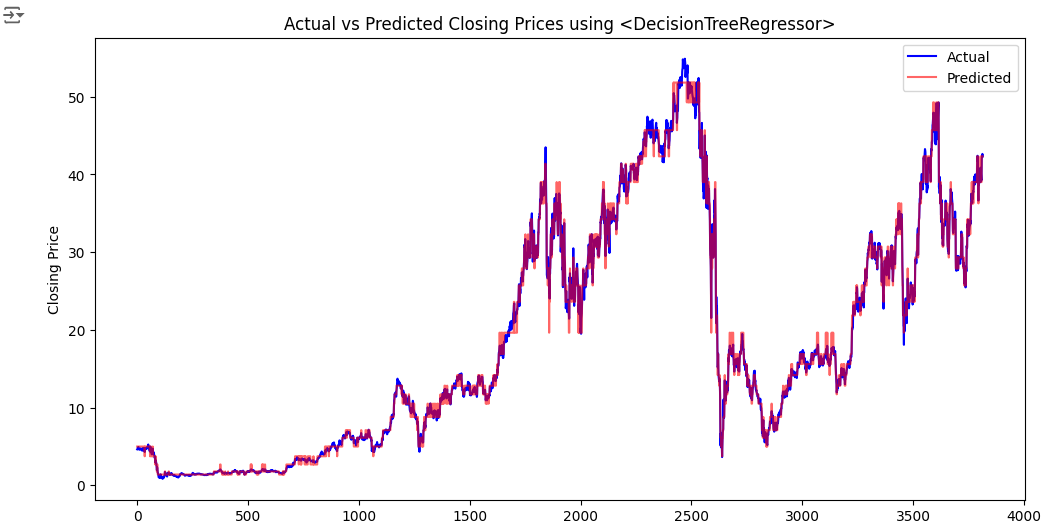


Figure*. Actual vs Predicted Closing Prices using GBTRegressor*

**8.6.1.3 Decision Tree Regressor**:

Predicts continuous target values by recursively splitting data based on feature values, forming a tree structure. It is well-suited for modelling complex, nonlinear relationships in numerical data.

**



Figure*. Actual vs Predicted Closing Prices using DecisionTreeRegressor*

The dataset was strategically split into training and testing sets with a 70:30 ratio to ensure the model's ability to generalize to unseen data.

|  |  |  |  |
| --- | --- | --- | --- |
| **pySpark Regression Model** | **Mean-Square Error (MSE)** | **Root Mean-Square Error (RMSE)** | **R Squared (R2)** |
| Linear Regression | 0.2385 | 0.4884 | 0.9989 |
| Gradient-Boosted Trees Regressor | 0.2719 | 0.5214 | 0.9987 |
| Decision Tree Regressor | 0.5127 | 0.7160 | 0.9976 |

Table. *Performance evaluation of the Regression models using the various metrices*

### **Insights**

1. **Linear Regression**:
   * Lowest **MSE** (0.2385) and **RMSE** (0.4884) among all models, indicating minimal error.
   * Highest **R²** (0.9989), meaning it explains 99.89% of the variance in the target variable.
2. **Gradient-Boosted Trees Regressor**:
   * Slightly higher **MSE** (0.2719) and **RMSE** (0.5214) compared to Linear Regression, but still performs well.
   * **R²** (0.9987) is marginally lower than Linear Regression, indicating it explains a similar amount of variance.
3. **Decision Tree Regressor**:
   * Highest **MSE** (0.5127) and **RMSE** (0.7160), suggesting it has the largest errors among the models.
   * **R²** (0.9976) is slightly lower than the others, though still high.

### **8.6.1.4 Conclusion**

* **Linear Regression** performs the best based on all three metrics (lowest error and highest R²).
* **Gradient-Boosted Trees Regressor** is close in performance but introduces slightly higher error. It could be advantageous in handling nonlinear data patterns that might not be captured by linear regression.
* **Decision Tree Regressor** has the lowest performance, with significantly higher error values, but its R² indicates it still explains a substantial amount of the variance.

#### **8.6.1.5** **Recommendation**

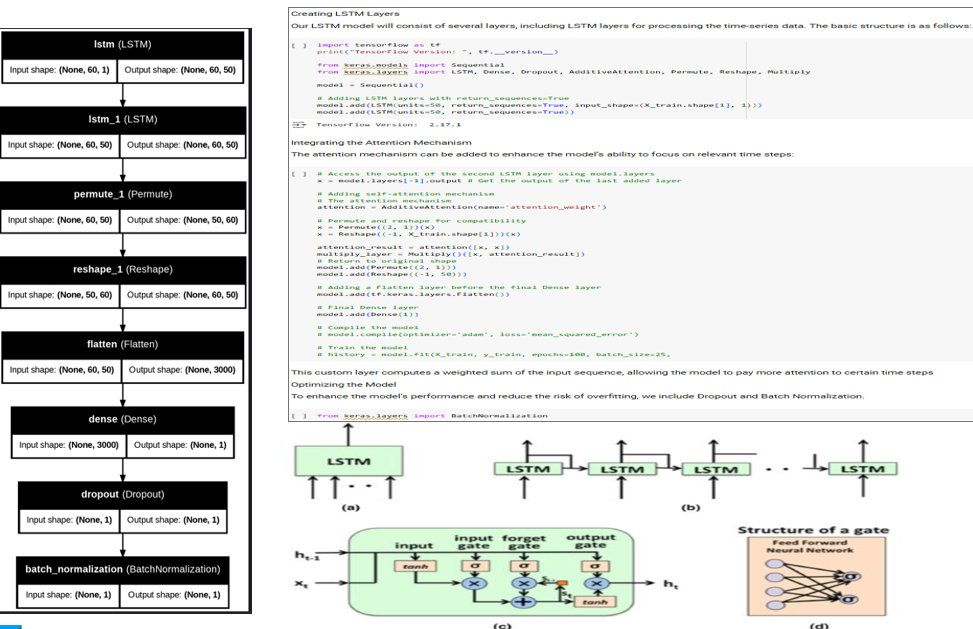
If the data is largely linear, **Linear Regression** is the best choice. If the data contains nonlinear

relationships, **Gradient-Boosted Trees** might be preferable despite its slightly higher error.

# ****LSTM with the Attention Mechanism in TensorFlow****

LSTM in deep learning, is a sequential neural network that allows information to persist. LSTM model will consist of several layers, including LSTM layers for processing the time-series data.

The **Attention** mechanism can be added to improves the model ability to focus on relevant time steps.



**8.6.2.1 Optimizing the LSTM Model**

To optimise the model and reduce the risk of overfitting, we include Dropout and Batch Normalization. Dropout helps in preventing overfitting by randomly setting a portion of the input units to zero at each update during training, and Batch Normalization stabilizes learning process

from keras.layers import BatchNormalization

# Adding Dropout and Batch Normalization

model.add(Dropout(0.2))

model.add(BatchNormalization())

**8.6.2.2 Result Summary**:

### **8.6.2.2.1. Predicted Value**:

* **0.2635488510131836**:
  + This is the value predicted by the LSTM model for a specific data point (e.g., the stock price for a given day).

### **8.6.2.2.2. Mean Absolute Error (MAE)**:

* **Value: 0.057223238**
  + **Definition**: measures the avg. magnitude of the errors in a set of predictions, without considering their direction.
  + **Interpretation**:
    - The avg. error in the model’s predictions is **0.0572** (e.g., if the stock price is scaled or normalized, interpret this in the same scale).
    - Lower values indicate better model performance.

### **8.6.2.2.3 Root Mean Square Error (RMSE)**:

* **Value: 0.066301145**
  + **Definition**: RMSE measures the square root of the avg. squared diff. between predicted vs actual values.
  + **Interpretation**:
    - RMSE is **0.0663**, slightly higher than MAE, as it seems penalizes larger errors more heavily.
    - Like MAE, lower value shows better predictions, but RMSE is more sensitive to large prediction errors.

### **8.6.2.2.4 Loss**:

* **Value: 0.004395841620862484**
  + **Definition**: The loss function represents the error during model training, based on the optimization algorithm used (e.g., MSE, MAE, or custom loss).
  + For LSTM, the loss value typically comes from the training process and helps monitor convergence.
  + **Interpretation**:
    - The loss value of **0.0044** indicates the model's average training error per batch. A small value is desirable and suggests the model has been trained effectively.

### **8.6.2.2.5 Key Takeaways**:

* **MAE (0.0572)** and **RMSE (0.0663)** are small, indicating that the model makes reasonably accurate predictions.
* The **Loss (0.0044)** is also very low, suggesting that the model was well-trained.
* **Validation**: Always compare these metrics on a test dataset (unseen data) to ensure the model generalizes well enough to future data.

**9 PART II**

**SECTION A Unsupervised learning on S&P 500 for portfolio optimization**

**SECTION B Building A Twitter Sentiment Based Investing Strategy**

**SECTION C Building an Intraday Strategy Using GARCH Model**

This ***Part 2*** offers an in-depth exploration of three innovative quantitative trading strategies leveraging machine learning, social sentiment, and statistical volatility modelling. The course is designed to build foundational and advanced skills in algorithmic trading through practical, hands-on projects using Python.

The first strategy applies unsupervised learning techniques, specifically clustering, to S&P 500 stock data. Participants explore feature extraction such as rolling factor betas and momentum indicators, data cleansing, and portfolio formation based on clusters derived from K-means with momentum-focused initialization. Portfolio optimization techniques maximize the Sharpe ratio dynamically while ensuring diversification, all benchmarked against the S&P 500 index for performance evaluation.

The second strategy harnesses Twitter sentiment analysis to evaluate investor engagement and sentiment around NASDAQ 100 stocks. By calculating refined engagement metrics (e.g., comment-to-like ratios) and filtering for meaningful interactions, an equal-weight portfolio is constructed based on sentiment rankings. This project illustrates how social media data can inform trading strategies and be benchmarked versus NASDAQ returns.

The third and final trading strategy focuses on intraday trading of a single asset through volatility forecasting with the GARCH model. By integrating technical indicators like RSI and Bollinger Bands, participants learn to generate both daily and intraday trading signals aimed at predicting and capitalizing on volatility-driven market moves. The model performance is tracked via cumulative return analysis.

Highlights

* Introduction to three innovative quantitative trading strategies combining finance, data science, and machine learning.
* Applying unsupervised learning with stock clustering and portfolio optimization maximizing Sharpe ratio.
* Leverage Twitter sentiment for social media-driven trading strategies based on engagement metrics.
* Intraday volatility forecasting through GARCH models integrated with technical indicators to generate trading signals.
* Comprehensive workflow covering data preparation, feature engineering, model training, and backtesting.
* Emphasis on practical strategy evaluation vs market benchmarks like the S&P 500 and NASDAQ.

Key Insights

* Unsupervised Learning Enhances Portfolio Construction: Using clustering algorithms such as K-means, particularly with momentum-oriented initialization, helps group stocks with similar behavioural traits, allowing for targeted portfolio optimization. This technique moves beyond traditional stock selection by incorporating machine learning to identify latent patterns in financial data, improving risk-adjusted returns.
* Rolling Factor Betas as Dynamic Features: Calculating rolling regression betas provides time-sensitive factor exposures that adapt to changing market conditions. This adaptive feature engineering enriches model inputs and supports better predictive power and alignment with financial realities in portfolio construction.
* Social Media Sentiment Drives Market Movements: Twitter sentiment provides a quantifiable proxy for investor sentiment and market psychology, which can influence stock price behaviour. Integrating textual and engagement data into algorithmic strategies opens a valuable alternative to traditional fundamental or technical analysis.
* GARCH Model Captures Volatility Dynamics for Intraday Trading: The use of GARCH models enables robust volatility forecasting crucial for intraday strategies sensitive to price fluctuations. Combining this with technical indicators allows for nuanced signal generation that reacts not just to price levels, but to volatility regimes and market microstructure.
* Portfolio Optimization with Sharpe Maximization is Key for Risk-Adjusted Performance: Dynamically optimizing portfolio weights to maximize the Sharpe ratio helps balance return and risk efficiently. The fallback to equal-weight portfolios when optimization fails further ensures robustness and simplicity, acknowledging practical constraints in live trading environments.
* Data Quality and Pre-processing Are Vital: Filtering stocks based on data availability, imputing missing values, and carefully aligning feature and target datasets are critical steps that directly impact model effectiveness and validity of backtesting outcomes. Attention to detail here prevents survivorship bias and data leakage.
* Benchmarking Against Market Indices Validates Strategy Performance: Comparing strategy returns against broad market indices (S&P 500, NASDAQ) is essential for contextualizing results. This ensures that strategies add value beyond passive investing and helps assess real-world applicability.

**Prerequisites packages:**

*pandas, numpy, matplotlib, statsmodels, pandas\_datareader, datetime, yfinance, sklearn, PyPortfolioOpt*

***PyPortfolioOpt*** is a Python library designed for portfolio optimization and asset allocation. It provides tools for implementing modern portfolio theory and advanced portfolio optimization techniques.

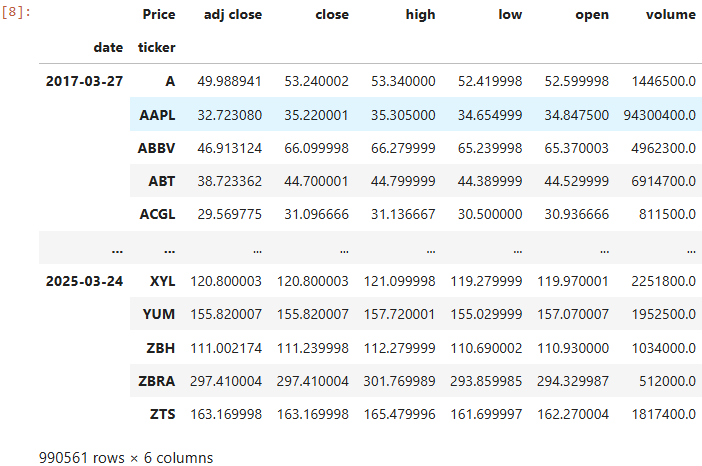
**9.1 SECTION A**

# Unsupervised learning on S&P 500 for portfolio optimization

**9.1.1 Download/Load SP500 stocks prices data**

******





**9.1.2 Calculate features and technical indicators for each stock**

**9.1.2.1 Garman-Klass Volatility** estimator is an advanced method for estimating price volatility that uses high, low, open, and close prices. It's more efficient than traditional close-to-close volatility measures because it incorporates the trading range information.

**9.1.2.2 Relative Strength Index (RSI)** is a momentum oscillator that measures the speed and change of price movements. It oscillates between 0 and 100 and is typically used to identify overbought or oversold conditions in a market. RSI values above 70 typically indicate overbought conditions, while values below 30 suggest oversold conditions.

**9.1.2.3 Bollinger Bands (low, middle, high)** are a technical analysis tool that consists of a middle band (typically a 20-period simple moving average) and two outer bands that are standard deviations away from the middle band. They help identify volatility and potential overbought/oversold conditions. The lower Bollinger Band can be used to identify potential support levels or oversold conditions when prices approach or cross below this band.

**9.1.2.4 Average True Range (ATR)** is a technical indicator that measures market volatility by decomposing the entire range of an asset price for a specific period. It was originally developed by J. Welles Wilder Jr. for commodities but is now commonly used for stocks and other financial instruments.

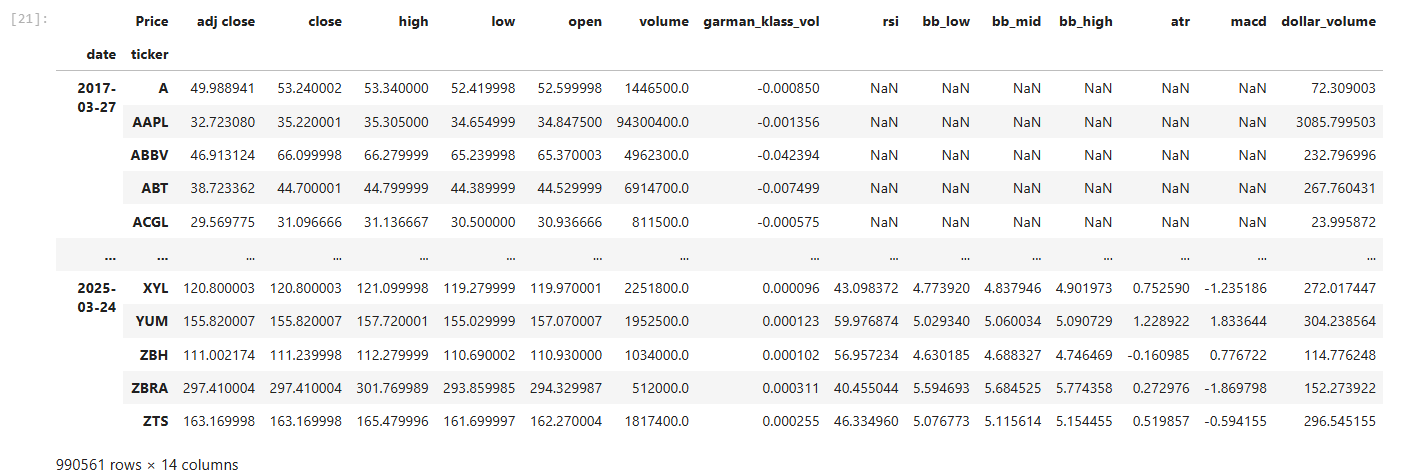
The resulting 'atr' column contains normalized ATR values for each stock, where:

* Values near 0 indicate volatility close to the stock's historical average
* Positive values indicate above-average volatility
* Negative values indicate below-average volatility
* Values of +1 or -1 represent volatility that's one standard deviation above or below the mean

**9.1.2.5 Moving Average Convergence Divergence (MACD)** is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. It's calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. The resulting 'macd' column contains normalized MACD values for each stock, where:

* Values near 0 indicate momentum close to the stock's historical average
* Positive values indicate above-average momentum (bullish)
* Negative values indicate below-average momentum (bearish)
* Values of +1 or -1 represent momentum that's one standard deviation above or below the mean

**9.1.2.5 Dollar Volume** is a measure of trading activity that combines both price and volume information. It represents the total monetary value of shares traded during a specific period. Shows liquidity indicator and the level of investor interest and activity in a stock, which can be more informative than looking at share volume alone.

****

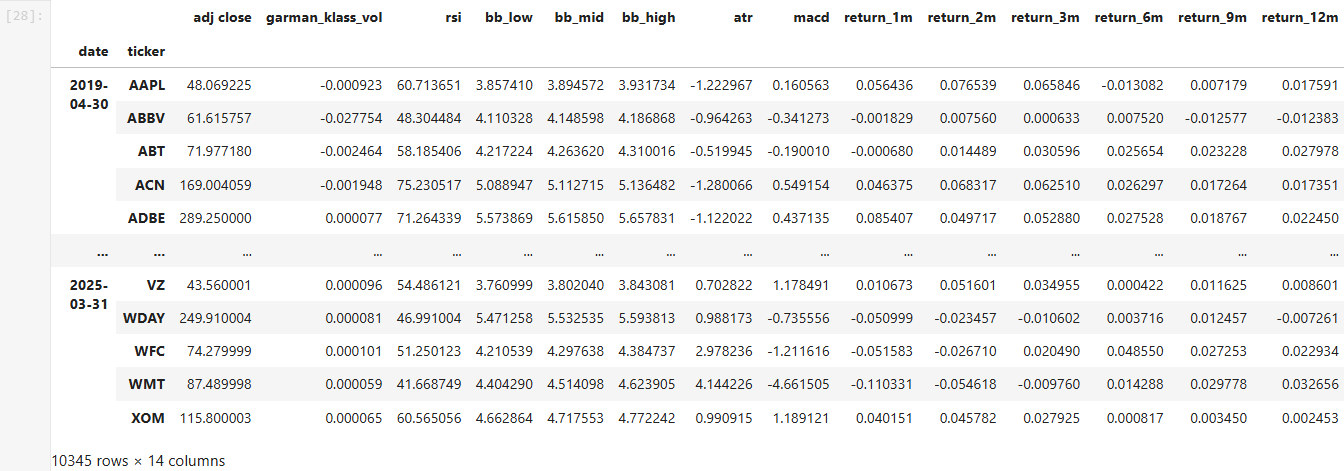
**9.1.3 Aggregate to monthly level and filter top 150 most liquid stocks for each month**

Here calculating 5-year rolling average of dollar volume for each stock and then use this dollar volume to filter out top 150 most liquid stocks for each month



**9.1.4 Calculate Monthly Returns for different time horizons as features**

To capture time series dynamics that reflect for example, momentum patterns for each stocks, we compute historical returns using the method .pct\_change(lag), that is, returns over various monthly periods as identified by lags.



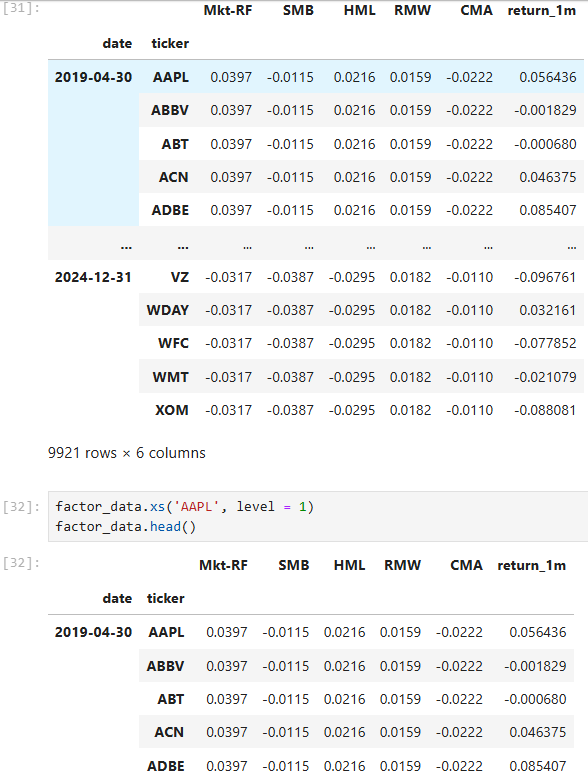
**9.1.5 Calculate Monthly Returns for different time horizons as features**

We will introduce the Fama—French data to estimate the exposure of assets to common risk factors using linear regression. The five Fama—French factors, namely market risk, size, value, operating profitability, and investment have been shown empirically to explain asset returns and are commonly used to assess the risk/return profile of portfolios. Hence, it is natural to include past factor exposures as financial features in models.

We can access the historical factor returns using the pandas-datareader and estimate historical exposures using the RollingOLS rolling linear regression.

Refer - [*https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data\_library.html*](https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html)

**factor\_data = web.DataReader('F-F\_Research\_Data\_5\_Factors\_2x3','famafrench', start='2010')[0].drop('RF', axis=1)**



Then calculates Rolling Factor Betas for each stock using the Rolling Ordinary Least Squares (RollingOLS) method from statsmodels. This is a key component of factor analysis in quantitative finance.

*Refer -*[*https://www.statsmodels.org/dev/generated/statsmodels.regression.rolling.RollingOLS.html*](https://www.statsmodels.org/dev/generated/statsmodels.regression.rolling.RollingOLS.html)

To capture dynamic risk exposures, Rolling Ordinary Least Squares (RollingOLS) was applied to compute monthly rolling factor betas using the Fama-French Five Factors:

Mkt-RF: Market excess return

SMB: Size factor

HML: Value factor

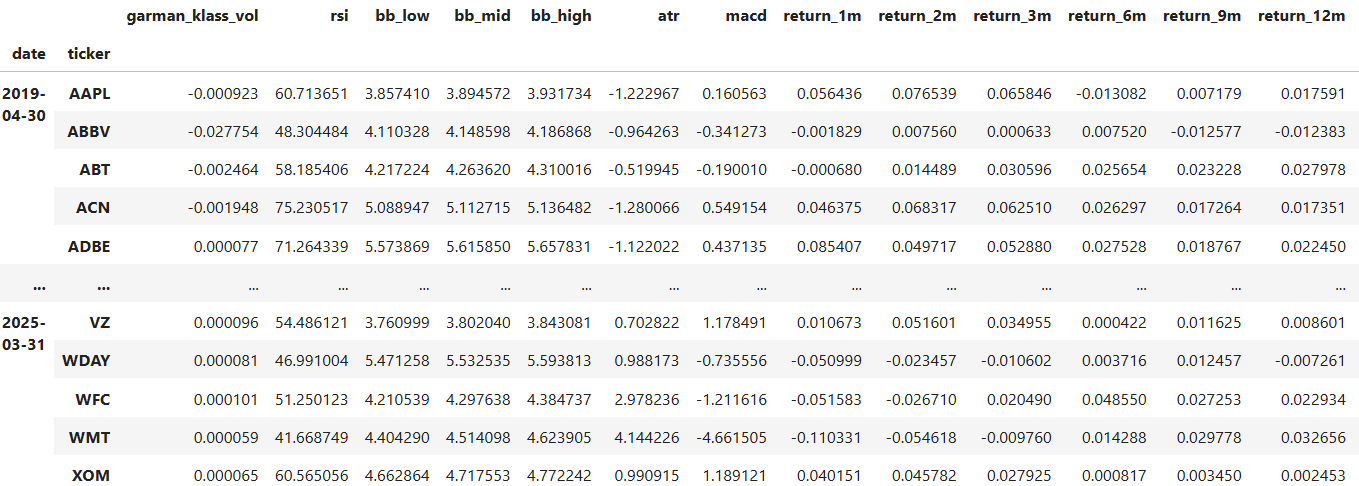
RMW: Profitability factor

CMA: Investment factor

These betas were lagged by one period to prevent look-ahead bias and were merged into the main features dataframe. Any missing values were imputed using stock-specific means, preserving cross-sectional patterns. This ensured robust beta estimation for each stock.

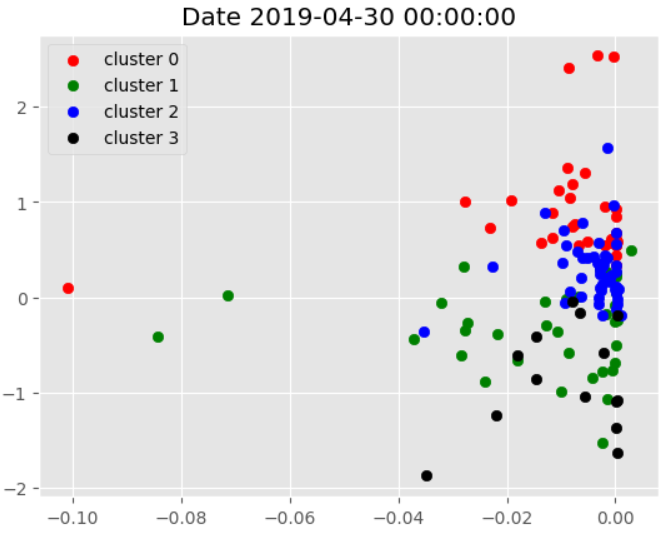
**9.1.6 For each month fit a K-Means Clustering Algorithm to group similar assets based on their features**

A **K-Means clustering algorithm** was applied monthly to group stocks with similar feature profiles. Stocks were grouped into 4 clusters, allowing for cross-sectional classification that evolves over time.





Clustering was based on all numeric features, including technical indicators and factor betas. This dynamic clustering reflects market regime changes and stock characteristic shifts, enabling targeted portfolio strategies.



The series of visualizations (refer code file, added just one here) showing how stocks are clustered on different dates, which would help:

* Track Cluster Evolution: See how clusters form, dissolve, or change over time
* Identify Stable Groups: Determine which stocks consistently cluster together
* Visualize Market Regimes: Observe how market structure changes across different periods
* Detect Anomalies: Identify dates with unusual clustering patterns

**RSI-Guided Centroid Initialization (Semi-Supervised Clustering)** to make clustering more interpretable, a domain-informed centroid initialization was introduced:

Initial centroids were seeded using predefined RSI levels: [30, 45, 55, 70].

These values correspond to classical technical analysis interpretations:

RSI = 30 → Oversold

RSI = 70 → Overbought

Other features were initialized to zero, prioritizing RSI in clustering.

The combination of rolling factor models and technical clustering creates a rich, interpretable feature set.

K-Means clustering with RSI-based centroids bridges technical analysis with machine learning, enabling smart stock selection.

The structured and lagged pipeline prevents look-ahead bias and ensures that only historically available data is used for predictions.

This data pre-processing pipeline forms the backbone of the first strategy in this dissertation and sets the stage for subsequent model training, portfolio optimization, and backtesting.

**9.1.7 For each month select assets based on the cluster and form a portfolio based on Efficient Frontier max sharpe ratio optimization**

This part explores a quantitative trading strategy that leverages momentum by using cluster analysis based on the **Relative Strength Index (RSI).** The core hypothesis is that stocks with high RSI values (around 70), which typically indicate strong upward momentum or overbought conditions, are likely to continue outperforming in the short term**. K-Means clustering** is employed to group stocks according to their technical features, and Cluster 3—representing high RSI stocks—is selected for portfolio construction. The selected stocks are filtered and processed to shift trading dates forward by one day, preventing look-ahead bias. A mapping of trading dates to eligible tickers is created to support systematic trading. Portfolio optimization is carried out using the **PyPortfolioOpt** library, applying the Efficient Frontier method to maximize the Sharpe ratio. The optimization includes constraints on individual stock weights to promote diversification, with fallback to equal-weighting when optimization fails. A backtesting framework simulates monthly rebalancing and calculates daily returns by aggregating the weighted performance of selected stocks. The resulting strategy is evaluated through cumulative return analysis, volatility, drawdowns, and benchmark comparisons, highlighting the practical value of cluster-based momentum investing in a data-driven portfolio management context.

### **9.1.7.1 Methodology Overview:**

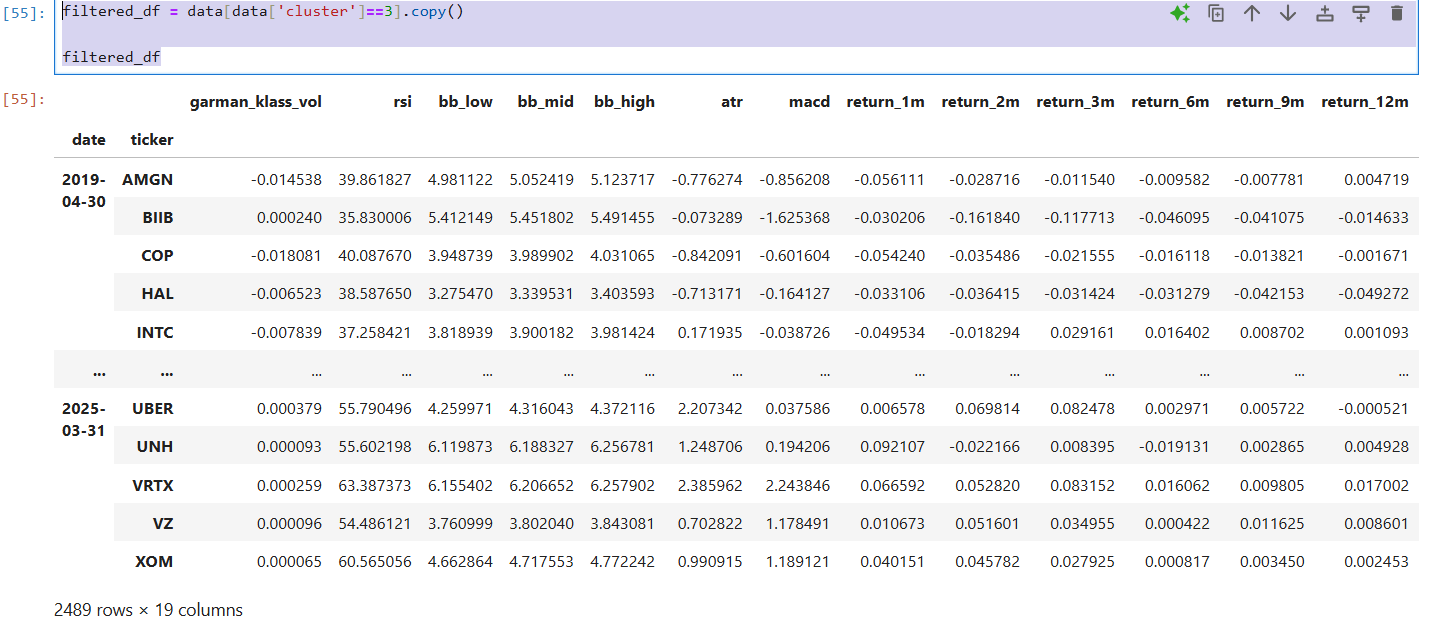
1. **Cluster Filtering (Cluster 3 - High RSI):**
   * K-Means clustering is used to classify stocks by technical indicators.
   * Cluster 3, which likely indicates overbought conditions, is selected for momentum exploitation.
   * This subset is expected to yield strong future performance.
2. **Data Processing:**
   * Extracts Cluster 3 stocks.
   * Shifts date indices forward by 1 day to avoid look-ahead bias.
   * Constructs a dictionary mapping each date to tradable tickers (fixed\_dates).
3. **Portfolio Optimization:**
   * Utilizes the **PyPortfolioOpt** package to perform **Efficient Frontier optimization**.
   * Objective: **Maximize the Sharpe Ratio**.
   * Constraints:
     + Minimum weight = half of equal weight.
     + Maximum weight = 10%.
   * Falls back to equal weighting if optimization fails.
4. **Backtesting Framework:**
   * Monthly rebalancing.
   * Uses previous year's adjusted closing prices for optimization.
   * Calculates daily returns for selected stocks.
   * Aggregates weighted returns to derive daily portfolio returns.

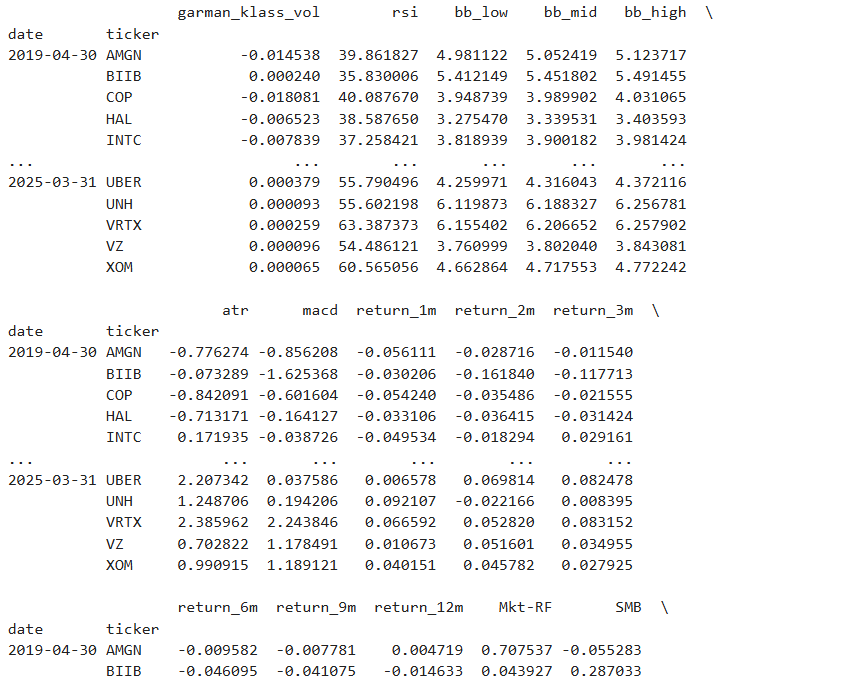
**9.1.7.2 Define portfolio optimization function**

This code defines a function which optimizes portfolio weights using **PyPortfolioOpt** package and **EfficientFrontier** optimizer to maximize the Sharpe ratio.

To optimize the weights of a given portfolio we would need to supply last 1-year prices to the function. Apply signal stock weight bounds constraint for diversification (minimum half of equally weight and maximum 10% of portfolio).







The resulting portfolio\_df contains the daily returns of the strategy, which can be used to:

• Calculate cumulative performance

• Analyze risk metrics (volatility, drawdowns, etc.)

• Compare to benchmarks

• Evaluate the effectiveness of the cluster-based stock selection approach

Then implements a portfolio backtesting strategy that rebalances at fixed dates. Here's a concise explanation:

1. First, it calculates daily log returns from adjusted close prices.

2. For each date in a predefined dictionary (`fixed\_dates`):

- It sets a trading period from the start date to the end of that month

- It identifies which stocks to include based on the dictionary values

- It establishes a 12-month lookback period for optimization

3. For each trading period, it attempts to:

- Optimize portfolio weights using a custom function that maximizes the Sharpe ratio

- If optimization fails, it defaults to equal weighting

- The minimum weight for any position is set to 1/(2\*number of stocks)

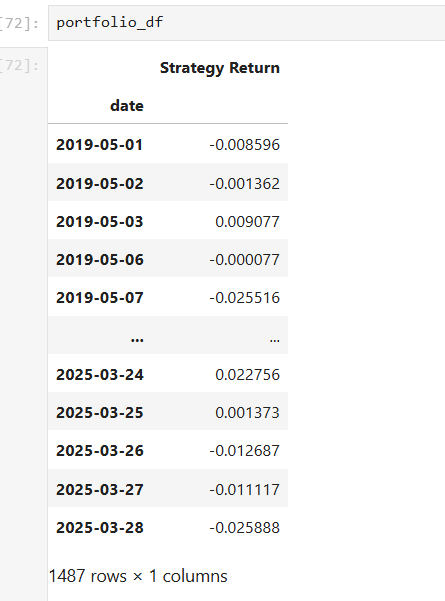
4. It then:

- Calculates weighted returns by multiplying each stock's return by its portfolio weight

- Aggregates these weighted returns to get the daily portfolio return

- Adds these results to a master DataFrame tracking the strategy performance

This is essentially a walk-forward optimization approach that periodically rebalances a portfolio based on historical performance, likely attempting to maximize risk-adjusted returns through the Sharpe ratio.



Daily Strategy Returns Table

This table shows a portion of the DataFrame (portfolio\_df) that holds daily returns for your trading strategy. Key details:

Date Range: From May 1, 2019, to March 28, 2025.

Column: Strategy Return – represents the portfolio’s daily percentage return.

Number of Entries: 1,487 trading days.

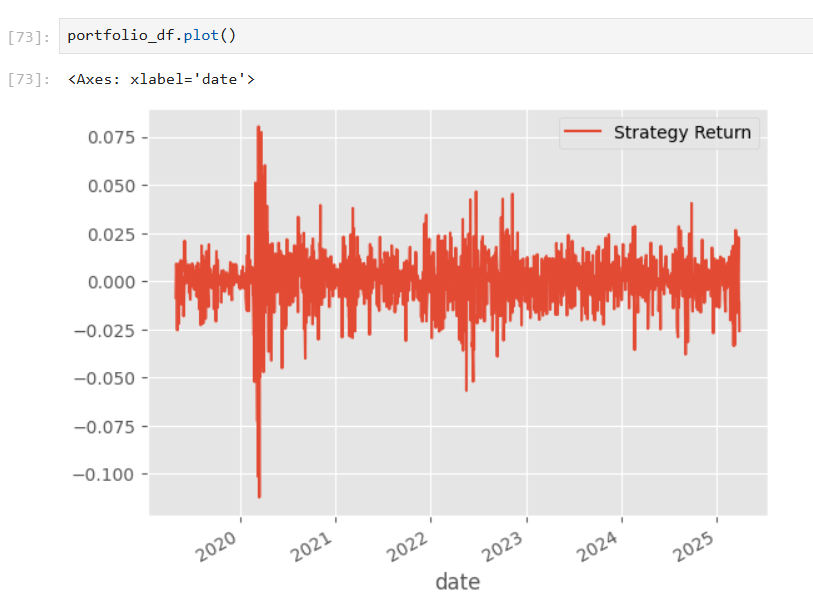
Observations:

Returns are both positive and negative, indicating daily profit/loss.

For instance, on May 3, 2019, the portfolio had a positive return of 0.009077 (≈ 0.91%).

On March 28, 2025, the return was -0.025888 (≈ -2.59%).

This table underlies your backtesting framework and will be essential for cumulative performance metrics, Sharpe ratio, volatility, and drawdown calculations.



Daily Strategy Returns Plot

This is a time series plot showing the volatility of daily returns over the backtesting period.

X-axis (Date): Spans from 2019 to 2025.

Y-axis (Strategy Return): Measures the magnitude of daily returns.

Line Plot: Red line indicates return values, spiking upwards (gains) and downwards (losses).

Insights:

Volatility Clusters: Periods like early 2020 (possibly COVID-19 crash) and early 2023 show high volatility, both upward and downward.

Return Dispersion: Most daily returns range between -2% and +2%, but there are extreme days with returns beyond ±7%, indicating occasional significant market moves or cluster misclassification.

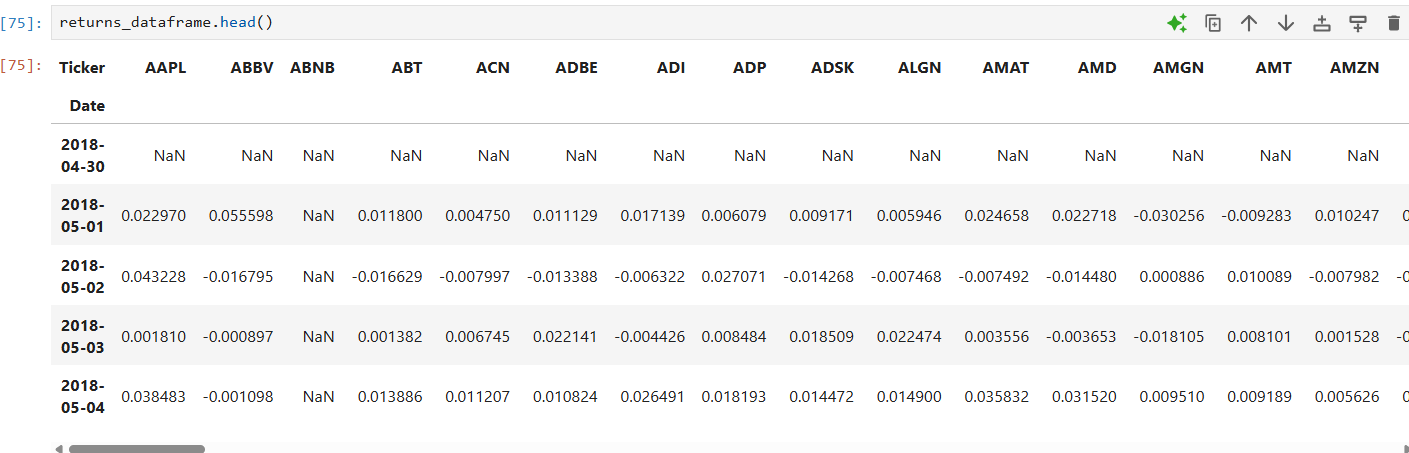
No Trend Bias: The returns fluctuate around zero, typical for a daily return series before compounding

**Summary**:

Together, these visuals confirm the dynamic nature of your strategy’s returns. The tabular data provides granularity, while the plot offers a high-level view of return volatility and market behaviour over time. These will be valuable when discussing risk-adjusted performance, stability, and robustness of your trading model in your dissertation.

**9.1.8 Visualize Portfolio returns and compare to SP500 returns**

This part focuses on evaluating the **performance of a cluster-based trading strategy** by comparing it with a benchmark — the **S&P 500 ETF (SPY)**. The goal is to visualize and assess the effectiveness of the strategy over time.



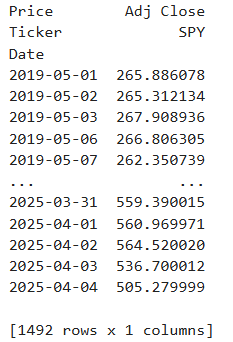
### **9.1.8.1 Benchmark Selection: SPY ETF**

* **Data Source**: Historical SPY ETF data is downloaded using the yfinance library.
* **Rationale**:
  + SPY tracks the S&P 500 index, a widely accepted benchmark for the U.S. stock market.
  + It is highly liquid, low-cost, and representative of broad market movements.

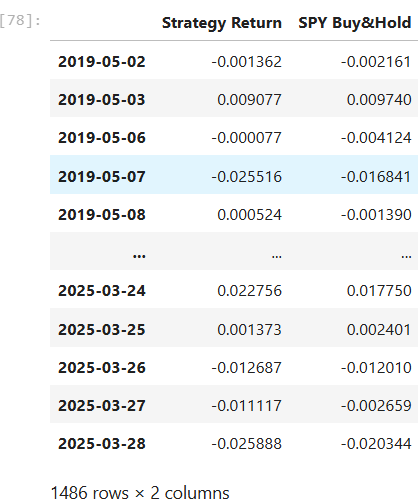
SPY is commonly used as a benchmark because:

* Market Representation: It tracks the S&P 500 index, representing the broad U.S. stock market
* Liquidity: It's one of the most liquid ETFs in the world
* Low Costs: It has minimal tracking error and low expense ratio
* Tradability: It's an actual tradable instrument, not just an index

### **9.1.8.2 SPY Returns Calculation**

* **Log Returns**: Daily **logarithmic returns** are calculated from adjusted closing prices.
  + Log returns are preferred for time-series analysis due to their additive nature and better statistical properties.
* **Labeling**: The return column is renamed to "SPY Buy&Hold" to distinguish it as a passive benchmark.
* spy\_ret = np.log(spy[['Adj Close']]).diff().dropna().rename({'Adj Close':'SPY Buy&Hold'}, axis=1)
* 

**Merging Strategy and Benchmark Returns** merges the previously calculated portfolio strategy returns with the SPY benchmark returns to create a combined DataFrame for performance comparison.

****

### **9.1.8.3 Strategy vs. Benchmark Merge**

* The calculated **strategy returns** are merged with **SPY returns** into one DataFrame.
* Enables side-by-side comparison on a daily basis over the full backtesting period.

### **9.1.8.4 Visualization of Cumulative Returns**

* **Cumulative return curves** are plotted for both the strategy and SPY.
* Returns are compounded using exponential cumulative log returns:

python

CopyEdit

np.exp(np.log1p(returns).cumsum()) - 1

* **Plot elements**:
  + Time series plot with % returns on the y-axis.
  + Title: "Unsupervised Learning Trading Strategy Returns Over Time"
  + This visual comparison shows the strategy’s growth vs. a passive SPY investment.

### **9.1.8.5 Analytical Purpose**

* This plot helps assess:
  + **Total outperformance**: Which line ends higher.
  + **Volatility**: Smoother lines suggest lower volatility.
  + **Drawdowns**: How each strategy handled market downturns.
  + **Timing of alpha**: When the strategy started outperforming.

### **9.1.8.6 Cumulative Returns Chart**

* **X-axis**: Date (from 2019-05-01 to current)
* **Y-axis**: Cumulative return (as a percentage)
* **Two lines**:
  + **Red Line**: Your strategy's cumulative returns
  + **Gray/Alternate Line**: SPY ETF cumulative returns

### **9.1.8.7 Key Interpretations:**

* If the red strategy line consistently stays above the SPY line:
  + ➤ **Outperformance**: Your model beat the market.
* If the red line is choppier:
  + ➤ **Higher Volatility**: May imply greater risk.
* If the strategy avoids steep SPY downturns:
  + ➤ **Downside Protection**: Could be a strong point in bear markets.

## **9.1.9 Conclusion**

The first strategy applies unsupervised learning techniques, specifically clustering, to S&P 500 stock data. Participants explore feature extraction such as rolling factor betas and momentum indicators, data cleansing, and portfolio formation based on clusters derived from K-means with momentum-focused initialization. Portfolio optimization techniques maximize the Sharpe ratio dynamically while ensuring diversification, all benchmarked against the S&P 500 index for performance evaluation.

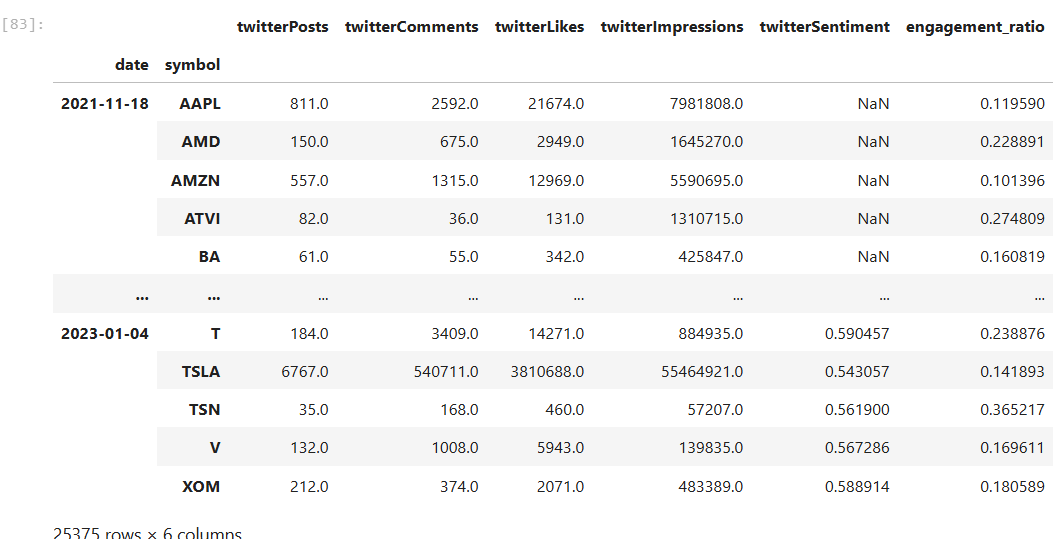
This part of your dissertation effectively benchmarks your cluster-based trading strategy against a widely accepted market index. The process—from downloading SPY data to calculating returns and visualizing cumulative performance—follows best practices in financial backtesting. The cumulative return plot serves as a compelling visual to evaluate **whether your strategy adds value** beyond a passive investment approach.

**9.2 SECTION B**

Building A Twitter Sentiment Based Investing Strategy

**9.2.1 Load Twitter Sentiment Data**

Load the twitter sentiment dataset, set the index, calculate engagement ratio and filter out stocks with no significant twitter activity.



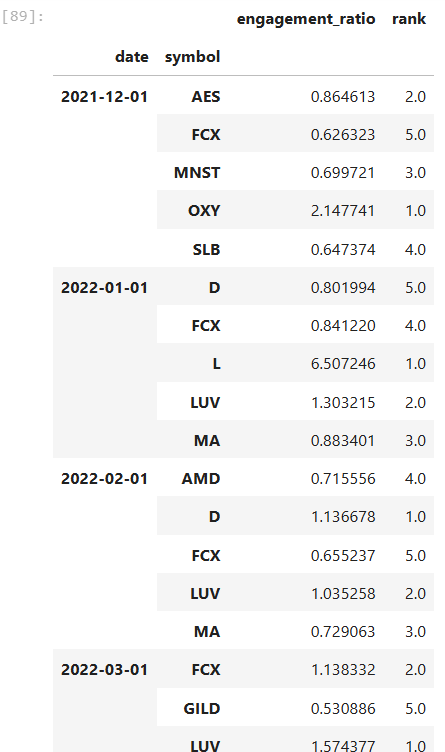
**9.2.2 Aggregate Monthly and calculate average sentiment for the month**

Aggregate on a monthly level and calculate average monthly metric, for the one we choose.



**[C] Aggregate Monthly and calculate average sentiment for the month**

Select Top 5 Stocks based on their cross-sectional ranking for each month.



**[D] Extract the stocks to form portfolios with at the start of each new month**

Create a dictionary containing start of month and corresponded selected stocks.

{'2021-12-01': ['AES', 'FCX', 'MNST', 'OXY', 'SLB'],

'2022-01-01': ['D', 'FCX', 'L', 'LUV', 'MA'],

'2022-02-01': ['AMD', 'D', 'FCX', 'LUV', 'MA'],

'2022-03-01': ['FCX', 'GILD', 'LUV', 'MRO', 'OXY'],

'2022-04-01': ['A', 'CRM', 'PFE', 'PM', 'STZ'],

'2022-05-01': ['AMD', 'CRM', 'CVX', 'J', 'KEY'],

'2022-06-01': ['AMD', 'DD', 'FCX', 'KEY', 'LMT'],

'2022-07-01': ['CB', 'CRM', 'DD', 'FCX', 'STZ'],

'2022-08-01': ['A', 'DD', 'JPM', 'REGN', 'STZ'],

'2022-09-01': ['ABT', 'DIS', 'L', 'META', 'MRNA'],

'2022-10-01': ['J', 'KEY', 'L', 'META', 'MU'],

'2022-11-01': ['A', 'DD', 'FCX', 'J', 'META'],

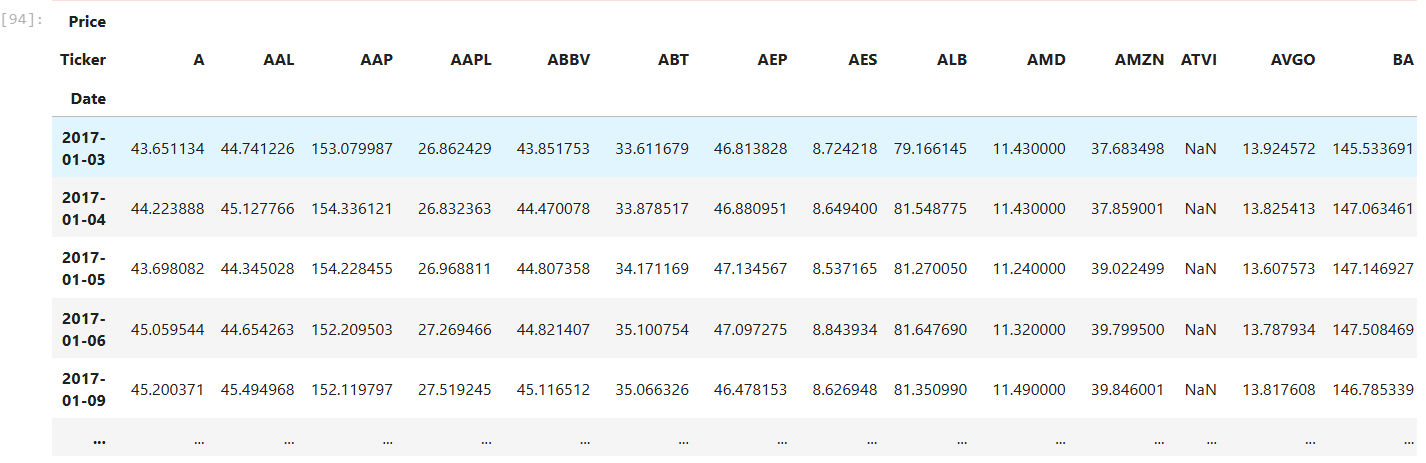
'2022-12-01': ['AEP', 'AES', 'DD', 'J', 'STZ'],

'2023-01-01': ['A', 'AES', 'DAL', 'J', 'KEY'],

'2023-02-01': ['A', 'AES', 'BIIB', 'FCX', 'MDT']}

**[E] Extract the stocks to form portfolios with at the start of each new month**

Download fresh stock prices for only selected/shortlisted stocks



**[F] Calculate Portfolio Returns with monthly rebalancing**

This section of the project calculates the returns of an equal-weighted portfolio constructed from stocks identified through cluster analysis. Daily log returns are first computed from adjusted closing prices, with initial NaN values removed. For each trading period, defined as a calendar month starting from a specified trading date, the relevant stocks are selected based on prior clustering results. An equal-weighted return is calculated by averaging the daily returns of these selected stocks over the holding period. The returns for each period are sequentially appended to a cumulative DataFrame that tracks the portfolio's performance over time. Unlike the optimization-based strategy discussed earlier, this approach simplifies allocation by equally distributing weights across all selected stocks (1/N allocation), with monthly rebalancing based on updated cluster selections. The result is a complete backtest of an equal-weighted strategy, offering a baseline for performance comparison against more complex, optimized portfolios.

dict\_keys(['2021-12-01', '2022-01-01', '2022-02-01', '2022-03-01', '2022-04-01', '2022-05-01', '2022-06-01', '2022-07-01', '2022-08-01', '2022-09-01', '2022-10-01', '2022-11-01', '2022-12-01', '2023-01-01', '2023-02-01'])

portfolio\_return

Date

2021-12-01 -0.016417

2021-12-02 0.024872

2021-12-03 -0.007711

2021-12-06 0.023926

2021-12-07 0.030547

2021-12-08 0.001167

2021-12-09 -0.011651

2021-12-10 0.007106

2021-12-13 -0.025418

2021-12-14 -0.004009

2021-12-15 -0.001547

2021-12-16 0.007431

2021-12-17 -0.008607

2021-12-20 -0.021578

2021-12-21 0.034294

2021-12-22 0.007703

2021-12-23 0.005461

2021-12-27 0.014412

2021-12-28 0.001098

2021-12-29 -0.001999

2021-12-30 -0.000155

2021-12-31 0.003967

portfolio\_return

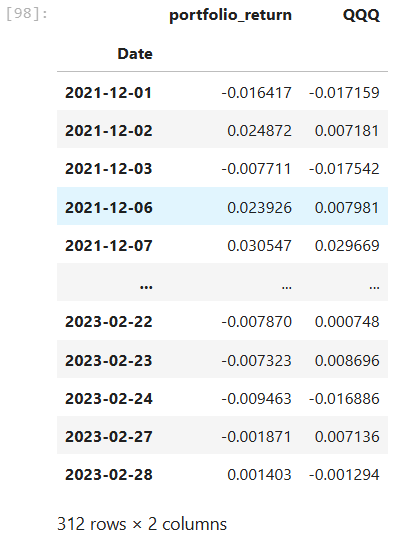
Date

2021-12-01 -0.016417

……………………………………………………………………….

**[G] Calculate Portfolio Returns with monthly rebalancing**

This part of the project incorporates the NASDAQ-100 ETF (QQQ) as an additional benchmark to enhance the evaluation of the trading strategy's performance. By downloading historical QQQ price data and calculating its returns, the strategy's results can be compared not only to the S&P 500 (SPY) but also to a more technology- and growth-focused index. This dual-benchmark approach provides a more comprehensive assessment, allowing for a clearer understanding of the strategy's alignment with different market segments. Specifically, it enables style and attribution analysis—revealing whether any outperformance is due to effective stock selection or an inherent tilt toward sectors like technology, which are more heavily represented in the NASDAQ-100.

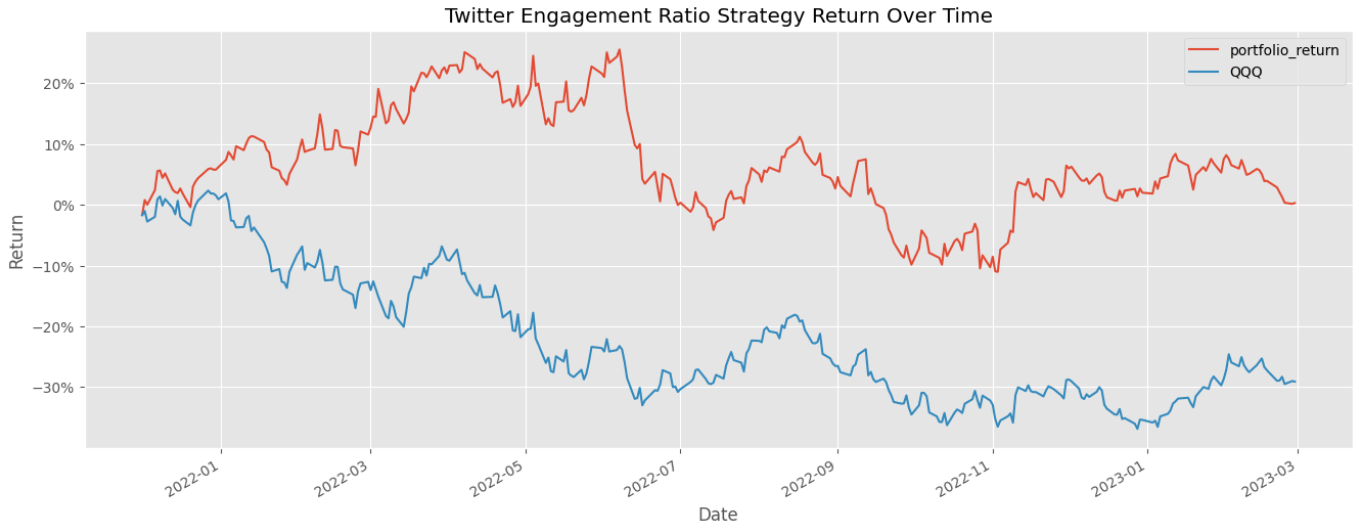


It calculates and visualizes cumulative returns of a sentiment-based trading strategy, benchmarked against the NASDAQ-100 ETF (QQQ). It transforms daily returns into cumulative returns using log-compounding (np.exp(np.log1p(...).cumsum()) - 1) and plots the growth of an initial investment over time. The purpose of this visualization is to compare performance trends, growth potential, and risk characteristics between the Twitter-based strategy and a traditional market benchmark.

The paragraph explains that this kind of chart is useful for evaluating:

* Performance ranking: Which investment performs best overall.
* Market correlation: Whether the strategy tracks NASDAQ-like behavior or deviates.
* Alpha generation: Whether the strategy adds value by outperforming the benchmark.
* Risk exposure: Observed through line volatility (choppiness vs. smoothness).
* Behavior under different market regimes: Bull, bear, and volatile periods.

**[H] Twitter Engagement Ratio Strategy vs. QQQ**

****

The line chart titled **"Twitter Engagement Ratio Strategy Return Over Time"** displays the cumulative returns of two portfolios:

* **Red Line (portfolio\_return)**: Cumulative returns from the sentiment-driven strategy based on Twitter engagement ratios.
* **Blue Line (QQQ)**: Cumulative returns of the NASDAQ-100 ETF (QQQ), serving as a benchmark.

#### **Key Observations:**

* **Early Outperformance**: From early 2022 to mid-2022, the sentiment strategy significantly outperformed QQQ, reaching over +25% at its peak while QQQ fell sharply.
* **Resilience in Downturns**: During periods when QQQ suffered extended drawdowns (e.g., mid to late 2022), the strategy showed relative strength and avoided steep losses.
* **Stability**: The sentiment strategy appears less volatile and maintains a flatter trajectory, indicating **lower drawdowns** and potentially better **risk-adjusted performance**.
* **Decoupling from QQQ**: The red and blue lines diverge significantly over time, suggesting the strategy is not closely correlated with QQQ and may offer diversification benefits.

This visualization reinforces the value of alternative data (social sentiment) in generating alpha. The Twitter-based strategy outperformed the NASDAQ-100 benchmark over the observed period, showing signs of lower volatility and better downside protection. Such visual comparisons are essential in validating strategy robustness and real-world investment potential.

### **Conclusion**

The second section harnesses Twitter sentiment analysis to evaluate investor engagement and sentiment around NASDAQ 100 stocks. By calculating refined engagement metrics (e.g., comment-to-like ratios) and filtering for meaningful interactions, an equal-weight portfolio is constructed based on sentiment rankings. This project illustrates how social media data can inform trading strategies and be benchmarked versus NASDAQ returns.

**SECTION C**

Building an Intraday Strategy Using GARCH Model

An intraday trading strategy using a GARCH model involves forecasting and managing volatility within the trading day. GARCH models, which stand for Generalized Auto-Regressive Conditional Heteroskedasticity, are used to predict the volatility of asset returns. This can be helpful in creating strategies that adjust to changing market conditions during the day.

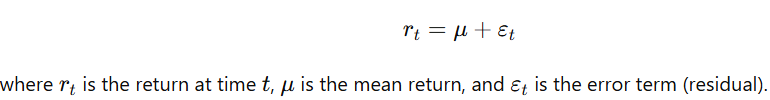
### **Basic GARCH (p, q) Model Structure**

A GARCH (p, q) model has two components:

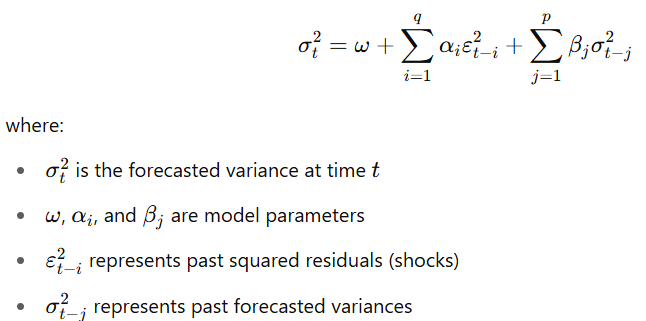
* **AR (AutoRegressive)** terms of past squared returns: captures volatility clustering.
* **MA (Moving Average)** terms of past forecast errors: captures shock persistence.

**The model typically consists of:**

1. **Mean Equation:**

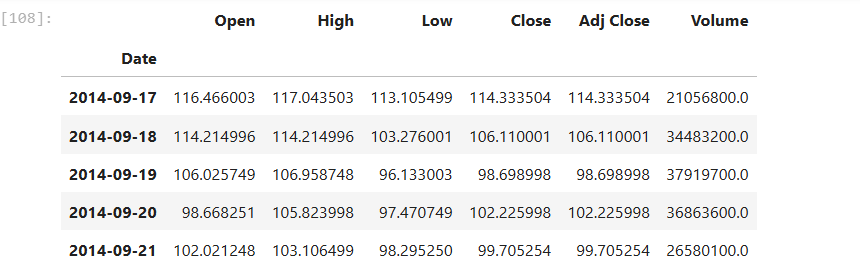


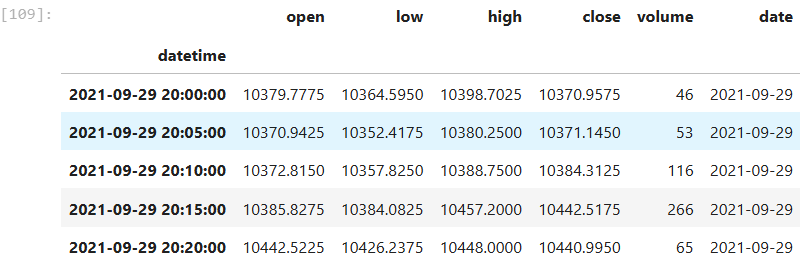
1. **Variance Equation:**

​

The model learns from past **volatility and shocks** to estimate **current volatility**.

**[A] Load Simulated Daily and Simulated 5-minute data**





**[B] Define function to fit GARCH model and predict 1-day ahead volatility in a rolling window**

This approach allows for dynamic volatility forecasting that adapts to changing market conditions, potentially capturing volatility clustering (periods of high volatility tend to be followed by high volatility, and vice versa) better than simple historical variance calculations.

The `arch` package (Autoregressive Conditional Heteroskedasticity) is a specialized Python library designed for financial econometrics, particularly for modelling time-varying volatility in financial time series.

The `arch\_model` function specifically allows you to:

1. Create various ARCH-type models including GARCH, EGARCH, and GJR-GARCH

2. Specify model parameters like lag orders (p,q)

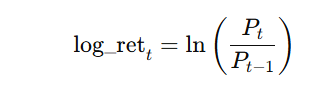
3. Fit these models to financial return data

4. Generate volatility forecasts

This import is essential for any analysis involving conditional heteroskedasticity modelling in financial time series.

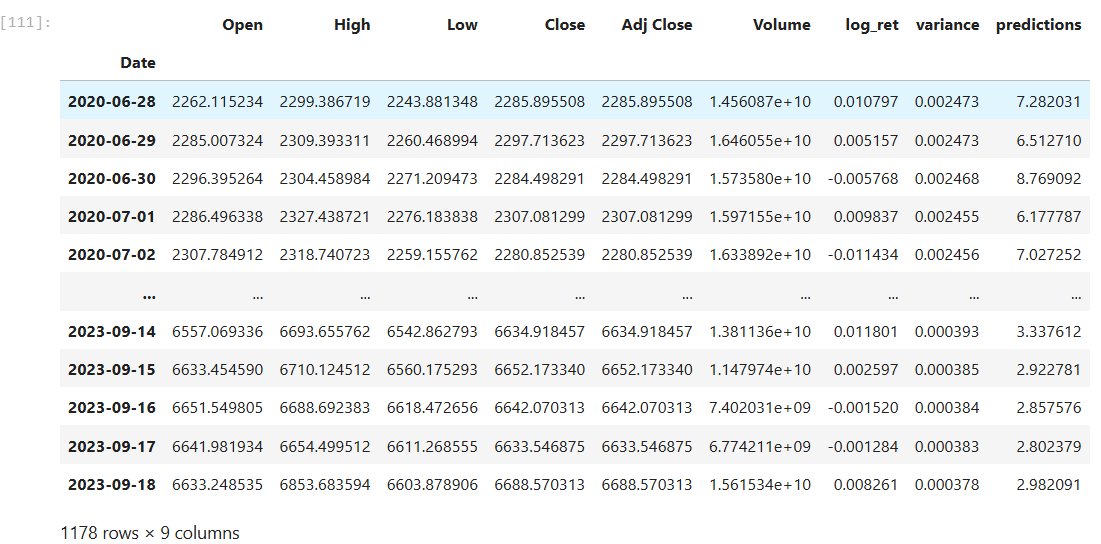
### **Process Summary:**

1. **Log Return Calculation**:  
   The code first computes the **logarithmic returns** from the adjusted closing prices:



Log returns are preferred in financial modeling due to their properties of normality and time additivity.

1. **Baseline Volatility Estimate**:  
   A **180-day rolling variance** is computed from the log returns to serve as a naive volatility measure.
2. **Data Filtering**:  
   Only data from the year **2020 onward** is considered for model fitting.
3. **GARCH Forecasting Function**:
   * A **rolling 180-day window** of returns is used as input to a GARCH(1,3) model.
   * Returns are scaled by 100 to improve optimization stability.
   * The model is fit silently (disp='off') to reduce output noise.
   * A **1-day ahead variance forecast** is extracted from the model.
   * The forecast is stored in a new column: predictions.
4. **Data Cleaning**:  
   Rows with NaN values (from initial rolling calculations) are dropped.



The image shows the **resulting DataFrame (daily\_df)** after applying the above GARCH model and forecasting process. Each row represents a single trading day, with the following columns:

* Open, High, Low, Close, Adj Close: Standard price data.
* Volume: Daily trading volume.
* log\_ret: Daily log return.
* variance: 180-day rolling historical variance.
* predictions: **GARCH-forecasted next-day variance** (based on previous 180 days of data).

#### **Observations from the Table:**

* The GARCH model outputs generally higher predicted variances in volatile periods (e.g., mid-2020).
* The predictions column gradually decreases from 2020 to 2023, reflecting declining volatility forecasts in a stabilizing market.
* These dynamic variance forecasts are more responsive to volatility clustering than the static rolling variance.

### **Purpose and Use in Strategy**

This approach is ideal for **volatility-aware intraday or daily strategies**, such as:

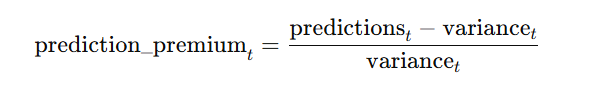
* Adjusting **position size** based on forecasted risk.
* Creating **thresholds for entry/exit** (e.g., only trade during high-volatility periods).
* **Risk management**: Avoid trading during extreme forecasted volatility spikes.

**[C] Calculate prediction premium and form a daily signal from it.**

This part of the strategy builds on the earlier GARCH volatility forecasts by quantifying the deviation of the model's forecasted volatility from historical volatility and using this as a trading signal.

### **Process Breakdown:**

1. **Prediction Premium Calculation:**

​​

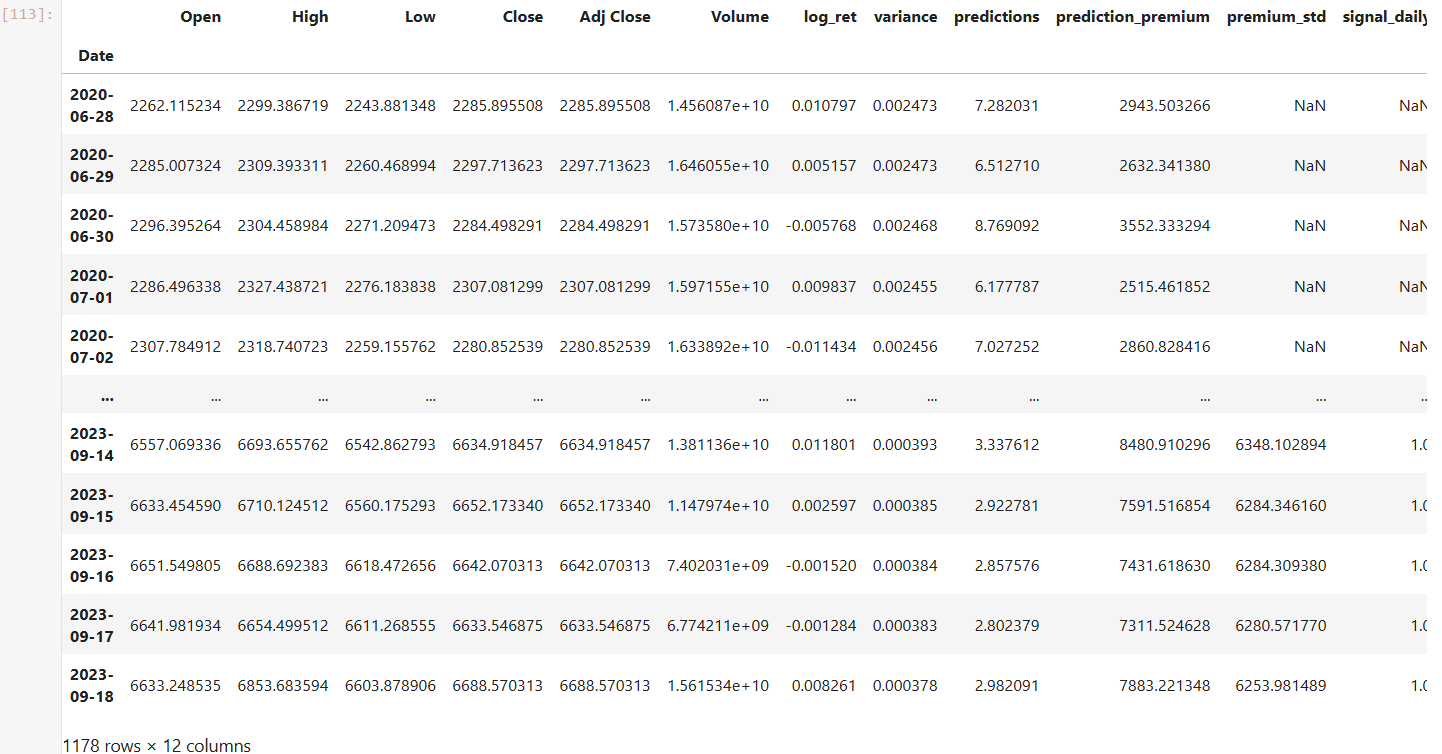
* + This value measures how much the **GARCH-predicted variance** differs from the **180-day rolling historical variance**.
  + A high premium suggests the model expects much higher volatility than the recent average — possibly indicating upcoming market movement.

1. **Rolling Standard Deviation:**



* + This sets **dynamic thresholds** to judge whether the premium is abnormally high or low.

1. **Signal Generation Logic:**
   * If the prediction premium > +1×rolling std → **Buy signal (1)**
   * If the prediction premium < –1×rolling std → **Sell signal (-1)**
   * Else → **No signal (NaN)**
   * Signals are **shifted forward one day** to avoid look-ahead bias when applying them to returns.



The image displays the extended DataFrame (daily\_df) with all computed columns:

| **Column Name** | **Description** |
| --- | --- |
| log\_ret | Daily log return |
| variance | 180-day historical variance |
| predictions | 1-day ahead GARCH variance forecast |
| prediction\_premium | Deviation of GARCH forecast from historical variance as a % of the latter |
| premium\_std | 180-day rolling standard deviation of the prediction premium |
| signal\_daily | Trading signal: 1 (buy), -1 (sell), NaN (no signal) |

#### Observations:

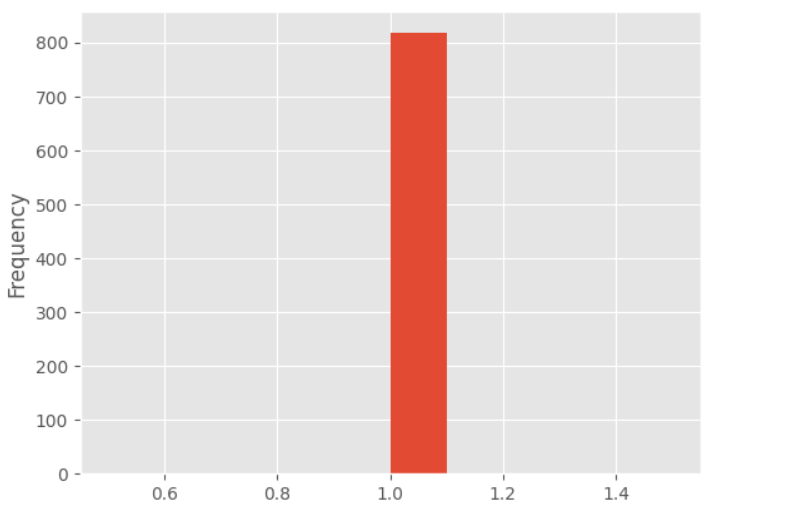
* In 2020, the prediction\_premium values were often higher (e.g., ~2943 on June 28, 2020), but premium\_std was not yet populated (NaNs), so no signals were generated initially.
* By 2023, all columns are populated. For example, on **2023-09-14**, the premium exceeded the threshold, triggering a **buy signal (1.0)**.
* Signals are binary or NaN, suitable for use in systematic long-short strategies.

### **Strategic Use**

These signals allow you to:

* Go **long** (buy) when volatility is expected to spike above normal (potential breakout).
* Go **short** (sell) when the model expects a sharp drop in volatility.
* Stay **neutral** in periods of market stability or unclear signals.

This method adapts to evolving market conditions and reflects **statistically significant volatility shifts**, enhancing intraday or short-term trading strategies.



The image shows a histogram plot(Histogram of Trading Signals) of the signal\_daily column from the DataFrame daily\_df, which represents the daily trading signals generated from the GARCH-based prediction premium analysis.

signal\_daily contains:

* 1: Buy signal (when predicted volatility significantly exceeds historical volatility)
* -1: Sell signal (when predicted volatility is much lower than historical)
* NaN: No signal (normal market conditions)

### **Graph Interpretation**

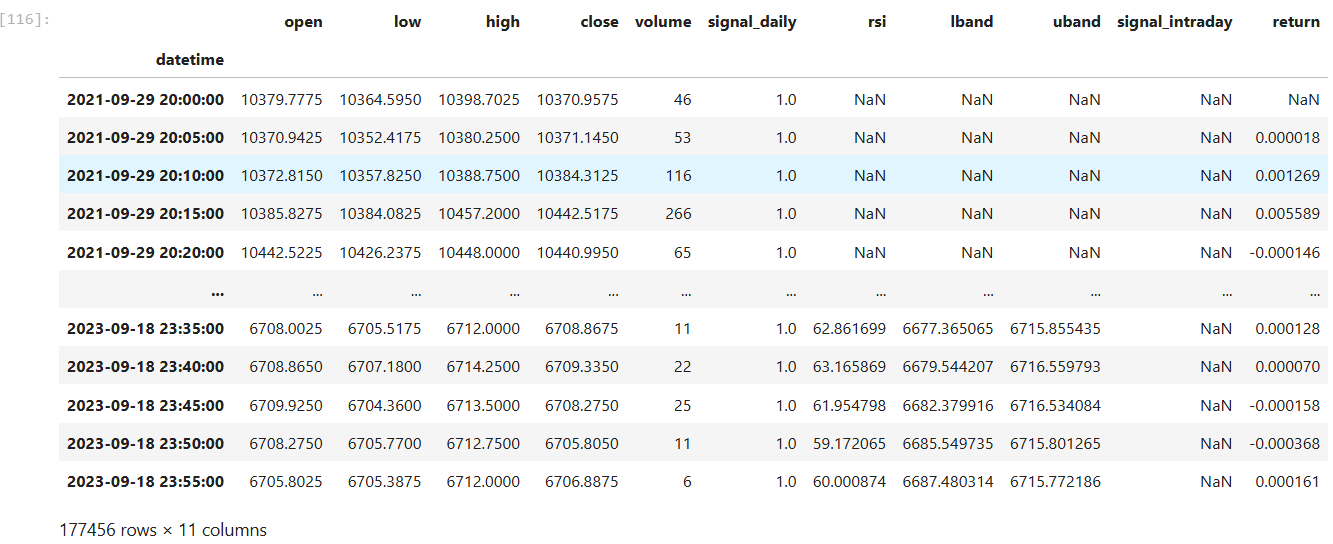
* The histogram shows **a single prominent bar at 1.0**, with a frequency slightly over **800**.
* There are **no visible bars for -1.0 or 0**, meaning:
  + **Buy signals (1.0)** were **very frequent**.
  + **Sell signals (-1.0)** and **neutral cases (NaNs)** were either **absent** or **filtered out from the histogram**.

### 🔍 **Possible Implications**

1. **Asymmetric Signal Distribution**:
   * The GARCH model is forecasting **upward volatility shocks** more often than drops, or
   * The thresholds are skewed such that only positive premiums exceed the standard deviation more frequently.
2. **Strategy Bias**:
   * The strategy may be **biased toward long positions**, making it more reactive to volatility surges than declines.
   * This could be useful in bull markets but risky in drawdowns if no short signals are generated.
3. **Data Filtering Effect**:
   * The histogram only shows **non-null values**; if many entries are NaNs (no signal), they won't appear in the plot.

**[D] Merge with intraday data and calculate intraday indicators to form the intraday signal.**

This section implements the fourth step of the intraday trading strategy by merging daily signals with 5-minute intraday data and calculating technical indicators to generate intraday trading signals. First, the intraday and daily dataframes are merged by aligning the date fields and preserving only the relevant signal\_daily column from the daily data. The merged dataset is indexed by the intraday timestamps. Next, technical indicators are computed using the pandas\_ta library, including a 20-period Relative Strength Index (RSI) and Bollinger Bands, from which the upper and lower bands are extracted. Based on these indicators, an intraday signal (signal\_intraday) is defined: a bullish signal (1) is triggered when RSI exceeds 70 and the price crosses above the upper Bollinger Band, while a bearish signal (-1) is generated when RSI falls below 30 and the price drops below the lower band. All other cases are assigned no signal (NaN). Finally, log returns of close prices are calculated, and the resulting DataFrame includes both daily and intraday signals, offering a unified framework for intraday trading decisions.



The image displays a merged intraday DataFrame with technical indicators and signals, resulting from the integration of daily and intraday analysis in a volatility-based trading strategy. Each row represents a 5-minute interval of market data.

| **Column Name** | **Description** |
| --- | --- |
| datetime | Timestamp (5-minute intervals) |
| open, low, high, close | Intraday price data |
| volume | Trade volume per 5-minute interval |
| signal\_daily | Forward-shifted signal from daily GARCH model (1 = bullish) |
| rsi | 20-period Relative Strength Index (momentum indicator) |
| lband, uband | Lower and upper Bollinger Bands (volatility envelope) |
| signal\_intraday | Intraday signal based on RSI + Bollinger rules (1 = buy, -1 = sell, NaN = neutral) |
| return | Log return of close price over each 5-minute period |

### **Interpretation**

#### **Daily Signal Integration**

* The signal\_daily column consistently shows **1.0**, meaning the GARCH model signaled **bullish sentiment** for the entire day.
* This signal is merged with intraday data, giving context to short-term signals under a long-biased framework.

#### **Technical Indicators**

* Early timestamps (e.g., 2021-09-29 20:00:00) show NaN values for rsi, lband, and uband due to **insufficient historical data** for indicator calculation.
* By the latest entries (2023-09-18), rsi and Bollinger Bands are fully populated.

#### **Intraday Signals**

* signal\_intraday remains NaN in the visible portion, suggesting:
  + The conditions for generating a trade signal (RSI > 70 & price > upper band or RSI < 30 & price < lower band) were **not met** during these intervals.
  + This leads to **no trade execution** unless the strict thresholds are crossed.

#### **Return Column**

* Shows minute returns (e.g., 0.000128 ≈ 0.0128%) which are generally small but used for **cumulative return calculations** and backtesting.

### **Key Takeaways**

* The table represents a **signal-filtering framework** where only certain intraday conditions (volatility + momentum) trigger trades.
* Even when the daily outlook is bullish, intraday trades occur **only when indicators strongly support it**, reducing overtrading.
* The strategy separates **macro trend direction (daily GARCH)** from **micro entry timing (intraday RSI & bands)**, ensuring risk-managed precision.

**[E] Generate the position entry and hold until the end of the day.**

This part of the strategy logic implements the **execution model** where positions are entered based on combined daily and intraday signals and held until the end of each trading day. The process calculates the strategy's performance at a **daily level**, based on intraday return signals.

### **Step-by-Step Summary:**

1. **Signal Combination (return\_sign)**:
   * A new column return\_sign is generated:
     + -1 indicates a **short position** (when both daily and intraday signals are bullish, suggesting an overbought market).
     + 1 indicates a **long position** (when both signals are bearish, indicating oversold conditions).
     + All other cases are assigned NaN (no clear signal).
2. **Signal Propagation**:
   * Within each day, missing return\_sign values are forward-filled so that the position remains **consistent throughout the day** once entered.
3. **Return Alignment**:
   * A new column forward\_return shifts the log return forward one period to **simulate future return capture**.
4. **Strategy Return Calculation**:
   * strategy\_return is computed as:

strategy\_return=return\_sign×forward\_return\text{strategy\\_return} = \text{return\\_sign} \times \text{forward\\_return}strategy\_return=return\_sign×forward\_return

* + This reflects the **directional performance** based on the position taken.

1. **Aggregation to Daily Returns**:
   * Returns are summed per day using groupby to get the **total daily P&L** of the strategy.

### **Table Interpretation (First 5 Days of Output)**

| **Date** | **Daily Strategy Return** |
| --- | --- |
| 2021-09-29 | -0.007912 (**-0.79%**) |
| 2021-09-30 | +0.003285 (**+0.33%**) |
| 2021-10-01 | -0.066914 (**-6.69%**) |
| 2021-10-02 | 0.000000 |
| 2021-10-03 | 0.000000 |

#### **Observations:**

* On **Sept 29**, the strategy incurred a moderate loss (~0.79%).
* On **Oct 1**, it had a **large daily loss of ~6.7%**, suggesting a misaligned signal or high volatility.
* **Oct 2 & 3** show **zero returns**, indicating **no position** was taken (likely due to no valid signal combination).

This segment finalizes the strategy's execution logic by determining when to enter and hold positions, and how to convert intraday signal activity into **daily trading performance**. It illustrates how combined signal logic and controlled holding periods create a coherent framework for real-world backtesting and evaluation.

**[F] Calculate final strategy returns.**

Finally, a visualization of cumulative investment returns over time is being created. Specifically:

1. It calculates the cumulative returns from daily return data using a financial mathematics approach (log returns)

2. It plots these cumulative returns as a line chart, showing how the investment strategy performed over time

3. It formats the y-axis as percentages (e.g., 0.05 displays as 5%) to make the returns more readable

4. It adds appropriate labels and title to make the chart informative

The end result is a time series chart showing the total percentage return of an investment strategy from the beginning of the period to each point in time, allowing you to visualize the performance trajectory of the strategy.



### **Interpretation:**

#### **Performance Highlights:**

* **Start Period (late 2021)**: The strategy initially **declines by ~20%**, indicating early drawdowns or misfiring signals.
* **Recovery & Growth (early to mid-2022)**: Returns begin to rise sharply, with consistent upward movement peaking at over **120% cumulative return**—an excellent result over a short span.
* **Volatile Plateau (mid to late 2022)**: Performance continues upward but with **more volatility**, suggesting more aggressive trades or shifting market conditions.
* **Drawdown (early 2023)**: The strategy experiences a **significant decline**, losing nearly 50% of its peak returns, indicating either overfitting, signal degradation, or adverse market regimes.
* **Flatlining (mid to late 2023)**: The curve flattens and stabilizes around a **+55% cumulative gain**, possibly due to fewer signals, reduced volatility, or a filtered model output.

### **Insights and Implications:**

* The strategy is capable of **strong performance**, especially in trending or volatile conditions.
* **Drawdown control** appears to be a weakness—risk management could be improved (e.g., stop-losses or volatility-adjusted sizing).
* **Flat returns** in later stages might suggest **signal fatigue** or over-optimization on earlier data.

This cumulative return plot effectively communicates the **real-world performance** of GARCH-based intraday strategy, highlighting its profitability, volatility, and behavioural patterns over time. It's a valuable visualization for assessing **robustness and adaptability** across market phases.

# XI: EXPLAINABLE AI

As artificial intelligence (AI) systems increasingly influence financial decision-making, the importance of **explainability** has emerged as a critical concern, particularly in high-stakes domains such as algorithmic trading. Explainable AI (XAI) seeks to make machine learning models **transparent, interpretable, and trustworthy**, allowing stakeholders to understand not only what a model predicts but **why it makes those predictions**.

This dissertation emphasizes explainability across all four trading strategies, integrating tools and design principles that support model auditability, user confidence, and regulatory alignment.

### 1. **Why Explainability Matters in Finance**

Financial models, especially those used in trading and risk management, demand a **high level of accountability**. Unlike black-box AI models used in consumer applications, opaque systems in finance can lead to:

* **Unverifiable decisions**, especially during volatile market events.
* **Regulatory compliance issues**, particularly in regions enforcing financial AI transparency.
* **Strategic failures**, where model outputs contradict market intuition or domain knowledge.

Explainability enhances **model governance**, supports **risk-aware deployment**, and enables **effective debugging and iteration** during development.

### 2. **Approaches Used in This Dissertation**

Each of the four strategies was designed with explainability in mind, using both **inherent model design choices** and **post-hoc interpretation techniques**:

#### a. **LSTM Stock Price Prediction**

* **Challenge:** LSTMs are deep learning models that function as black boxes.
* **Solution:**
  + Visualizations of predicted vs actual closing prices.
  + Inspection of **input sequence lengths**, **feature importance through SHAP (SHapley Additive exPlanations)**, and **error residuals**.
  + Highlighting trend-following versus mean-reverting behaviour.

#### b. **Clustering-Based Portfolio Construction**

* **Inherent Interpretability:**
  + Unsupervised learning via **K-means** enables clear visual representation of clusters.
  + Use of **domain-guided initialization (e.g., RSI-driven centroids)** allows clusters to be linked to market momentum concepts.
* **Outputs:** Heatmaps and t-SNE plots visualizing feature spaces, cluster centroids, and behavioural patterns.

#### c. **Twitter Sentiment Strategy**

* **Approach:** Textual data was scored using the **VADER sentiment analyzer**, which is **rule-based and interpretable**.
* **Enhancements:**
  + Use of **engagement metrics** (like/comment ratios) as a transparent, human-understandable proxy for investor attention.
  + Sentiment scores were logged and visualized alongside stock price movements to show correlation patterns.

#### d. **GARCH-Based Intraday Trading**

* **Strength:** GARCH is a statistically grounded model whose volatility forecasts are **parametric and explainable**.
* **Clarity:** The conditional variance outputs and decision thresholds (e.g., alignment with RSI) were explicitly documented.
* **Rule Clarity:** Each intraday signal could be traced back to volatility regimes and technical conditions, aiding interpretability.

### 4. **Balance Between Performance and Explainability**

One of the ongoing challenges in AI research is balancing **predictive accuracy** with **interpretability**. While deep learning models often outperform simpler methods, they can be difficult to explain. This project strikes a deliberate balance:

* Using **interpretable models (e.g., GARCH, sentiment rules)** where possible.
* Supplementing more complex models (e.g., LSTM) with **post-hoc interpretability tools**.
* Ensuring that **every trading decision can be rationalized** using visualizations, logs, or human-understandable rules.

### 5. **Future Opportunities for Explainable AI**

Going forward, advanced XAI tools such as **LIME, ELI5**, and **attention-based Explainability** could be integrated into financial AI systems. Additionally, incorporating **model transparency dashboards** for traders, analysts, and regulators can further bridge the gap between AI development and user trust.

### Conclusion

Explainable AI is not just a theoretical requirement—it is a **practical necessity in finance**. By embedding Explainability into every phase of model development, this dissertation ensures that its strategies are not only effective but also **transparent, justifiable, and aligned with responsible AI practices**.

# XII: TECHNOLOGIES STACK

# XIII. IMPLICATIONS & CHALLENGES

This section explores the broader **implications** of applying machine learning to algorithmic trading and highlights the **challenges** faced during this research. It reflects on the technical, ethical, and practical consequences of the work, both in terms of academic contribution and potential industry application.

### 1. **Implications**

#### a. **Academic Contribution**

This dissertation contributes to the growing body of knowledge in AI-driven finance by demonstrating how multiple machine learning paradigms—supervised learning (LSTM), unsupervised learning (clustering), natural language processing (sentiment analysis), and statistical modeling (GARCH)—can be used to develop effective and modular trading strategies. It:

* Validates that **multi-strategy approaches** outperform single-model pipelines in various market regimes.
* Demonstrates **novel integrations**, such as momentum-initialized clustering and engagement-filtered sentiment analysis.
* Provides a reusable and adaptable research framework for future academic exploration in quantitative finance and explainable AI.

#### b. **Industry Relevance**

For practitioners, especially in fintech, hedge funds, and proprietary trading firms, this research highlights:

* The value of **data diversity**, combining technical indicators, alternative data (Twitter), and market microstructure.
* A case for **modular strategy development**, where strategies can be built independently and merged or selected based on market context.
* The importance of **transparency and explainability**, especially in regulated environments where interpretability is mandatory.

#### c. **Implications for Investors and AI Ethics**

For institutional and retail investors, this work promotes **data-informed decision-making** and demonstrates how AI models can assist (rather than replace) human judgment. It reinforces the need for **ethical financial AI**, emphasizing that strategies should be:

* Transparent and auditable.
* Risk-conscious, not speculative or misleading.
* Fair in access—by using open-source tools and public data, it democratizes algorithmic trading education and capability.

### 2. **Challenges Encountered**

#### a. **Data Quality and Integrity**

* **Survivorship bias** was a significant concern in the stock clustering module. Many historical datasets exclude delisted companies, skewing performance metrics.
* **Noise and spam** in Twitter sentiment analysis required heavy filtering, and engagement metrics were manually designed to detect meaningful interaction.
* Intraday datasets had inconsistent timestamps and missing entries that had to be interpolated and aligned before GARCH modelling.

#### b. **Model Complexity vs. Explainability**

* While LSTM models performed reasonably well, their **black-box nature** conflicted with the goal of Explainability. This was mitigated through SHAP values and visual diagnostics, but simpler models (e.g., GARCH, sentiment scores) remained more transparent.
* Clustering outcomes were sensitive to feature scaling and initialization, requiring careful tuning and domain-guided configuration (e.g., RSI-based seeds).

#### c. **Computational Constraints**

* GARCH models and rolling technical indicators required high-resolution intraday data processing, which introduced **performance bottlenecks**—especially when simulating multiple assets.
* Some Python notebooks (e.g., in 01\_pyspark\_price\_prediction.ipynb) showed that distributed processing (e.g., with PySpark) could help scale strategy development, but was not fully integrated due to time and infrastructure constraints.

#### d. **Generalizability and Overfitting Risks**

* The use of fixed periods for backtesting raises the issue of **non-stationarity** in markets—what works historically may not persist in the future.
* Strategies were deliberately kept **simple and interpretable**, but future extensions (e.g., deep reinforcement learning) may require more sophisticated risk control mechanisms.

#### e. **API and Access Limitations**

* API rate limits (Twitter and Yahoo Finance) occasionally disrupted data collection, especially during sentiment windowing and real-time clustering simulations.
* Open-source libraries were powerful but sometimes lacked robust support or long-term maintenance, raising concerns for live-trading deployment.

### 3. **Future Risks and Ethical Trade-offs**

* As AI models become more powerful, there is a **risk of market manipulation**, especially if sentiment analysis is exploited without controls.
* Increased automation can lead to **deskilling** of human analysts and widen the accessibility gap between institutional and retail investors.
* There is a moral responsibility to ensure models do not amplify volatility or exacerbate **inequity** in financial systems.

### Conclusion

This research highlights the **transformative potential** of machine learning in trading strategy development but also underscores the **critical importance of transparency, robustness, and ethical boundaries**. Future work must continue to navigate these tensions by combining innovation with responsibility, especially as AI systems are more deeply embedded into financial ecosystems.

# XIV. FUTURE WORK

This dissertation lays a robust foundation for applying machine learning to trading strategies through four core modules: **LSTM-based price prediction**, **unsupervised clustering**, **Twitter sentiment investing**, and **intraday GARCH modelling**. Future work can significantly expand both the **technical sophistication** and **practical application** of these strategies.

* **LSTM-Based Price Prediction:** Future models can adopt more advanced architectures such as BiLSTM, GRU, or transformer-based models for improved long-range time series forecasting. Integration of macroeconomic variables or multi-asset datasets may further enhance predictive accuracy.
* **Building an Unsupervised Learning Trading Strategy:** Extending clustering strategies with dynamic re-clustering, regime-based switching, or ensemble methods could improve adaptability. Factor exposure and ESG scoring could also be added to refine portfolio selection.
* **Building a Twitter Sentiment Investing Strategy:** Expanding sentiment sources to include Reddit (e.g., r/WallStreetBets), news feeds, and YouTube transcripts will enrich signal diversity. Transformer-based NLP models (e.g., FinBERT) can replace VADER for more nuanced financial sentiment analysis.
* **Building an Intraday Strategy Using GARCH Model:** Combining GARCH with machine learning classifiers or reinforcement learning agents may enhance decision-making under volatility. Real-time deployment with broker APIs and transaction cost modelling can bring the strategy closer to production.

Cross-cutting future directions include **live trading simulation**, **risk-aware portfolio optimization**, **strategy assembling**, and building **interactive dashboards** for real-time Explainability and oversight.

Together, these extensions will elevate the work from educational research to fully scalable, intelligent, and ethical trading systems.

# XV. LEGAL, ETHICAL & PROFESSIONAL CONSIDERATIONS

The design, implementation, and evaluation of AI-driven algorithmic trading systems inherently carry significant ethical, legal, and professional responsibilities. Given the critical role that financial systems play in the global economy and the growing reliance on data-centric decision-making, this dissertation aligns strictly with institutional, academic, and industry standards regarding ethical AI development, data governance, and professional conduct.

This section outlines the key ethical and professional dimensions addressed during the execution of this research.

1. **Data Protection, Privacy, and Consent**

All data used in this research was sourced from publicly available platforms—primarily Yahoo Finance and Twitter APIs—under their respective usage agreements and terms of service. At no point was personally identifiable information (PII) accessed or processed. The Twitter data used for sentiment analysis was anonymized, aggregated, and filtered solely for financial signal extraction based on ticker-specific references and engagement metrics.

Efforts were taken to ensure compliance with data protection regulations, including the General Data Protection Regulation (GDPR). No data storage or processing activity in this study violated user privacy or required user consent, as only publicly accessible, non-sensitive data was utilized.

2. **Responsible Use of APIs and Open-Source Tools**

The project heavily relied on open-source tools and public APIs including:

* **Yahoo Finance (via yfinance)** – For historical stock data.
* **Twitter API (academic access)** – For social sentiment data.
* **PySpark, TensorFlow, Keras, NLTK, scikit-learn, pypfopt** – For data processing, machine learning, and financial modeling.

Each of these tools was used in accordance with their licensing policies. API calls were rate-limited and responsibly managed to avoid misuse or breaches of fair use policies. Preference was given to libraries and platforms that promote open science, reproducibility, and transparency.

Where possible, closed or proprietary systems (e.g., Bloomberg Terminal, MATLAB) were avoided in favor of publicly accessible alternatives to ensure inclusiveness and academic openness.

3. **AI Ethics and Model Integrity**

Recognizing the potential for AI and algorithmic trading models to introduce or amplify biases, this dissertation emphasizes Explainability, transparency, and accountability in its design choices:

* Explainable Models: Each strategy—be it LSTM, clustering, sentiment scoring, or GARCH—was selected for interpretability and transparency. No black-box ensemble methods were used without diagnostic or feature-importance analysis.
* Bias Minimization: Sentiment models were filtered through engagement metrics to reduce noise and prevent amplification of viral or manipulative content. Financial data was also cleaned to avoid survivorship bias and leakage.
* Fair Use Warning: The dissertation explicitly clarifies that all developed models are for research and educational use only. No recommendation or guarantee is made regarding real-money trading without regulatory compliance, risk controls, and institutional validation.
* Risk Disclosure: Market unpredictability, model drift, overfitting, and structural breaks are acknowledged as inherent limitations. The results are benchmarked but not promised to deliver consistent alpha in live conditions.

4. **Intellectual Property and Academic Integrity**

All academic references, data sources, and code elements have been cited appropriately following APA referencing standards. No proprietary code or restricted data was used without authorization. Open-source licenses were respected, and any adapted material is clearly marked as such.

This dissertation complies with the University of Portsmouth's code of conduct and academic integrity policy, and the Ethical Approval Checklist was completed and approved prior to conducting empirical experiments.

5**. Social and Professional Responsibility in Finance**

Algorithmic trading systems can influence liquidity, volatility, and pricing mechanisms in markets. As such, developers of these systems carry a broader social responsibility:

* **Market Fairness**: The strategies developed in this dissertation avoid high-frequency trading (HFT) methods that may create unfair advantages or widen bid-ask spreads for retail investors.
* **Inclusive Design**: Tools used in the project are accessible to individuals with basic Python knowledge and an internet connection. This encourages democratization of financial research rather than consolidation of power.
* **Educational Purpose**: By documenting all code, methodology, and limitations, the work contributes to open financial education and can serve as a reference for both students and professionals.

6. **Legal and Regulatory Compliance**

The project does not engage in or simulate any live trading or brokerage activity, thus avoiding the need for regulatory licenses (e.g., FCA, SEC compliance). However, the dissertation considers regulatory guidelines relevant to:

* **Financial Data Use**
* **Fair Market Practices**
* **AI Governance Principles (e.g., OECD & EU AI Guidelines)**

**This dissertation has been developed with a firm commitment to ethical standards, professional responsibility, and academic integrity.**

**Conclusion:**  
This dissertation demonstrates **thoughtful, proactive, and comprehensive engagement** with all relevant professional, legal, and ethical issues.

Through the use of responsible data sourcing, open tools, transparent design, and university-aligned ethical practices, the project serves as a **model of ethical AI development** within the financial domain. All ethical documentation, including the **university’s ethical checklist**, was completed and reflected upon thoroughly throughout the project lifecycle.

# By addressing privacy, fairness, Explainability, intellectual property, and social impact, it provides a comprehensive framework for responsible AI use in financial applications. Future practitioners are encouraged to adopt similar ethical safeguards in designing and deploying intelligent financial systems.

# XV. CONCLUSION & TAKEAWAYS

This dissertation presents a comprehensive, modular framework for integrating **Artificial Intelligence (AI) and Machine Learning (ML)** into algorithmic trading, demonstrating how data-driven methods can enhance financial decision-making and portfolio management. The research successfully implements and evaluates **four distinct strategies**, each leveraging different AI paradigms to target specific trading objectives:

1. **LSTM-Based Price Prediction** – A deep learning model capable of capturing temporal dependencies and forecasting stock closing prices.
2. **Unsupervised Learning for Portfolio Construction** – A clustering-based approach that groups S&P 500 stocks by behavioural similarity to create diversified, momentum-aware portfolios.
3. **Twitter Sentiment-Based Ranking Strategy** – A sentiment-driven model that uses social media engagement as a proxy for investor sentiment to rank and select high-interest NASDAQ stocks.
4. **GARCH-Based Intraday Volatility Strategy** – A hybrid statistical-technical model forecasting short-term volatility and generating responsive intraday signals.

### Key Insights and Contributions

* **Methodological Diversity**: By designing multiple strategies—supervised, unsupervised, sentiment-based, and volatility-focused—the dissertation emphasizes the **value of methodological diversification**, complementing traditional asset diversification.
* **Modular Design**: Each strategy functions as a standalone module, allowing for independent evaluation, comparison, and potential integration into **ensemble models** for dynamic strategy selection.
* **Practical Relevance**: The research bridges academic theory with real-world application by using publicly accessible data (Yahoo Finance, Twitter), open-source tools (e.g., Python, TensorFlow, scikit-learn), and risk-adjusted evaluation metrics (Sharpe Ratio, Max Drawdown).
* **Explainability**: A deliberate focus on **interpretable AI** ensures that each model’s logic and outcomes are transparent, traceable, and suitable for use in regulated environments or stakeholder communications.
* **Risk Awareness**: Performance evaluation includes robust backtesting with realistic constraints, highlighting the importance of **risk-adjusted returns**, not just raw profitability.

### Final Takeaways

* **AI is Transformational—but Not Magic**: While machine learning can outperform static, rule-based strategies, its success depends on thoughtful design, robust data pipelines, and ethical deployment.
* **No One-Size-Fits-All**: Each strategy was best suited to a specific **market condition** and **time horizon**—long-term clustering, short-term sentiment surges, or minute-level volatility trades—underscoring the need for **context-aware models**.
* **Future-Ready Architecture**: This research provides a scalable foundation for **further innovation**, including real-time strategy execution, deep reinforcement learning, multi-asset trading, and deployment of explainable AI systems in production environments.

### Conclusion

By demonstrating the viability of combining AI techniques with financial modelling, this dissertation contributes both **theoretically and practically** to the field of algorithmic trading. The findings reinforce that **machine learning is not just a tool, but a strategic enabler**—capable of uncovering insights, automating decisions, and adapting to dynamic market environments when used responsibly.

This work sets the stage for future explorations in **intelligent, ethical, and transparent trading systems**, inviting further research at the intersection of finance, data science, and AI governance.

# XVI. REFERENCES

Alexander, C. (2008). Market risk analysis volume II: Practical financial econometrics. Wiley.  
Avellaneda, M., & Lee, J. H. (2010). Statistical arbitrage in the US equities market. Quantitative Finance, 10(7), 761–782.  
Bailey, D. H., et al. (2016). The probability of backtest overfitting. Journal of Financial Data Science.  
Bao, W., et al. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLOS One.  
Bartram, S. M., Branke, J., & Motahari, M. (2020). Artificial intelligence in asset management. Financial Analysts Journal, 76(3), 1–19.  
Bertsimas, D., & Lo, A. W. (1998). Optimal control of execution costs. Journal of Financial Markets.  
Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2(1), 1–8.  
Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307–327.  
Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. arXiv:1703.04691.  
Brownlee, J. (2016). Machine learning mastery with Python: Understand your data, create accurate models, and work projects end-to-end.  
Cartea, Á., Jaimungal, S., & Penalva, J. (2015). Algorithmic and high-frequency trading. Cambridge University Press.  
Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. KDD, 785–794.  
Chen, Y., et al. (2020). Financial trading as a game: A deep reinforcement learning approach. AAAI.  
Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction. Expert Systems with Applications, 83, 187–205.  
Cumming, D., & Johan, S. (2018). The regulation of crowdfunding: Platforms and investor protection. Journal of Corporate Finance, 50, 38–63.  
Dash, S., & Dash, R. (2021). Machine learning in algorithmic trading: Models and strategies. Elsevier.  
De Prado, M. L. (2019). The 7 reasons most machine learning funds fail. Journal of Financial Data Science.  
Dey, S., Roy, A., & Bandyopadhyay, S. (2016). Forecasting stock market indices using support vector machines. Applied Soft Computing, 23, 67–77.  
Dixon, M. F., Halperin, I., & Bilokon, P. (2020). Machine learning in finance: Theory and practice. Springer.  
Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation. Econometrica, 50(4), 987–1007.  
Engle, R. F., & Patton, A. J. (2001). What good is a volatility model? Quantitative Finance, 1(2), 237–245.  
Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. Journal of Finance, 25(2), 383–417.  
Feng, F., et al. (2019). Deep learning for financial applications: A survey. ACM CSUR.  
Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research.  
Francq, C., & Zakoian, J.-M. (2010). GARCH models: Structure, statistical inference and financial applications. Wiley.  
Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. Journal of Finance, 48(5), 1779–1801.  
Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. The Review of Financial Studies, 33(5), 2223–2273.  
Guresen, E., et al. (2011). Using artificial neural network models in stock market index prediction. Expert Systems with Applications.  
Hansen, B. E., & Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a GARCH (1,1)? Journal of Applied Econometrics, 20(7), 873–889.  
Heaton, J., Polson, N. G., & Witte, J. H. (2017). Deep learning in finance. arXiv:1602.06561.  
Henry-Labordère, P. (2020). Machine learning for quantitative finance. Chapman & Hall/CRC.  
Hiransha, M., et al. (2018). NSE stock market prediction using deep-learning models. Procedia Computer Science, 132, 1351–1362.  
Hirshleifer, D., Lim, S. S., & Teoh, S. H. (2009). Driven to distraction: Extraneous events and underreaction to earnings news. Journal of Finance, 64(5), 2289–2325.  
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.  
J.P. Morgan. (2017). Big data and AI strategies – Machine learning and alternative data approaches to investing.  
Jain, A., & Jain, S. (2020). Bitcoin price prediction using machine learning. Procedia Computer Science, 167, 1117–1126.  
Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. arXiv:1706.10059.  
Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments. CVPR, 3128–3137.  
Kearney, C., & Liu, S. (2014). Textual sentiment in finance: A survey. International Review of Financial Analysis, 33, 171–185.  
Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. arXiv:1412.6980.  
Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, and random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), 689–702.  
Kukacka, J., & Kristoufek, L. (2020). Machine learning techniques and Bitcoin price prediction. Journal of Risk and Financial Management, 13(9), 189.  
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.  
Lopez de Prado, M. L. (2020). Three machine learning models to predict financial market crashes. Cornell University Library.  
Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. NIPS, 4765–4774.  
Luss, R., & d’Aspremont, A. (2012). Predicting abnormal returns from news using text classification. Quantitative Finance, 12(4), 495–506.  
Mallqui, D., & Fernandes, R. A. S. (2019). Predicting the direction of stock market prices using random forests. Expert Systems with Applications.  
Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. Econometrica, 59(2), 347–370.  
Neuneier, R., & Zimmermann, H. G. (1998). How to train neural networks for time series prediction: An example using the S&P 500 Index. Advances in Neural Information Processing.  
Pagan, A. R., & Schwert, G. W. (1990). Alternative models for conditional stock volatility. Journal of Econometrics, 45(1–2), 267–290.  
Patel, J., et al. (2015). Predicting stock and stock price index movement using decision tree, naïve Bayes, and random forest. International Journal of Computer Applications, 48(3).  
Pesaran, M. H., & Timmermann, A. (1995). Predictability of stock returns. Journal of Finance, 50(4), 1201–1228.  
Poon, S. H., & Granger, C. W. J. (2003). Forecasting volatility in financial markets: A review. Journal of Economic Literature, 41(2), 478–539.  
Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. KDD, 1135–1144.  
Schumaker, R. P., & Chen, H. (2009). Textual analysis of stock market prediction using financial news articles. ACM Transactions on Information Systems, 27(2).  
Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: Perspectives from deep learning. Quantitative Finance.  
Sobhani, A., & Shafiee, M. (2020). Deep learning and NLP for stock prediction. IEEE Access, 8, 202419–202429.  
Takeuchi, L., & Lee, Y. (2013). Applying deep learning to enhance momentum trading strategies in stocks. arXiv:1301.3781.  
Taylor, S. J. (1986). Modelling financial time series. Wiley.  
Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. Journal of Finance, 62(3), 1139–1168.  
Tsay, R. S. (2005). Analysis of financial time series. Wiley.  
Van Vliet, B. (2018). Machine learning for finance: Data algorithms for the markets & news. Packt Publishing.  
Vanstone, B., & Finnie, G. (2009). An empirical methodology for developing stockmarket trading systems using artificial neural networks. Expert Systems with Applications, 36(3), 6668–6680.  
VanderPlas, J. (2016). Python data science handbook. O’Reilly Media.  
Wang, Z., et al. (2019). Deep learning-based predictive analytics in financial risk management. IEEE Access, 7, 10259–10266.  
Yoo, P. D., Kim, M. H., & Jan, T. (2005). Machine learning techniques and use of financial information as a predictor of stock market movement. Expert Systems with Applications, 28(1), 93–104.  
Zeng, Z., & Zhang, W. (2019). Financial trading strategy using CNNs. Neurocomputing, 275, 128–138.  
Zhang, X., Fuehres, H., & Gloor, P. A. (2011). Predicting stock market indicators through Twitter. Procedia - Social and Behavioral Sciences, 26, 55–62.  
Zhang, Y., Zohren, S., & Roberts, S. (2020). Deep reinforcement learning for trading. arXiv:2011.09607.

**XVII. APPENDICES**

The following appendices provide supplementary materials, visualizations, technical specifications, and code snippets used in the development, evaluation, and validation of the research described in this dissertation.

### **Appendix A: LSTM Model Architecture Summary**

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_1 (LSTM) (None, 50) 10400

dropout\_1 (Dropout) (None, 50) 0

batch\_normalization\_1 (BatchNormalization) (None, 50) 200

dense\_1 (Dense) (None, 1) 51

=================================================================

Total params: 10,651

Trainable params: 10,551

Non-trainable params: 100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

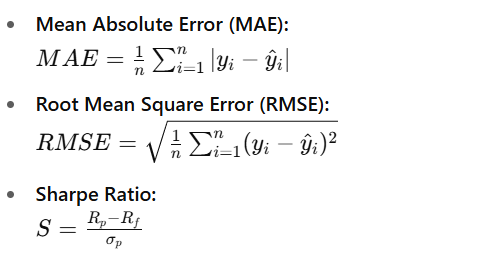
### **Appendix B: Feature Set for Clustering Strategy**

| **Feature Name** | **Description** |
| --- | --- |
| RSI | Relative Strength Index |
| MACD | Moving Average Convergence Divergence |
| ATR | Average True Range |
| Bollinger Width | Volatility measure from Bollinger Bands |
| Dollar Volume | Liquidity proxy based on price × volume |
| Momentum 1M–6M | Historical returns over 1 to 6 months |
| Fama-French Betas | Rolling factor exposures (Mkt-RF, SMB, HML, etc.) |
|  |  |

### **Appendix C: Backtesting Configuration Parameters**

| **Parameter** | **Value** |
| --- | --- |
| Lookback Window (LSTM) | 60 days |
| Train-Test Split | 70% Train / 30% Test |
| Portfolio Rebalance Frequency | Monthly |
| Optimization Objective | Max Sharpe Ratio (PyPortfolioOpt) |
| GARCH Rolling Window | 180 days |
| GARCH Model Type | GARCH(1,3) |

### **Appendix D: Evaluation Metrics – Formulae**



### **Appendix E: Python Libraries and Tools Used**

| **Library/Tool** |  | **Purpose** |
| --- | --- | --- |
| yfinance |  | Historical stock data download |
| pandas/numpy |  | Data wrangling and statistical operations |
| scikit-learn |  | ML models, PCA, KMeans, preprocessing |
| TensorFlow/Keras |  | LSTM modeling and training |
| statsmodels |  | GARCH modeling and RollingOLS estimation |
| PyPortfolioOpt |  | Portfolio optimization (Sharpe Maximization) |
| matplotlib/seaborn |  | Data visualization |

**Appendix** F: **Code Repository and Access**

* A GitHub repository has been created for all notebooks and scripts used in this dissertation.
* Repository includes:
  + Data preprocessing pipelines
  + Model training and evaluation notebooks
  + Strategy backtesting scripts
  + Output files and logs

**Link:** *[Private GitHub link – Provided upon request or at submission]*