ST. XAVIER'S COLLEGE (EMPOWERED AUTONOMOUS INSTITUTION), Mumbai-1.

MSc. Big Data Analytics (Part 2) – 2025 – 26

SEMESTER III

COURSE NAME – Research Project

1.Project Abstract:

This project explores portfolio optimization using the **Black-Litterman model**. The model integrates **market-implied returns with subjective investor views** to generate balanced investment strategies. It addresses the limitations of traditional models like Markowitz by allowing flexibility, improved diversification, and incorporation of investor insights.

Here instead of investor views we use **random views**.

2. Contents

List of Tables

List of Figures:

PORTFOLIO OPTIMIZATION USING BLACK-LITTERMAN MODEL

3. Objective:

The goal of this project is to help people make smarter decisions about how to invest their money across different stocks. Instead of just using past stock data like most models do, our system also lets investors share what they personally believe will happen in the market — for example, if they think a particular stock will go up.

We are using a method called the **Black-Litterman model**, which takes both:

- What the market says about stock performance, and
- What the investor thinks (and how confident they are about it).

By combining both these things, the model gives **better, more balanced investment suggestions** that make more sense in real life. It avoids risky or strange decisions that other models sometimes suggest.

Our aim is to:

- Let investors include their opinions in the decision-making.
- Show how much those opinions affect the final result.
- Give clear charts and visuals to explain how money should be spread across stocks.

- Compare our method with older models to prove that it gives more stable and realistic results.

In short, we want to build a smart and flexible tool that helps people — especially big investors — invest their money in a way that's both data-driven and aligned with real-world thinking.

Problem Statement / Background

In the world of investing, one of the biggest challenges is deciding **how to split money across different stocks** to get the best possible returns while reducing risk. Many people and organizations use models like the **Markowitz Mean-Variance Optimization**, which look at **past stock data** to make these decisions.

However, these traditional models have some important problems:

- They can give **unrealistic or extreme results**, like putting too much money in just one or two stocks.

- They don't allow investors to include **their own opinions or insights** about future stock performance.

- They only rely on **past data**, which doesn't always reflect what's going to happen in the future.

In real life, investors — especially professionals and institutions — often have their own research, insights, or market views that they want to include in their decision-making. But current models don't give them a way to do that effectively.

This project focuses on solving that problem using the Black-Litterman model, which allows us to:

- Mix market data with expert or investor opinions,

- Control how much those opinions should influence the final decision (based on confidence), and

- Create investment suggestions that are more stable, realistic, and tailored to what the investor believes.

By doing this, we aim to build a smarter system that helps investors make better, more balanced decisions — something that is especially useful in today's fast-changing and unpredictable financial markets.

4. Scope Of The Project:

This project is focused on designing and implementing a portfolio optimization framework using the Black-Litterman model, which integrates both market-based data and subjective investor views to provide balanced investment recommendations. The scope of this project is defined by the following key areas:

1. Data Collection and Cleaning-

Fetching historical stock prices for 20+ NIFTY 50 stocks using NSE website.

Collecting benchmark index data and market capitalizations.

2. Market-Implied Return Estimation-

Computing prior (baseline) expected returns from market data using reverse optimization techniques.

Visual analysis of historical correlations between selected stocks.

3. Investor Views Integration-

Manually defining expected returns for certain stocks.

Assigning confidence levels to each view to influence the model appropriately

Using Python code to merge these views with market priors

4. Black-Litterman Model Implementation-

Combining subjective views and market returns to generate posterior expected returns.

Using these to create optimized asset allocations.

5. Portfolio Optimization-

Using the Sharpe Ratio as the performance metric

Optimizing weights to achieve maximum return for a given level of risk

Visualizing results with pie charts, bar charts, and correlation heatmaps.

6. Model Comparison and Evaluation-

Comparing output from the Black-Litterman model with the traditional Markowitz model.

Highlighting benefits like improved diversification, reduced overfitting, and flexibility.

7. Insights and Interpretation

Exploring how changing confidence levels affect final allocations

Generating clear, visual explanations of portfolio composition and performance

5. Approach:

5.1 Data Wrangling

- Fetch historical stock prices and market capitalization using NSE data.
- Clean, preprocess, and align stock return data.

5.2 Generate Market-Implied Priors

- Use reverse optimization to derive implied excess returns from market capitalizations.
- Visualize correlation matrices and market priors.

5.3 Incorporate Views

- Define investor views (e.g., "Stock A will outperform Stock B").
- Assign confidence levels to these views.

5.4 Black-Litterman Posterior Returns

- Combine market priors with views to generate posterior expected returns.
- Derive the new covariance-adjusted return distribution.

5.5 Portfolio Optimization

- Optimize portfolio using the maximum Sharpe Ratio.

- Compare results with the Markowitz model.

5.6 Visualization & Analysis

- Generate visual outputs: pie charts, risk-return plots, allocation comparison.

- Evaluate differences in asset distribution and performance metrics.

6. Expected Outcomes / Delivarables:

6.1 Well-Defined Research Report

- Explanation of portfolio optimization concepts.
- Comparative analysis between **Markowitz Model** and **Black-Litterman Model**.
- Detailed discussion on the integration of investor views and market data.

6.2 Python Codebase (Jupyter Notebook)

- Full pipeline implemented including:
    - Data acquisition using NSE
    - Calculation of market-implied returns
    - Integration of subjective views and confidence levels
    - Posterior return estimation
    - Portfolio optimization (Sharpe Ratio maximization)
    - Comparison with Markowitz optimization

6.3 Visualizations

- Pie/bar charts showing asset allocations under both models
- Return-risk scatter plots

- Confidence level vs. portfolio allocation effect

6.4 Insightful Analysis

- Effect of confidence levels on portfolio allocation
- Discussion on stability, realism, and diversification of portfolios
- Scenarios showing differences in allocation under varying views

6.5 Final Presentation / Viva Deliverables

- Slide deck summarizing methodology, models, results, and insights
- Clear articulation of how Black-Litterman improves upon traditional approaches
- Q&A readiness for research questions, mathematical intuition, and real-world applicability

6.6 Project Documentation

- Abstract + Introduction
- Research questions & methodology
- Result discussion & comparative tables
- References and data source links

6.7 Collaborative Contributions

- Individual contributions documented across:
    - Data cleaning and preparation
    - Model development and coding
    - Result interpretation and visualization

TIMELINE-

TASKS:

1. Data collection and preprocessing
2. Market-implied return calculation
3. Define investor views and confidence
4. Black-Litterman model implementation
5. Portfolio optimization and Sharpe Ratio
6. Visualizations and model comparison
7. Report writing and presentation prep
8. Final review, polishing, and submission

TOOLS AND TECHNOLOGIES-

Programming Language: Python

Libraries: pandas, numpy, matplotlib, seaborn

Platform: Jupyter Notebook

Dataset Source: NSE (https://www.nseindia.com/)

DATA CLEANING

7. Column Descriptions:

1. **Symbol**: This is the name or code of the stock you're looking at. It’s like a nickname for the company in the stock market.

2. **Series**: This tells you the type of stock or security. For example, it could be "EQ" for equity which is regular stocks.

3. **Date**: The specific date on which the stock information is recorded. The trading date for each entry is shown. It is usually in the format DD-MM-YYYY Used for time series analysis.

4. **Prev Close**: The price of the stock at the end of the previous trading day. Think of it as yesterday’s closing price. It is a baseline for comparing today’s performance.

5. **Open Price**: This is the price of the stock when the market opens on that specific day. The stock’s price at the start of the trading day. It gives insight into investor sentiment before trading begins.

6. **High Price**: The highest price the stock reached during that day. Useful for measuring intra-day volatility.

7. **Low Price**: The lowest price the stock dropped to during that day. The Together with "High", it defines the day’s price range.

8. **Last Price:** The most recent price at which the stock was traded during the day. It updates in real-time while the market is open and represents the last executed trade, not necessarily the closing price.

9. **Close Price**: The price at which the stock ended its trading for the day. It's often used as the most important reference point for that day.

10. **VWAP (Volume Weighted Average Price):** This is like an average price of the stock, but it considers how much was traded at each price. So, it's more accurate because it's based on the volume of trades. Traders use VWAP to evaluate whether they're getting a good price.

11. **Total Traded Quantity:** This shows how many shares of the stock were bought and sold in total during the day.

12. **Turnover ₹**: The total value of all the trades made during the day, measured in Indian Rupees (₹). It's the sum of all the shares traded multiplied by their prices.

13. **No. of Trades**: This tells you how many individual buying or selling actions were made for that stock.

14. **Deliverable Qty**: This is the number of shares that were actually "delivered" to the buyers at the end of the trading day, meaning these shares were not sold again during the day. It reflects actual ownership transfer.

15. **% Dly Qt to Traded Qty**: This tells you what percentage of the total traded shares were actually delivered (i.e., bought for long-term holding) compared to the total traded quantity. A higher percentage means more people are buying to hold, while a lower percentage suggests a lot of short-term trading.

8. Why are we using the closing price column?

The closing price is used as the target variable, representing the final trading price of the stock for each day. It reflects the market consensus value at the end of each trading session and is a key indicator for forecasting stock performance.

9.List of companies:

➔ **Banks & Finances:**

1. HDFC
2. ICICI
3. SBI

➔ **Tech:**

1. TCS
2. INFOSYS

➔ **Conglomerates:**

1. ITC
2. Reliance

➔ **Engineering & Construction:**

1. LARSEN & TOUBRO (L&T)

➔ **Telecommunication:**

1. BHARTI AIRTEL

➔ **Consumer Goods:**

1. HINDUSTAN UNILEVER

Market Cap Of Each Company:

- RELIANCE = Total Market Cap (₹ Cr.) 18,81,013.70
- TCS = Total Market Cap (₹ Cr.) 10,97,438.31
- HDFC = Total Market Cap ₹15,54,269 Cr
- INFY = Total Market Cap (₹ Cr.) 6,26,838.20
- HINDUSTAN = Total Market Cap (₹ Cr.) 5,93,036.83
- ICICI = Total Market Cap (₹ Cr.) 10,59,993.55

- ITC = Total Market Cap (₹ Cr.) 5,15,597.91
- BHARTI AIRTEL = Total Market Cap (₹ Cr.) 10,90,242.54
- L&T = Total Market Cap (₹ Cr.)4,99,049.17
- SBI = Total Market Cap (₹ Cr.) 1,84,432.64

10. Data collection and Pre-processing:

1. Library Imports:

- pandas is used for data manipulation and reading CSV files.
- os handles file paths and directory navigation.
- matplotlib and seaborn are used for visualizing data.

2. Company List and Data Aggregation:

- A list of major Indian companies is defined.
- The script loops through each company's folder to:
- Find all .csv files (representing daily trading data).

3. Reading and Combining Multiple CSVs:

- Reads multiple CSV files from a folder containing NIFTY50 stock data.
- Combines them into a single DataFrame called niftycombine.

4. Data Preview:

- The combined dataset is loaded.
- The first few rows are previewed using head().

- The column names are printed for inspection.

5. Date Conversion and Sorting

- Converts the Date column to proper datetime format.
- Sorts all rows by date in ascending order.
- Reformats the date to dd-mm-yyyy.

6. Visualization of Results:

- Objective: Present findings clearly and insightfully.
- Plot predicted vs actual price over time
- Visualize feature importance
- Present returns comparison across companies.

7. Exploratory Data Analysis (EDA):

- Line plots of closing prices to understand trends
- Volume vs price analysis
- Distribution plots of returns
- Correlation heatmaps between features

11. Date Formating:

We converted the date format from "DD-MMM-YYYY" (e.g., 02-Jan-2024) to "DD-MM-YYYY" (e.g., 02-01-2024) to standardize the format for consistency and easier processing in subsequent analysis.

12. Checking missing values:

The obtained data from the NSE (National Stock Exchange) website was already complete and ready to use. When we first looked through it, we didn't find any empty or missing values in any of the columns. Since the data was clean and well-organized, we didn't need to do anything extra to fix or fill in missing information.

Stock market data, especially from trusted sources like the NSE, is usually very reliable. That's because for every trading day, the stock exchange keeps a proper record of things like the opening price, closing price, highest and lowest prices, and how many shares were traded. These numbers come directly from actual buying and selling in the market, so there's really no chance for missing values unless something goes wrong which is very rare.

13. Data type Inconsistency:

Even though the data was complete, some of the columns were not in the right format for analysis.

For example, the ‘Closing Price’ (which shows the final price of a stock at the end of each day) was saved as plain text (string) instead of a number (float). Because of this, we couldn't do any mathematical and graphical analysis properly. Since the Black-Litterman model requires the input data to be in numerical format, we converted it into a numerical format (float) that works well with decimal (float) values as well.

Also, the ‘Date’ column was actually stored as plain text too (string). To easily sort the data by date and to pick out specific months or years, we changed it into the “datetime” format (dd/mm/yyyy) which is better for analysis.

14. How duplicate rows are handled:

The data for 6 years is combined on the date column from 1st jan 2019 to 31 Dec 2024.

Stock market data usually doesn’t contain duplicate records, even when we merge data from different files. This is because each row in the dataset represents a specific stock’s activity on a specific trading day which makes it unique by default.

Even though we have data for multiple days, each date will have different values for prices, volume, and other details, since the stock market changes every day. The stock prices may differ even by 0.1 value. So, for example, the data for August 1st will never be exactly the same as August 2nd or 3rd.

Because of this, even after combining or merging multiple datasets like different companies, days, months or years, the data remains unique and doesn't repeat, as long as we are not accidentally merging the exact same file more than once.

15. Outlier Analysis :

To identify anomalies in the historical price data, a visual inspection was performed using both closing price trends and daily return series. A line chart of HDFC's closing price revealed an abrupt shift in July 2023, where the price oscillated between approximately ₹1,600 and ₹600 within short intervals. In a liquid, large-cap stock such as HDFC, such rapid and repeated large-scale changes are highly unusual under normal market conditions.

The daily returns plot reinforced this finding. While typical daily returns remained within the ±5% range, the flagged period showed extreme spikes exceeding ±50%. These spikes are inconsistent with organic market movements and are likely attributable to corporate actions such as stock splits or mergers or unadjusted data entries from the source.

Such anomalies, if left unaddressed, can distort return calculations and inflate volatility estimates, directly impacting models like Black-Litterman that rely on accurate historical covariance structures. Therefore, the identified segment was classified as a data anomaly requiring adjustment before proceeding with portfolio optimization.

DATA MODELLING

16. Introduction

● Objective: Portfolio optimization using Black-Litterman model.

● Approach: Combination of Exploratory Data Analysis (EDA), and Portfolio

Optimization using the Black-Litterman model.

17. Dataset Details

17.1 Column Descriptions:

Source: NSE India (nseindia.com)

Companies Included:

● Banks & Finance: HDFC, ICICI, SBI

● Tech: TCS, Infosys

● Conglomerates: ITC, Reliance

● Engineering & Construction: L&T

● Telecommunication: Bharti Airtel

● Consumer Goods: Hindustan Unilever

17.2 Important Columns:

1. Symbol – Stock code.
2. Series – Type of stock/security.
3. Date – Trading date.
4. Prev Close – Previous day’s closing price.
5. Open Price, High Price, Low Price – Daily price movements.
6. Last Price – Latest traded price.
7. Close Price – Price at the end of the day (target variable).
8. VWAP – Volume Weighted Average Price.
9. Total Traded Quantity, Turnover ₹, No. of Trades – Market activity indicators.
10. Deliverable Qty & % Dly Qt to Traded Qty – Ownership transfer metrics.

18. Step-by-Step Process

1. Importing Libraries

○ Imported pandas for data handling, os for directory navigation, matplotlib and seaborn for visualization.

○ Role: These libraries provide the foundation for reading, cleaning, and visualizing stock data.

2. Market Capitalization Dictionary

○ Created a Python dictionary mcaps containing the market capitalization (in ₹) for each stock in the portfolio.

○ Purpose: Used later to calculate market portfolio weights (Wmkt) in the Black-Litterman model.

3. Data Collection

○ Loaded CSV files containing historical daily trading data for each stock.

○ Combined them into a single DataFrame for analysis.

○ Standardized date formats and sorted data chronologically.

18.1 Stock Prices of Companies

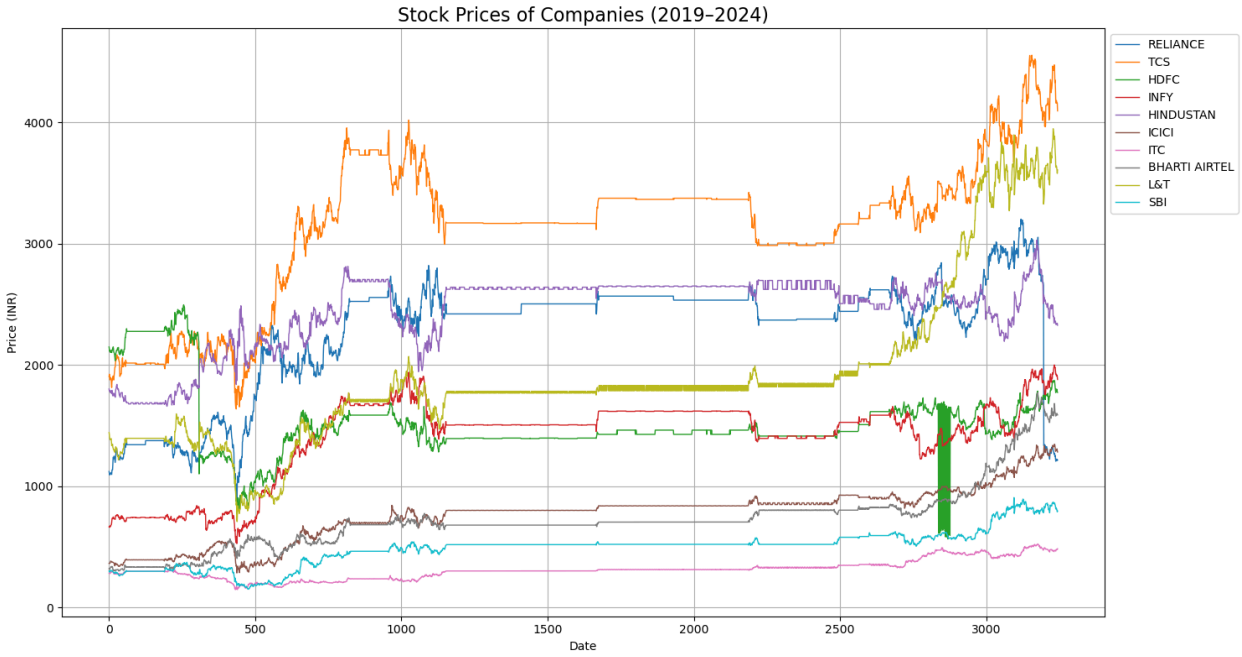


fig. 1 stock prices of companies

Displays historical price trends for all portfolio companies from 2019-2024.

Visualizes price movements, identifies trends and patterns, compares relative performance across stocks, and shows market cycles and correlations over the 5-year period.

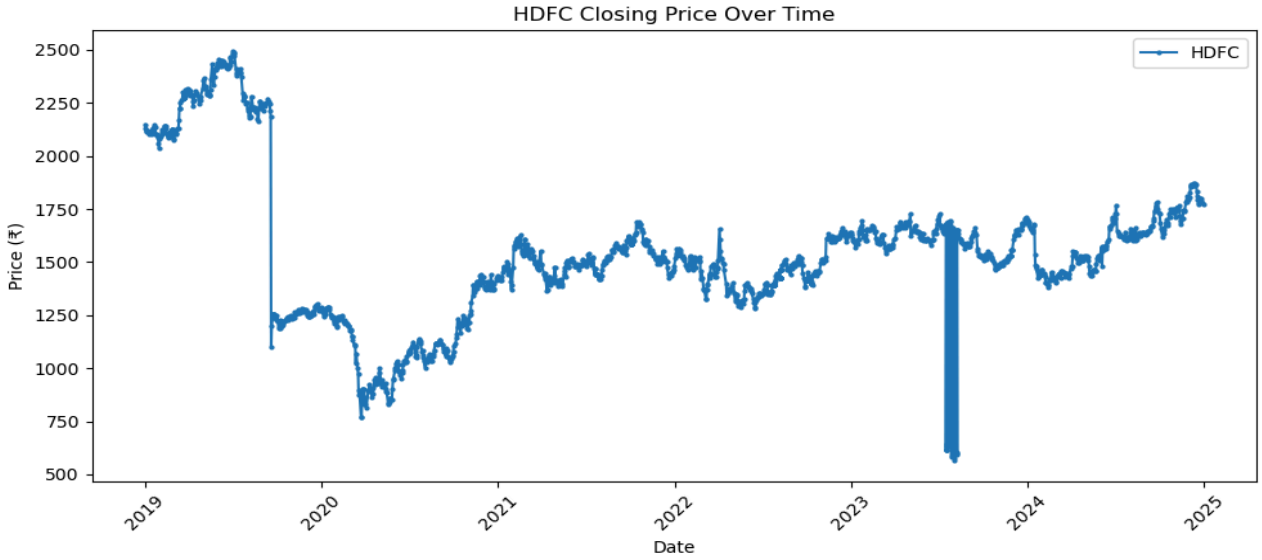


fig. 2 HDFC closing price over time

The above visualization is useful for spotting sudden, unrealistic price jumps that may indicate corporate actions or data errors. In HDFC's case, a clear abrupt shift between two distinct price ranges can be seen around mid-2023, confirming the presence of an outlier series that needs cleaning before portfolio analysis.

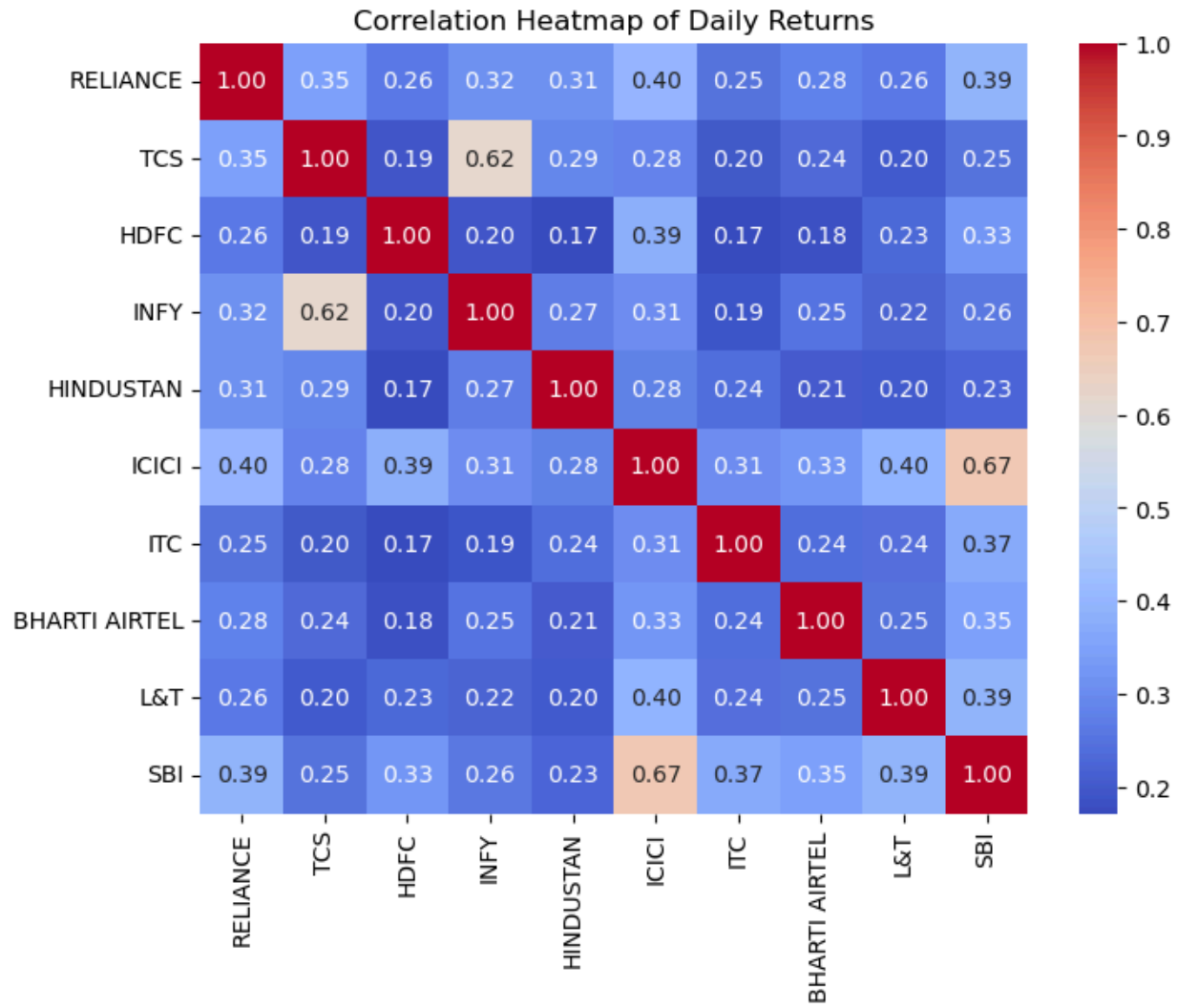


fig. 3 correlation heatmap of daily return

Here is a clear explanation of a correlation heatmap for stocks:

- A correlation heatmap is like a friendship chart for stocks, showing their relationships through colors.

- A value close to 1 (red) means two stocks move together almost all the time, indicating strong positive correlation.
- A value close to 0 means there is no relationship—stocks move independently without influencing each other.
- A value close to -1 means stocks often move in opposite directions, showing strong negative correlation.
- This visualization helps spot if a portfolio is too dependent on stocks moving together, which indicates low diversification and higher risk.

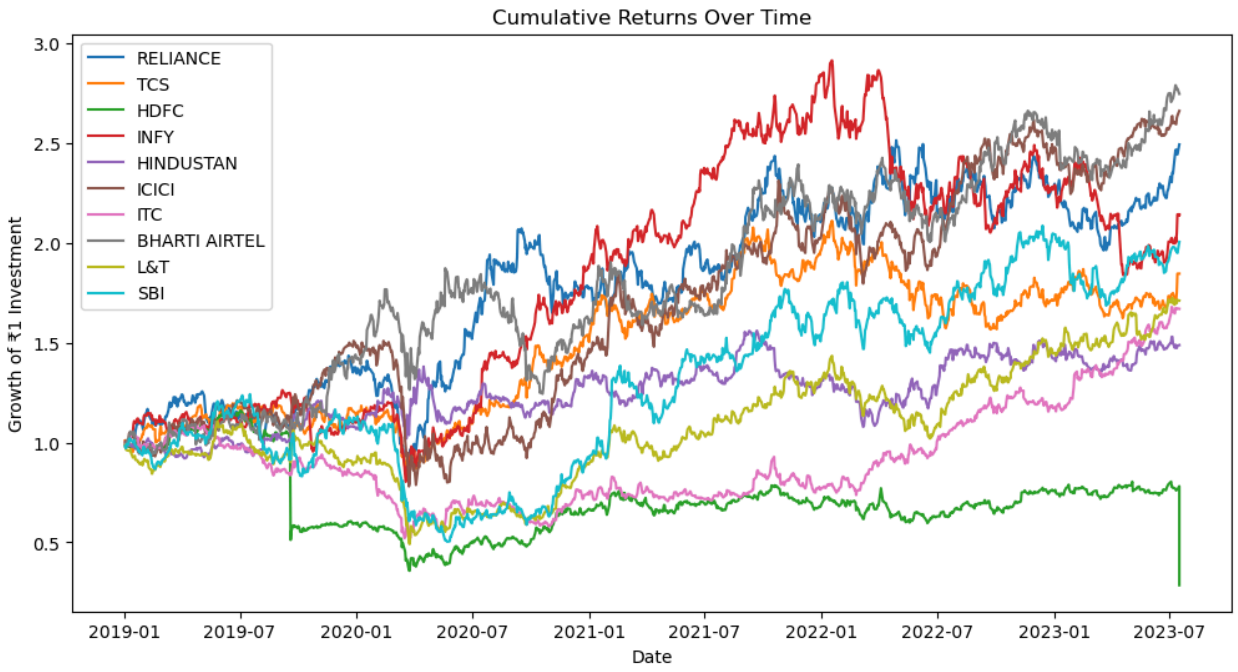


fig.4 Cumulative returns over time

Cumulative Returns Plot Tracks the growth of a ₹1 investment in each asset over time. Purpose: Visualizes long-term performance, compares asset returns, and identifies outperforming stocks. Shows compounding effects and relative performance trends across the portfolio.

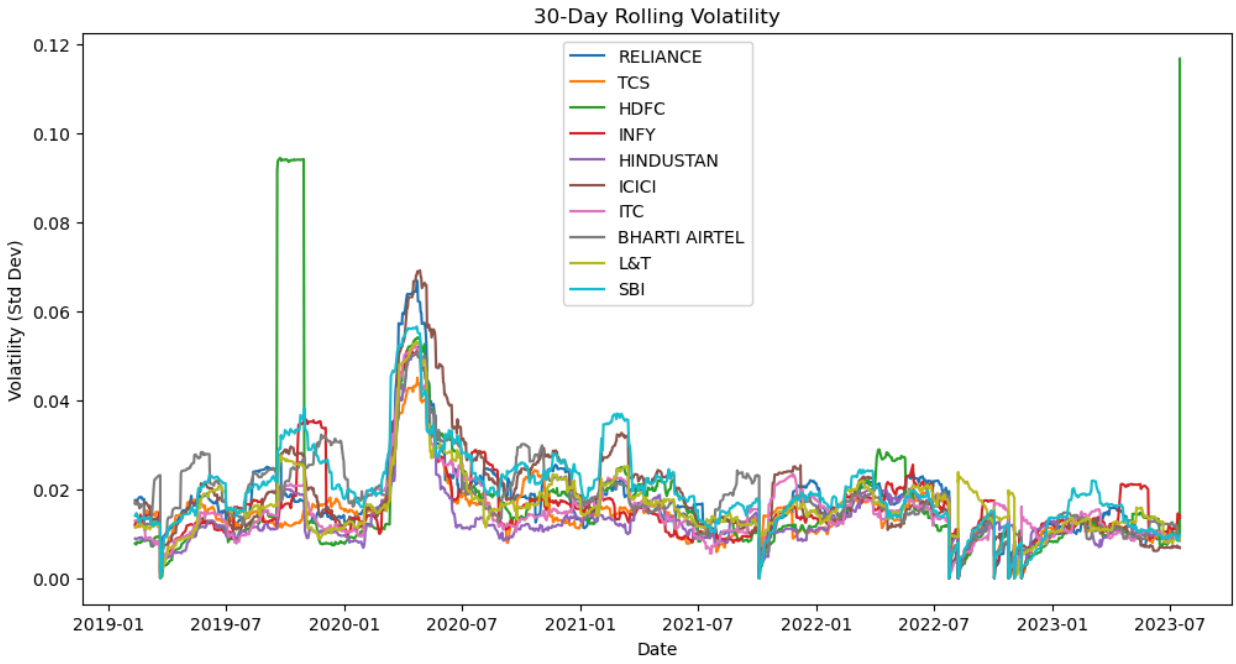


fig.5 30-day rolling volatility

Rolling Volatility Plot Shows 30-day rolling volatility for all portfolio assets over time. Purpose: Visualizes how risk levels change over time, identifies periods of high market turbulence, and compares volatility patterns across different stocks. Helps assess dynamic risk characteristics and time-varying correlation structures.

19. Black-Litterman Model Setup

○ **Formula:**

$\pi = \delta \cdot \Sigma \cdot W_{mkt}$

1. π (Implied Returns)

● The market's own "hidden" expectation of returns, based on current prices.

● We take this as a neutral starting point before adding our own predictions.

2. δ (Risk Aversion Coefficient)

● Shows how much investors dislike risk.

● High δ → Very cautious investors → Need higher returns for taking risks.

● Low δ → More risk-tolerant investors → Accept lower returns for more risk.

How δ is calculated from the stock prices

1. Takes current market prices of assets.
2. Computes market capitalization weights from those prices.
3. Uses historical return variance (or a covariance matrix) and the market's expected excess returns.
4. Solves for δ using the formula:

$\delta = w_m^T \Sigma w_m / \mu_m$

Where:

δ = market-implied risk aversion

○ w_m = market weights

○ Σ = covariance matrix of returns

○ μ_m = expected excess return of the market

3. Σ (Covariance Matrix)

● Measures how risky each asset is and how they move together:

○ Diagonal → Risk of each asset (variance).

○ Off-diagonal → Relationship between assets (covariance).

● This makes the matrix more stable and reliable.

A plain sample covariance matrix from historical data can be noisy, especially when you have many assets and not much data.
Noisy covariance matrices can make portfolio optimization unstable.
We calculate the shrunk covariance matrix of your portfolio's asset returns using the Ledoit-Wolf shrinkage method.
Shrinkage means:
Σ shrunk = $(1-\alpha)$ Σ sample + α Σ target
Where:
● Σ sample = normal covariance matrix from data
● Σ target = a simpler, more stable matrix (often the identity matrix scaled by variance)
● α = shrinkage intensity (automatically chosen by Ledoit-Wolf method)

The Ledoit-Wolf algorithm optimally picks α to minimize estimation error.

The result is less sensitive to outliers and small-sample noise, which makes it better for portfolio optimization, risk management, and Black-Litterman inputs.

4. Wmkt (Market Portfolio Weights)
● The percentage of each asset in the whole market.
● Example: In S&P 500, Apple might be 6%, Microsoft 5%, etc.
● Reflects how the market is actually invested today.
● We use the Market capitalization to represent this weight

5. Sharpe Ratio Calculation
○ **Formula:**
Sharpe Ratio=Rp−Rf / σp
■ Rp: Portfolio return
■ Rf: Risk-free rate (e.g., Treasury bill rate)
■ σp: Portfolio risk (standard deviation of returns)

❖ Interpretation:

■ 1.0 → Poor

■ 1–1.99 → Good

■ 2–2.99 → Very Good

■ ≥ 3.0 → Excellent

20. Ratios:

20.1 Sharpe Ratio

● What it measures: How well an investment performs relative to its risk.

● Formula: [(Return of Asset) - (Risk-Free Rate)]/ Standard Deviation

● What it tells you: It compares the excess return (how much extra you earn over a risk-free investment) to the risk (volatility). A higher Sharpe ratio means you are getting more return per unit of risk.
● What it does: This tells you how much return you're getting for each unit of risk you take. It looks at both gains and losses and compares them to the overall volatility (or up-and-down movement) of your investment.
● How it helps: When you are deciding between two investments, the one with a higher Sharpe ratio is better because it gives you more return for less risk. It helps you pick investments that are more efficient.

To compute a Sharpe Ratio, typically these parameters are used:

- **Risk-free rate** (Rf)
- **Expected return of the portfolio/asset** (Rp)
- **Standard deviation (volatility) of portfolio returns** (σp)
- Sometimes **daily/weekly returns** are annualized depending on frequency

20.2 Treynor Ratio

● What it measures: How well an investment performs relative to the risk taken, but it uses market risk (beta) instead of total risk (standard deviation).

● Formula: [(Return of Asset) - (Risk-Free Rate)]/ Beta

● What it tells you: It focuses on systematic risk (market risk) and is useful if you are comparing investments in relation to how they move with the market. A higher Treynor ratio means the asset is giving you more return for the market risk you are taking.
● What it does: This ratio is like the Sharpe ratio, but it focuses only on the market risk—the risk that comes from how your investment moves with the overall market.
● How it helps: If you're building a diversified portfolio (where the risks are spread out), the Treynor ratio helps you choose investments that give you better returns relative to the risk from the market. It's like saying, "How much return am I getting for the risk I take from the market itself?"

20.3 Sortino Ratio

● What it measures: Like the Sharpe ratio, but it only looks at downside risk (losses), not total volatility.

● Formula: [(Return of Asset) - (Risk-Free Rate)]/ Downside Deviation

● What it tells you: It focuses on minimizing negative returns (losses), so it's more concerned about how much risk is involved when the asset performs badly. A higher Sortino ratio indicates that the asset is more rewarding for the downside risk it's carrying.
● What it does: This one is like the Sharpe ratio, but it focuses only on the downside risk—the risk of losing money. It ignores the upside (the good times) and focuses on avoiding losses.
● How it helps: If you're more worried about losing money than just how much your investment moves up and down, the Sortino ratio helps you pick investments that are less likely to lose money. It's perfect for those who want to protect against big losses.

20.4 Calmar Ratio

● What it measures: How much return you get for each unit of risk, but specifically maximum drawdown risk (largest loss from peak to trough).

● Formula: Average Annualized Return / Maximum Drawdown

● What it tells you: It's useful for evaluating investments that have had significant downturns, like hedge funds. A higher Calmar ratio means the asset is returning more for the amount of loss it has endured during its worst periods.
● What it does: This ratio compares the return of an investment to its biggest loss (the biggest drop from its highest point to its lowest point).
● How it helps: If you are worried about major drops in your portfolio value, the Calmar ratio helps you find investments that give you a good return while avoiding big losses. It's great for people who want to protect their portfolio from severe downturns.

Summary of Key Differences:
● Sharpe Ratio: Total risk (volatility) vs. return.
● Treynor Ratio: Market risk (beta) vs. return.
● Sortino Ratio: Downside risk vs. return (focus on negative volatility).
● Calmar Ratio: Maximum drawdown risk vs. return.

In short:
● Sharpe is about overall volatility.
● Treynor is about how the investment moves with the market.
● Sortino is about downside risk.
● Calmar is about worst-case scenario drawdowns.

20.5 Ratio comparison Table:

Ratio	Risk Measure	Formula	Focus	Best For
Sharpe Ratio	Total risk (volatility)	[(Return of Asset) - (Risk-Free Rate)]/ Standard Deviation	Return relative to overall risk (volatility)	Comparing general risk-adjusted returns
Treynor Ratio	Market Risk (beta)	[(Return of Asset) - (Risk-Free Rate)]/ Beta	Return relative to market risk (systematic risk)	Comparing return based on market risk (beta)
Sortino Ratio	Downside risk (negative volatility)	[(Return of Asset) - (Risk-Free Rate)]/ Downside Deviation	Return relative to downside risk (negative returns)	Evaluating return with focus on negative volatility

20.6 How These Ratios Help You Optimize Your Portfolio?

● Balancing Risk and Reward: Each ratio helps you balance risk and return in a different way. By using them together, you can build a portfolio that matches your goals and how much risk you're willing to take.

● Choosing the Right Investments:

  ○ Use the Sharpe ratio to see which investment is giving you the best return for the least amount of risk.

  ○ Use the Treynor ratio to see how your investment behaves in relation to the market. If
  you are focused on market risk, this helps.

○ Use the Sortino ratio if you want to protect your portfolio from losses, focusing on
minimizing downside risk.
○ Use the Calmar ratio if you're worried about how much you could lose from the worst
case and want to avoid big drawdowns (huge losses).

● Building a Safer Portfolio: If you care more about avoiding big losses, focus on the Sortino or Calmar ratios. If you want the best balance of risk and return, the Sharpe ratio is great. If you're thinking about how your investments move with the market, then the Treynor ratio helps you focus on market risk.
● Adjusting Over Time: As your investments change, you can keep track of these ratios to make sure your portfolio is still doing well relative to the risk it's taking. If an investment has a low ratio, it might be time to replace it with something better.

In short, these ratios give you tools to understand how risky your investments are and how much reward you're getting for that risk. By using them, you can create a portfolio that matches your risk tolerance and financial goals—whether you are focusing on avoiding losses, getting the best returns, or handling market risks.

21.Use of Random Views in the Model

● Instead of getting analyst forecasts or investor opinions, you let the computer randomly assign positive or negative expectations within a range (–5% to +10%).

● This way, you can test how the Black–Litterman model reacts when different views are plugged in.

● We did this, to see how the Black–Litterman model changes the portfolio when you give it different opinions about stocks.

● By using random numbers, we can test the flexibility of the model — how it balances real market data with whatever "views" you feed it.

● It's like giving the model fake "investor guesses" and then watching how the recommended stock allocations shift.

● We let the computer pretend to be an investor, randomly guessing which stocks will go up or down. These fake guesses are called "random views." They are not real forecasts but help us test and demonstrate how the Black–Litterman model works when it blends opinions with market data.

22.Key EDA findings

1. Price Trends (2019–2024)

○ Line plots showed how each company's stock price evolved over time.

○ Some stocks displayed steady long-term growth (e.g., Reliance, TCS).

○ Others were more volatile, with sharp ups and downs (e.g., SBI, ITC).

○ A few showed declining or stagnant trends in certain periods.

2. Anomalies and Outliers

○ HDFC showed a clear abrupt jump in mid-2023, shifting between two very different price ranges.

○ This was linked to a corporate action (like a stock split or merger), not actual market behavior.

○ Such anomalies were cleaned to avoid misleading portfolio models (otherwise volatility and risk would appear artificially inflated).

3. Correlation Patterns

○ Heatmap analysis showed how strongly stock prices moved together.

○ High correlations suggest limited diversification benefits if both are included in a portfolio.

○ Some pairs were strongly positively correlated (moving together), while others showed weak correlations, offering diversification opportunities.

4. Cumulative Returns

○ A hypothetical ₹1 invested in each stock highlighted long-term winners vs. laggards.

○ Certain companies significantly outperformed others over the 2019–2024 period.

○ Others underperformed or had flat cumulative returns.

5. Rolling Volatility (30-day window)

○ Visualized how risk changed over time.

○ Stocks had spikes in volatility during market stress periods (e.g., COVID-19 in 2020, global uncertainty events).

○ Some remained relatively stable compared to more turbulent peers.

23. Results and Insights

1. Stable Covariance Matrix

○ Ledoit-Wolf shrinkage improved robustness against small-sample noise.

2. Market-Implied Returns

○ Provided a neutral starting point before integrating investor opinions.

3. Risk Diversification

○ Correlation analysis guided the inclusion of low-correlated assets.

4. Sharpe Ratio

○ Showed that the optimized portfolio outperformed the market in risk-adjusted terms.

24. Key Outcomes

- Developed a stable covariance matrix using Ledoit-Wolf shrinkage.
- Derived implied returns from market data without directly forecasting them.
- Identified risk-return trade-offs via the Sharpe Ratio.
- Produced a diversified portfolio recommendation.

25. Limitation

1.Black–Litterman Model

● Relies on investor views: Difficult to quantify subjective beliefs and confidence levels in real-world settings.

- ○ Quantifying views: Translating beliefs like "Reliance will outperform TCS by 2%" into precise numbers is not straightforward.
- ○ Confidence levels: Investors must specify how certain they are about each view. This is subjective and can bias results.

● Sensitive to chosen parameters (risk aversion δ, tau, confidence levels).

● Computationally more complex than Markowitz.

2.Dependence on Priors (Equilibrium Market Portfolio):

● The model starts with an assumed equilibrium portfolio (often derived from market-cap weights).

● If this starting point is not realistic, the final portfolio can be misleading.

3.Parameter Sensitivity:

● The results can be highly sensitive to:

- ○ The choice of the market equilibrium portfolio.
- ○ The confidence levels assigned to investor views.
- ○ Estimation of covariance matrix.

4.Interpretation of Views:

● Translating subjective investor opinions into precise numerical views (absolute or relative return expectations) is often difficult and subjective.

5.Not Universally Accepted Inputs:

● There's no standard way of choosing tau (τ), the uncertainty parameter that links the prior and views.

● Different choices of τ lead to different optimized portfolios.

6.Computational Intensity:

● For large asset universes, solving the equations becomes computationally demanding compared to simpler models.

26.Challenges

1. Data Cleaning

● Combining different CSVs may have formatting issues.

  ○ We converted the date format from "DD-MMM-YYYY" (e.g., 02-Jan-2024) to DD-MM-YYYY (e.g., 02-01-2024) to standardize the format for consistency and easier processing in subsequent analysis.

2.Outliers

● Events like the HDFC–HDFC Bank merger create sudden structural shifts in stock prices. If not adjusted properly, these appear as outliers, misleading the model's return calculations and risk estimates. Handling corporate actions consistently is a key challenge.

3.Overfitting Risk:

● If too many detailed views are added, the model may overfit to subjective expectations rather than capturing broad, robust market structure.

4.Estimating Covariance Matrix Accurately:

● Real-world financial returns are noisy and often non-stationary, so the covariance estimates may not reflect future risk.

5.Incorporating Realistic Views:

● Investors may have qualitative views (e.g., "Tech will outperform Energy") but converting them into precise expected returns is challenging.

6.Confidence Level Assignment:

● Assigning how confident an investor is about each view (which determines the weight it has in the model) is subjective and may lead to bias.

7.Data Requirements:

● Needs reliable and high-quality data on asset returns, market capitalization, and covariance – which may not always be available for emerging markets or alternative assets.

27.Stationarity Tests and Seasonality

- adfuller() was imported, to check for stationarity before ARIMA/SARIMAX modeling.
- The price series shows a clear upward trend suggesting non-stationarity.
- The rolling mean drifts upward over time. The rolling standard deviation also changes. Both indicate non-stationarity.
- Augmented Dickey-Fuller (ADF) Test:
    - Test Statistic = -1.78
    - p-value = 0.39 (> 0.05)
    - Critical values:
        - ➢ 1%: -3.43
        - ➢ 5%: -2.86
        - ➢ 10%: -2.57
- Fail to reject null hypothesis Reliance series is non-stationary.

28.Splitting the data into training and testing data

- The data was divided into Train and Test split
    - **80%** of the total length of the time series is considered as a **training** set.
    - Remaining **20%** is determined as the size of the **testing** set.
- Used **integer-location based indexing** to split the series

Why Chronological Split?

For time series forecasting, **chronological order must be preserved**, we cannot randomly split data like in regular machine learning, as it would cause data leakage which means using information from the future (test set) while training your model. This is unrealistic, because in real forecasting, you never know future prices when building your model.

29.ACF/PACF plots to justify ARIMA order selection

- ACF (Autocorrelation Function):
    - The ACF plot shows significant spikes at lag 1 and lag 2, after which the autocorrelation values quickly decline.
    - This pattern suggests the presence of a Moving Average (MA) component of order 2.

- PACF (Partial Autocorrelation Function):
    - The PACF plot also shows significant spikes at lag 1 and lag 2, then cuts off.
    - This behavior indicates the presence of an Autoregressive (AR) component of order 2.

- Interpretation Based on the plots:
    - AR order (p) = 2
    - Differencing order (d) = 1 (from stationarity test)
    - MA order (q) = 2

- Thus, the selected model ARIMA(2,1,2) is justified for Reliance.

30. Time Series Modeling

30.1 ARIMA

The ARIMA model (AutoRegressive Integrated Moving Average) is one of the most widely used statistical models for time series forecasting. It combines three components:

1. AR (AutoRegressive) – captures the relationship between an observation and its past values.

2. I (Integrated) – applies differencing to make the time series stationary (remove trends).

3. MA (Moving Average) – models the relationship between an observation and past forecast errors.

   Formula:

   ARIMA(p,d,q)

   where,

   - p: Number of autoregressive (AR) terms
   - d: Number of times the series must be differenced to achieve stationarity
   - q: Number of moving average (MA) terms

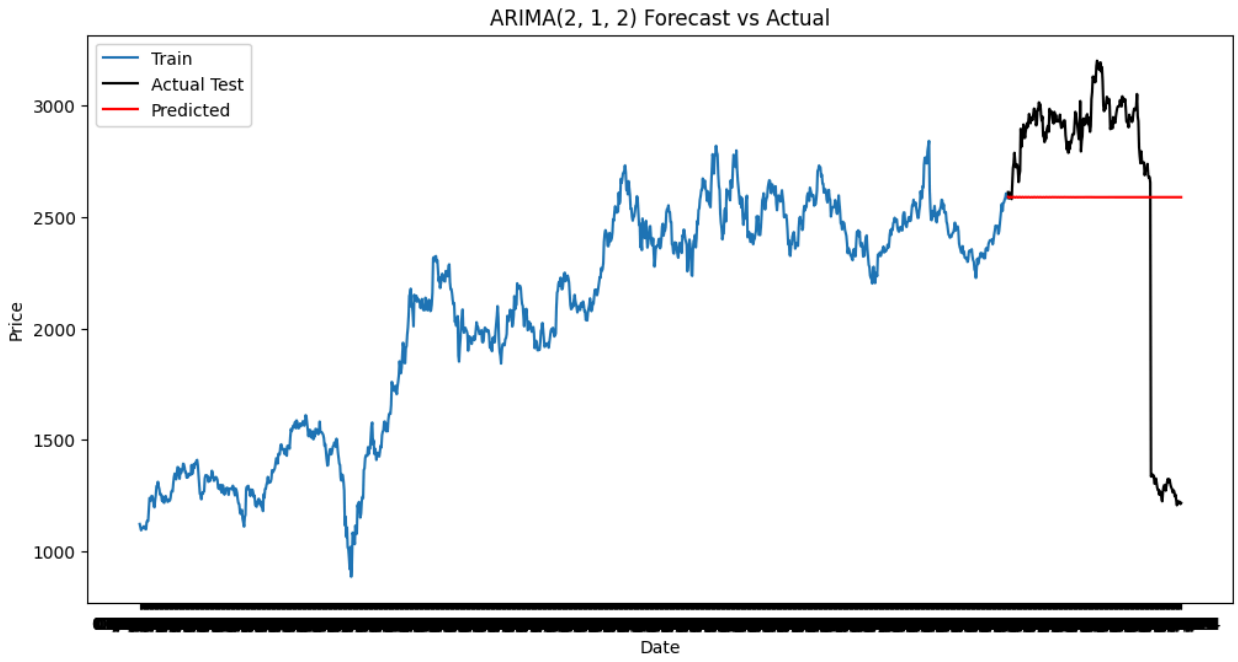


fig.6 ARIMA forecast vs actual

30.1.1 Why is ARIMA Unsuitable for Portfolio Optimization?

- ARIMA forecasts a single asset's future price/return (time-series focus) whereas, Black-Litterman optimizes capital allocation across multiple assets (cross-sectional focus).

- **Univariate vs. Multivariate:**
  - ARIMA models each asset in isolation, ignoring correlations.
  - Portfolio optimization relies on covariances between assets for risk management.

- **Model Input vs. Output:**
  - Black-Litterman uses historical covariance and market-implied returns as inputs whereas, ARIMA produces forecasts, not covariance matrices or equilibrium

returns.

- **Handling of Investor Views:**
    - ARIMA is purely data-driven and cannot integrate subjective views.
    - Black-Litterman combines market data with investor beliefs using a Bayesian approach.
- **Practical Drawbacks of ARIMA:**
    - Relies on past patterns, often unreliable in dynamic markets.
    - Requires tuning (p,d,q) for each asset, making it impractical at scale.
    - Ignores volatility dynamics (needs ARCH/GARCH for that).

30.2. SARIMA

SARIMA (Seasonal AutoRegressive Integrated Moving Average) is a statistical time series forecasting model that extends ARIMA by adding the ability to model seasonal patterns (repeating cycles like monthly, quarterly, or yearly effects).

Formula:

SARIMA(p,d,q)(P,D,Q)s

The general model is:

ΦP(Ls) ϕp(L) (1−L)d (1−Ls)D yt=ΘQ(Ls) θq(L) ε□

Where:

- y_t = value of the series at time ttt
- L = lag operator (shifts data back by 1 period, e.g. $Ly_t = y_{t-1}$
- d = order of non-seasonal differencing
- D = order of seasonal differencing
- p,q= non-seasonal AR and MA orders
- P,Q= seasonal AR and MA orders
- s = length of seasonality (e.g., 12 for monthly data with yearly seasonality)
- $\phi_p(L)$ = non-seasonal AR polynomial
- $\theta_q(L)$ = non-seasonal MA polynomial
- $\Phi P(Ls)$ = seasonal AR polynomial
- $\Theta Q(Ls)$ = seasonal MA polynomial
- ε_t = error (white noise)

ACF and PACF plot to find p, d, q values of differenced series

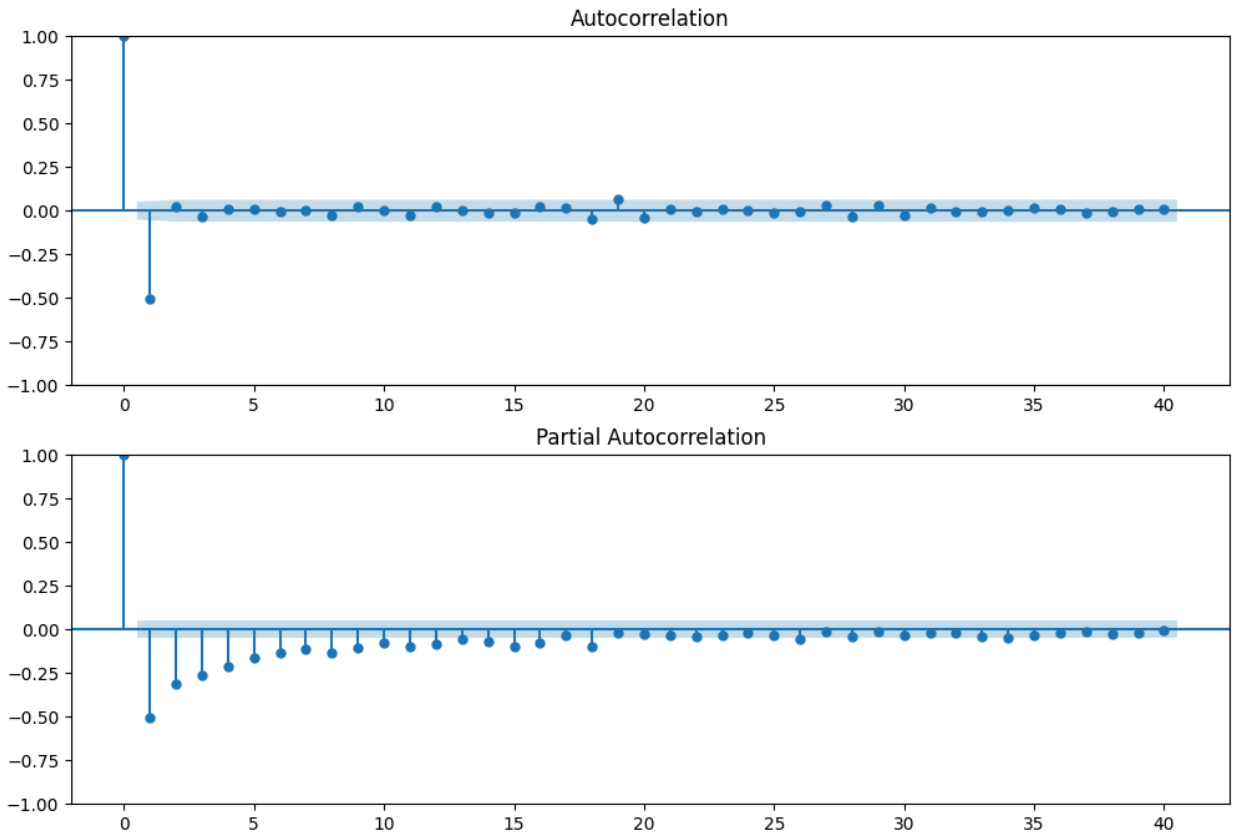


fig.7 ACF and PACF plots for p,d,q

ACF (top plot):

Big negative spike at lag 1, then everything else is close to zero. This is the textbook signature of an MA(1) process.

PACF (bottom plot) Negative spike at lag 1, then gradual tapering off. This is also consistent with an MA(1) model.

Conclusion:

Differencing was necessary which gives d = 1.

ACF/PACF indicate q = 1, p = 0.

ACF and PACF plots to identify seasonal patterns

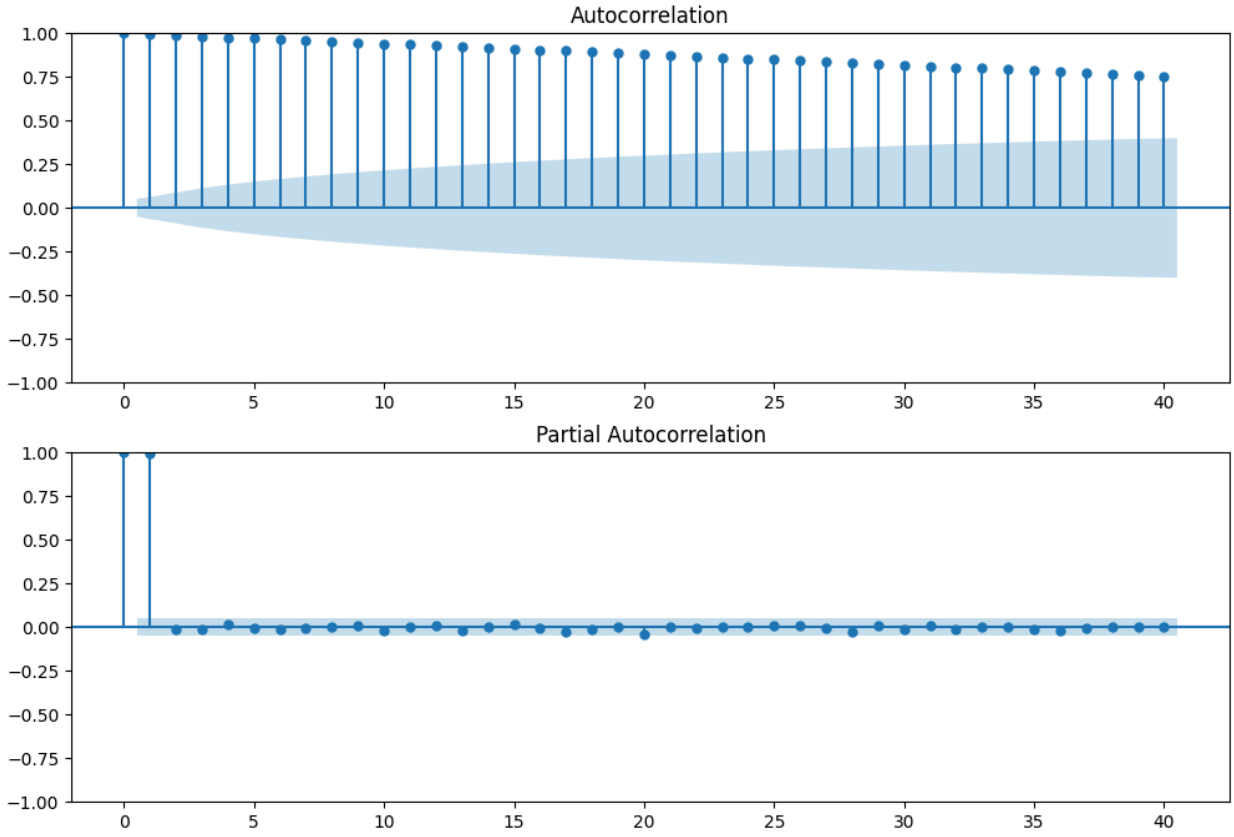
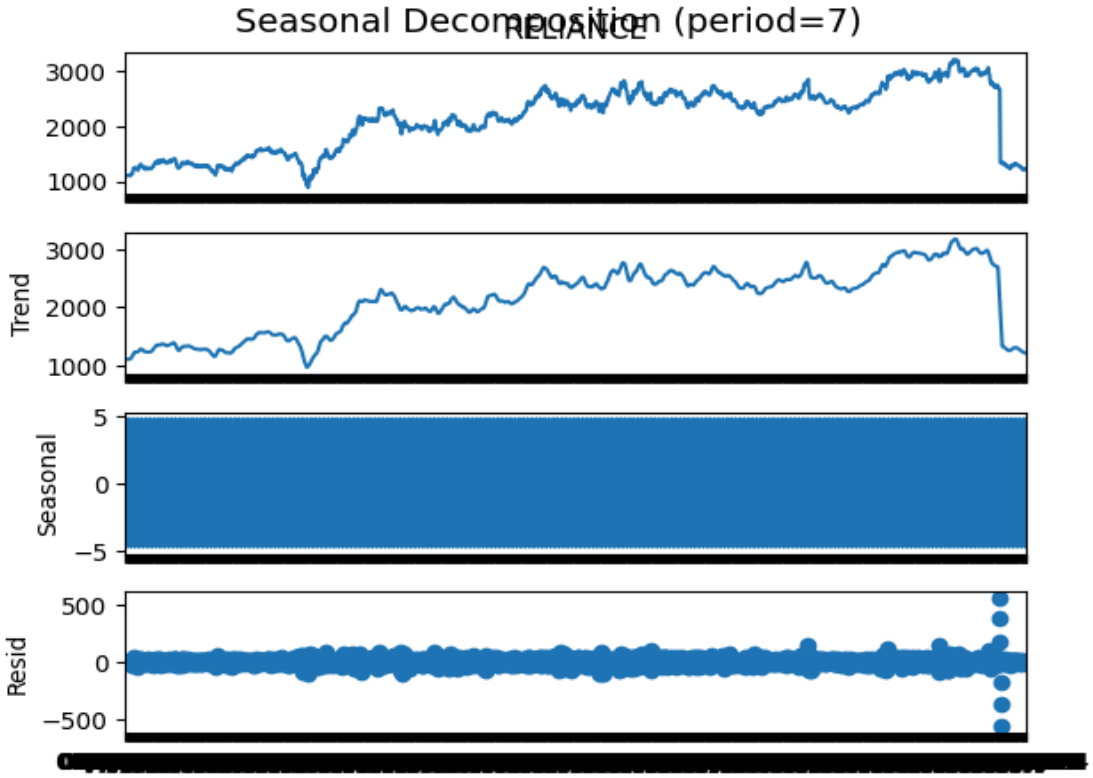
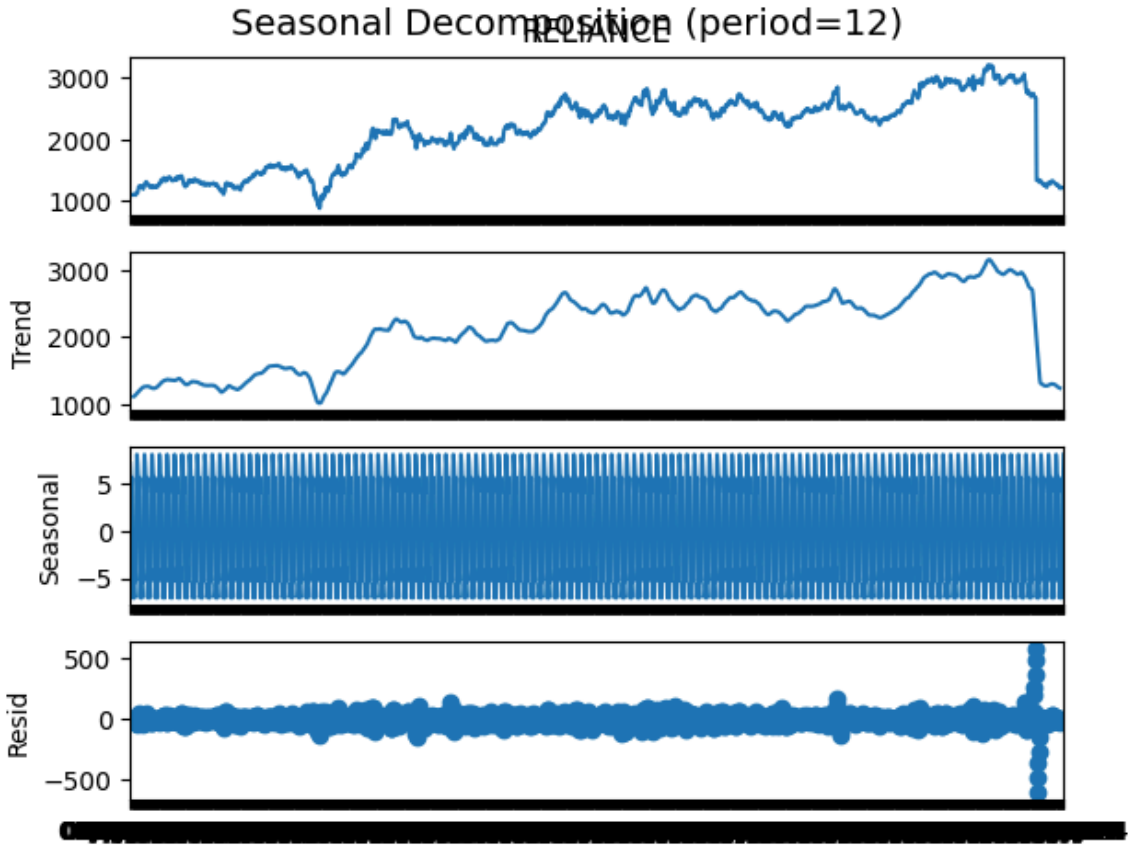


fig.8 ACF and PACF plots for P,D,Q

No obvious seasonal spikes (e.g., at 12, 24, etc.), so P = D = Q = 0. So your series is best described by: SARIMA(0,1,1)(0,0,0,s).

Checking the value of s(seasonal period)


Seasonal Decomposition (period=7)
RELIANCE
3000
2000
1000
Trend
3000
2000
1000
Seasonal
5
0
−5
Resid
500
0
−500


Seasonal Decomposition (period=12)
RELIANCE
3000
2000
1000
Trend
3000
2000
1000
Seasonal
5
0
−5
Resid
500
0
−500

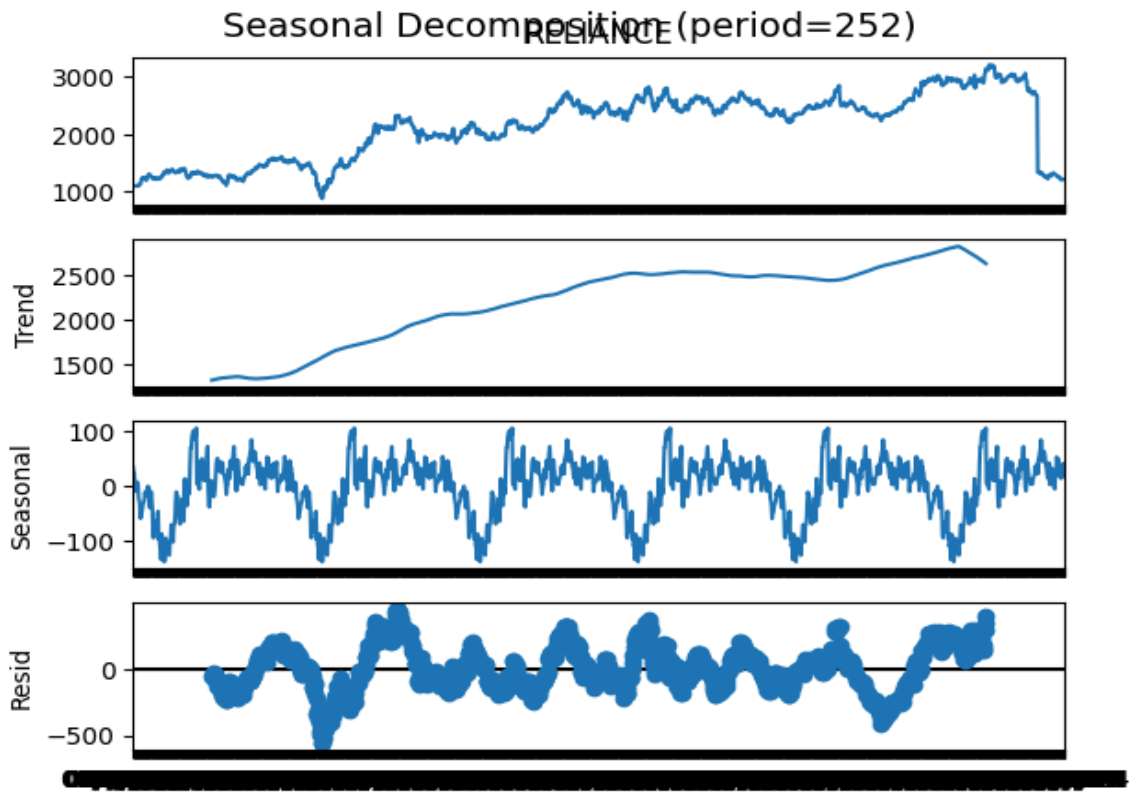


fig.9 decomposition plot to check seasonality

Decomposition with m=12 (monthly cycle)

The seasonal component looks very noisy and repetitive, not a clear economic cycle.

This suggests no real 12-monthly seasonality in daily Reliance stock prices.

Decomposition with m=252 (yearly trading days)

The seasonal component shows some structured up-and-down cycles.

This is much more plausible — stocks often exhibit annual effects (e.g., fiscal year cycles, market behavior patterns).

The trend and seasonal parts make sense here.

Conclusion:

m=252 (yearly seasonality) makes the most sense.

m=12 is not appropriate for daily stock data.

m=5 (weekly) could be tested, but decomposition didn't show much support.

SARIMA Model Output:

SARIMA Model Evaluation Metrics:

MSE = 0.00

MAE = 0.01

RMSE = 0.03

R^2 = -0.01

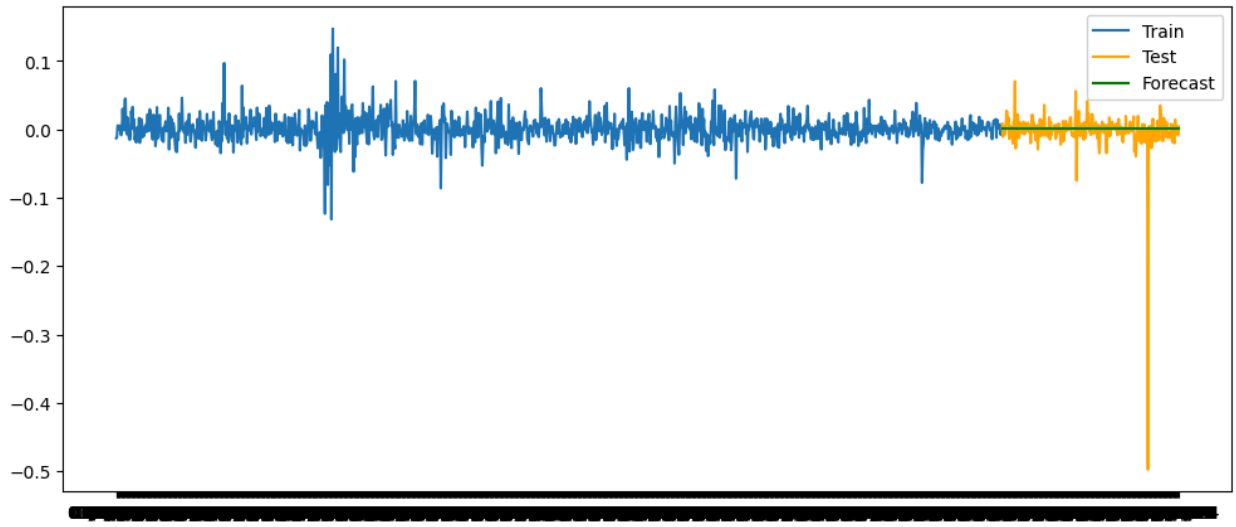


fig.10 SARIMA(0,1,1)(0,0,0,252)model output of Forecast vs Actual Returns on Date

30.2.1 Why SARIMA Fails on Stock Data?

- **Non-stationarity beyond differencing**
    - SARIMA assumes the series can be made **stationary** (constant mean, variance) using differencing.
    - Stock prices are usually a **random walk** (unit root process), meaning they don't have a fixed mean or variance.
    - Even after differencing, the variance and shocks don't stabilize well.
- **No strong seasonality**
    - SARIMA is powerful when there's **clear seasonality** (monthly sales, yearly cycles, weather).
    - Stock prices don't follow clean seasonal cycles → they are influenced by news, macro events, policies, investor sentiment.
- **High noise & volatility clustering**
    - Stock data contains **random noise and volatility clustering** (sudden spikes/drops).
    - SARIMA assumes residuals are white noise → but in finance, they're not (ARCH/GARCH models work better for volatility).
- **Structural breaks**
    - In 6 years of stock data, you'll see regime changes (COVID crash, policy shifts, elections, etc.).

- SARIMA assumes the underlying process is stable → this breaks down in financial markets.

- **Poor predictive power**
  - At best, SARIMA captures **short-term autocorrelation** (like momentum).
  - For stock returns, autocorrelation is very weak — making SARIMA no better than a naive forecast (yesterday's price ≈ today's price).

30.3 SARIMAX

SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous factors) is an extension of SARIMA.

- SARIMA models only the past values of the series + seasonal effects.
- SARIMAX adds exogenous variables (X) — i.e., other independent factors that can influence the series.
- This makes it more flexible for forecasting when outside factors matter.
- The exogenous variables included in our model are "TCS" and "INFY"

Formula :

$$y_t = SARIMA(p,d,q)(P,D,Q)s + \beta X_t + \varepsilon_t$$

Where:

- y_t = target time series
- SARIMA(p,d,q)(P,D,Q)s = seasonal ARIMA component

- X□ = exogenous (external) variables (e.g., interest rates, oil prices, weather, GDP, etc.)
- β = coefficients for exogenous variables
- ε□= error term

SARIMAX Model Output:

MSE = 0.0016

MAE = 0.0204

RMSE = 0.0399

R^2 = -0.3291

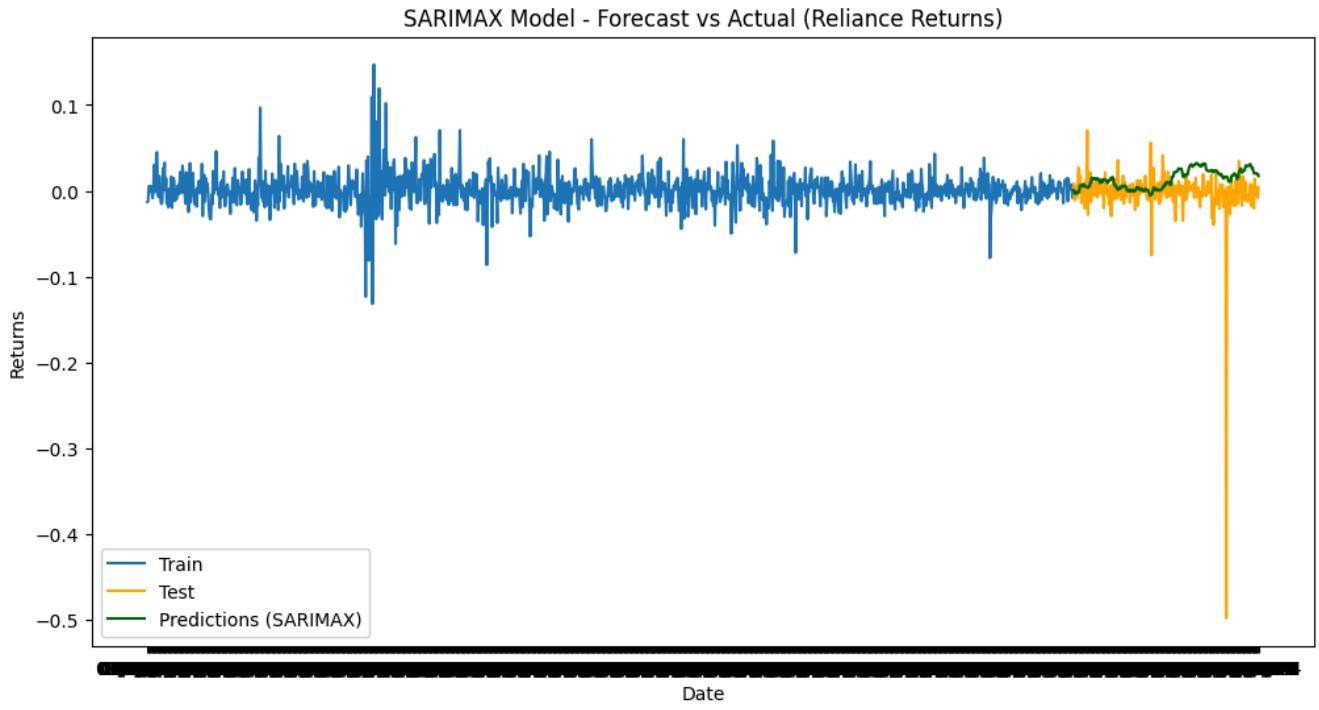


Fig.11 SARIMAX(0,1,1)(0,0,0,252)model output of Forecast vs Actual Returns on Date

30.3.1 Why does SARIMAX fail in stock market data?

- **Stock prices are nearly random walks**

    - The Efficient Market Hypothesis (EMH) says prices reflect all available info.
    - Tomorrow's price is basically today's price + random shock.
    - SARIMAX (like SARIMA) assumes there are **predictable patterns**, which often don't exist in raw stock prices.

- **Exogenous variables don't explain much**
    - You can add GDP, interest rates, oil prices, etc., but stock movements are often driven by **sentiment, news, and sudden shocks** that aren't in the exogenous dataset.
    - The relationship between macro variables and stock prices is **weak, nonlinear, and unstable**.

- **Non-stationarity + structural breaks**
    - In 6 years you'll see events like COVID-19, elections, policy shifts, war, etc.
    - SARIMAX assumes stable relationships, but markets **change regimes** (bull vs bear markets).

- **Short sample size for complex dynamics**
    - 6 years = ~1500 daily points or ~72 monthly points.
    - That's **too little** for stable SARIMAX fitting, especially if you add exogenous regressors.

- **High noise-to-signal ratio**

- Stock data is dominated by noise.

- SARIMAX tries to find patterns in noise → leading to **overfitting** with poor out-of-sample forecasts.

- **Linear model in a nonlinear world**

  - SARIMAX assumes a **linear relationship** between past values + exogenous features.

  - Stock prices have **nonlinear dependencies** (momentum bursts, volatility clusters) that SARIMAX cannot capture.

30.4 ARCH Model

The ARCH model, introduced by Robert Engle (1982), is used to model and forecast time-varying volatility in time series data. It assumes that the current period's variance depends on the squared errors (shocks) from previous periods. It is widely used in finance for modeling asset return volatility.

Formula:

$$y_t = \mu + \epsilon_t, \quad \epsilon_t = \sigma_t z_t, \quad z_t \sim N(0,1)$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2$$

Where:

- y_t: Time series value
- μ: Mean (constant or deterministic trend)
- ϵ_t: Error term (innovation) at time t
- σ_t^2: Conditional variance at time t
- z_t: White noise process (E[zt]=0, Var(zt)=1)
- $\alpha_0 > 0$, $\alpha_i \geq 0$: Parameters to be estimated

- q: Order of the ARCH model (number of past squared shocks)

Explanation

The variance ($\sigma\square^2$) at time t depends on past squared errors.

If past shocks ($\epsilon\square-i^2$) were large, the variance today will be high, meaning higher volatility.

Captures volatility clustering – periods of high volatility followed by high volatility, and low by low.

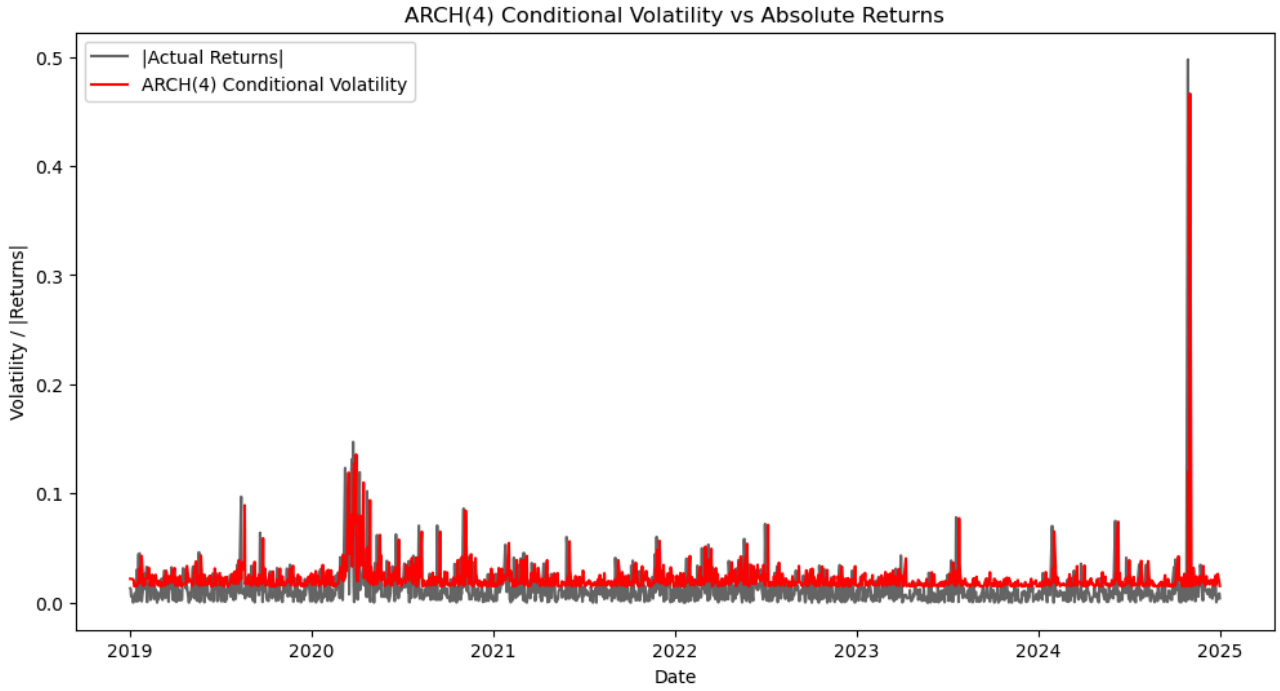


Fig. 12 ARCH(4) Conditional Volatility vs Absolute Returns

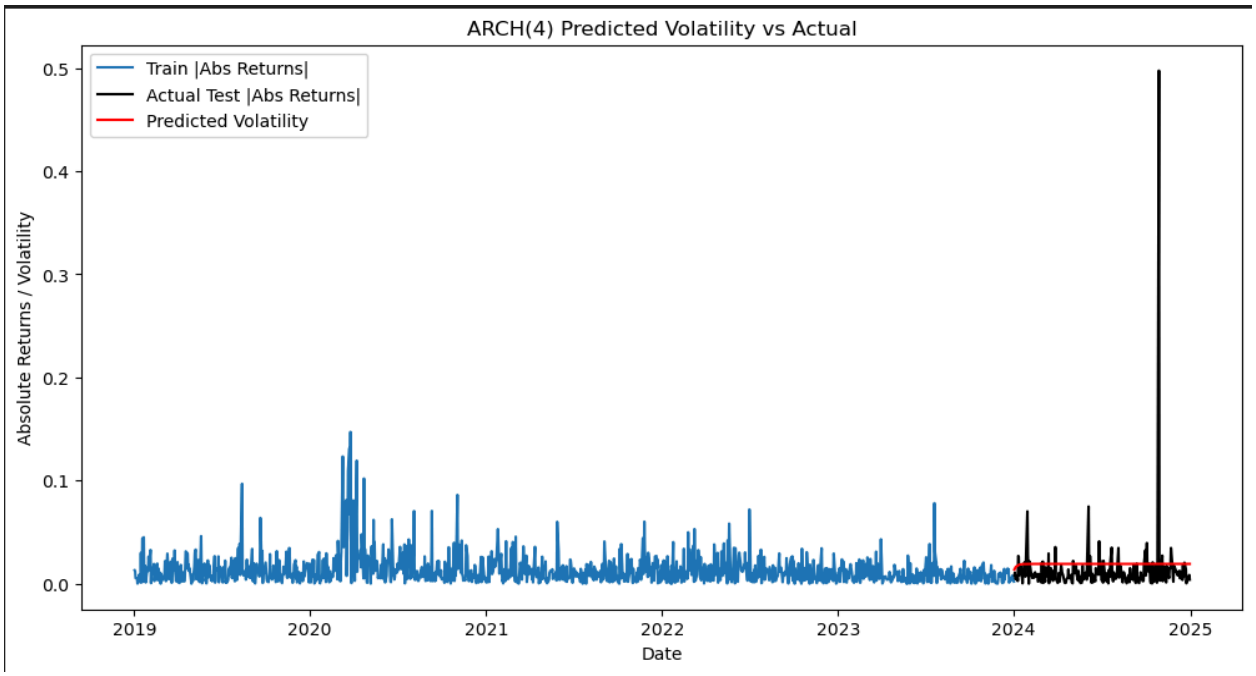


Fig.13 ARCH(4) Predicted Volatility vs Actual Volatility

30.4.1 Why is it NOT implied in the project?

ARCH(4) is not suitable for your project because:

1. **Poor Prediction**: The predicted volatility (red) diverges significantly from actual returns (black) after 2024, indicating inaccurate forecasting.

2. **Misaligned Volatility**: The predicted volatility doesn't match actual returns, especially during periods of extreme volatility.

3. **Inability to Capture Outliers**: The model fails to account for large spikes in returns, which are crucial for accurate volatility modeling.

Drawback of ARCH itself: The pure ARCH model has a major limitation: it often requires a very high order q to capture the persistence of volatility shocks in financial data, making it cumbersome. This weakness is precisely why the GARCH model was developed.

30.5 GARCH Model

The GARCH model, introduced by Bollerslev (1986), is a generalization of the ARCH model. It includes both past squared errors (ARCH terms) and past conditional variances (GARCH terms) to model volatility more efficiently with fewer parameters.

Formula:

GARCH(p, q) model:

$y_t = \mu + \epsilon_t, \epsilon_t = \sigma_t z_t, z_t \sim N(0,1)$

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

Where:

- p: Order of GARCH terms (past variances)
- q: Order of ARCH terms (past squared shocks)
- β_t: Coefficients for past variances

Other notations are as defined above.

Explanation

Combines ARCH effects (past squared shocks) and GARCH effects (past variances).

More flexible and parsimonious than a high-order ARCH model.

If p=0, the GARCH reduces to ARCH.

Widely used in financial time series like stock returns, exchange rates, etc.

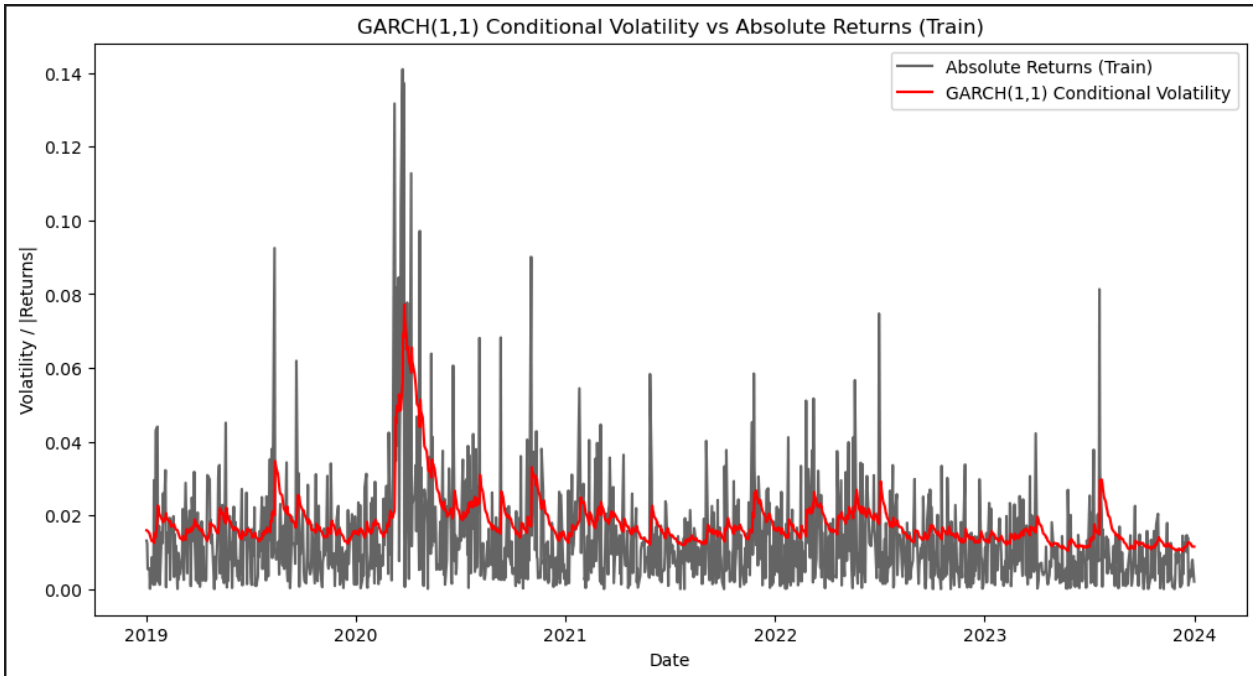


Fig.14 GARCH(1,1) Conditional Volatility vs Absolute Returns

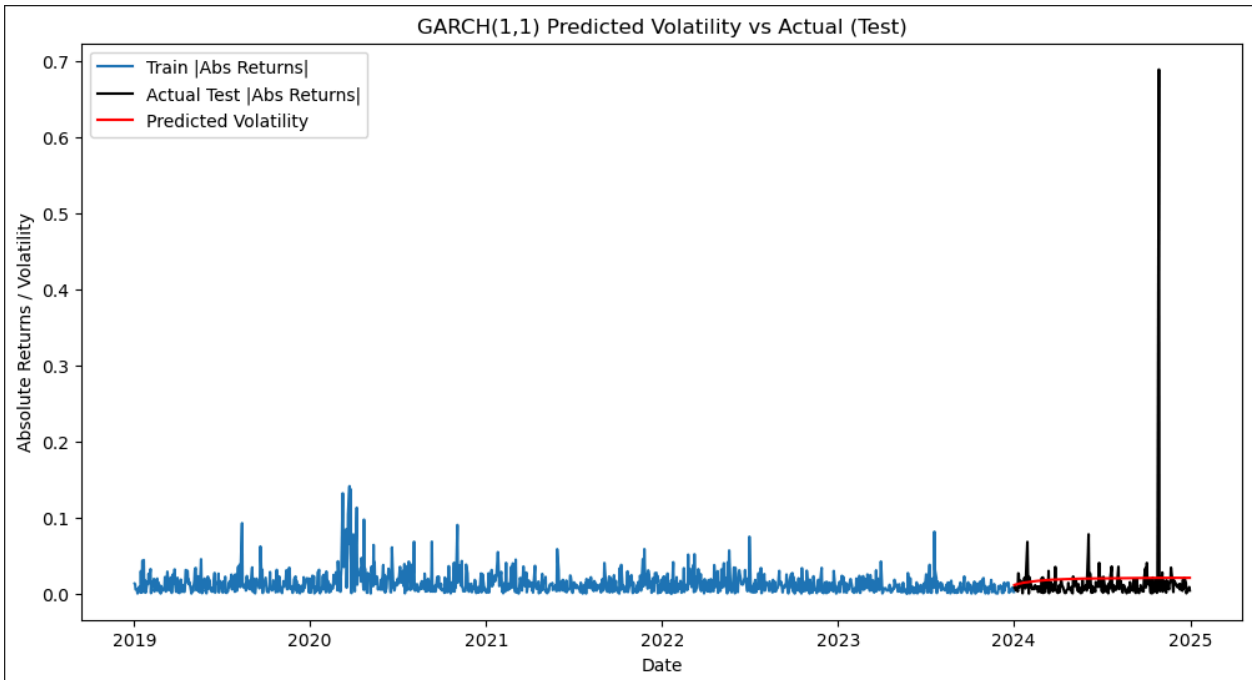


Fig 15 GARCH(1,1) Predicted Volatility vs Actual Volatility

30.5.1 Why is GARCH NOT used for Stock Market Data?

The graphs show that **GARCH(1,1) is not suitable** because:

- **In training (Graph 1):** It smooths volatility and misses sharp spikes.
- **In testing (Graph 2):** Predicted volatility stays flat while actual returns show large spikes → poor out-of-sample performance.
- **Reason:** GARCH assumes stable volatility dynamics, but your data shows **structural breaks, jumps, and heavy tails**, which GARCH cannot capture.

Key Difference Between ARCH and GARCH

ARCH: Uses only past squared shocks.

GARCH: Uses both past squared shocks and past conditional variances (more efficient).

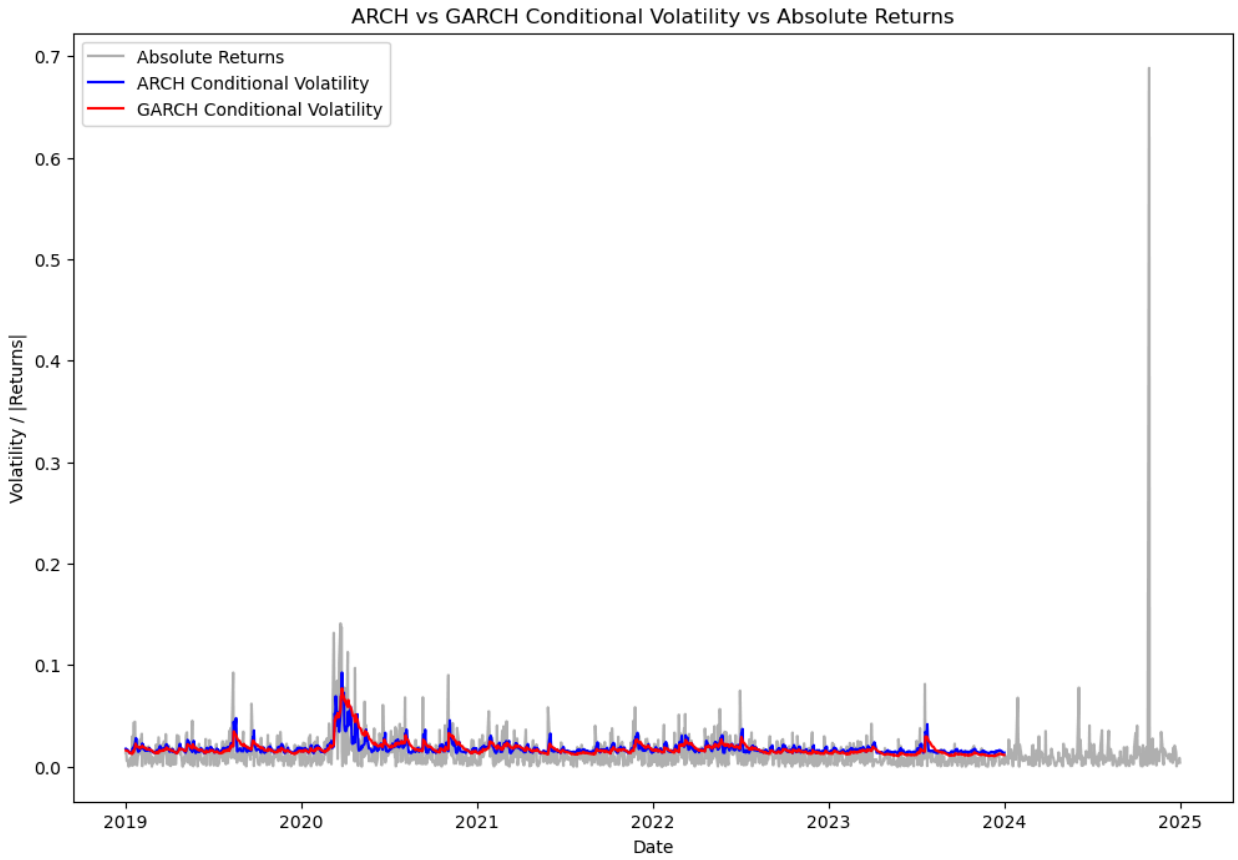


Fig 16 ARCH vs GARCH: Conditional Volatility vs Absolute Returns

ARCH reacts to short-term shocks but is noisy, while GARCH smooths volatility and captures persistence better. However, both fail to model extreme spikes (e.g., 2025), showing limitations under structural breaks and fat-tailed returns.

30.6 VAR

A VAR (Vector Autoregression) model in time series analysis is a statistical model used to analyze and forecast multiple time series that influence each other. It's a multivariate extension of the univariate autoregressive model, where each variable in the system is modeled as a linear function of its own past values and the past values of all other variables in the system. VAR models are particularly useful in economics and finance for understanding dynamic relationships, making forecasts, and performing policy analysis

Mathematically, a VAR(p) model with 'p' lags can be represented as:

Y□=c+Φ1Y□−1+Φ2Y□−2+⋯+ΦpY□-□+ε□

Here,

- Yt: This represents the value of the time series at time t.
- c: This represents the constant intercept term in the model.
- Φ1,Φ2,...,Φp: These represent the autoregressive coefficients for lags 1, 2, ..., p, respectively.
- Y□−1,Y□−2,...,Y□-□: These represent the values of the time series at lags 1, 2, ..., p before time t.
- ε□: This represents the error term at time t.

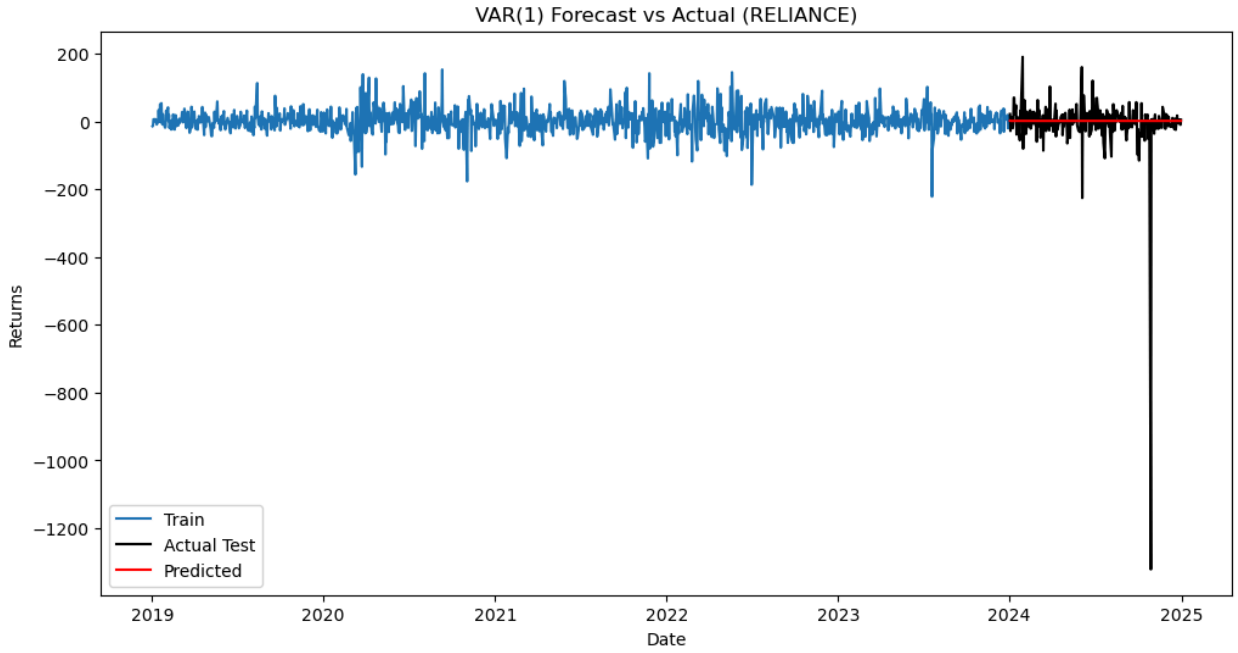


Fig.17 VAR(1) Forecast vs Actual Returns

30.6.1 Why can VAR not be used in this model?

Based on the graph, VAR may **not** be suitable due to:

- **Large Outliers**: The forecast shows extreme spikes, indicating outliers that VAR struggles to handle.

- **Diverging Forecasts**: Predicted returns deviate significantly from actual returns, suggesting poor model performance.

- **High Volatility**: The data appears volatile and possibly non-stationary, which VAR models are sensitive to.

30.6.2 Model Output Comparison Table:

Models	R^2	MSE	MAE
ARIMA	-0.0025	403589.01	497.50
ARCH	-0.04524	0.00110	0.01348
GARCH	-0.01863	0.001983	0.01440
SARIMA	-0.01	0.00	0.01
SARIMAX	-0.3291	0.0016	0.0204
VAR	-0.005318	8573.23251	31.76287

31. A simple implementation of Black-Litterman:

31.1 Data Loading and Preparation

- We collected historical stock prices from NSE (NIFTY50).
- The datasets were cleaned, and dates were changed to date-time format.
- Focus was placed on the **most recent 6 months** to assess recent market behavior.
- Daily returns (percentage change in prices) were calculated for each stock.

31.2 Evaluating Stock Performance (Risk-Adjusted Ratios)

- For each stock, we calculated four key performance ratios:
    - **Sharpe Ratio**: Return compared to overall risk.
    - **Sortino Ratio**: Return compared to downside (bad) risk.
    - **Treynor Ratio**: Return compared to how much the stock moves with the market.
    - **Calmar Ratio**: Return relative to the largest loss seen.
- Returns and risk values were **annualized for standard comparison**.
- **NIFTY 50** was used as the benchmark for market movement.
- A **7% annual risk-free rate** was assumed for all calculations.

31.3 Preparing Data for Black-Litterman

- We included **market capitalizations** of all stocks.

- Calculated:

    - **Covariance matrix** (to understand how stocks move together) using a more stable method (Ledoit-Wolf shrinkage).

    - **Risk aversion factor** (delta) to gauge market sentiment.

    - **Market-implied returns** based on market capitalization weights.

31.4 Black-Litterman Model with Random Views

- Since we didn't have expert opinions, we created **random views** for each stock (expected return range between –2% and +10%).

- A **confidence matrix (omega)** was added to represent uncertainty in these views.

- The **Black-Litterman model** then combined market data and these views to estimate adjusted (posterior) returns for each stock.

31.5 Portfolio Optimization

- Used **PyPortfolioOpt's Efficient Frontier** method to find the **best stock allocation**.

- Objective: **Maximize the Sharpe ratio** (best return-to-risk balance).

- Added a small penalty (L2 regularization) to avoid extreme weights.

- Produced a final portfolio with percentage weights for each stock.

31.6 Testing the Portfolio (Out-of-Sample)

- Data was split: older data for training, **last 6 months for testing**.
- The portfolio was tested on this unseen period to ensure realistic performance.
- Risk-adjusted metrics (Sharpe, Sortino, Treynor, Calmar) were recalculated for:
    - Each individual stock
    - The optimized portfolio
- A **comparison table** was created to show which performed best.

31.7 Visual Insights

- **Correlation heatmap** showed how stocks are related to each other.

- **Portfolio weight chart** displayed final allocations.

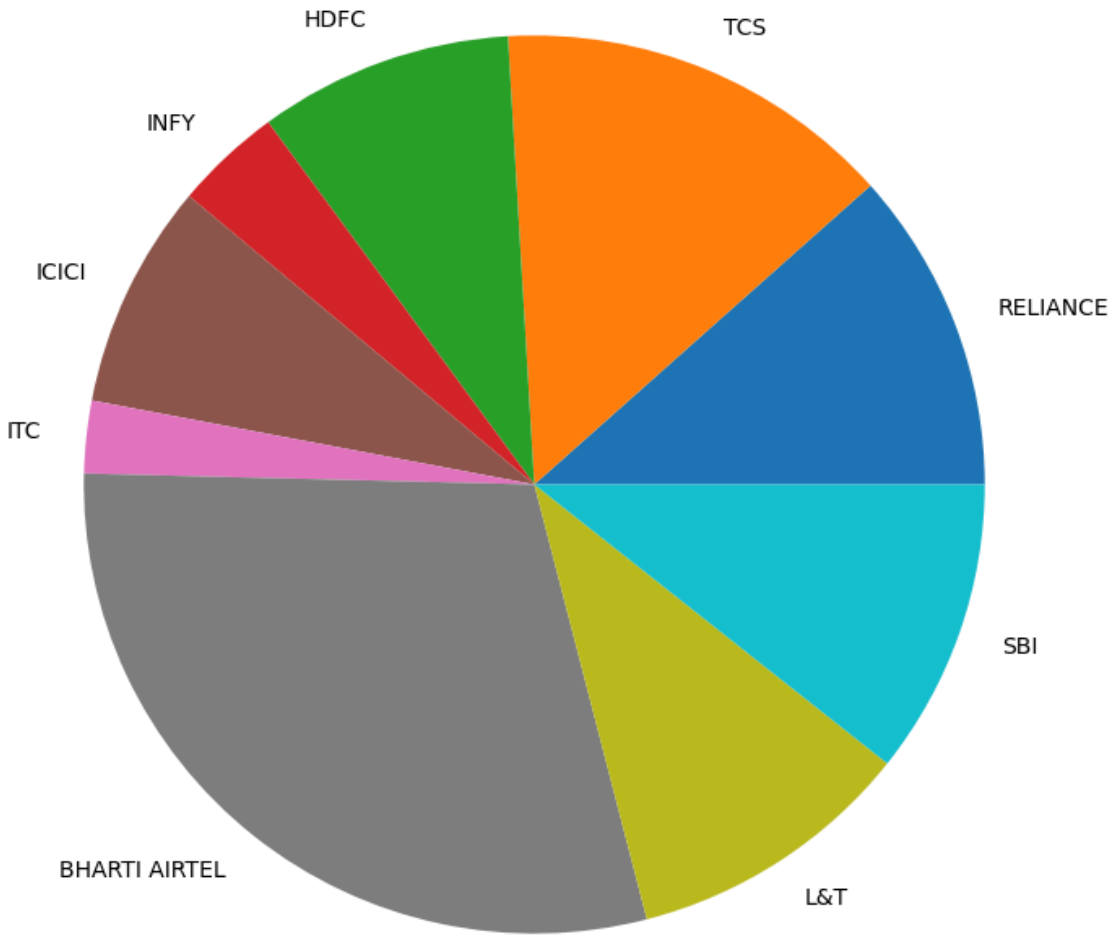


Fig 18 optimal portfolio allocation pie chart

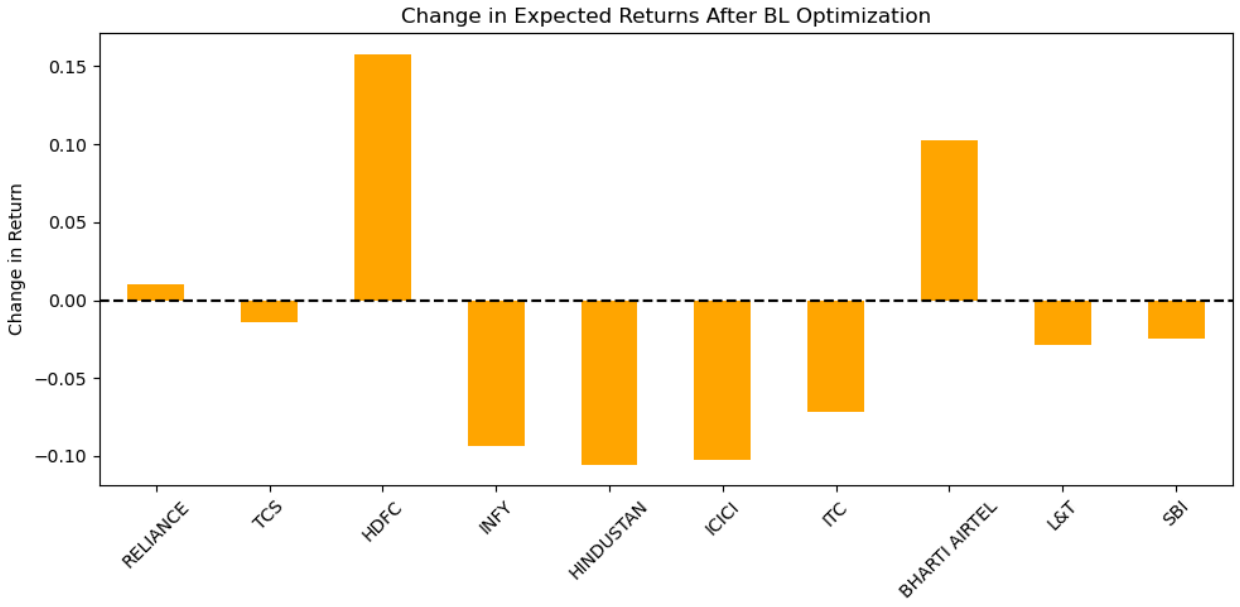
Optimal Portfolio Allocation Pie Chart This pie chart visualizes the optimal asset allocation derived from the Black-Litterman model and mean-variance optimization, showing how the portfolio should be distributed across different assets.

Key Insights:

Diversification Level: Many small slices indicate Well-diversified portfolio Few large slices indicate Concentrated portfolio

View Implementation: Assets with positive views typically receive higher allocations Assets with negative views may be underweighted or excluded

Risk-Return Balance: Allocation reflects both return expectations and risk characteristics Regularization ensures no extreme concentrations



This plot shows how the **expected returns** of each stock changed after incorporating investor views via the Black-Litterman model.

- **Positive values (above zero line):** Expected return increased after optimization.

  HDFC and **Bharti Airtel** gained the most in expected return → market/prior + your views combined to strongly favor them.
  Reliance had a slight positive adjustment.

- **Negative values (below zero line):** Expected return decreased after optimization.

Infosys (INFY), **Hindustan**, and **ICICI** had the largest drop in expected returns, meaning the BL model adjusted them downward (possibly due to negative or weaker views relative to priors).
TCS, ITC, L&T, SBI also declined slightly.

31.8 Key Highlights

- Built a **dynamic portfolio using the Black-Litterman model** without relying on expert analysts.

- Evaluated the portfolio using **realistic, out-of-sample testing**.

- Provided **both individual stock insights and overall portfolio performance**.

32. Which model fits the best?

- Prior (Market-Implied Model):
    - This model uses only market data without making any additional assumptions or adjustments.
    - It generally predicted **higher returns for most stocks**, as it reflects the market's natural outlook.
    - Best suited if your goal is to **follow market expectations directly** and take a more aggressive stance based on what the market already prices in.

- Posterior (Black-Litterman with Random Views)
    - This model started with the market's outlook but then **adjusted the returns using additional "views" (here, random views since expert opinions were unavailable)**.
    - As a result, the expected returns were **reduced for most stocks**, creating a more **conservative and balanced portfolio**.
    - While it may lower potential gains, it also **reduces risk and volatility**, which can improve the portfolio's **risk-adjusted performance (e.g., Sharpe or Sortino ratio)**.
    - Best suited when you want **better risk control** or have access to reliable expert insights.

32.1 Pros of the Prior (Market-Implied) Model:

1. **Simple and transparent** – Uses only market data without adding assumptions.

2. **Reflects real market expectations** – Directly based on capitalization and market consensus.

3. **Higher return potential** – Often projects higher returns for most stocks.

4. **Easy to implement** – No need for expert opinions or subjective adjustments.

32.2 Cons of the Prior (Market-Implied) Model:

1. **No risk adjustment from views** – Does not incorporate any additional information (e.g., analyst opinions, macroeconomic factors).

2. **Potentially over-optimistic** – Can overestimate returns, especially in volatile or declining markets.

3. **Lacks flexibility** – Cannot adapt easily if new market insights become available.

32.3 Pros of the Posterior (Black-Litterman with random views) model:

1. **Balances market data with views** – Combines market expectations with additional opinions (even if random here) to adjust returns.

2. **Improved risk control** – Produces more conservative portfolios by reducing extreme exposures.

3. **Reduces overconfidence in market trends** – Helps prevent over-optimistic allocations.

4. **More flexible** – Can easily incorporate credible expert views or forecasts.

32.4 Cons of the Posterior (Black-Litterman with random views) model:

1. **Dependent on quality of views** – If views are inaccurate or random (as in this case), results may deviate from reality.

2. **Can reduce return potential** – Often lowers expected returns, which may limit upside.

3. **More complex** – Requires additional parameters (views, confidence matrix, etc.).

4. **Interpretability challenges** – Harder for non-technical stakeholders to understand compared to pure market-based models.

33. Conclusion:

This project establishes a solid foundation for stock price forecasting using closing prices as the primary indicator. Through thorough EDA, strategic feature engineering, and robust modeling techniques, insights into market behavior were achieved. Future work may involve real-time data streaming, broader market indicators, and deployment of predictive dashboards.

34. References:

F. Black and R. Litterman, "Global asset allocation with equities, bonds, and currencies," Goldman Sachs Asset Management, Oct. 1991. [Online].

https://www.nseindia.com/ – Historical stock data via NSE

https://towardsdatascience.com/the-black-litterman-model-for-portfolio-optimization-6b3fcd33e662

https://www.investopedia.com/terms/b/black-litterman-model.asp