Project Documentation

For

Data-Driven Stock Analysis

Version 1.0

Prepared by

Name: Mohamed Niyas
Batch No: MAE1

Release Notice

This is Version 1.0 of Data-Driven Stock Analysis Project Documentation

Name	Batch No	Organization
Prepared By		
Mohamed Niyas	MAE1	Guvi

Version History:

Version No.	Date Issued	Remarks
1.0	23 Jan 2025	Initial Release

Table of Contents

NAME: MOHAMED NIYAS 1
BATCH NO: MAE1 1

1 INTRODUCTION 5

1.1.1 Purpose Of the Document 5
1.1.2 Scope 5
1.1.3 Business Use cases 6

2 DATA-DRIVEN STOCK ANALYSIS ARCHITECTURE DIAGRAM 7

2.1.1 Overview 7
2.1.2 Data Extraction: 7

3 INSTALLATION 8

3.1.1 Prerequisites 8

4 CLONE THE REPOSITORY 9

5 INSTALL DEPENDENCIES 10

5.1.1 requirements.txt file 10
5.1.2 install the dependencies: 10

6 MYSQL DATABASE SETUP 11

6.1.1 Create a MySQL Database: 11
6.1.2 Create the required Tables: 11

7 CONFIGURATION 12

7.1.1 Update Database Connection 12

8 RUN THE APPLICATION 13

8.1.1 Access the Application 13

9 FEATURES 14

9.1.1 Home Page 14
9.1.2 Home Page Wireframe 14
9.1.3 Data Cleaning 14
9.1.4 Search Page Wireframe 15
9.1.5 Volatility Analysis 15
9.1.6 Volatility Analysis Wireframes 16
9.1.7 Cumulative Return 17
9.1.8 Cumulative Return wireframe 19
9.1.9 Sector-wise Performance 20
9.1.10 Sector-wise Performance wireframe 21
9.1.11 Stock Price Correlation 22
9.1.12 Stock Price Correlation wireframe 23
9.1.13 Top 5 Gainers and Losers (Month-wise) 24
9.1.14 Top 5 Gainers and Losers (Month-wise) wireframe 25

10 INSIGHTS AND CHALLENGES 27

10.1.1 Insights 27
10.1.2 Challenges 27

11 TROUBLESHOOTING 28

11.1.1 Common Issues..... 28

1 Introduction

Welcome to **Data-Driven Stock Analysis**, The Stock Performance Dashboard project aims to deliver a robust and interactive platform for visualizing and analyzing the performance of Nifty 50 stocks over the past year. By leveraging advanced data processing and visualization tools, this project will provide valuable insights into stock trends, helping investors, analysts, and enthusiasts make informed decisions. The dashboard will utilize daily stock data, including open, close, high, low, and volume values, to generate comprehensive performance metrics and visualizations.

1.1.1 Purpose Of the Document

The purpose of this document is to outline the objectives, scope, and key components of the Stock Performance Dashboard project. It serves as a guide for the development team, stakeholders, and end-users, detailing the project's goals, functionalities, and expected outcomes. This document will ensure a clear understanding of the project's direction and provide a reference for all phases of development, from initial planning to final implementation.

1.1.2 Scope

The scope of the Stock Performance Dashboard project includes the following key components:

1. **Data Collection and Processing:**
    - Gather daily stock data for Nifty 50 stocks, including open, close, high, low, and volume values.
    - Clean and preprocess the data to ensure accuracy and consistency.
2. **Performance Analysis:**
    - Analyze stock performance to identify key metrics such as price changes, average prices, and volatility.
    - Generate insights into the top 10 best-performing (green) and worst-performing (red) stocks over the past year.
3. **Visualization:**
    - Develop interactive dashboards using Streamlit and Power BI to visualize stock performance trends.
    - Provide visual summaries of market performance, including the percentage of green vs. red stocks.
4. **User Interaction:**
    - Enable users to interact with the dashboards to explore stock performance data and insights.
    - Offer tools for investors to identify stocks with consistent growth or significant declines.

5. **Decision Support:**
    - Provide comprehensive insights to support investment decisions for both retail and institutional traders.
    - Include features for analyzing average prices, volatility, and overall stock behavior.

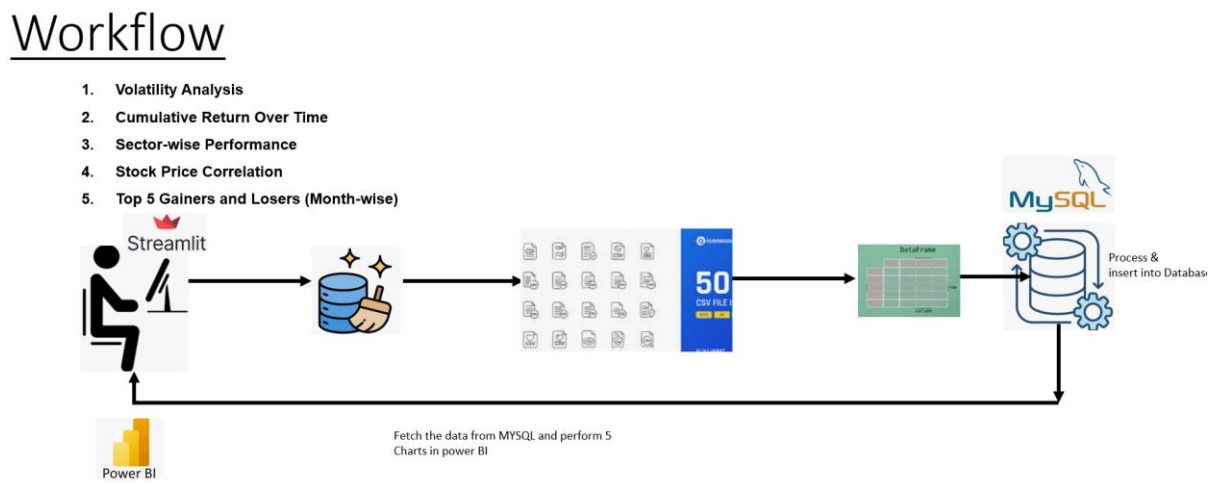
    By defining these components, the project aims to deliver a comprehensive tool that enhances the ability of users to analyze and understand stock performance trends effectively.

1.1.3 Business Use cases

- **Stock Performance Ranking:** Identify the top 10 best-performing stocks (green stocks) and the top 10 worst-performing stocks (red stocks) over the past year.
- **Market Overview**: Provide an overall market summary with average stock performance and insights into the percentage of green vs red stocks.
- **Investment Insights:** Help investors quickly identify which stocks showed consistent growth and which ones had significant declines.
- **Decision Support:** Provide insights on average prices, volatility, and overall stock behaviour, useful for both retail and institutional traders

2 Data-Driven Stock Analysis Architecture Diagram

2.1.1 Overview



2.1.2 Data Extraction:

- Data was provided in YAML format, organized by months.
- Within each month's folder, there are date-wise data entries.
- The task is to extract this data from the YAML file and transform it into a CSV format, organized by symbols
- This resulted in 50 CSV files after the extraction process, one for each symbol or data category

3 Installation

3.1.1 Prerequisites

- Python 3.7 or higher
- MySQL Server
- MySQL Workbench (optional, for database management)
- PowerBI

4 Clone the Repository

https://github.com/niyasatsg/Data-Driven-Stock-Analysis

5 Install Dependencies

5.1.1 requirements.txt file

Create a requirements.txt file with the following content:

pandas==2.2.3

requests==2.32.3

streamlit==1.41.1

plotly==5.24.1

SQLAlchemy==2.0.36

PyMySQL==1.1.1

streamlit-option-menu==0.4.0

5.1.2 install the dependencies:

pip install -r requirements.txt

6 MySQL Database Setup

6.1.1 Create a MySQL Database:

```
CREATE DATABASE Stockanalysis;
```

6.1.2 Create the required Tables:

```
CREATE TABLE `volatilityanalysis` (
  `Ticker` varchar(10) NOT NULL,
  `Volatility` float DEFAULT NULL,
  PRIMARY KEY (`Ticker`)
)
CREATE TABLE `cumulativereturnanalysis` (
  `Ticker` text,
  `Cumulative` double DEFAULT NULL
)
CREATE TABLE `sectorperformance` (
  `Sector` text,
  `Average Yearly Return` double DEFAULT NULL
)
CREATE TABLE `gainers` (
  `id` int NOT NULL AUTO_INCREMENT,
  `month` varchar(7) NOT NULL,
  `ticker` varchar(10) NOT NULL,
  `percentage_return` decimal(10,2) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `month` (`month`,`ticker`)
)
CREATE TABLE `losers` (
  `id` int NOT NULL AUTO_INCREMENT,
  `month` varchar(7) NOT NULL,
  `ticker` varchar(10) NOT NULL,
  `percentage_return` decimal(10,2) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `month` (`month`,`ticker`)
)
```

7 Configuration

7.1.1 Update Database Connection

Update the database connection string in your code:

engine = create_engine('mysql+pymysql://root:yourpassword@localhost:3306/ StockAnalysis)

8 Run the Application

streamlit run app.py

8.1.1 Access the Application

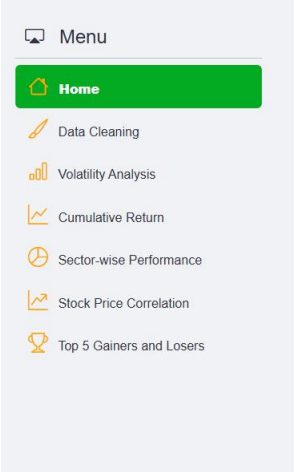
Open your web browser and go to http://localhost:8501.

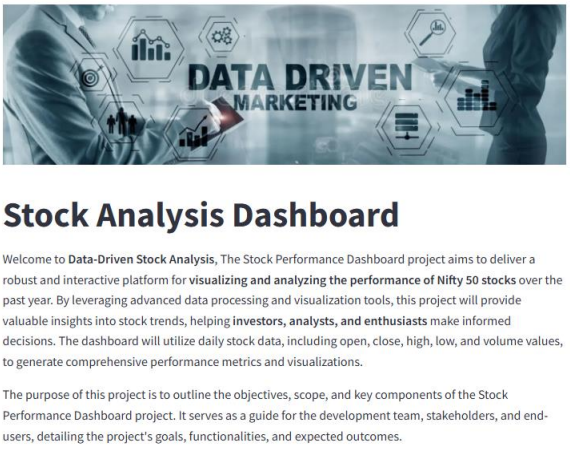
9 Features

9.1.1 Home Page

- Introduction to Data-Driven Stock Analysis.

9.1.2 Home Page Wireframe





9.1.3 Data Cleaning

- **Data Source**: The data is provided in YAML format, organized by months. Each month's folder contains date-wise data entries.
- **Objective**: The main task is to extract data from the YAML files and transform it into CSV format, organized by symbols.
- **Output**: The extraction process will result in 50 CSV files, each corresponding to a specific symbol or data category.

Steps Involved

Data Extraction:

- Read and parse the YAML files.
- Navigate through the monthly folders and date-wise entries to gather the required data.

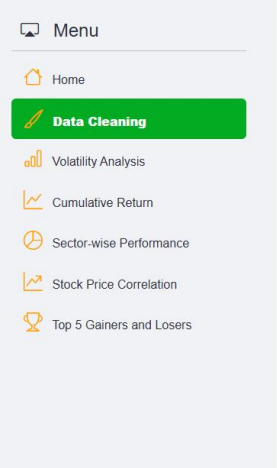
Data Transformation:

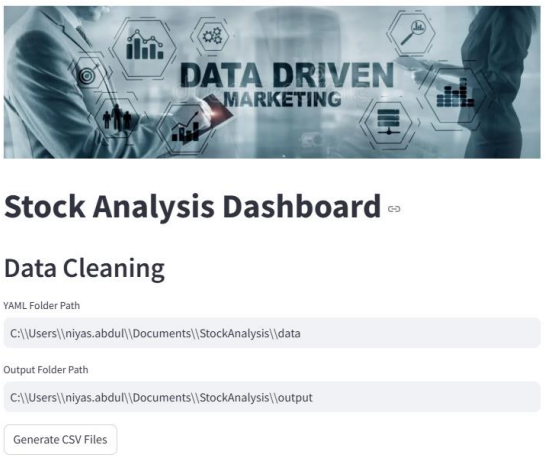
- Convert the extracted data into a structured format suitable for CSV.
- Organize the data by symbols to ensure each CSV file contains data for a specific symbol.

CSV Generation:

- Create 50 CSV files, one for each symbol or data category.
- Ensure the CSV files are properly formatted and contain all relevant data.

9.1.4 Data Cleaning Wireframe





9.1.5 Volatility Analysis

- **Objective**: To visualize the volatility of each stock over the past year by calculating the standard deviation of daily returns.
- **Purpose**: Understanding volatility helps in assessing the risk associated with each stock. Higher volatility indicates greater risk, while lower volatility suggests a more stable stock.

Methodology

1. **Data Collection**:
    - Gather daily stock price data for the past year.
    - Calculate daily returns based on the collected data.
2. **Volatility Calculation**:
    - Compute the standard deviation of daily returns for each stock to measure volatility.
3. **Visualization**:
    - Create a bar chart to display the top 10 most volatile stocks.
    - The x-axis represents the stock tickers, and the y-axis shows the volatility (standard deviation).

Insights

- **Risk Assessment**: The analysis provides valuable insights into the risk levels of different stocks, aiding in investment decisions.

- **Top Volatile Stocks**: Highlighting the most volatile stocks helps investors identify potential high-risk, high-reward opportunities.

Tools and Libraries Used

- **Python**: The primary programming language for the analysis.
- **Pandas**: For data manipulation and calculation of daily returns.
- **Matplotlib/Seaborn**: For creating the bar chart visualization.

Challenges and Solutions

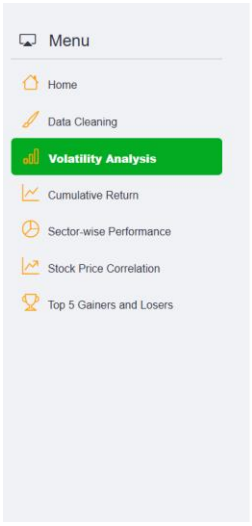
- **Data Accuracy**: Ensuring the accuracy of daily stock prices is crucial for reliable volatility calculations.
- **Visualization Clarity**: Designing clear and informative visualizations to effectively communicate the findings.

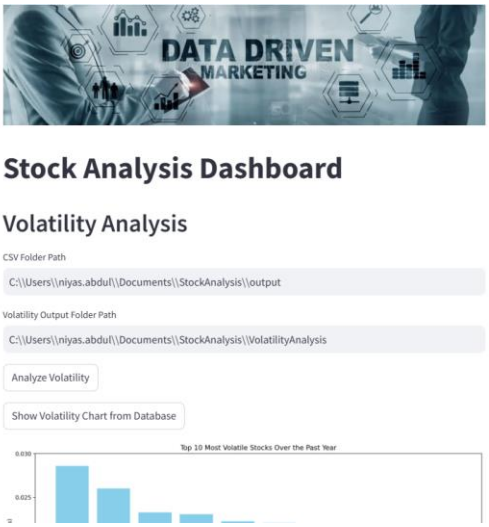
Future Enhancements

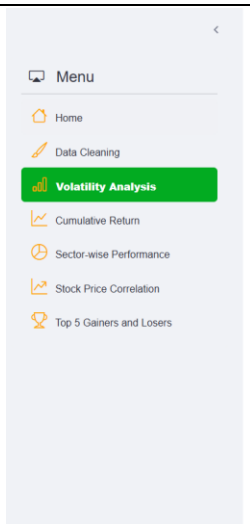
- **Extended Analysis**: Including additional metrics such as beta and Sharpe ratio for a more comprehensive risk assessment.
- **Interactive Dashboards**: Developing interactive dashboards for real-time volatility tracking and analysis.

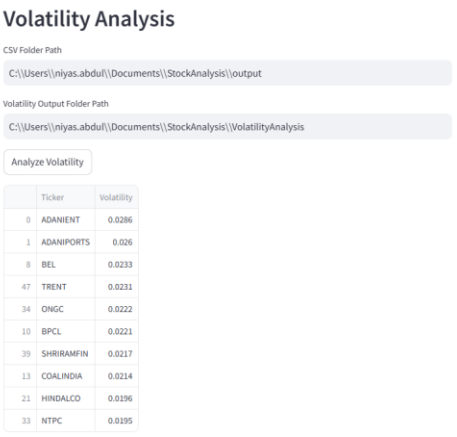
9.1.6 Volatility Analysis Wireframes

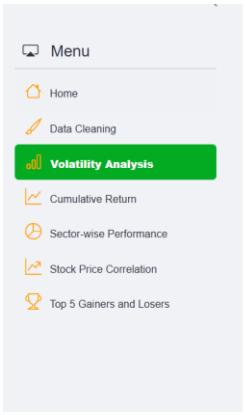
- Wireframe:



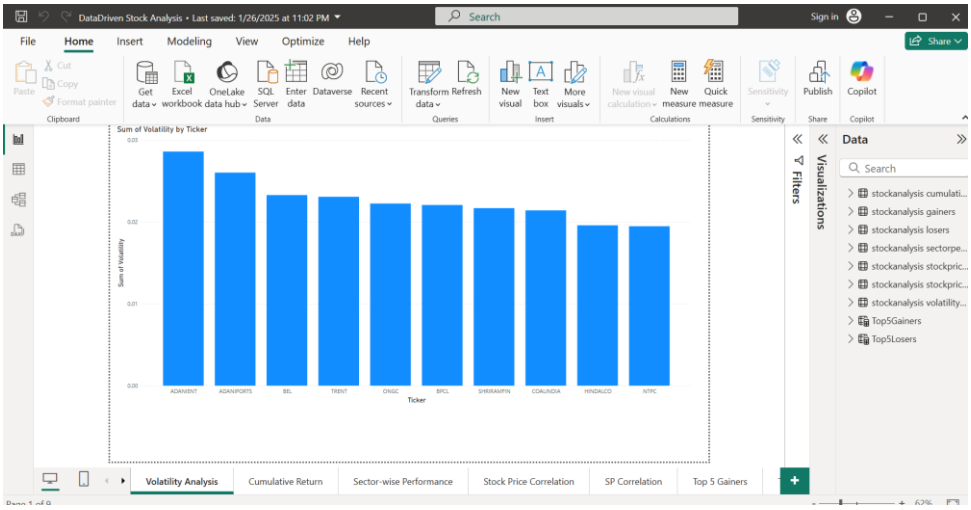








9.1.7 Volatility Analysis in PowerBI



9.1.8 Cumulative Return

- **Objective**: To illustrate the cumulative return of each stock from the beginning to the end of the year.
- **Purpose**: Cumulative return is a key metric for visualizing overall performance and growth over time, enabling users to compare the performance of different stocks.

Methodology

1. **Data Collection**:
    - Obtain daily stock price data from the start to the end of the year.
    - Calculate the cumulative return for each stock based on the daily price changes.
2. **Cumulative Return Calculation**:
    - Compute the cumulative return by aggregating daily returns over the specified period.

Visualization:

- Create a line chart to display the cumulative returns of the top 5 performing stocks.
- The x-axis represents the timeline (days of the year), and the y-axis shows the cumulative return percentage.

Insights

- **Performance Comparison**: The analysis allows users to compare the performance of different stocks over the year.
- **Growth Trends**: Identifying stocks with positive performance trends helps in making informed investment decisions.

Tools and Libraries Used

- **Python**: The primary programming language for the analysis.
- **Pandas**: For data manipulation and calculation of cumulative returns.
- **Matplotlib/Seaborn**: For creating the line chart visualization.

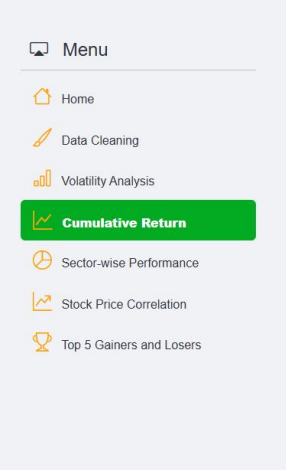
Challenges and Solutions

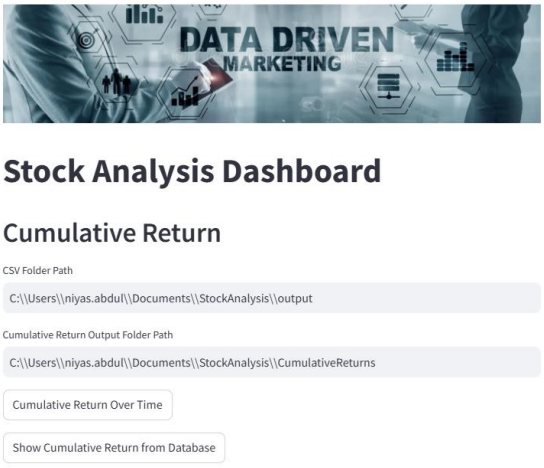
- **Data Consistency**: Ensuring consistent and accurate data collection throughout the year.
- **Visualization Clarity**: Designing clear and informative visualizations to effectively communicate the performance trends.

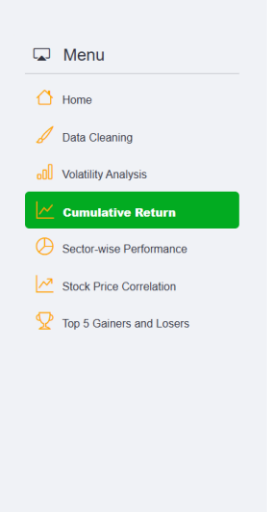
Future Enhancements

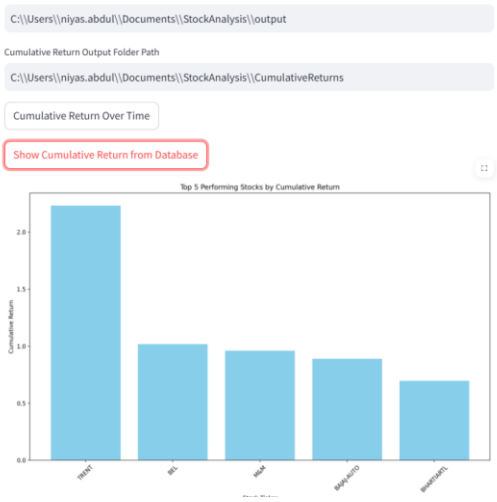
- **Extended Timeframes**: Analyzing cumulative returns over multiple years for long-term performance insights.
- **Interactive Visualizations:** Developing interactive charts for dynamic exploration of stock performance data

9.1.9 Cumulative Return wireframe

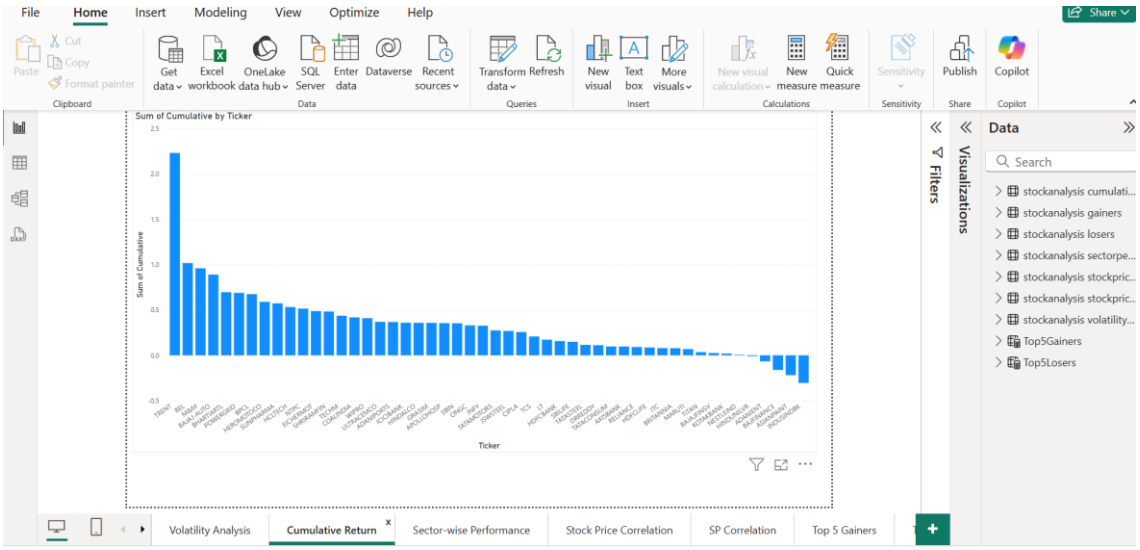








9.1.10 Cumulative Return in PowerBI



9.1.11 Sector-wise Performance

- **Objective**: To provide a detailed breakdown of stock performance by sector, using sector data shared in CSV format.
- **Purpose**: Sector performance analysis helps investors and analysts gauge market sentiment in specific industries, such as IT, Financials, Energy, etc.

Methodology

1. **Data Collection**:
    - Import sector data from the provided CSV file.
    - Aggregate stock performance data by sector.
2. **Performance Calculation**:
    - Calculate the average yearly return for stocks within each sector.
    - Ensure accurate aggregation and calculation to reflect true sector performance.
3. **Visualization**:
    - Create a bar chart to display the average yearly return by sector.
    - The x-axis represents different sectors, and the y-axis shows the average yearly return.

Insights

- **Market Sentiment**: The analysis provides insights into how different sectors are performing, reflecting market sentiment and trends.

- **Investment Decisions**: Identifying high-performing sectors can guide investment strategies and decisions.

Tools and Libraries Used

- **Python**: The primary programming language for the analysis.
- **Pandas**: For data manipulation and calculation of average returns.
- **Matplotlib/Seaborn**: For creating the bar chart visualization.

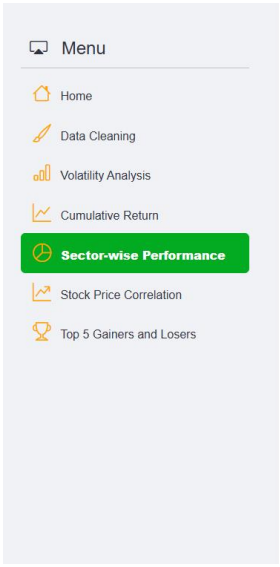
Challenges and Solutions

- **Data Integration**: Ensuring seamless integration of sector data with stock performance data.
- **Visualization Accuracy**: Designing clear and accurate visualizations to effectively communicate sector performance.

Future Enhancements

- **Sector Comparison**: Including additional metrics such as median return and volatility for a more comprehensive sector analysis.
- **Interactive Dashboards**: Developing interactive dashboards to explore sector performance dynamically.

9.1.12 Sector-wise Performance wireframe

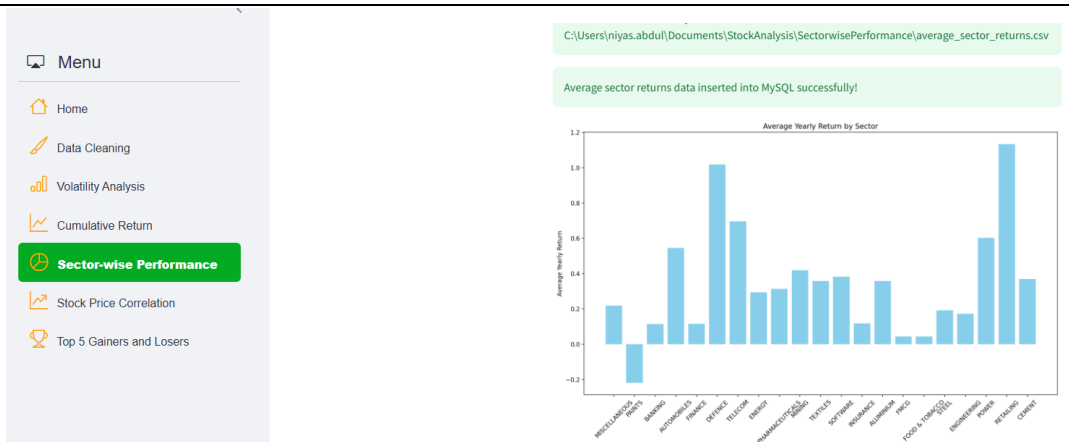


Stock Analysis Dashboard

Sector-wise Performance

Average Yearly Return by Sector:

	Sector	Average Yearly Return
0	MISCELLANEOUS	0.2185
1	PAINTS	-0.2194
2	BANKING	0.1138
3	AUTOMOBILES	0.5453
4	FINANCE	0.1158
5	DEFENCE	1.0176
6	TELECOM	0.696
7	ENERGY	0.2939
8	PHARMACEUTICALS	0.3138
9	MINING	0.4185



9.1.13 Sector-wise Performance in PowerBI

9.1.14 Stock Price Correlation

- **Objective**: To visualize the correlation between the stock prices of different companies.
- **Purpose**: Understanding stock price correlations helps identify if certain stocks tend to move together, which can be indicative of market trends or sector performance.

Methodology

1. **Data Collection**:
    - Gather daily closing prices for the stocks under analysis.
    - Ensure data consistency and accuracy for reliable correlation calculations.
2. **Correlation Calculation**:
    - Compute the correlation coefficients between the closing prices of various stocks.
    - Use statistical methods to determine the strength and direction of the correlations.
3. **Visualization**:

- Create a heatmap to display the correlation between stock prices.
- Darker colors on the heatmap represent higher correlations, making it easy to identify strongly correlated stocks.

Insights

- **Market Trends**: The analysis reveals how stocks move in relation to each other, providing insights into broader market trends.
- **Investment Strategies**: Identifying correlated stocks can help in diversifying portfolios and managing risk.

Tools and Libraries Used

- **Python**: The primary programming language for the analysis.
- **Pandas**: For data manipulation and calculation of correlation coefficients.
- **Seaborn/Matplotlib**: For creating the heatmap visualization.

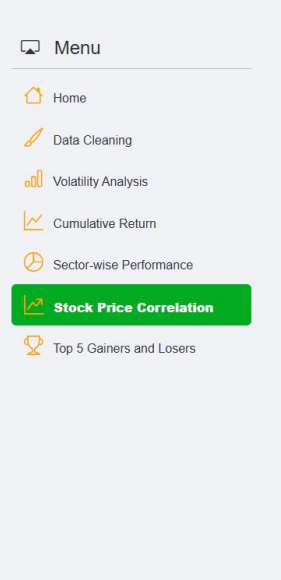
Challenges and Solutions

- **Data Quality**: Ensuring high-quality data is crucial for accurate correlation analysis.
- **Visualization Clarity**: Designing a clear and informative heatmap to effectively communicate the correlation results.

Future Enhancements

- **Dynamic Analysis**: Implementing real-time correlation analysis to track changes over time.
- **Extended Metrics**: Including additional metrics such as rolling correlations to capture dynamic relationships between stocks.

9.1.15 Stock Price Correlation wireframe

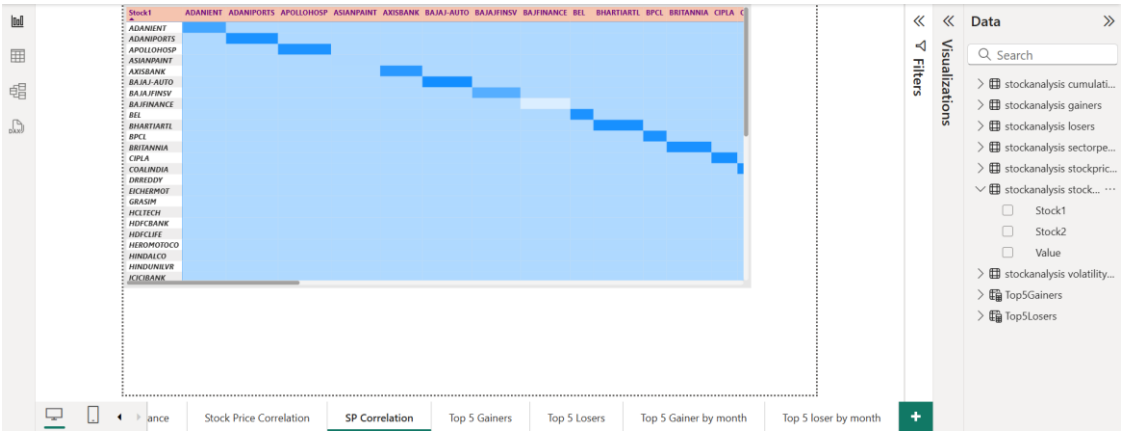


Stock Analysis Dashboard

Stock Price Correlation

	ADANIENT	ADANIPORTS	APOLLOHOSP	ASIANPAINT	AXISBANK	BAJAJ-AUTO	BAJAJFINSV
ADANIENT	1	0.8458	0.5948	-0.1865	0.537	0.6346	0.0801
ADANIPORTS	0.8458	1	0.8151	-0.2465	0.7702	0.8877	0.2151
APOLLOHOSP	0.5948	0.8151	1	-0.1579	0.6074	0.8938	0.5091
ASIANPAINT	-0.1865	-0.2465	-0.1579	1	0.001	-0.1626	0.4634
AXISBANK	0.537	0.7702	0.6074	0.001	1	0.7428	0.3346
BAJAJ-AUTO	0.6346	0.8877	0.8938	-0.1626	0.7428	1	0.5164
BAJAJFINSV	0.0801	0.2151	0.5091	0.4634	0.3346	0.5164	1
BAJFINANCE	-0.4373	-0.4861	-0.4109	0.6094	-0.1552	-0.3229	0.3855
BEL	0.5976	0.9004	0.7521	-0.2463	0.863	0.8727	0.2483
BHARTIARTL	0.4823	0.82	0.8655	-0.178	0.7676	0.951	0.518

9.1.16 Stock Price Correlation in PowerBI



9.1.17 Top 5 Gainers and Losers (Month-wise)

- **Objective**: To provide detailed monthly breakdowns of the top-performing and worst-performing stocks.
- **Purpose**: This analysis helps users observe granular trends and identify which stocks are gaining or losing momentum on a monthly basis.

Methodology

1. **Data Collection**:
    - Gather daily stock price data for each month.
    - Calculate the monthly returns for each stock based on the collected data.
2. **Performance Analysis**:
    - Identify the top 5 gainers and top 5 losers for each month based on percentage return.
    - Rank the stocks to determine the best and worst performers.
3. **Visualization**:
    - Create a set of 12 bar charts, one for each month, showing the top 5 gainers and losers.
    - The x-axis represents the stock tickers, and the y-axis shows the percentage return.

Insights

- **Trend Identification**: The analysis provides insights into monthly performance trends, helping users understand market dynamics.

- **Momentum Tracking**: Identifying stocks with consistent performance patterns aids in making informed investment decisions.

Tools and Libraries Used

- **Python**: The primary programming language for the analysis.
- **Pandas**: For data manipulation and calculation of monthly returns.
- **Matplotlib/Seaborn**: For creating the bar chart visualizations.

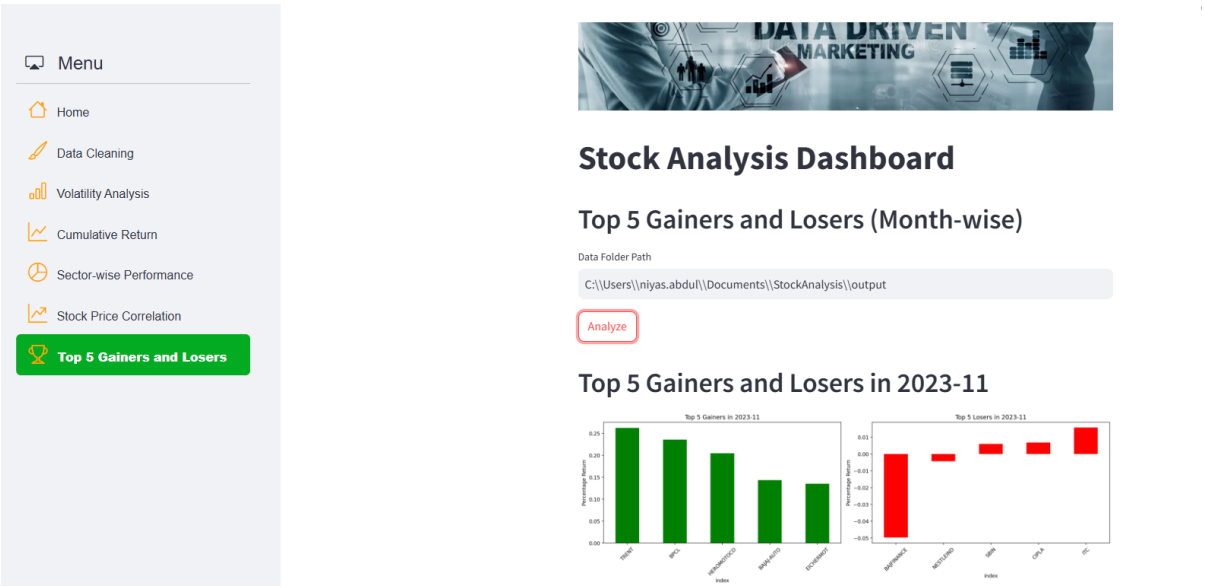
Challenges and Solutions

- **Data Volume**: Handling large volumes of daily stock data requires efficient data processing techniques.
- **Visualization Clarity**: Ensuring that visualizations are clear and informative to effectively communicate monthly performance trends.

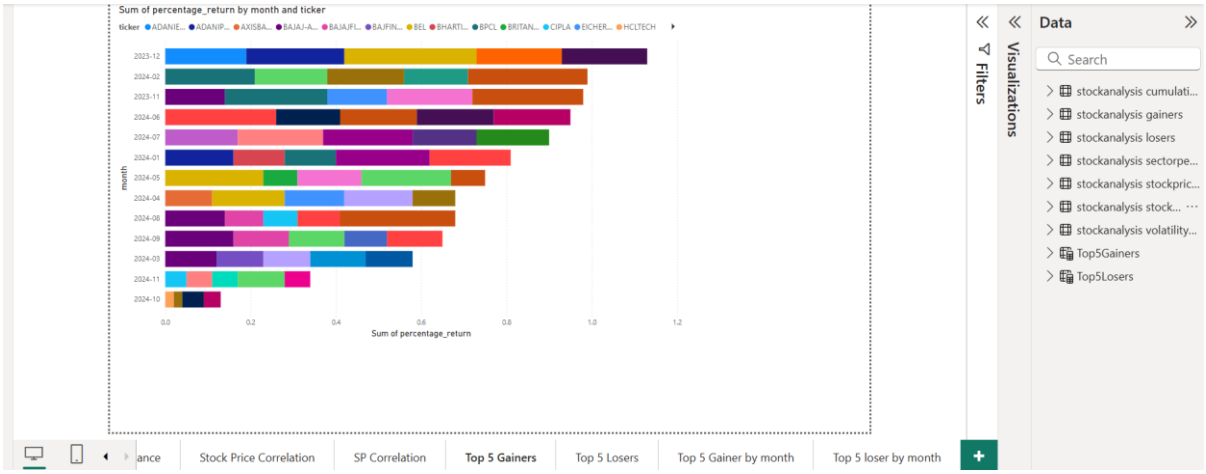
Future Enhancements

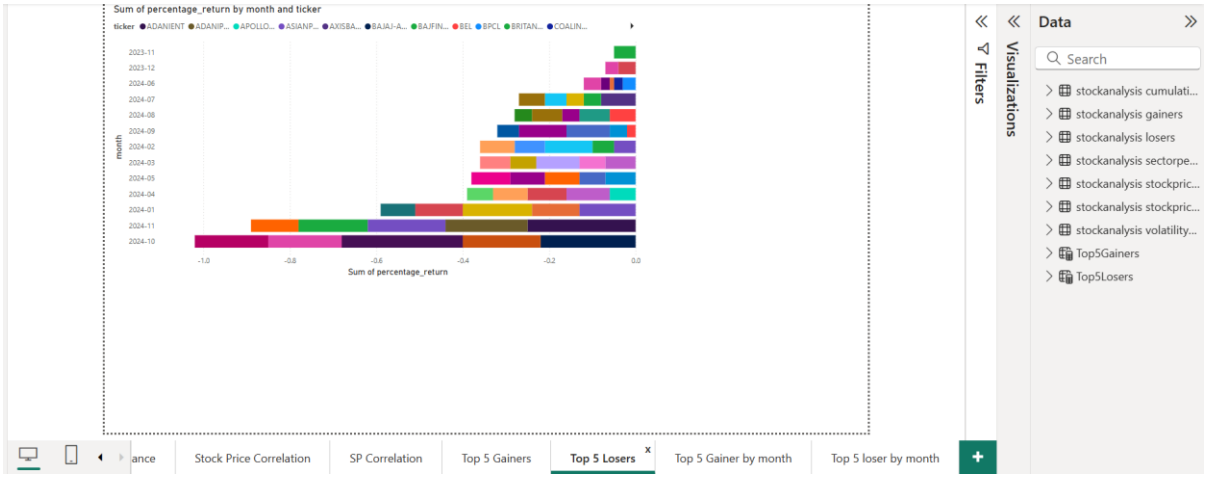
- **Automated Updates**: Implementing automated scripts to update the monthly performance breakdowns with new data.
- **Interactive Dashboards**: Developing interactive dashboards for dynamic exploration of monthly performance data.

9.1.18 Top 5 Gainers and Losers (Month-wise) wireframe



9.1.19 Top 5 Gainers and Losers (Month-wise) PowerBI





10 Insights and Challenges

10.1.1 Insights

- **Comprehensive Visualization**: The dashboard provides a detailed visualization of the Nifty 50 stocks' performance over the past year.
- **Daily Stock Data Analysis**: Analyzes daily stock data, including open, close, high, low, and volume values.
- **Data Cleaning and Processing**: Ensures the data is clean and processed for accurate analysis and visualization.
- **Key Performance Insights**: Generates key insights into stock performance, highlighting significant trends and patterns.
- **Top-Performing Stocks**: Visualizes the top-performing stocks based on price changes, helping users identify high-growth opportunities.
- **Average Stock Metrics**: Provides average metrics for stocks, offering a broader view of market performance.
- **Interactive Dashboards**: Utilizes Streamlit and Power BI to create interactive dashboards, enhancing user engagement and decision-making.

10.1.2 Challenges

- ***Connection error:*** I experienced Connection error issues when connecting to my database. “Connection error Is Streamlit still running?”. This issue was resolved after installing sqlalchemy
- ***Integration with Tools:*** Seamlessly integrating Streamlit and Power BI to provide a cohesive user experience.
- ***Error Handling:*** Comprehensive error handling was necessary to manage different types of exceptions, including IntegrityError and DataError.

11 Troubleshooting

11.1.1 Common Issues

- ***Database Connection Error**:* Ensure your MySQL server is running and the connection string is correct