



ALTERNATIVE DATA INTEGRATION AND CORRELATION ANALYSIS IN STOCK MARKET BEHAVIOR

Group A



Group Members:

- 16242 – Chathuni Amasha
- 16252 – Nethmi Sansala
- 16210 – Hiruni Hasara
- 16220 – Meedum Keerthisiri

Table of Contents

1. Introduction.....	2
2. Objectives	3
3. System Design and Architecture	3
4. Data Ingestion Layer.....	4
5. Automation Layer	5
6. Data Storage Layer	5
7. Data Integration Layer	6
8. Analysis & Modeling Layer.....	7
8.1 Predictive Modeling and Forecasting.....	7
8.2 Correlation Analysis.....	8
8.3 Causation Analysis (Granger Causality)	9
9. Visualization and deployment Layer.....	10
9.1 Streamlit App.....	10
9.2 Dockerization	11
9.3 Cloud Deployment	12
10. Challenges Faced	14
11. Conclusion	14
12. References.....	15
13. Appendix	15

Table of Figures

Figure 1:Project workflow diagram	4
Figure 2:Data ingestion functions.....	4
Figure 3:Cloud Scheduler configuration functions	5
Figure 4:Firestore collections.....	5
Figure 5:preprocess_data() function.....	6
Figure 6:Cleaned CSV files in GCS	6
Figure 7:integrate_data() function	6
Figure 8:Integration CSV file in GCS	7
Figure 9:run_analysis() function	7
Figure 10:Analysis csv files in GCS.....	8
Figure 11:Example csv file of price forecasting results	
Figure 12:Example csv file of granger causality results	9
Figure 13:Example csv file of correlation matrix	10

Figure 14:creating docker image	11
Figure 15:Streamlit dashboard view 01.....	13
Figure 16:Streamlit dashboard view 02.....	13
Figure 17:Streamlit dashboard view 03.....	13
Figure 18:Streamlit dashboard view 04.....	14

1. Introduction

The stock market is a complex and dynamic system influenced by a broad range of factors, including company fundamentals (earnings, revenue, balance sheets), macroeconomic indicators (interest rates, inflation, unemployment), global events (geopolitical tensions, pandemics, policy changes), and public sentiment (investor confidence, social media trends). Traditional financial datasets, such as stock price history, trading volumes, and company reports provide a valuable foundation for analysis. However, they often fail to capture the behavioral and external elements that drive market volatility in real time.

To address these limitations, researchers and practitioners increasingly rely on alternative data sources. These include online search trends (e.g., Google Trends), macroeconomic databases (e.g., FRED), real-time financial APIs (e.g., Finnhub, Yahoo Finance), and even social media sentiment. Alternative data enables analysts to uncover hidden correlations between investor interest, economic signals, and stock price fluctuations, offering a more holistic view of market behavior.

In this project, we focus on integrating both traditional and alternative datasets to create a unified pipeline for analysis. The goal is not only to explore correlations but also to leverage predictive modeling technique using SARIMAX to forecast stock movements. By combining structured financial data with behavioral and economic indicators, we aim to bridge the gap between traditional financial analysis and modern data-driven approaches.

A significant aspect of our work lies in the automation and scalability of the data pipeline. Using Google Cloud services (Cloud Functions, Firestore, Cloud Storage, and Cloud Scheduler), we established a fully automated system that ingests, preprocesses, and stores data in near real-time. This ensures that the dataset is consistently updated without manual intervention, addressing the challenge of time-sensitive financial data.

The final deliverable of our project is an interactive Streamlit dashboard. The dashboard serves as a comprehensive tool where users can:

- Visualize stock trends over time,
- Explore correlation heatmaps between multiple factors, and
- View model predictions of future stock movements.

To ensure accessibility and scalability, the dashboard is containerized using Docker and deployed on Google Cloud Run, making it available as a cloud-hosted application. This deployment choice allows the system to scale automatically with demand, ensures high availability, and supports continuous integration of new features and data sources.

Overall, our project demonstrates the value of integrating alternative data into stock market research and the importance of building cloud-native, automated solutions for financial analytics. It provides both academic insights into correlation patterns and practical tools for decision-makers who seek to understand and predict stock market dynamics in a rapidly changing world.

2. Objectives

The main objectives of the project are:

- To integrate stock market data from multiple APIs (Finnhub, Yahoo Finance, Google Trends, FRED).
- To automate real-time and periodic data ingestion using Google Cloud services.
- To preprocess and clean datasets for correlation, causation analysis, and modeling.
- To conduct correlation and causation analysis between alternative data and stock performance.
- To train predictive models for forecasting stock behavior.
- To develop a Streamlit dashboard for visualization.
- To containerize and deploy the dashboard using Docker and Google Cloud Run.

3. System Design and Architecture

The project was designed in a modular and layered architecture, ensuring scalability, flexibility, and fault tolerance. The system follows a data pipeline architecture that transforms raw stock market data into meaningful insights, visualizations, and predictions.

1. Data Ingestion Layer
2. Automation Layer
3. Data Storage Layer
4. Data Integration Layer
5. Analysis & Modeling Layer

6. Visualization and deployment Layer

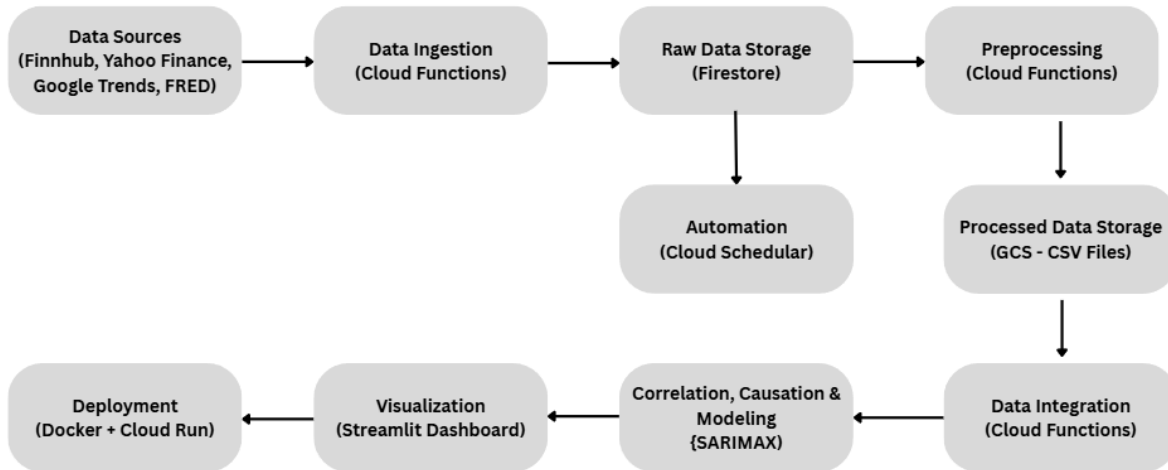


Figure 1: Project workflow diagram

4. Data Ingestion Layer

The first step involves fetching raw data from multiple APIs:

- Finnhub API – Real-time stock quotes, company fundamentals, and financial indicators.
- Yahoo Finance (yfinance) – Historical stock prices and company-level market data.
- Google Trends (pytrends) – Public search interest data related to company names and stock symbols.
- FRED API – Macroeconomic indicators such as CPI, GDP growth, and unemployment rate.

Each data source is accessed using a Google Cloud Function, written in Python, that handles API authentication, request handling, and data parsing. This modular approach allows us to add or remove APIs without disrupting the overall pipeline.

<input type="checkbox"/>	<input checked="" type="checkbox"/>	Name ↑	Deployment type	Req/sec ?	Region	Authentication ?	Ingress ?	Last deployed	Deployed by
<input type="checkbox"/>	<input checked="" type="checkbox"/>	fetch-finnhub-data	(-.) Function	0	us-central1	Public access	All	2 Aug 2025	Cloud Run functions
<input type="checkbox"/>	<input checked="" type="checkbox"/>	fetch-fred-data	(-.) Function	0	us-central1	Public access	All	3 Aug 2025	Cloud Run functions
<input type="checkbox"/>	<input checked="" type="checkbox"/>	fetch-google-trends	(-.) Function	0	us-central1	Public access	All	3 Aug 2025	Cloud Run functions
<input type="checkbox"/>	<input checked="" type="checkbox"/>	fetch-yfinance-data	(-.) Function	0	us-central1	Public access	All	3 Aug 2025	Cloud Run functions

Figure 2: Data ingestion functions

5. Automation Layer

Since financial data changes frequently, it is important to have up-to-date data. Therefore, to ensure that the datasets remain current, the above data ingestion functions have been scheduled using Google Cloud Scheduler. These functions are triggered automatically on an hourly basis, eliminating the need for manual intervention. This approach guarantees that financial data is ingested regularly, keeping them fresh and up to date.

Name ↑	Status of last execution	Region	State	Description	Frequency	Target	Last run	Next run	Last updated	Actions
analysis	✔ Success	us-central1	Enabled		0 **** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/run_analysis	5 Sept 2025, 18:00:00	5 Sept 2025, 19:00:00	4 Sept 2025, 14:56:50	⋮
fetch-finnhub-schedule	✔ Success	us-central1	Enabled		0 **** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/fetch_finnhub_data	5 Sept 2025, 19:00:00	5 Sept 2025, 20:00:00	3 Aug 2025, 03:01:49	⋮
fetch-fred-daily	✔ Success	us-central1	Enabled		0 3 *** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/fetch_fred_data	5 Sept 2025, 03:00:00	6 Sept 2025, 03:00:00	3 Aug 2025, 03:25:52	⋮
fetch-google-trends-hourly	✔ Success	us-central1	Enabled		0 **** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/fetch_google_trends	5 Sept 2025, 19:00:00	5 Sept 2025, 20:00:00	3 Aug 2025, 11:06:24	⋮
fetch-yahoo-hourly	✔ Success	us-central1	Enabled		0 **** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/fetch_yfinance_data	5 Sept 2025, 19:00:00	5 Sept 2025, 20:00:00	3 Aug 2025, 11:33:09	⋮
integrate_hourly	✔ Success	us-central1	Enabled		0 **** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/integrate_data	5 Sept 2025, 19:00:00	5 Sept 2025, 20:00:00	1 Sept 2025, 15:23:51	⋮
preprocess_hourly	✔ Success	us-central1	Enabled		0 **** (Asia/Colombo)	URL : https://us-central1-stock-data-analysis-467504.cloudfunctions.net/preprocess_data	5 Sept 2025, 19:00:00	5 Sept 2025, 20:00:00	8 Aug 2025, 09:36:51	⋮

Figure 3:Cloud Scheduler configuration functions

6. Data Storage Layer

We used a two-tier storage strategy:

1. **Google Firestore** – Stores raw ingested data from each API in separate collections as stock_quotes, yfinance_data, fred_data, google_trends. Data from all four companies (Apple, Tesla, Amazon, Microsoft) is continuously stored in these collections as hourly updates.

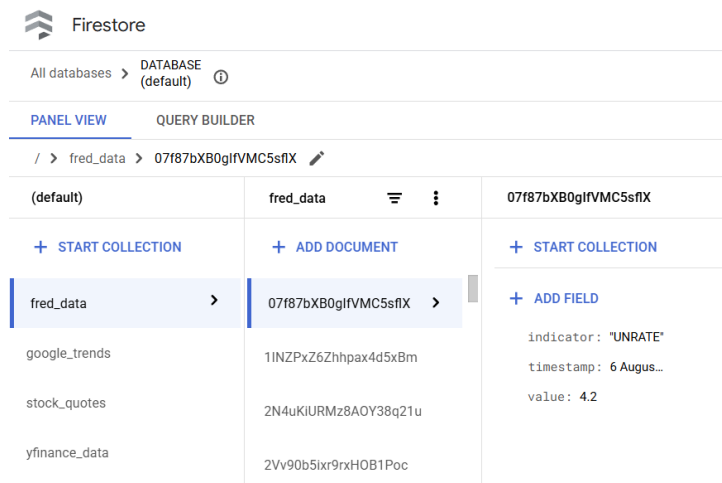


Figure 4:Firestore collections

After that, the `preprocess_data()` Cloud Function was implemented to clean the datasets. The preprocessing at this stage involved: removing null and duplicate entries, aligning timestamps across APIs for the ease of integration, and converting Firestore dictionaries into Pandas Data Frames. This function was scheduled using the cloud scheduler so that this will preprocess data each hour.

<input type="checkbox"/>	<input checked="" type="checkbox"/>	preprocess-data	(...) Function	0	us-central1	Public access	All	8 Aug 2025
--------------------------	-------------------------------------	---------------------------------	----------------	---	-------------	---------------	-----	------------

Figure 5: `preprocess_data()` function

2. **Google Cloud Storage (GCS)** – Stores above cleaned and preprocessed data in the form of CSV files for efficient retrieval and analysis.

Name	Size	Type	Created ?	Storage class	Last modified	Public access ?	Version history ?
cleaned_fred.csv	4.8 KB	text/csv	5 Sept 2025, 19:00:11	Standard	5 Sept 2025, 19:00:11	Not public	—
cleaned_stock.csv	318.3 KB	text/csv	5 Sept 2025, 19:00:11	Standard	5 Sept 2025, 19:00:11	Not public	—
cleaned_trends.csv	10.4 KB	text/csv	5 Sept 2025, 19:00:11	Standard	5 Sept 2025, 19:00:11	Not public	—
cleaned_yahoo.csv	404.3 KB	text/csv	5 Sept 2025, 19:00:11	Standard	5 Sept 2025, 19:00:11	Not public	—

Figure 6: Cleaned CSV files in GCS

This approach allows us to preserve both the original raw data and the cleaned versions.

7. Data Integration Layer

After preprocessing, a data integration step was carried out to merge the datasets from all four sources (Finnhub, Yahoo Finance, Google Trends, and FRED). The data was aligned based on a common timestamp to ensure synchronization across sources, especially since new data is ingested on an hourly basis.

<input checked="" type="checkbox"/>	integrate-data	(...) Function	0	us-central1	Public access	All	4 days ago	Cloud Run functions
-------------------------------------	--------------------------------	----------------	---	-------------	---------------	-----	------------	---------------------

Figure 7: `integrate_data()` function

For storage and analysis, two levels of integrated outputs were generated:

1. **Company-specific integrated files:** Each company (Apple, Tesla, Amazon, Microsoft) has its own integrated file containing all relevant features for that particular company.
2. **Combined integrated file:** A single file named `integrated_all` contains the complete dataset across all four companies, with company-level records organized per timestamp.

To ensure that newly ingested data is consistently incorporated, the `integrate_data()` function was also scheduled to run on an hourly basis using Google Cloud Scheduler. This automation guarantees that each batch of incoming data is integrated promptly, keeping both the company-

specific files and the combined *integrated_all* dataset continuously updated without requiring manual intervention.






<input type="checkbox"/>	Name	Size	Type	Created ②	Storage class	Last modified	Public access ②	Version history ②
<input type="checkbox"/>	 AAPL.csv	184.6 KB	text/csv	5 Sept 2025, 19:00:06	Standard	5 Sept 2025, 19:00:06	Not public	—
<input type="checkbox"/>	 AMZN.csv	183.4 KB	text/csv	5 Sept 2025, 19:00:06	Standard	5 Sept 2025, 19:00:06	Not public	—
<input type="checkbox"/>	 MSFT.csv	180.4 KB	text/csv	5 Sept 2025, 19:00:06	Standard	5 Sept 2025, 19:00:06	Not public	—
<input type="checkbox"/>	 TSLA.csv	183.1 KB	text/csv	5 Sept 2025, 19:00:07	Standard	5 Sept 2025, 19:00:07	Not public	—
<input type="checkbox"/>	 integrated_all.csv	731.1 KB	text/csv	5 Sept 2025, 19:00:07	Standard	5 Sept 2025, 19:00:07	Not public	—

Figure 8: Integration CSV file in GCS

8. Analysis & Modeling Layer

After integration and company-wise separation of datasets, the next layer of our workflow focused on advanced analysis and predictive modeling. This stage was designed to extract meaningful insights from the integrated data, generate forecasts, and identify explanatory factors influencing stock market behavior.

To operationalize this process, we developed an automated function named *run_analysis()*, which is scheduled to run hourly using Google Cloud Scheduler. This ensures that analytical outputs remain continuously updated as new data is ingested and integrated.


	run-analysis	(~) Function	0	us-central1	Public access	All	1 day ago	Cloud Run functions
---	------------------------------	--------------	---	-------------	---------------	-----	-----------	---------------------

Figure 9: *run_analysis()* function

For each company, this function generates three key outputs:

1. **Forecast File:** Contains 1-week ahead forecasts (7 days) generated using the SARIMAX model, which captures autoregressive and seasonal patterns in stock prices.
2. **Correlation File:** Stores Pearson correlation matrices quantifying linear relationships between stock prices, Google Trends scores, macroeconomic indicators, and company fundamentals.
3. **Causation File:** Stores results from the Granger Causality test, highlighting causal dependencies between stock prices and external features like search trends, and macroeconomic factors.

These outputs are saved separately for each company (Apple, Tesla, Amazon, Microsoft) and are refreshed hourly. This automation ensures that both the predictive models and explanatory analyses adapt dynamically to evolving market conditions.

















Name	Size	Type	Created [?]	Storage class	Last modified	Public access [?]	Version history [?]
 AAPL_causality.csv	1.6 KB	text/csv	5 Sept 2025, 19:00:20	Standard	5 Sept 2025, 19:00:20	Not public	—
 AAPL_correlation.csv	5.6 KB	text/csv	5 Sept 2025, 19:00:19	Standard	5 Sept 2025, 19:00:19	Not public	—
 AAPL_forecast.csv	5.3 KB	text/csv	5 Sept 2025, 19:00:21	Standard	5 Sept 2025, 19:00:21	Not public	—
 AAPL_sarimax.pkl	650.4 KB	application/octet-stream	5 Sept 2025, 19:00:21	Standard	5 Sept 2025, 19:00:21	Not public	—
 AMZN_causality.csv	1.6 KB	text/csv	5 Sept 2025, 19:00:14	Standard	5 Sept 2025, 19:00:14	Not public	—
 AMZN_correlation.csv	5 KB	text/csv	5 Sept 2025, 19:00:13	Standard	5 Sept 2025, 19:00:13	Not public	—
 AMZN_forecast.csv	5.4 KB	text/csv	5 Sept 2025, 19:00:15	Standard	5 Sept 2025, 19:00:15	Not public	—
 AMZN_sarimax.pkl	753.2 KB	application/octet-stream	5 Sept 2025, 19:00:15	Standard	5 Sept 2025, 19:00:15	Not public	—
 MSFT_causality.csv	1.6 KB	text/csv	5 Sept 2025, 19:00:18	Standard	5 Sept 2025, 19:00:18	Not public	—
 MSFT_correlation.csv	5.7 KB	text/csv	5 Sept 2025, 19:00:17	Standard	5 Sept 2025, 19:00:17	Not public	—
 MSFT_forecast.csv	5.4 KB	text/csv	5 Sept 2025, 19:00:19	Standard	5 Sept 2025, 19:00:19	Not public	—
 MSFT_sarimax.pkl	727.7 KB	application/octet-stream	5 Sept 2025, 19:00:19	Standard	5 Sept 2025, 19:00:19	Not public	—
 TSLA_causality.csv	1.6 KB	text/csv	5 Sept 2025, 19:00:16	Standard	5 Sept 2025, 19:00:16	Not public	—
 TSLA_correlation.csv	5 KB	text/csv	5 Sept 2025, 19:00:15	Standard	5 Sept 2025, 19:00:15	Not public	—
 TSLA_forecast.csv	5.4 KB	text/csv	5 Sept 2025, 19:00:17	Standard	5 Sept 2025, 19:00:17	Not public	—
 TSLA_sarimax.pkl	666.5 KB	application/octet-stream	5 Sept 2025, 19:00:17	Standard	5 Sept 2025, 19:00:17	Not public	—

Figure 10: Analysis csv files in GCS

8.1 Predictive Modeling and Forecasting

We implemented a forecasting model to predict stock prices for each of the four companies (Apple, Tesla, Amazon, Microsoft) using their integrated datasets:

1. SARIMAX – A time-series model suitable for capturing autoregressive and seasonal components in stock prices.

For each company, we generated 1-week ahead forecasts (next 7 trading days) based on historical data and alternative features.

8.2 Correlation Analysis

To understand the relationships between stock prices and influencing factors, we performed correlation analysis on each company's dataset.

1. Pearson correlation matrices were used to measure linear relationships between stock closing prices, Google Trends scores, macroeconomic indicators, and company fundamentals.
2. Heatmaps were generated for visualization, helping us identify which features were strongly associated with stock movement.

For example:

1. Tesla's stock showed higher correlation with Google Trends search activity.
2. Microsoft's stock was more correlated with macroeconomic factors like interest rates.

8.3 Causation Analysis (Granger Causality)

While correlation helps identify associations, it does not prove causation. To address this, we applied Granger Causality tests on each company's dataset.

1. Granger causality examines whether past values of one time series (e.g., Google Trends interest in "Tesla") help predict the future values of another (e.g., Tesla's stock price).
2. We tested multiple pairs of features (search trends, macroeconomic indicators, stock prices) to identify causal drivers of market movements.

Examples:

1. Google Trends data Granger-caused Tesla's stock price, indicating that search interest was a leading indicator.
2. Macroeconomic variables like interest rates Granger-caused Microsoft's stock prices, showing dependency on broader economic conditions.
3. Apple's price was more self-driven, with stronger autocorrelation compared to causation by external features.

Below are some sample output files of the analysis.

step	forecast_c
1	232.73857
2	232.82596
3	232.82462
4	232.82915
5	232.83858
6	232.84155
7	232.82062
8	232.82864
9	232.82326
10	232.82445
11	232.82222
12	232.82452
13	232.82387
14	232.82448
15	232.82452
16	232.82452
17	232.82211
18	232.82315
19	232.81482
20	232.95605
21	233.02806
22	232.7955
23	232.68634
24	232.33192
25	232.32718

Figure 11:Example csv file of price forecasting results

predictor	lag	p_value
pc	1	0.23166
pc	2	0.02493
pc	3	0.01473
pc	4	0.0064
pc	5	0.0135
l	1	0.85927
l	2	0.33538
l	3	0.26706
l	4	0.07305
l	5	0.10928
h	1	0.00319
h	2	2e-05
h	3	2e-05
h	4	4e-05
h	5	1e-05
d	1	0.23166
d	2	0.02493
d	3	0.01473
d	4	0.0064
d	5	0.0135
o	1	0.16729
o	2	0.02088
o	3	0.01266
o	4	0.0045

Figure 12:Example csv file of granger causality results

c	pc	l	h	d	o	t	dp	High	Volume	Stock Splits	Open	Close	Low	Dividends	trend_score	CPI	AUCSL	GDP	UNRATE
1	0.52721088	0.90705486	0.96351135	0.46875399	0.84132271	0.69863567	0.46428565	0.96076473	-0.436753		0.84895626	0.99816251	0.90597439	0.77527169	-0.285117	0.77250428	0.29818616	0.391697367	
0.52721088	1	0.67106676	0.65539779	-0.503463	0.76286909	0.45615370	-0.507483	0.64889319	-0.016329		0.75261147	0.52569595	0.66284037	0.53393063	-0.030776	0.51301642	0.37286956	0.709991903	
0.90705486	0.67106676	1	0.92581630	0.22470339	0.95840118	0.66538054	0.21324014	0.92229593	-0.489297		0.95720329	0.90549923	0.99273015	0.1313218	-0.245946	0.70209021	0.36993429	0.1368755	
0.96351135	0.65539779	0.92581630	1	0.29839725	0.91208506	0.65902116	0.29312791	0.99466476	-0.32717		0.91705782	0.96147784	0.92285383	0.3070781	-0.236535	0.76443472	0.34058470	0.80604549	
0.46875399	-0.503463	0.22470339	0.29839725	1	0.06243466	0.23619071	0.99963564	0.30236623	-0.427119		0.08085968	0.46846045	0.23215639	0.108876074	-0.25252	0.23674566	-0.095685		
0.84132271	0.76286909	0.95840118	0.91208506	0.06243466	1	0.64767640	0.04994658	0.90493786	-0.414374		0.98984248	0.83904152	0.94823418	0.64028542	-0.187179	0.71494599	0.39550958	0.7432192	
0.69863567	0.45615370	0.66538054	0.65902116	0.23619071	0.64767640	1	0.22909323	0.65437298	-0.460932		0.64028052	0.69849101	0.66034826	0.2706456	-0.215652	0.79532978	0.69966010	0.13572854	
0.46428565	-0.507483	0.21324014	0.29312791	0.99963564	0.04994658	0.22909323	1	0.29719815	-0.413391		0.06884068	0.46399072	0.22093614	0.684884538	-0.25004	0.23054389	-0.099574		
0.96076473	0.64889319	0.92229593	0.99466476	0.30236623	0.90493786	0.65437298	0.29719815	1	-0.306082		0.91752546	0.96219560	0.91966477	0.59248058	-0.243706	0.75849344	0.33706305	0.1195112	
-0.436753	-0.016329	-0.489297	-0.32717	-0.427119	-0.414374	-0.460932	-0.413391	-0.306082	1		-0.409654	-0.438373	-0.510349		0.22770876	-0.399863	-0.187337		
0.84895626	0.75261147	0.95720329	0.91705782	0.08085968	0.98984248	0.64028052	0.06884068	0.91752546	-0.409654		1	0.85048936	0.95870882	0.7855889	-0.198967	0.71603724	0.38957467	0.60643758	
0.99816251	0.52569595	0.90549923	0.96147784	0.46846045	0.83904152	0.69849101	0.46399072	0.96219560	-0.438373		0.85048936	1	0.90780055	0.14067149	-0.28437	0.77286366	0.29812730	0.68142732	
0.90597439	0.66284037	0.99273015	0.92285383	0.23215639	0.94823418	0.66034826	0.22093614	0.91966477	-0.510349		0.95870882	0.90780055	1		-0.2434	0.70336504	0.36513705	0.73357625	
-0.285117	-0.030776	-0.245946	-0.236535	-0.25252	-0.187179	-0.215652	-0.25004	-0.243706	0.22770876	0.1951629	-0.198967	-0.28437	-0.2434		1	-0.187904	-0.169814		
0.77250428	0.51301642	0.70209021	0.76443472	0.23674566	0.71494599	0.79532978	0.23054389	0.75849344	-0.399863		0.71603724	0.77286366	0.70336504	0.3978583	-0.187904	1	0.34089740	0.594207407	
0.29818616	0.37286956	0.36993429	0.34058470	-0.095685	0.39550958	0.69966010	-0.099574	0.33706305	-0.187337		0.38957467	0.29812730	0.36513705	0.73357625	-0.169814	0.34089740	1		

Figure 13: Example csv file of correlation matrix

9. Visualization and deployment Layer

The final stage of our project was to design and deploy an interactive dashboard that allows users to explore the insights derived from the data pipeline and analysis. This component was critical in making the results both accessible and interpretable, enabling visualization of trends, correlations, and predictive outcomes.

9.1 Streamlit App

We developed a Streamlit-based dashboard because of its simplicity, interactivity, and seamless integration with Python libraries such as Pandas, Matplotlib, and Plotly. The dashboard was designed with two main pages, each providing different levels of insights.

Stock Market Overview Page

This page provides a high-level market snapshot across all four companies (Apple, Tesla, Amazon, Microsoft) along with key economic indicators. The following components are included:

1. **Current Market Snapshot:** Displays the real-time (hourly) stock prices of the four companies, along with GDP and unemployment rate values.
2. **Historical Price Trends:** A line chart visualizing the price movements of all four companies for quick comparison.
3. **Market Volume Visualization:** A horizontal bar chart comparing the trading volume of the four companies.

This overview enables users to monitor the overall stock market status.

Company Overview Page

This page provides a detailed view of a user-selected company, offering both descriptive and analytical insights:

1. **Company Profile:** A short description of the selected company.

2. **Price Summary:** Displays previous close price, current price, percentage change between them, highest and lowest price within the latest hour, and trading volume.
3. **Price Trend Visualization:** An interactive area chart showing the stock price movement for the selected company.
4. **Return Distribution:** A KDE (Kernel Density Estimation) plot representing the return distribution, helping assess the stock's stability and volatility.
5. **Correlation Heatmap:** Highlights relationships between the company's stock price, search trends, macroeconomic indicators, and fundamentals.
6. **Causal Graph:** Visualizes causality results (from Granger Causality analysis), showing which features have predictive power over stock prices.
7. **Forecasting:** Displays the 7-day ahead forecast for the selected company generated by the SARIMAX model.

The dashboard was updated automatically on an hourly basis, in line with the automated data pipeline. This ensured that users always accessed the latest analysis results.

9.2 Dockerization

To make the dashboard portable and deployment-ready, we containerized the Streamlit app using Docker.

Key steps in Dockerization:

1. Dockerfile was created to define the environment, dependencies, and execution instructions.
2. The requirements.txt file contained Python dependencies such as streamlit, pandas, plotly, matplotlib, scikit-learn and statsmodels.
3. The Docker image was built and tested locally to confirm that the app runs correctly.

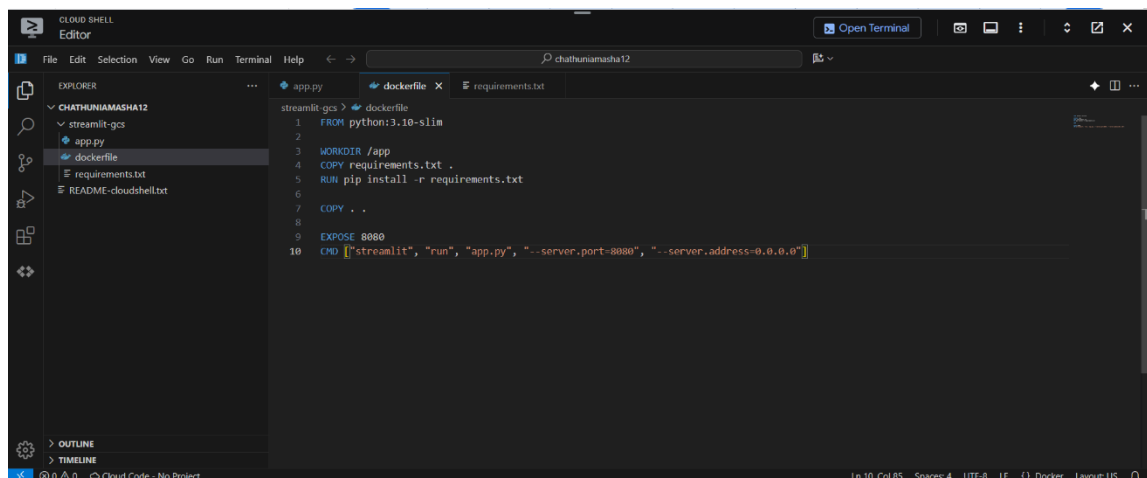


Figure 14:creating docker image

This ensured that the environment was lightweight, reproducible, and consistent, regardless of where the container was run.

9.3 Cloud Deployment

After successful Dockerization, the containerized dashboard was deployed to the cloud for public access:

1. Push to Container Registry (GCR):

1. The Docker image was tagged and pushed to Google Container Registry (GCR).
1. This allowed seamless integration with other Google Cloud services.

2. Deploy to Cloud Run:

1. The container was deployed on Google Cloud Run, a fully managed serverless platform.
2. Cloud Run handled auto-scaling, so the app could dynamically adjust to multiple users.

3. Configuration:

1. HTTPS endpoint was automatically generated, making the dashboard securely accessible via web browser.
2. Permissions were configured to allow public access.

The deployment ensured that the dashboard was highly available, scalable, and secure, aligning with industry practices for cloud-native applications.

The final deployed dashboard:

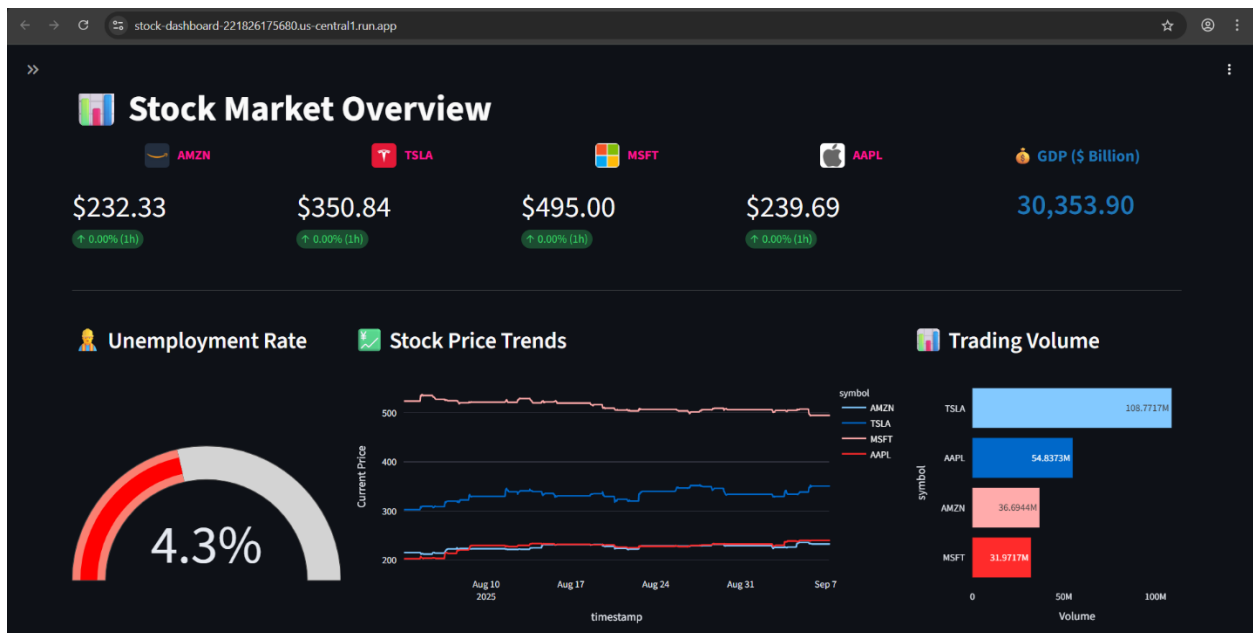


Figure 15:Streamlit dashboard view 01

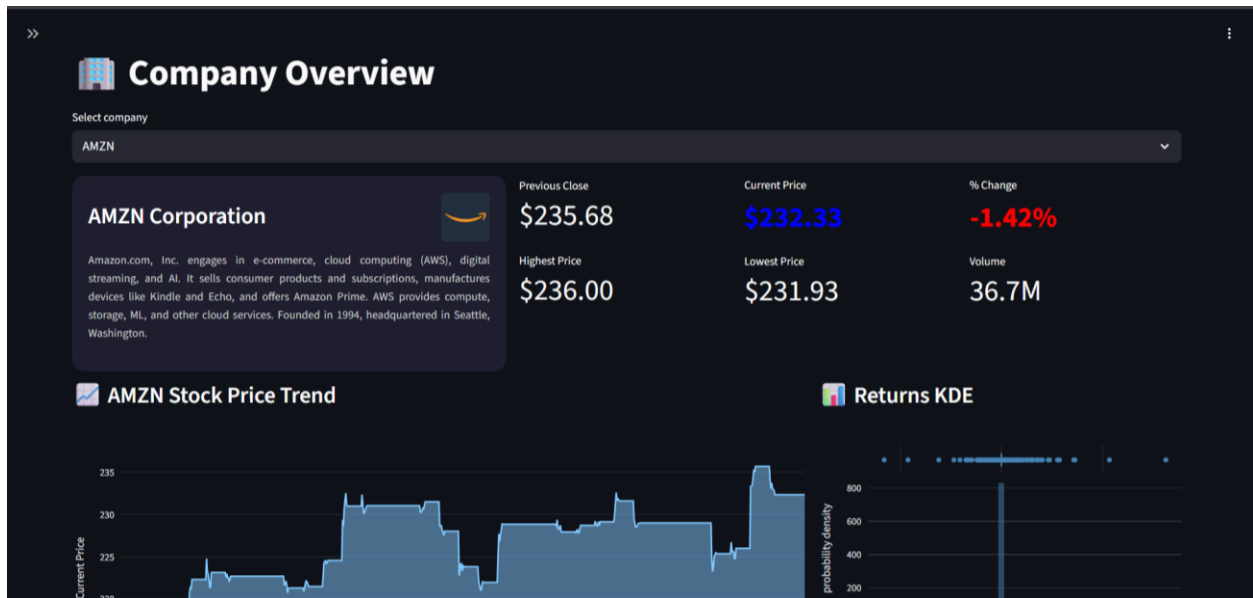


Figure 16:Streamlit dashboard view 02



Figure 17:Streamlit dashboard view 03

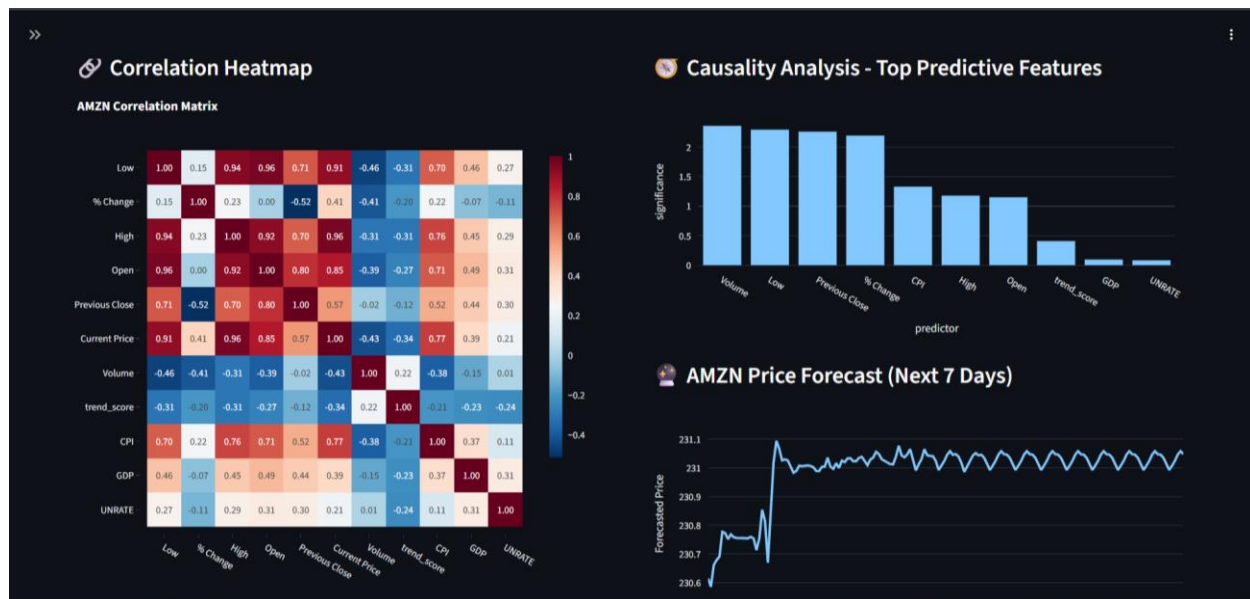


Figure 18:Streamlit dashboard view 04

10. Challenges Faced

- Timestamp alignment between multiple APIs.
- API rate limits for free-tier data.
- Optimizing Docker container size to improve deployment performance.
- Maintaining consistency across different update frequencies.

11. Conclusion

The project successfully implemented a fully automated, cloud-based stock market analysis system, demonstrating both technical rigor and practical utility. One of its key strengths was the integration of diverse data sources, combining traditional financial metrics such as stock prices, trading volumes, and technical indicators with alternative data including news, sentiment analysis, and other relevant market signals. This holistic approach enabled the system to capture a broader spectrum of market influences, which in turn improved the robustness and predictive power of the machine learning models.

Another important aspect of the project was the application of machine learning techniques for predictive insights. By leveraging historical and real-time data, the model was able to identify

patterns, trends, and potential future movements in stock prices, providing users with actionable information. The inclusion of machine learning added a layer of intelligence that went beyond static analysis, allowing for more dynamic and responsive predictions that could adapt as new data became available.

The development of a user-friendly, interactive dashboard was also a major achievement. The dashboard served as the interface through which users could explore the analyzed data and model predictions in real time. Features such as interactive charts, trend visualizations, and customizable views enhanced accessibility, making complex financial analysis understandable to a wider audience, including traders, analysts, and even non-experts interested in market trends.

Overall, the project not only demonstrated strong technical execution in areas like data integration, modeling, and cloud deployment but also emphasized practical usability and accessibility. By combining predictive analytics with an intuitive interface, the system provides a valuable tool for decision-making in financial markets, bridging the gap between complex data science techniques and everyday application for end-users.

12. References

<https://www.udemy.com/course/mastering-google-cloud-platform-gcp-for-python-developers/?srsltid=AfmBOopVc1WEGdwcBJWgF-OCu6gipzgypWD2SpGHeyvLYtegQCrw93AU&couponCode=25BBPMXNVD25CTRL>

13. Appendix

Deployed Streamlit Dashboard (Cloud Run):

<https://stock-dashboard-221826175680.us-central1.run.app/>

Google Drive Repository (Code Files & Dataset):

https://drive.google.com/drive/folders/1_Ha56-7d30Vv1PGELMWdOz2xjtolZUsl?usp=sharing