Impact of Commodities on Stocks

1. Project Introduction and Methodology

Project Outline and Goal

The main objective of this project was to analyze the impact of commodity price fluctuations—specifically crude oil, gold, and natural gas—on stock market performance, focusing on indices such as the S&P 500. The problem being addressed is the need to understand the relationship between commodity prices and stock markets, which is crucial for investors and analysts in making informed decisions.

The goal of this project is to build a data pipeline that automates the process of retrieving daily updated data from the S&P 500, correlating it with real-time commodity price data, and performing analysis to uncover any patterns or dependencies between these two data sources.

Technologies used for this project

- Python for data extraction, cleaning, and analysis.
- SQLite as the database to store processed data.
- Power BI and matplotlib for final data visualization.
- Apache Airflow for automating the ETL pipeline.
- Google Colab as the platform for development

Source Systems
Alpha Vantage API - for daily updated and historical data on commodities.
Kaggle - for daily updated stock market data (S&P 500 stocks and company information).

Method of Retrieval
Alpha Vantage API calls for real-time commodity price data.
Kaggle's daily updated dataset for S&P 500 stocks is downloaded using their API.

Data Destination
Cleaned and transformed data is stored in an SQLite database.
The data pipeline is automated using Apache Airflow to handle the daily updates and transformation process.

Data Velocity and Variety:
Data is retrieved daily, and the data includes numeric fields like commodity and stock prices and categorical fields sector, company names.

Overall Architecture

The architecture used is designed to handle real-time and daily updated data:

- Data Extraction: Commodity prices are extracted using the Alpha Vantage API, while stock data is pulled from Kaggle using its API to ensure the stock data is always up to date.
- Data Transformation: Data cleaning and transformation are performed in Python, ensuring both datasets are aligned by date.
- Data Storage: SQLite is used for storing the merged datasets, making querying for analysis easy.
- ETL Automation: Apache Airflow is used to automate the daily retrieval, transformation, and loading of the datasets.
- Visualization: Power BI and matplotlib are used for creating visual reports

2. Project Work Review

ETL Process

We started the project by setting up data extraction processes for both commodity prices and stock market data. Using the Alpha Vantage API, we retrieved daily updated commodity prices, and through an API integration with Kaggle, we fetched the latest stock market data for the S&P 500. Initially, these steps were performed manually in Python, and then we proceeded with data cleaning and transformation

The cleaning process involved handling missing values, such as filling gaps in company details for the stock data and removing rows with missing price information. The transformation included merging data, analyzing and modifying columns so that it will be ready to be served.

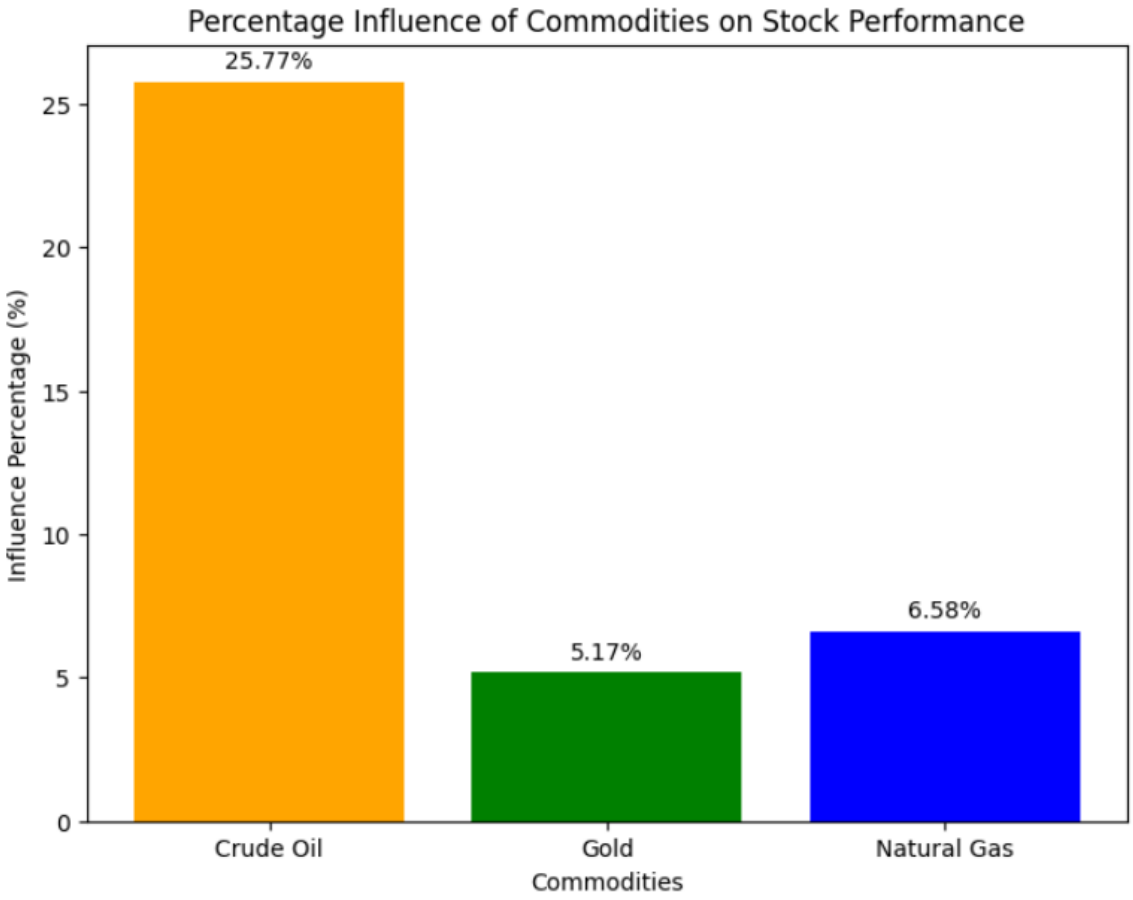
Automation with Apache Airflow

Once we successfully built the extraction, cleaning, and transformation processes in Python, the next critical step was to automate the entire process using Apache Airflow. We took all the code for data extraction, as well as the cleaning and transformation codes, and incorporated them into Airflow tasks. This setup allowed for a fully automated ETL pipeline
The Airflow DAG (Directed Acyclic Graph) was configured to run daily:

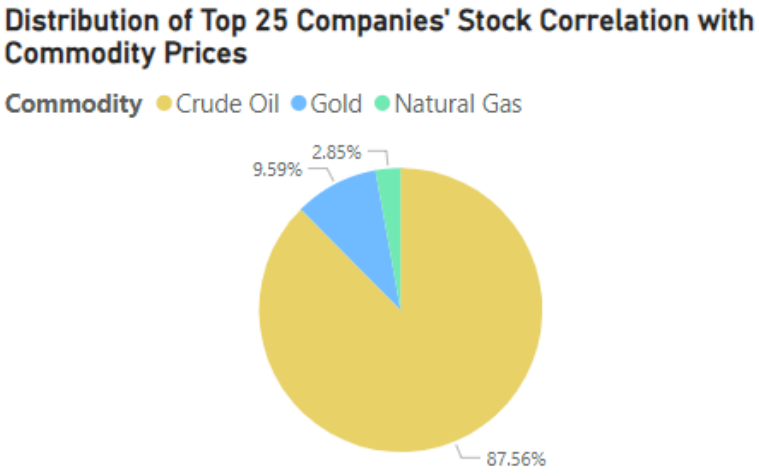
1. Task 1: Data Extraction
2. Task 2: Data cleaning and Transformation
3. Task 3: Companies Correlation Table
4. Task 4: Transformed data Load to SQLite

Data Visualizations

Regression Analysis (matplotlib)



Power BI



Top 25 Companies Stock Correlation with Commodity prices (Power BI)

Company Name	Correlation	Commodity	Sector
Newmont Corporation	0.56	Gold	Basic Materials
ConocoPhillips	0.49	Crude Oil	Energy
Hess Corporation	0.47	Crude Oil	Energy
Marathon Oil Corporation	0.47	Crude Oil	Energy
Exxon Mobil Corporation	0.46	Crude Oil	Energy
EOG Resources, Inc.	0.45	Crude Oil	Energy
Devon Energy Corporation	0.44	Crude Oil	Energy
Chevron Corporation	0.44	Crude Oil	Energy
Diamondback Energy, Inc.	0.43	Crude Oil	Energy
Occidental Petroleum Corporatio	0.42	Crude Oil	Energy
Halliburton Company	0.41	Crude Oil	Energy
APA Corporation	0.39	Crude Oil	Energy
Schlumberger N.V.	0.39	Crude Oil	Energy
Valero Energy Corporation	0.35	Crude Oil	Energy
Smurfit WestRock plc	0.35	Gold	Consumer Cyclical
Phillips 66	0.33	Crude Oil	Energy
Marathon Petroleum Corporation	0.32	Crude Oil	Energy
Baker Hughes Company	0.31	Crude Oil	Energy
Coterra Energy Inc.	0.30	Crude Oil	Energy
ONEOK, Inc.	0.29	Crude Oil	Energy
EQT Corporation	0.29	Crude Oil	Energy
Freeport-McMoRan, Inc.	0.28	Crude Oil	Basic Materials
Caterpillar, Inc.	0.27	Crude Oil	Industrials
Targa Resources, Inc.	0.27	Crude Oil	Energy
EQT Corporation	0.27	Natural Gas	Energy

Collaboration and Communication

Lakshmi Alekhya Tatampudi: Focused on Data Extraction, Transformation, Analysis
Krishna Swathi Peetha: Focused on Automation, Visualizations

For communication we met in person, had zoom calls and texted each other about progress

Challenges Faced

- Data Loss During Cleaning: Cleaning the stock and commodity data led to significant data loss due to missing values. We addressed this by adding default values, such as filling missing company locations with 'Unknown' and setting missing employee counts to 0, while removing rows only if essential data like stock prices were missing.

- API Call Limit: The Alpha Vantage API had request limits, which we overcame by using multiple email addresses to create additional API keys, allowing us to continue retrieving daily commodity price data without interruptions.

Most Challenging Part

One of the challenging parts was automating the ETL pipeline with Apache Airflow. Since we were working with daily updated data from Kaggle, we had to ensure that the Airflow DAG was correctly configured to run every day, pull the latest data, and integrate it with the real-time commodity data.

Another challenge was ensuring that missing or incomplete stock data from the daily updates didn't affect the overall results. We learned how to handle missing data correctly without losing valuable information.

Actual Work vs Expected Work

- The actual work differed from the expected work with the use of matplotlib for one visualization. Initially we thought of using Power BI for visualizations, but we also had to use python for one visualization i.e., for regression analysis (which is needed for knowing the dependency of commodity prices on overall stock performance)

- Initially, we expected to work with historical stock data, but the project took a different direction by getting daily updated stock data from Kaggle. This change enhanced the project's relevance, as it allowed the analysis to be conducted on the latest available data, providing more timely insights.

3. Retrospective

Future Developments

In the future, we would improve the project by:

- Using a cloud-based database for better scalability, as it can handle larger datasets more efficiently than SQLite.
- Implementing real-time data streaming using tools like Apache Kafka to continuously collect and analyze stock and commodity data without delays.
- Adding more advanced error handling and monitoring in Apache Airflow to automatically detect and fix issues in the data pipeline.
- In terms of data analysis - performing lag analysis

Critical Takeaways

The key lessons from this project are:

- Handling missing data effectively is important to avoid losing valuable insights, so filling gaps with default values is often better than discarding incomplete data.
- Synchronizing datasets from different sources, especially with daily updates, requires careful attention to ensure that they align properly for analysis.
- Automation is crucial when working with daily updated data to ensure accuracy and save time. Using Apache Airflow made the process more efficient.