DE Lab 4 Report

1. Identify a data pipeline design pattern that fits the project use case

In the context of financial data analysis that aims to predict gold prices while incorporating world events, a suitable data pipeline design pattern would be the **Lambda Architecture**. This architecture efficiently combines batch processing and stream processing, thereby allowing for both historical data analysis and real-time processing of world events that may impact gold pricing.

Characteristics of Lambda Architecture

Lambda Architecture consists of two main layers:

Batch Layer: This layer processes large volumes of historical data related to gold prices and relevant world events. The batch processing capability allows for comprehensive analytics and insights to be generated from accumulated data over time. This data can be used to train predictive models that account for long-term trends and seasonality in gold prices.

Speed Layer: The speed layer processes incoming data streams in real-time, allowing insights to be generated from immediate world events impacting gold prices. For instance, geopolitical events, economic indicators, and risks can be processed as they unfold, facilitating quick market responses. The integration of such real-time data can enhance the predictive capabilities of the models, making them more relevant and responsive to current market conditions.

Benefits of Using Lambda Architecture

The use of Lambda Architecture in this project offers several advantages:

Comprehensive Analysis: By utilizing both batch and stream processing, stakeholders can gain insights from historical data trends and react promptly to new information.

Flexibility and Scalability: It accommodates large datasets and high-velocity data streams, ensuring efficient data handling at scale. As the volume of financial data and world events increases, the infrastructure can be scaled to meet demand.

Accuracy and Completeness: This dual-layer system provides the ability to refine predictions over time. The batch layer ensures that large data volumes contribute to accuracy, while the speed layer enhances timeliness and relevance.

Conclusion:
We are using Lambda Architecture for our financial data analysis project on predicting gold prices to effectively analyze patterns from historical data while reacting to real-time events. This approach enables comprehensive predictive modeling and supports better decision-making in a dynamic financial environment.

2. Collect data from online sources:

We first took datasets from kaggle and gfg but some of the cons of using that are:

Data Inaccuracy: The Kaggle dataset showed inconsistencies, such as gold prices being in the range of 100-200 USD, which is significantly lower than verified sources like newspapers reporting 2600 USD.

Unrealistic Price Relationships: The dataset implied that silver prices were higher than gold prices, which contradicts known market data.

Missing Data: For some commodities like Rhodium, the dataset had numerous missing values or incorrect zeros due to the illiquidity in its market.

Outdated Information: The data from Kaggle spanned only from 2011 to 2019, lacking the recent updates necessary for reliable forecasting in 2024.

Limited Customization: The pre-existing dataset did not allow for real-time updates or the ability to scrape and customize data, which was essential for your dynamic analysis

Why we switched to **web scraping**:

- Web scraping ensures you collect up-to-date, accurate data directly from reliable sources, resolving inconsistencies and outdated information found in Kaggle datasets.
- It allows gathering specific financial data, filling in missing values and tailoring the dataset to include exactly the variables needed for the analysis (e.g., gold, silver, S&P 500, Dow Jones).

3. Collect data (Scrap from sources):

-> **Purpose of Data Collection**:
The data was collected from multiple financial sources to analyze the relationships between various economic factors and gold prices. The sources include stock indices, commodity futures, exchange rates, and bond rates.

-> Data Sources:
Yahoo finance does not allow automated access so web scraping using tools like beautifulsoup is not possible. consequently, we switched to using **finance api**
The data was retrieved from **Yahoo Finance** using the `yfinance` library. It includes financial data for:

- S&P 500 Index (`^GSPC`)
- Dow Jones Index (`^DJI`)
- Eldorado Gold Corporation (`EGO`)
- EUR/USD Exchange Rate (`EURUSD=X`)
- Brent Crude Oil Futures (`BZ=F`)
- Crude Oil WTI Futures (`CL=F`)
- Silver Futures (`SI=F`)
- US Bond Rate (`^TNX`)
- Platinum Futures (`PL=F`)
- Palladium Futures (`PA=F`)
- Gold Miners ETF (`GDX`)
- Oil ETF USO (`USO`)
- US Dollar Index (`DX-Y.NYB`

-> Time Period:
The data was collected from July 30, 2007 to October 18, 2024, with a daily frequency (`interval='1d'`).

-> Data Validation:

The script checks whether the data downloaded for each ticker is empty or not. If data is found, it is saved as a CSV file for further analysis. If no data is available, the script logs that no data was found for the respective ticker.

-> Data Storage:
Each dataset is saved as a separate CSV file for modularity and easy access during analysis. The naming convention is `'{ticker_name}_data.csv'`, ensuring clarity when accessing the datasets.

->Conclusion:
By using a well-documented library like `yfinance` and saving each dataset to individual CSV files, the data collection process ensures reliability and accuracy. Additionally, the consistent time range and frequency across datasets ensure comparability in the subsequent analysis.

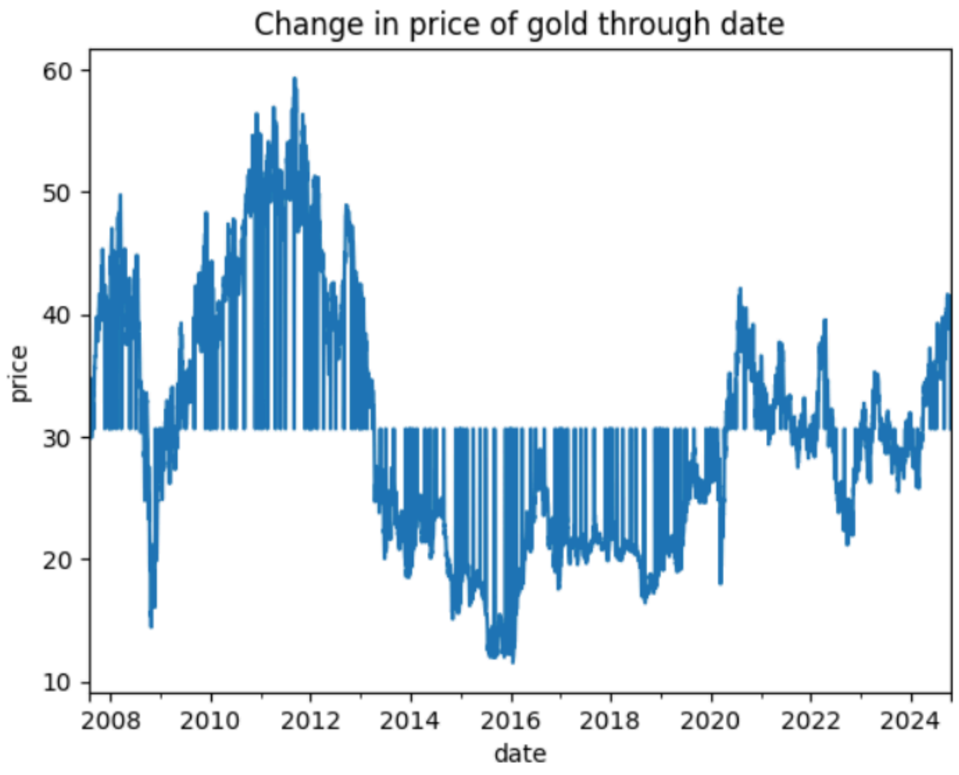
4. Data Wrangling of data collected:

- First we did Data Pre-processing :

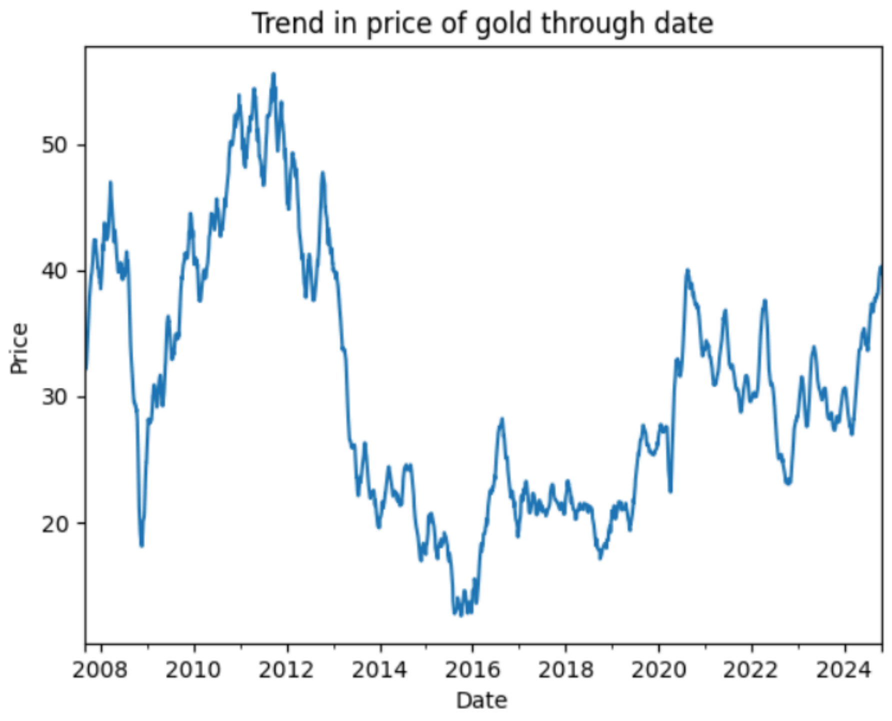
  -> **Handling Missing Values**:
  The dataset is first checked for missing or null values using `isna()`. Any missing values are then addressed using the forward and backward fill method (`ffill, bfill`), where NaN values are replaced with the last valid data point, ensuring no gaps in the dataset.

  ->**Correlation Analysis**:
  A correlation matrix is calculated to understand the relationships between different features in the dataset. This is visualized using a heatmap, allowing us to identify strong positive or negative correlations between variables, which is essential for feature selection and analysis.

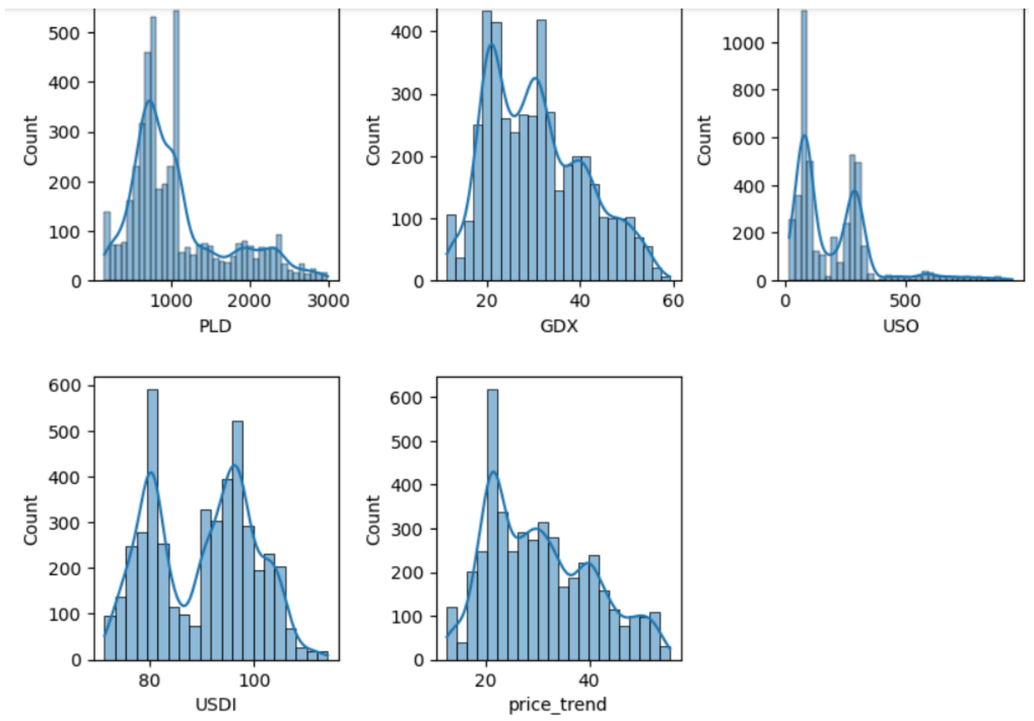
- Data Wrangling:
  ->A line plot was generated to visualize the **change in the price of gold (GDX) over time**, helping to observe trends and fluctuations across the dataset.



-> A rolling mean with a window size of 20 was applied to smooth out fluctuations in gold prices, and a line plot was generated to visualize the **overall price trend over time**.

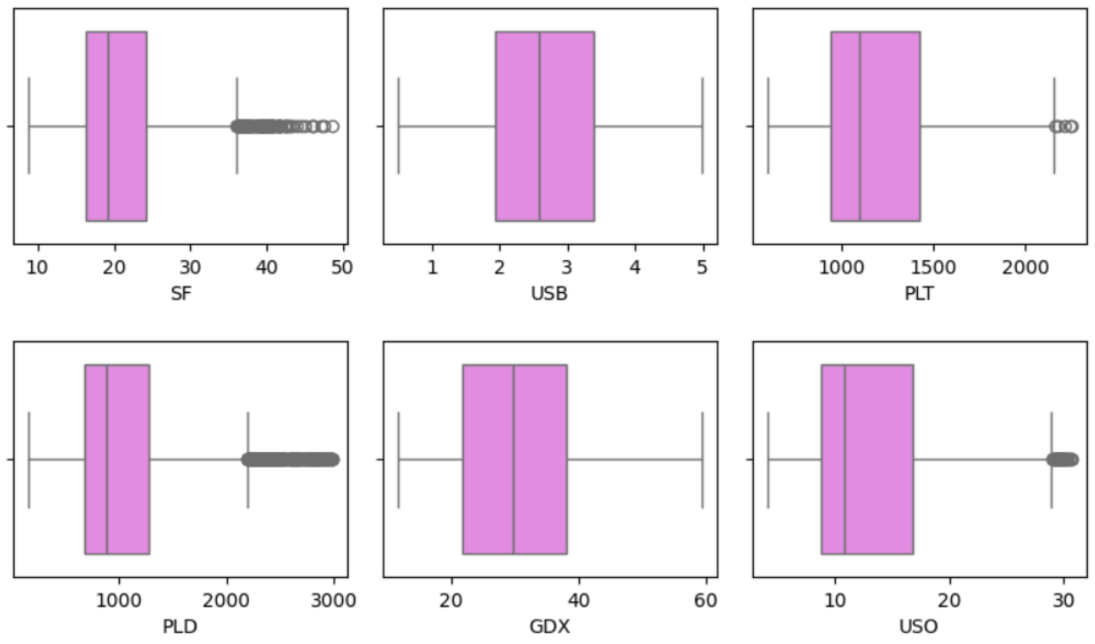


->**Histograms** with KDE were plotted for each numeric column to visualize the distribution of the data across various features.



-> **Skewness** was calculated for each numeric column to identify asymmetries in the data, and a square root transformation was applied to reduce skewness in the "USO" column.

->Box plots were then generated to visualize the distribution and detect **potential outliers** across the remaining numeric features.



5. ## Establish that your data respects the data reliability framework that you designed for Lab_3 Assignment

Data Governance:

Data Ownership & Accountability:We assigned specific team member to manage gold price data, ensuring its accuracy.
Data Quality & Integrity: We ensured that the data remains clean and trustworthy upon ingestion via checking for missing values, consistency, etc.

Data Collection Standards:

Consistent Units & Formats: We collected data from Yahoo Finance, for instance, using the same units (USD per ounce) and uniform timestamps (UTC).
Automation with Python: Python is used to automate data scraping and validation.

Data Cleaning and Validation:

Missing data was handled by forward and backward filling from previous and subsequent values.
Outliers(e.g. Covid period (March 2020 to Mid 2021)) are adjusted by changing any values above the top 5% to the value at the 95th percentile and any values below the bottom 5% to the value at the 5th percentile.

Version Control:

- Saved each modified dataset as a separate CSV file with the modification date in the filename (e.g., decrypted_data_union_2024_10_20.csv).
- Transitioned from a Kaggle dataset to custom-scraped data.
- Maintained an audit log to document changes for transparency.
- Ensured reliable tracking of dataset modifications without overwriting previous versions.

Data Security:

We initially encrypted the data before saving it to the CSV file to ensure that the information remains secure and inaccessible to unauthorized users. After decrypting the data for preprocessing and wrangling, we re-encrypted it to maintain its security.