Milestone 1: Data Collection, Preprocessing, and Exploratory Data Analysis (EDA)

Sai Pande(37696687)

INTRODUCTION

The financial market is a complex system influenced by various economic factors, including stock prices, commodity prices, real estate trends, and economic recessions. In this report I aims to explore the relationships between these datasets—S&P 500 stocks, gold prices, real estate sales, and U.S. recessions—to see potential correlations and insights. These connections can help investors, policymakers, and analysts make informed decisions.

Gold is often considered a safe-haven asset during market downturns, while real estate trends reflect economic stability. The stock market, influenced by company performance and macroeconomic conditions, plays a crucial role in financial cycles. By preprocessing and analyzing these datasets, I want to identify patterns, assess market behavior, and understand how different sectors react to economic changes. This project highlights the interconnectedness of financial markets and aims to provides valuable insights into economic trends.

Github link -> https://github.com/SaiPande/cap5771sp25-project

DATASETS I USED FOR THE PROJECT –

Kaggle Dataset links ->

S&P500 -> https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks?select=sp500_stocks.csv

Gold -> https://www.kaggle.com/datasets/ahmadkarrabi/gold-price-archive-2010-2023-dataset

Real Estate -> https://www.kaggle.com/datasets/utkarshx27/real-estate-sales-2001-2021-gl

Recession -> https://www.kaggle.com/datasets/shubhaanshkumar/us-recession-dataset

US Housing Price -> https://www.kaggle.com/datasets/utkarshx27/real-estate-sales-2001-2021-gl

1. sp500_companies.csv ->

Contains details about S&P 500 companies, such as their exchange, sector, and financial data.

- **Columns & Data Types:**
  - **This dataset has 502 rows and 16 columns.**
  - The Columns are ->
    Exchange, Symbol, Shortname, Longname, Sector, Industry, City, State, Country, Longbusinesssummary -> *Strings (Object)*

    Currentprice, Marketcap, Ebitda, Revenuegrowth, Fulltimeemployees, Weight -> *Float64*

2.sp500_stocks.csv -> Contains daily stock prices for S&P 500 companies.

- **This dataset has 1,891,536 rows and 8 columns.**
- The Columns are ->

  Date, Symbol of *Strings (Object)*

  Adj Close, Close, High, Low, Open, Volume -> *Float64*

3. **sp500_index.csv ->** Tracks historical S&P 500 index values over time.

- **This dataset has 2,517 rows and 2 columns.**
- The Columns are ->

  Date -> *String (Object)*

  S&P500 -> *Float64*

4. **Real_Estate_Sales_2001-2021_GL.csv ->** Contains property sales data across various towns from 2001 to 2021.

The dataset has 782,759 rows and 9 columns.
The Columns are ->

Date Recorded, Town, Address, Property Type, Residential Type -> Strings (Object)

List Year -> Integer (int64)

Assessed Value, Sale Amount, Sales Ratio -> Float (float64)

5. **US_Recession.csv** -> Contains monthly economic indicators of the US, including GDP, unemployment rate, and recession indicators.
    - **This dataset has 248 rows and 20 columns.**
    - The columns are ->

        Date -> Object (String)

        GDP, Unemployment Rate, Price_x, INDPRO, CPI, Rate, BBK_Index, Housing_Index, Treasury bond yields (3 Mo to 30 Yr) -> Float64

        Recession -> Int64 (Binary: 1 for recession, 0 for no recession)

6. **GOLD.csv ->** Contains historical gold price data with technical indicators.
    - **This dataset has 98,065 rows and 9 columns**.
    - The columns are ->

        Date, Time → String (Object)

        Open, High, Low, Close, Volume, RSI14, SMA14 → Float64

SUMMARY OF PRE-PROCESSING

1. S&P 500 Companies (sp500_companies.csv)

- **Missing Value Handling:**
    - ebitda, revenuegrowth, fulltimeemployees → Filled with median
    - state → Filled with "Unknown"
- **Outlier Handling:** Applied IQR method for ebitda, revenuegrowth, marketcap
- **Transformation:** Applied log transformation for positive values in financial columns
- **Normalization:** Scaled financial columns using MinMaxScaler
- **Output:** Saved as sp500_companies_cleaned.csv

2. S&P 500 Stocks (sp500_stocks.csv)

- **Missing Value Handling:** Dropped rows where all key stock price columns were NaN (adj close, close, high, low, open, volume)
- **Merging:** Merged with sp500_index.csv on date
- **Feature Engineering:**
    - Converted date to datetime format
    - Extracted year, month, day, day_of_week
- **Column Renaming:** Renamed s&p500 to index
- **Output:** Saved as sp500_cleaned.csv

3. Gold Prices (GOLD.csv)

- **Date Handling:** Converted time column to datetime and renamed it to date
- **Feature Engineering:**
    - Extracted year, month, day, day_of_week, hour
- **Outlier Handling:** Applied IQR method for open, high, low, close
- **Normalization:** Scaled rsi14 and sma14 using MinMaxScaler
- **Output:** Saved as GOLD_cleaned.csv

4. Real Estate Sales (Real_Estate_Sales_2001-2021_GL.csv)

- **Missing Value Handling:** Dropped rows with NaN in key columns (Date Recorded, Assessed Value, Sale Amount, etc.)
- **Date Handling:** Converted Date Recorded to datetime
- **Feature Engineering:**
    - Extracted year, month, day, day_of_week
- **Outlier Handling:** Applied IQR method to Assessed Value, Sale Amount, Sales Ratio
- **Normalization:** Scaled numeric columns using MinMaxScaler
- **Encoding:**
    - Label encoded Town

    - One-hot encoded Property Type & Residential Type
- **Output:** Saved as RealEstate_cleaned.csv

5. US Recession (US_Recession.csv)

- **Column Standardization:** Trimmed whitespace from column names
- **Missing Value Handling:**
    - Dropped rows with NaN in key economic indicators (GDP, Rate, Recession)
    - Filled NaN values in numeric columns with median
    - Filled NaN in categorical columns with mode
- **Outlier Handling:** Applied IQR method to numeric columns
- **Normalization:** Scaled numeric columns using MinMaxScaler
- **Encoding:** Label encoded categorical columns
- **Output:** Saved as US_Recession_cleaned.csv

Final Output Files:

sp500_companies_cleaned.csv

```python
# PROCESS S&P 500 COMPANIES

companies['ebitda'] = companies['ebitda'].fillna(companies['ebitda'].median())
companies['revenuegrowth'] = companies['revenuegrowth'].fillna(companies['revenuegrowth'].median())
companies['state'] = companies['state'].fillna("Unknown")
companies['fulltimeemployees'] = companies['fulltimeemployees'].fillna(companies['fulltimeemployees'].median())

def cap_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])

financial_cols = ['ebitda', 'revenuegrowth', 'marketcap']
for col in financial_cols:
    if col in companies.columns:
        cap_outliers_iqr(companies, col)

scaler = MinMaxScaler()
for col in financial_cols:
    if col in companies.columns:
        companies[col] = np.where(companies[col] > 0, np.log1p(companies[col]), 0)
        companies[col] = scaler.fit_transform(companies[col].values.reshape(-1, 1))

companies.to_csv("sp500_companies_cleaned.csv", index=False)
print("S&P 500 Companies dataset processed successfully!")
display(companies.head())
companies.info()
```

✓ 0.0s

```
S&P 500 Companies dataset processed successfully!
c:\Users\saipa\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\arraylike.py:399: RuntimeWarning: invalid value encountered in log1p
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

	exchange	symbol	shortname	longname	sector	industry	currentprice	marketcap	ebitda	revenuegrowth	city	state	country	fulltimeemployees	longbusine
0	NMS	AAPL	Apple Inc.	Apple Inc.	Technology	Consumer Electronics	254.49	1.0	1.0	0.248140	Cupertino	CA	United States	164000.0	Apple manu
1	NMS	NVDA	NVIDIA Corporation	NVIDIA Corporation	Technology	Semiconductors	134.70	1.0	1.0	1.000000	Santa Clara	CA	United States	29600.0	NVIDIA provides
2	NMS	MSFT	Microsoft Corporation	Microsoft Corporation	Technology	Software - Infrastructure	436.60	1.0	1.0	0.621985	Redmond	WA	United States	228000.0	Microsoft develops

sp500_cleaned.csv

```python
# PROCESS S&P 500 STOCKS

columns_to_check = ['adj close', 'close', 'high', 'low', 'open', 'volume']
stocks = stocks.dropna(subset=columns_to_check, how='all')

merged_data = pd.merge(stocks, index, on='date', how='inner')

merged_data['date'] = pd.to_datetime(merged_data['date'])

merged_data['year'] = merged_data['date'].dt.year
merged_data['month'] = merged_data['date'].dt.month
merged_data['day'] = merged_data['date'].dt.day
merged_data['day_of_week'] = merged_data['date'].dt.dayofweek

merged_data.drop(columns=['date'], inplace=True)

merged_data.rename(columns={'s&p500': 'index'}, inplace=True)

merged_data.to_csv("sp500_cleaned.csv", index=False)
print("S&P 500 Stocks dataset processed successfully!")
display(merged_data.head())
merged_data.info()
```

✓ 3.4s

```
S&P 500 Stocks dataset processed successfully!
```

	symbol	adj close	close	high	low	open	volume	index	year	month	day	day_of_week
0	AOS	23.673809	27.674999	27.684999	27.200001	27.309999	852600.0	2078.54	2014	12	22	0
1	AOS	23.960384	28.010000	28.145000	27.590000	27.795000	973400.0	2082.17	2014	12	23	1
2	AOS	24.033092	28.094999	28.209999	27.900000	27.900000	233600.0	2081.88	2014	12	24	2
3	AOS	24.217010	28.309999	28.455000	28.170000	28.250000	360000.0	2088.77	2014	12	26	4
4	AOS	24.238392	28.334999	28.490000	28.195000	28.299999	391800.0	2090.57	2014	12	29	0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 423337 entries, 0 to 423336
Data columns (total 12 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   symbol       423337 non-null  object
 1   adj close    423337 non-null  float64
 2   close        423337 non-null  float64
 3   high         423337 non-null  float64
 4   low          423337 non-null  float64
 5   open         423337 non-null  float64
 6   volume       423337 non-null  float64
 7   index        423337 non-null  float64
```

GOLD_cleaned.csv

```python
# PROCESS GOLD DATA

gold['time'] = pd.to_datetime(gold['time'])
gold.rename(columns={'time': 'date'}, inplace=True)

gold['year'] = gold['date'].dt.year
gold['month'] = gold['date'].dt.month
gold['day'] = gold['date'].dt.day
gold['day_of_week'] = gold['date'].dt.dayofweek
gold['hour'] = gold['date'].dt.hour

for col in ['open', 'high', 'low', 'close']:
    cap_outliers_iqr(gold, col)

gold[['rsi14', 'sma14']] = scaler.fit_transform(gold[['rsi14', 'sma14']])

gold.to_csv("GOLD_cleaned.csv", index=False)
print("Gold dataset processed successfully!")
display(gold.head())
gold.info()
```

✓ 9.3s

Gold dataset processed successfully!

	date	open	high	low	close	rsi14	sma14	year	month	day	day_of_week	hour
0	2010-01-03 18:00:00	1098.45	1100.0	1098.05	1099.95	0.842004	0.044514	2010	1	3	6	18
1	2010-01-03 18:05:00	1100.00	1100.3	1099.45	1099.75	0.812047	0.044833	2010	1	3	6	18
2	2010-01-03 18:10:00	1099.70	1100.1	1099.30	1099.45	0.767804	0.045123	2010	1	3	6	18
3	2010-01-03 18:15:00	1099.50	1099.6	1098.50	1099.45	0.767804	0.045376	2010	1	3	6	18
4	2010-01-03 18:20:00	1099.40	1099.6	1098.90	1098.90	0.687633	0.045563	2010	1	3	6	18

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 986004 entries, 0 to 986003
Data columns (total 12 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   date         986004 non-null  datetime64[ns]
 1   open         986004 non-null  float64
 2   high         986004 non-null  float64
 3   low          986004 non-null  float64
 4   close        986004 non-null  float64
 5   rsi14        986004 non-null  float64
 6   sma14        986004 non-null  float64
 7   year         986004 non-null  int32
 8   month        986004 non-null  int32
 9   day          986004 non-null  int32
 10  day_of_week  986004 non-null  int32
```

RealEstate_cleaned.csv

```
Real Estate dataset processed successfully!
```

	Serial Number	List Year	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Non Use Code	Assessor Remarks	OPM remarks	...	day_of_week	Property Type_Four Family	Property Type_Residential	Property Type_Single Family	Property Type_Three Family	Property Type_Two Family
1	20002	2020	3	390 TURNPIKE RD	0.556807	0.711318	0.390944	NaN	NaN	NaN	...	4	False	True	False	False	False
3	200212	2020	4	5 CHESTNUT DRIVE	0.286131	0.295662	0.522301	NaN	NaN	NaN	...	1	False	True	False	False	False
10	210045	2021	10	89 LONG MEADOW RD	0.505807	0.930447	0.218495	NaN	NaN	NaN	...	4	False	True	False	False	False
11	210101	2021	11	43 LEDYARD AVE	0.244580	0.344025	0.338497	NaN	NaN	NaN	...	0	False	True	False	False	False
12	20139	2020	9	16 DEEPWOOD DRIVE	0.376562	0.553432	0.317037	NaN	NaN	NaN	...	2	False	True	False	False	False

5 rows × 24 columns

```
Index: 519474 entries, 1 to 1054157
Data columns (total 24 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Serial Number               519474 non-null  int64
 1   List Year                   519474 non-null  int64
 2   Town                        519474 non-null  int64
 3   Address                     519474 non-null  object
 4   Assessed Value              519474 non-null  float64
 5   Sale Amount                 519474 non-null  float64
 6   Sales Ratio                 519474 non-null  float64
 7   Non Use Code                115480 non-null  object
 8   Assessor Remarks            91468 non-null   object
 9   OPM remarks                 8473 non-null    object
 10  Location                    138589 non-null  object
 11  year                        519474 non-null  int32
 12  month                       519474 non-null  int32
 13  day                         519474 non-null  int32
 14  day_of_week                 519474 non-null  int32
 15  Property Type_Four Family   519474 non-null  bool
 16  Property Type_Residential   519474 non-null  bool
 17  Property Type_Single Family 519474 non-null  bool
```

US_Recession_cleaned.csv

```
US Recession dataset processed successfully!
```

	Unnamed: 0	Price_x	INDPRO	CPI	3 Mo	4 Mo	6 Mo	1 Yr	2 Yr	3 Yr	5 Yr	7 Yr	10 Yr	20 Yr	30 Yr	GDP	Ra
0	0.000000	0.008556	0.046856	0.000000	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	0.995788	0.981235	0.0	0.957474	0.00000	0.9963
1	0.002915	0.011880	0.051140	0.001183	0.996264	0.0	0.991458	0.984958	0.987719	0.986550	0.998219	0.993983	0.976998	0.0	0.953608	0.00000	0.9866
2	0.005831	0.014674	0.055858	0.005914	0.993773	0.0	0.976205	0.967509	0.962573	0.963743	0.969121	0.968712	0.950363	0.0	0.926546	0.00000	1.0000
3	0.008746	0.014574	0.053475	0.009463	0.976339	0.0	0.960342	0.943442	0.938012	0.946199	0.956057	0.961492	0.951574	0.0	0.934278	0.00273	0.9830
4	0.011662	0.006509	0.058643	0.015968	0.957659	0.0	0.939597	0.927798	0.930994	0.943860	0.971496	0.985560	0.987893	0.0	0.985825	0.00273	0.9805

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Unnamed: 0     344 non-null    float64
 1   Price_x        344 non-null    float64
 2   INDPRO         344 non-null    float64
 3   CPI            344 non-null    float64
 4   3 Mo           344 non-null    float64
 5   4 Mo           344 non-null    float64
 6   6 Mo           344 non-null    float64
 7   1 Yr           344 non-null    float64
 8   2 Yr           344 non-null    float64
 9   3 Yr           344 non-null    float64
 10  5 Yr           344 non-null    float64
 11  7 Yr           344 non-null    float64
 12  10 Yr          344 non-null    float64
 13  20 Yr          344 non-null    float64
 14  30 Yr          344 non-null    float64
 15  GDP            344 non-null    float64
 16  Rate           344 non-null    float64
 17  BBK_Index      344 non-null    float64
 18  Housing_Index  344 non-null    float64
 19  Recession      344 non-null    float64
dtypes: float64(20)
```

EXPLORATORY DATA ANALYSIS

Statistics Description of all the datasets

```python
# Apply to each dataset
descriptive_statistics(stocks, "S&P 500 Stocks")
descriptive_statistics(companies, "S&P 500 Companies")
descriptive_statistics(gold, "Gold")
descriptive_statistics(realestate, "Real Estate")
descriptive_statistics(recession, "US Recession")
```

✓ 1.3s

```
             6 Mo        1 Yr        2 Yr        3 Yr        5 Yr        7 Yr  \
count  344.000000  344.000000  344.000000  344.000000  344.000000  344.000000
mean     0.333465    0.341654    0.357312    0.374446    0.409782    0.426895
std      0.291224    0.288109    0.284409    0.280199    0.273699    0.267787
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.018304    0.033394    0.073392    0.102924    0.161520    0.190584
50%      0.295912    0.302046    0.318713    0.332164    0.340855    0.353791
75%      0.622636    0.619434    0.620468    0.633918    0.657067    0.666667
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

            10 Yr       20 Yr       30 Yr         GDP        Rate   BBK_Index  \
count  344.000000  344.000000  344.000000  344.000000  344.000000  344.000000
mean     0.436425    0.455780    0.448581    0.393398    0.328922    0.518590
std      0.257701    0.267940    0.250996    0.275032    0.296108    0.168814
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.206416    0.284976    0.228093    0.126811    0.013350    0.394527
50%      0.388317    0.420620    0.421392    0.427503    0.284587    0.512449
75%      0.647851    0.690085    0.645780    0.611902    0.631068    0.657545
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

       Housing_Index  Recession
count     344.000000      344.0
mean        0.296034        0.0
std         0.248620        0.0
min         0.000000        0.0
25%         0.043431        0.0
50%         0.297394        0.0
75%         0.461393        0.0
max         1.000000        0.0
```

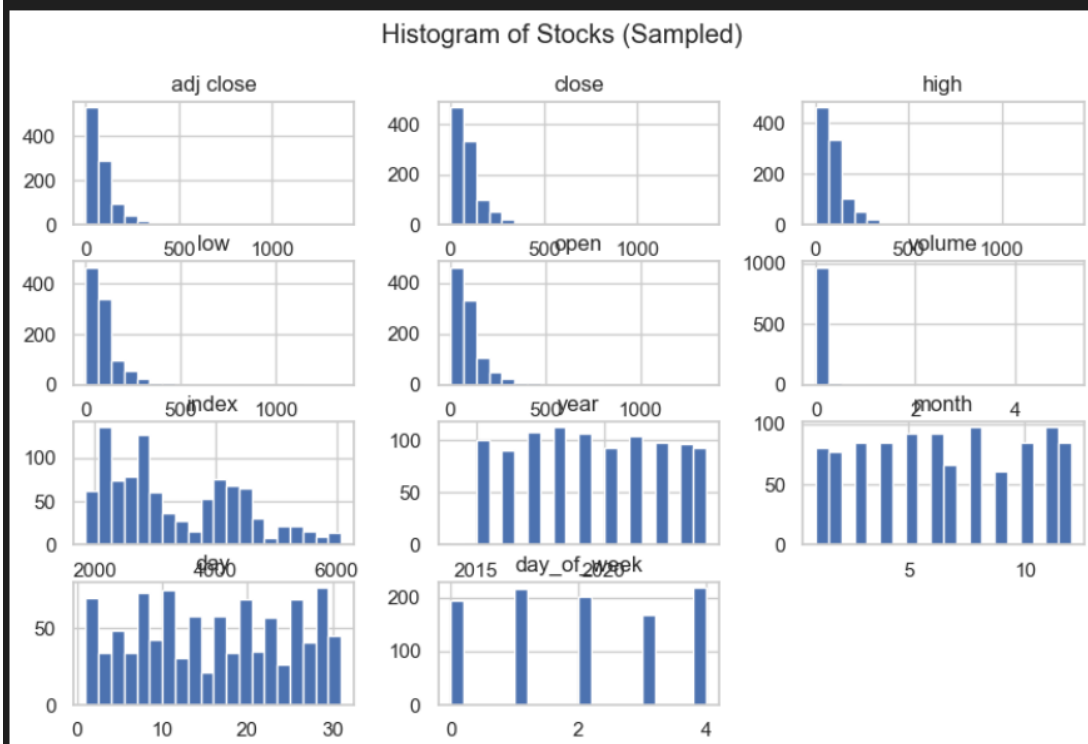
Visualization using Histogram

The histograms tells us that the price-related variables (adj close, close, high, low, and open) are right-skewed, with most values concentrated on the lower end and fewer extreme high values. Volume is heavily right-skewed, suggesting that most trading volumes are low, with occasional large spikes. The index shows a wider spread, reflecting long-term growth over time. The year histogram is evenly distributed, indicating a consistent sample across different years. Both month and day_of_week are relatively uniform, suggesting no strong seasonal or weekly patterns. Overall, the data shows skewed price and volume distributions, while time variables are more evenly spread.

```python
# Histograms with limited data
for df, name in zip([stocks, companies, gold, realestate, recession],
                    ["Stocks", "Companies", "Gold", "Real Estate"]):
    # Sample a limited number of rows
    sampled_df = df.sample(n=1000, random_state=42) if len(df) > 1000 else df

    # Histogram plot
    sampled_df.hist(bins=20, figsize=(10, 6))
    plt.suptitle(f"Histogram of {name} (Sampled)")
    plt.show()
```

✓ 5.5s



Correlation matrix of datasets

The correlation matrix tells us that that stock prices (adj close, close, high, low, and open) are perfectly correlated (1.00), indicating they move together.

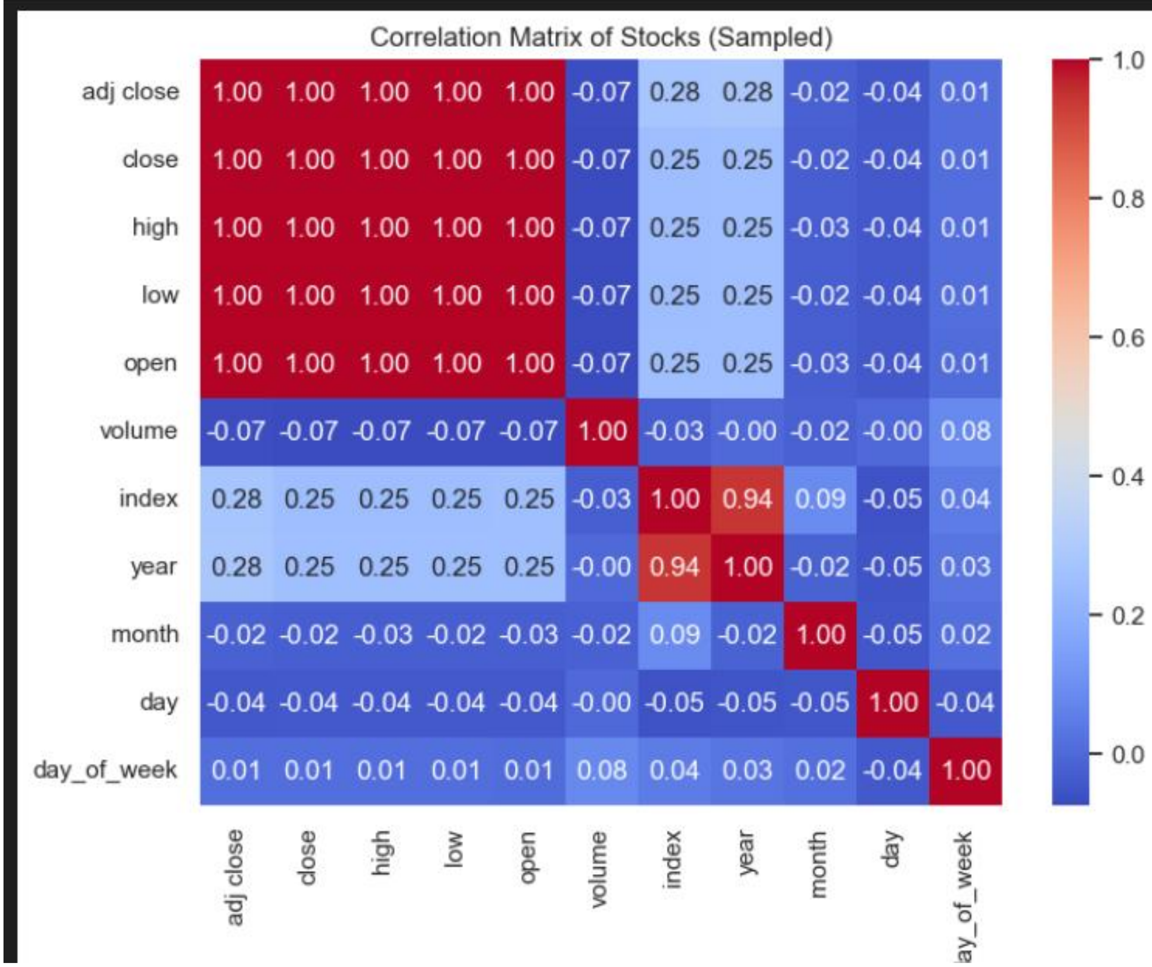
The index and year show a high positive correlation (0.94), reflecting the index's upward trend over time. Trading volume has little to no correlation with price variables, suggesting minimal impact on stock prices.

Time components like day_of_week, day, and month have weak or no correlations, except for the year, which aligns with long-term growth. Overall, stock prices exhibit strong internal relationships and upward trends, while volume and short-term time factors have limited influence.

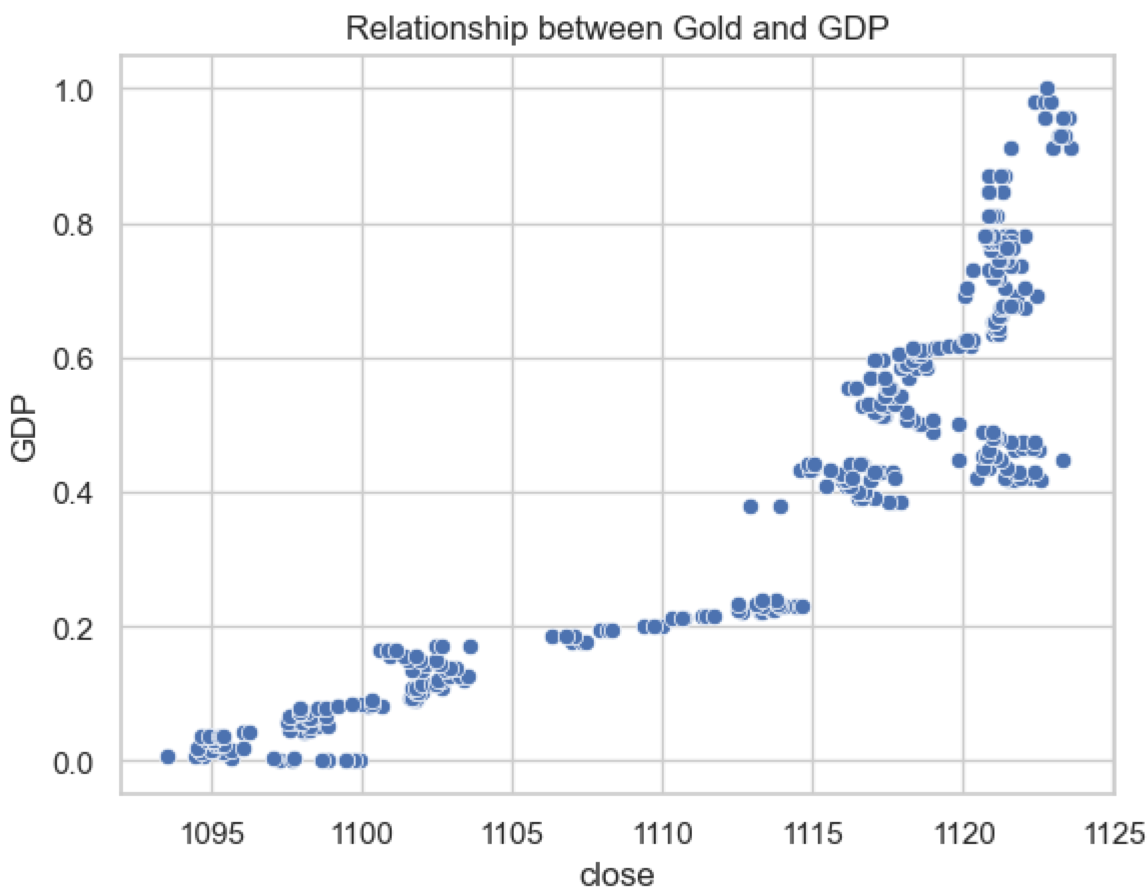
```python
# Correlation analysis with limited data
for df, name in zip([stocks, companies, gold, realestate, recession],
                    ["Stocks", "Companies", "Gold"]):
    # Sample a limited number of rows
    sampled_df = df.sample(n=1000, random_state=42) if len(df) > 1000 else df
    corr_matrix = sampled_df.corr(numeric_only=True)

    # Heatmap plot
    plt.figure(figsize=(8, 6))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
    plt.title(f"Correlation Matrix of {name} (Sampled)")
    plt.show()
```

✓ 2.1s



Correlation between Gold and GDP



This scatter plot shows the relationship between gold prices (x-axis) and GDP (y-axis). The pattern appears to be a positive correlation, as GDP increases with rising gold prices. However, the relationship looks somewhat segmented into clusters or plateaus, indicating potential periods of stability or specific economic conditions where both variables moved together.

This could suggest that during certain price ranges of gold, GDP remained relatively stable before jumping to another level. The clustered pattern might indicate:

- **Plateaus:** Economic phases where GDP didn't fluctuate much despite gold price changes.

- **Steep transitions:** Periods where a slight increase in gold prices led to a notable jump in GDP.

TECH STACK

1. Programming Language:

- Python

2. Libraries and Tools:

- **Pandas**: Data manipulation and analysis
- **NumPy**: Numerical operations
- **SciPy (zscore)**: Statistical analysis
- **Scikit-Learn (MinMaxScaler, LabelEncoder)**: Data normalization and encoding
- **Seaborn**: Data visualization
- **Matplotlib**: Plotting and visualization

3. Environment:

- Jupyter Notebook (based on the .ipynb file)
- Visual Studio Code (based on the screenshot of the file explorer)

4. Data Sources:

- CSV files (cleaned and raw datasets related to S&P 500, gold, real estate, and US recession) from Kaggle.

Next Milestones:

Milestone 2: February 21, 2025 - March 21, 2025

I tried my best to use the data efficiently but I still see that preprocessing is not up to the par, I wish to do it to refine my data and get better results. I will also be working on feature engineering, feature selection and data modeling timeline.

Milestone 3: March 24, 2025 - April 23, 2025

I will be evaluating my model on test sets and using matrices I used to train model; I plan to develop a dashboard and deliver the final report and project.

LLM

I use ChatGPT to check my grammar and describe my points in a better way.