International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Financial Predictions of Gold Prices using Time Series Analysis

Vedika Bansal

Grade 12, Mayo College Girls' School, Ajmer, Rajasthan, India
Email: *vedika1698[at]gmail.com*

Abstract: ***Gold has long been recognized as a safe-haven asset and a hedge against inflation, especially during times of economic uncertainty. This study aims to forecast gold prices using time series analysis, drawing on historical data from 2019 to 2024. Multiple forecasting models- ARIMA, SARIMA, Exponential Smoothing, and Prophet- were evaluated to determine the most reliable method. Given the presence of significant market volatility and outliers in the dataset, the Prophet model was selected for its robustness to irregularities and ability to model complex seasonality. The data underwent preprocessing. The final model forecasted daily gold prices for 2025, revealing a projected upward trend. This aligns with current macroeconomic indicators such as persistent inflation, geopolitical instability, and ongoing global conflicts- factors historically known to increase demand for gold. The study concludes that gold prices are likely to rise in 2025, reinforcing gold's role as a strategic investment asset. These insights can support better investment decision-making in uncertain economic climates.***

Keywords: Gold Prices, Time Series Forecasting, Prophet Model, ARIMA, SARIMA Exponential Smoothing, Train-Test Split, Data Preprocessing, Forecast Accuracy, Performance Metrics (MAE, RMSE, MAPE), Trends, Seasonality, Stationarity.

1. Introduction

1.1 Importance of Gold in Financial Markets

Gold has been a symbol of dignity, wealth and prestige of a family for thousands of years. Beyond that, gold acts as a financial asset and an indicator of a good economy of the country, contributing to the GDP of countries like India. Gold acts as a hedge against inflation i.e, it maintains or increases its value over time, offsetting the effects of inflation and making it a reliable and resilient long-term investment. It is also regarded as a safe-haven asset i.e, investors tend to move their capital into gold to preserve wealth at the time of economic uncertainty. Gold acts as a universal currency in times of crisis, when paper currencies lose trust.

1.2 Objectives of the Study

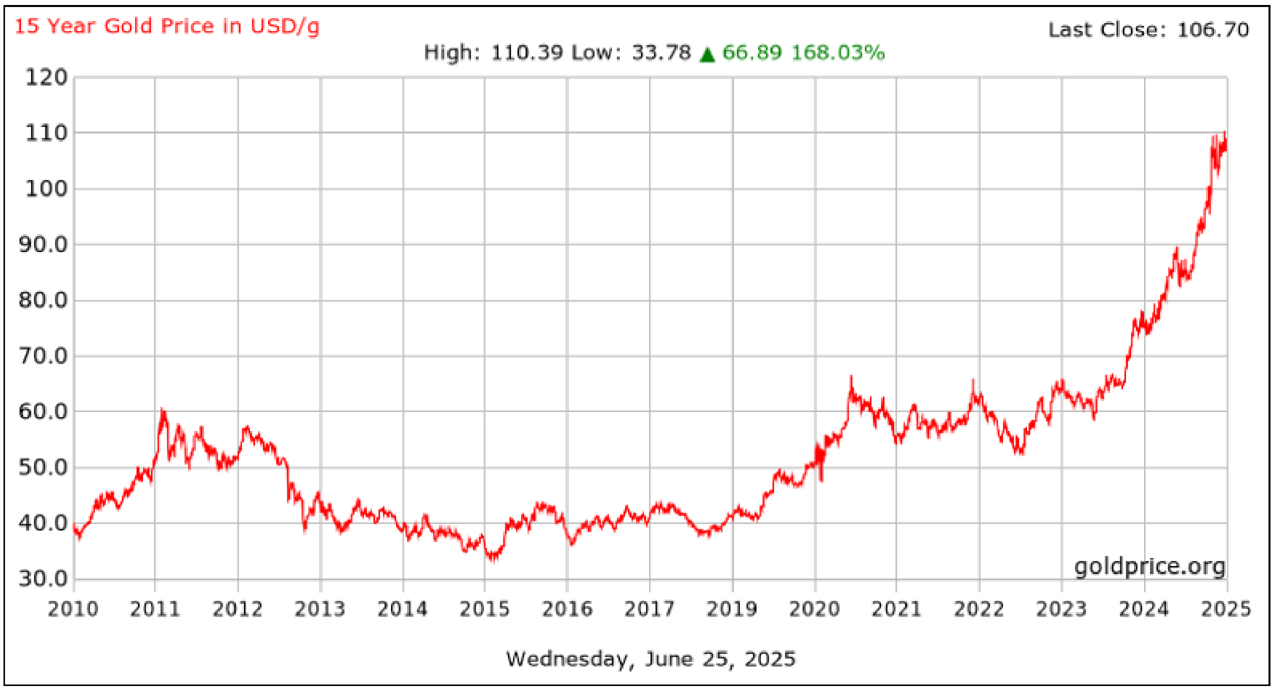
The primary objective of this study is to analyze the trends and patterns in gold prices in the past and develop a time series forecasting model to predict future price trends over the next five years. Given the intrinsic volatility in the value of gold, the study aims to provide insights that can help investors, who view gold as a safe-haven asset, identify optimal time of investment and enhance their decision-making.

1.3 Future Scope of Research

Future work can expand the model by incorporating macroeconomic indicators such as inflation, exchange rates, and global gold reserves to improve forecasting accuracy. Additionally, exploring ensemble models or deep learning approaches like LSTM may yield better long-term results.

2. Literature Review

2.1 Historical Trends in Gold Prices



Graph by: https://goldprice.org/gold-price-history.html

Between 2010 and 2020, the average annual gold price in India saw a significant rise. In 2010, the lowest average was recorded at ₹18,500 per 10 grams, while by 2020, it had increased to an average of ₹48,651 per 10 grams.

2.2 Factors Affecting Gold Prices

- Over the past decades, gold prices have exhibited significant fluctuations due to macroeconomic and geopolitical conditions, the pace of inflation, the amount of reserves, currency fluctuations, supply and demand considerations, and the cost of mining and refining gold.
- Gold prices have a positive correlation with its demand. When inflation is high, gold prices tend to rise as investors turn towards gold as they look for a safe-haven asset to protect their purchasing power and as an inflation hedge against the loss of value in traditional investments like bonds or savings. Gold prices rose in 1970 due to high inflation and economic uncertainty.
- Gold is priced in USD globally. When the dollar strengthens, gold becomes more expensive in other currencies and vice versa.
- During high geopolitical tensions gold prices tend to rise as investors look for a hedge against uncertainty. In the 2000s, gold prices rose steadily driven by geopolitical tensions and financial crises. In 2011, prices peaked due to global economic concerns, then declined as conditions stabilized. In the 2020s, gold prices reached new highs in response to the COVID-19 pandemic and subsequent economic uncertainties.
- Usually gold and stock markets follow a negative correlation. When stocks fall, gold often rises because investors shift towards safer assets such as gold. However, in the early 2020s (especially around 2020–2022), both gold and stock markets rose at the same time because during a global pandemic, COVID19, there was economic uncertainty, therefore, investors turned towards gold thereby increasing its price.
- Regional conflicts in the Middle East and the Russia-Ukraine war in the early 2020s was a period of global uncertainty which contributed to an increase in gold prices.

2.3 Previous Models Used to Predict Gold Prices and Their Drawbacks

2.3.1 Linear Regression Models-

- Use Period: 1970s- 2000s
- Application: To predict gold prices by assuming a constant linear relationship between gold prices and economic factors like inflation, demand and interest rates.
- Limitation: Gold markets are nonlinear, affected by global uncertainty and geopolitical shocks that linear models can't handle.

2.3.2 AR (Auto-Regressive Model)

- Use Period: 1970s – Present
- Application: Predicts future gold prices based on past price values.
- Limitation: Ignores external factors like inflation or geopolitical events; assumes past price patterns will persist.

2.3.3 MA (Moving Average Model)

- Use Period: 1970s – Present
- Application: Models gold prices using the average of past forecast errors.
- Limitation: Oversimplifies price movements and fails to capture market complexities.

2.3.4 ARIMA (Auto-Regressive Integrated Moving Average)

- Use Period: 1980s- 2010s
- Application: Made predictions based on past price patterns and autocorrelation
- Limitation: It assumed stationarity and stable trends.

2.3.5 SARIMA (Seasonal Auto-Regressive Integrated Moving Average)

- Use Period: 1990s – Present
- Application: Forecasts gold prices with seasonal patterns in addition to trend and noise components.
- Limitation: Assumes seasonality in gold prices, which may not always be present or significant.

2.3.6 SES (Single Exponential Smoothing)

- Use Period: 1950s – Present
- Application: Forecasts gold prices by giving more weight to recent observations.
- Limitation: Ineffective for data with trends or seasonal patterns; assumes a constant level over time.

2.3.7 DES (Double Exponential Smoothing /Holt's Linear Trend Method)

- Use Period: 1960s – Present
- Application: Extends SES by incorporating a trend component, making it more suitable for data with upward or downward movement like trending gold prices.
- Limitation: Fails when data has strong seasonality; cannot adjust for recurring annual or monthly cycles in demand or prices.

2.3.8 Holt-Winters Exponential Smoothing (Triple Exponential Smoothing)

- Use Period: 1970s – Present
- Application: Models level, trend, and seasonality in gold prices; can handle complex patterns in time series data.
- Limitation: Assumes seasonal patterns repeat consistently; may underperform during irregular economic shocks or structural breaks like financial crises.

3. Research Methodology

3.1 Time Series Analysis

Time series analysis is a way of analysing a sequence of observations collected through repeated measures of time over a very long period of time. It helps you understand the change in the past and present and what we can predict about the future. The basic aim is to summarize and study casual trends and patterns across a dataset(descriptive); study the impact of a single event (explanatory); and to forecast future trends and behaviours using historical data (predictions).

3.2 Data Collection

The data for this research has been collected from a publicly available dataset on Kaggle. It contains historical gold prices records over a span of 5 years, including daily values for attributes such as Open, High, Low, Close prices, and dates. https://www.kaggle.com/datasets/nisargchodavadiya/daily-gold-price-20152021-time-series -link to dataset

The data was chosen due to its reliability, comprehensiveness, and suitability for time series forecasting tasks. It was downloaded in CSV format and includes information that allows for exploration of seasonality, trends, and price volatility over time.

3.3 Data Cleaning and Processing

To ensure the integrity and accuracy of forecasting results, data preprocessing is a crucial step in time series analysis. Each aspect of this phase is to prepare the dataset for robust and reliable model training and evaluation.

3.3.1 Handling Missing Values

Time series models require a continuous and consistent sequence of observations. Missing values are problematic because they disrupt the evaluation. Missing data is handled by imputation, i.e., forward fill (replacing the missing values with the last known value), backward fill (replacing the missing values with the next known value), linear interpolation (estimating missing values by drawing a straight line between the nearest known values before and after the gap) or removal.

- If missing values are isolated and sparse, imputation is generally safe.
- If only a few values are missing, linear interpolation or forward fill are appropriate to use.
- If a large part of the data is missing, especially in significant segments, the data may be unreliable for modeling.
- If the missing data occurs in a non-target or less relevant feature, dropping the column altogether is safe.

In my dataset, there were null columns that served no purpose and were therefore dropped. There were no missing values in critical columns such as the target variable.

3.3.2 Outlier Detection and Treatment

Outliers in time series can result from anomalies, data entry errors, or true rare events. Traditional models like ARIMA and exponential smoothing are sensitive to outliers.
They can be handled using several methods:

- **Winsorization** involves capping extreme values at a defined percentile (e.g., 5th and 95th percentiles). This limits their influence without removing them entirely and is useful when the outliers are valid. It can distort actual trends or dynamics.
- **Removal based on statistical thresholds** often uses the Interquartile Range (IQR) method, where values lying beyond 1.5 times the IQR from the first or third quartile are considered outliers. This is commonly used when the outliers are likely due to error or are inconsistent with the rest of the data.
- **Transformation techniques** like logarithmic or square root transformation compress the scale of the data and reduce the disproportionate impact of outliers. These are effective when data is highly skewed. It does not remove any outliers, compresses all values proportionally and stabilizes variance, which improves model assumptions and forecasting accuracy.

I examined the dataset for outliers using the IQR method, but due to the presence of numerous outliers, reflecting actual market volatility in gold prices, it was not practical to remove them without losing valuable information. Therefore, no outliers were dropped.

Code for Checking Outliers

```
# IQR method to check for outliers in 'Close_Diff'
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Identify outliers
outliers = data[(data < lower_bound) | (data > upper_bound)]
```

Output:
Number of outliers: 1258

3.3.3 Date Parsing and Indexingi

Date-time formatting and proper indexing are essential to effectively capture temporal dependencies, trends, seasonal patterns, and maintain chronological order. The 'Date' column must be parsed into a recognized datetime format and set as the DataFrame index.

I ensured that the 'Date' column was converted into Python's datetime format and set as the index.

Code for Date Parsing and Indexing

```
# Convert 'Date' to proper datetime format (with UTC if needed)
data['Date'] = pd.to_datetime(data['Date'], utc=True)

# Remove timezone info but keep it as datetime64 (not .date)
data['Date'] = data['Date'].dt.tz_convert(None)

# Set 'Date' as index for time series operations
data.set_index('Date', inplace=True)
```

3.3.4 Normalization

Scaling is important when using models sensitive to input ranges. Without normalization, features with larger numeric values may disproportionately influence the model.

Min-Max Scaler- Rescales the data to a fixed range [0, 1]. The minimum becomes 0 and the maximum becomes 1. It is sensitive to outliers.

$$\mathbf{x_{scaled} = (x - x_{min}) / (x_{max} - x_{min})}$$

where,

- $\mathbf{x_{scaled}}$ is the standardized value.
- $\mathbf{x}$ is the actual value.
- $\mathbf{x_{min}}$ is the minimum value of that attribute.
- $\mathbf{(x_{max} - x_{min})}$ is the range.

Standard Scaler- Transforms the data to have mean equal to 0 and standard deviation equal to 1. It is sensitive to outliers.

$$\mathbf{x_{scaled} = (x - \mu) / \sigma}$$

where,

- $\mathbf{x_{scaled}}$ is the standardized value.
- $\mathbf{x}$ is the actual value.
- $\boldsymbol{\mu}$ is the mean of the feature.
- $\boldsymbol{\sigma}$ is the standard deviation of the feature.

Robust Scaler- Robust Scaler instead uses the median and interquartile range (IQR) to scale the data, making it robust to outliers, i.e., it is not heavily affected by extreme values.

$$\mathbf{x_{scaled} = (x - Median) / IQR}$$

where,

- $\mathbf{x_{scaled}}$ is the standardized value.
- $\mathbf{x}$ is the actual value.
- **Median** is the medium of the feature.
- **IQR** is the interquartile range of the feature.

I used the Robust Scaler. Unlike Min-Max Scaler or Standard Scaler, which can be heavily influenced by outliers, Robust Scaler uses the interquartile range to scale features. This makes it effective for gold prices time series, where sharp spikes and drops are common, by reducing the influence of outliers.

Code for Robust Scaler

```
scaler = RobustScaler()
scaled_values = scaler.fit_transform(data1[numerical_cols])
```

Gold Prices Scaling- excel sheet

3.4 Concept of Time Series

3.4.1 Stationarity

A time series is said to be stationary if its statistical properties such as mean, variance, and autocovariance, do not change over time.

Why is Stationarity Important?

It makes the data more predictable and easier to model accurately. Without stationarity, any relationships or patterns observed may be unreliable, as the behavior of the series could shift unpredictably.

- Models that require stationarity - ARIMA, SARIMA, and Simple Exponential Smoothing (SES)
- Models that handle non-stationarity internally- Holt's Linear Trend Method, Holt-Winters Method, Prophet

Test for Stationarity- A unit root in a time series indicates that the series is non-stationary – its statistical properties change over time. To check for a unit root, a simplified AR(1) model without the intercept **c** is used. It captures the persistence of the previous value.

Mathematical equation:

$Y_t = \rho Y_{t-1} + \epsilon t$

where,

- **Yt** is the current observation.
- Y_{t-1} is the previous observation.
- **ρ** is the autoregressive coefficient.
- **ϵt** represents the error term at time t (white noise).

Case 1: $|\rho| < 1$

The process is stationary, each value depends on the past but is pulled back toward the mean over time. This ensures that the effect of a shock **ϵt** diminishes as time passes and the variance of Y_t stabilizes to a finite value.

Case 2: $\rho = 1$

The process has a unit root and becomes:

$Y_t = Y_{t-1} + \epsilon t$

This is a random walk. Shocks **ϵt** do not fade as time passes, instead they accumulate and the series wanders without settling around a fixed mean and variance. Therefore, it is non-stationary.

Case 3: $|\rho| > 1$

The series is explosive.

$Y_t = \rho^t Y_0 + {}_{i=1}\Sigma^t \rho^{t-i} \epsilon t$

ρ grows exponentially with t so both the mean and variance blow up as time increases. Even small shocks are magnified rapidly.

There are three tests to check for stationarity.

a) Augmented Dickey-Fuller (ADF) Test

It checks for unit root. It finds out if the value today is just a continuation of yesterday's value (a random walk), or if it tends to return to a stable mean over time (stationarity).

Hypothesis

Null Hypothesis (H_0) is the default assumption ⇒ Non-Stationarity

Alternate Hypothesis (H_1) is what you want to test for ⇒ Stationarity

Mathematical Equations:

$\Delta y_t = \alpha + \beta t + \gamma\, y_{t-1} + {}_{i=1}\sum^p \delta_i \Delta y_{t-i} + \epsilon_t$

Where,

- Δy_t is the first difference of the series.
- **α** is the constant term (drift).
- **βt** is the time trend.
- **γ** is the coefficient of the lagged level term y_{t-1}. $\gamma = p-1$.
- δ_i is the coefficients of lagged differenced terms.
- **p** is the number of lag terms added to account for autocorrelation.
- ϵ_t is the error term.

Adding a drift term or intercept **α** allows us to check for stationarity around a constant level, not necessarily zero. If the data has a long-term upward or downward slope, adding a trend term **βt** captures that. This helps us test whether the residual series is still stationary, even if the level changes over time. By adding lagged values of **Δy**, we capture the autocorrelation shown by residuals (the difference between actual and predicted values) and remove it, making the test statistically sound. If autocorrelation exists, it means that the error today is influenced by the error in the past.

If $\gamma < 0$, then the series is stationary.

b) Kwiatkowski–Phillips–Schmidt–Shin (KPSS) Test

The KPSS test is a statistical test used to check whether a time series is stationary around a mean (level) or a deterministic trend.

Hypothesis

Null Hypothesis (H_0) ⇒ Stationarity

Alternate Hypothesis (H_1) ⇒ Non-Stationarity

Mathematical Equation:

The KPSS model breaks a time series into three parts:

$y_t = r_t + \delta_t + \epsilon_t$

Where,

- y_t is the observed time series at time **t**.
- δ_t is the deterministic trend.
- ϵ_t is the stationary error term (white noise).
- r_t is the random walk, defined as:
  $(r_t = r_{t-1} + u_t)$
  where,
  u_t is the stationary error in the random walk component.

We need to test whether the random walk component r_t is present. If it's not (i.e., variance of u_t is zero), the series is considered stationary.

KPSS test is used to confirm whether the resulting series is now stationary.

$$\mathbf{KPSS = (1 / T^2)\ {}_{t=1}\sum^T S_t^2 / \sigma^2}$$

Where,

- S_t is the cumulative sum of residuals from the regression of the time series y_t, either on a constant (if you're checking for level stationarity), or on a trend line (if you're checking for trend stationarity).
- S_t^2 is to emphasize the magnitude of deviation.
- σ^2 is the estimate of long-run variance of residuals.
- **T** is the total number of observations.

${}_{t=1}\sum^T S_t^2$ measures the drift in the residuals over time. A higher value of S_t^2 means that on average, the residuals have strayed away from zero consistently. σ^2 scales this by how much variability we expect (i.e., the long-run variance). Dividing by T^2 normalizes the test statistic.

A lower value of S_t^2 means the series is stationary.

c) Phillips–Perron (PP) Test

The PP test is used to test for unit roots in time series data, just like the ADF test. However, it takes a non-parametric approach, i.e., it keeps the model simple but fixes the problems after running the regression by modifying the calculated t-statistic, instead of introducing lagged difference terms, to handle:

- **Autocorrelation** (when errors are correlated over time)
- **Heteroscedasticity** (when error variance is not constant over time)

This makes it more robust when the error terms are not well-behaved.

Hypothesis
Null Hypothesis (H_0) ⇒ Non-Stationarity
Alternate Hypothesis (H_1) ⇒ Stationarity

Mathematical Formula:
$\Delta y_t = \alpha + \gamma y_{t-1} + \epsilon_t$

Where:

- Δy_t is the change in the series (first difference).
- y_{t-1} is the lagged level of the series.
- α is the constant or drift term.
- γ is the coefficient used to test for unit root.
- ϵ_t is the error term.

If $\gamma < 0$, then the series is stationary.

Code for Test for Stationarity

```
adf_result = adfuller(data_diff)
if adf_result[1] <= 0.05:
  print("ADF Result: Dataset is stationary")
else:
  print("ADF Result: Dataset is not stationary")

kpss_result = kpss(data_diff, regression='c', nlags='auto')
if kpss_result[1] <= 0.05:
  print("KPSS Result: Dataset is not stationary")
else:
  print("KPSS Result: Dataset is stationary")
```

Output:

```
ADF Statistic: -26.054054485962343
p-value: 0.0
ADF Result: Dataset is stationary
KPSS Statistic: 0.18464855795378315
p-value: 0.1
KPSS Result: Dataset is stationary
```

The p-value for ADF test is < 1, therefore, the dataset is stationary. To validate this finding, KPSS test was also conducted. The p-value for KPSS test is also < 1, therefore, the dataset is stationary.

3.4.2 White Noise

It is a completely random error with no predictable structure. White noise in time series refers to a sequence of random variables that have a mean of zero, constant variance and are uncorrelated with each other.

Code for Noise Detection

```
acf_vals = acf(residual.dropna(), nlags=30)
significant_autocorr = any(abs(acf_vals[1:]) > 0.2)
is_pure_noise = not significant_autocorr
```

3.4.3 Trends

A long-term direction in which the data is moving. In the case of gold, the gradual increase or decrease in the price of gold due to fluctuations in demand and supply.

Code for Trend Detection

```
trend = result.trend.dropna()
has_trend = trend.var() > 0.001
```

3.4.4 Seasonality

A short-term repetition of patterns at fixed periods. In India and certain other countries, gold prices increase sharply during festive seasons.

Code For Seasonality Detection

```
# Decompose the time series
result = seasonal_decompose(close_series, model='additive',
period=30)
seasonal = result.seasonal.dropna()
has_seasonality = seasonal.var() > 0.001
```

3.4.5 Cyclic Variations

Recurring, long-term fluctuations in the data, typically lasting more than a year. Gold prices rise during global crises (e.g., 2008 financial crisis, COVID-19) when investor confidence in other markets falls, and decline when economic confidence and interest rates rise.

Code for Cyclic Variations Detection

```
rolling_mean                                                    =
close_series.rolling(window=90).mean().dropna()
has_cyclicity = rolling_mean.var() > 0.001
```

3.4.6 Noise

Unpredictable or irregular variations in the data that cannot be attributed to trend, seasonality, or cycles. Sharp changes in the price of gold due to sudden geopolitical events.

Models	Trend	Seasonality	Cyclic Variation	Noise	White Noise
ARIMA	✅	❌	✅ (if periodic)	❌	❌
SARIMA	✅ (with seasonality)	✅ (with trend)	❌	❌	❌
Simple Exponential Smoothing	❌	❌	❌	❌	❌
Holt's Linear Trend Method	✅	❌	❌	❌	❌
Holt-Winters Method	✅ (with seasonality)	✅ (additive) (with trend-additive and multiplicative)	❌	❌ ❌	❌ ❌
Prophet	✅ (with seasonality or cyclic variation)	✅ (with trend)	✅	❌	❌

3.5 Data Analysis

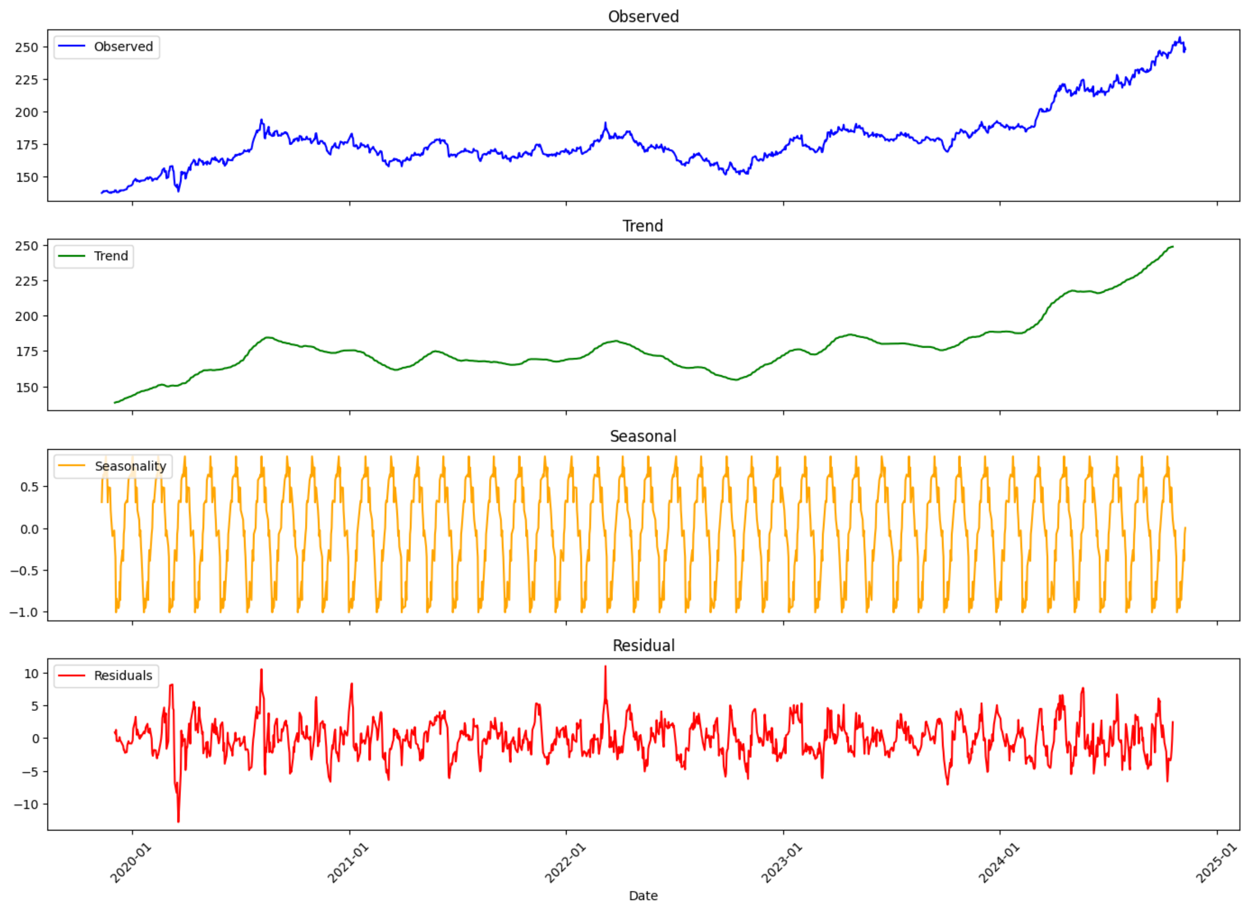
3.5.1 Analysing Time Series Components

It is very important to analyse your data to understand which model will work best on your dataset. I analyzed the structure and components of the time series data - trend, seasonality, cyclic patterns, and residual noise - to better understand the behavior of gold prices over time and work out the best model for my data.

Time Series Components in My Data

- Trend Present : True
- Seasonality Present : True
- Cyclicity Present : True
- Pure Noise : False

Graph



3.5.2 Correlation Matrix

Additionally, I constructed a correlation matrix to investigate the linear relationships between different features in the dataset. This helped identify which variables may be influential in predicting gold prices and ensured there was no multicollinearity among the predictors, i.e., two or more independent variables (features) in your dataset are highly correlated with each other which can be a problem in predictive modeling.

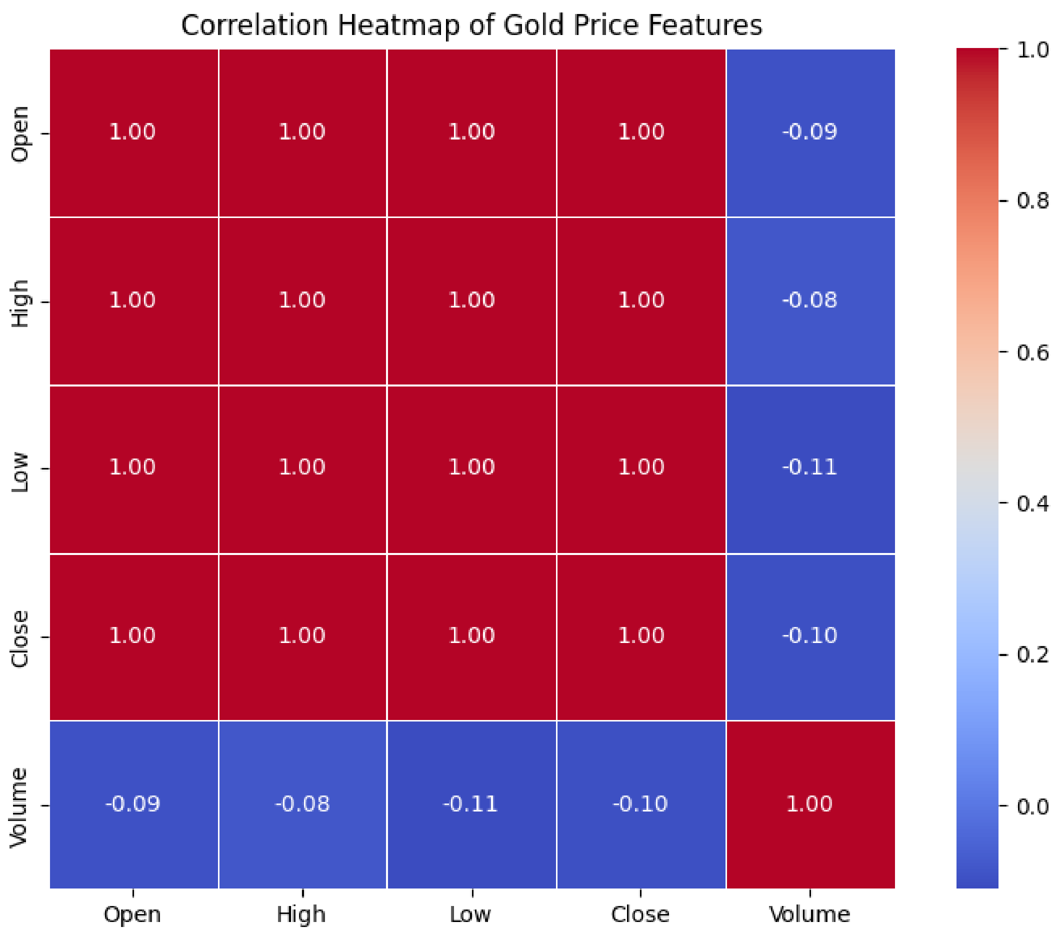
Output:

```
Correlation matrix:
            Open      High       Low     Close    Volume
Open    1.000000  0.999473  0.999334  0.998779 -0.094791
High    0.999473  1.000000  0.999232  0.999433 -0.083908
Low     0.999334  0.999232  1.000000  0.999500 -0.110670
Close   0.998779  0.999433  0.999500  1.000000 -0.098528
Volume -0.094791 -0.083908 -0.110670 -0.098528  1.000000
```

Code for Computing Correlation Matrix

```
#Define the numerical columns
numerical_cols = ['Open', 'High', 'Low', 'Close', 'Volume']

#Calculate the correlation matrix
correlation_matrix = data1[numerical_cols].corr()
```



The relationship among Open, High, Low and Close is a strong positive correlation as the day's opening, highest, lowest and closing price are all strongly linked. There is a weak negative correlation between Volume and Price columns (Open, High, Low, Close). There is a slight tendency that when volume increases, prices might slightly fall.

3.6 Train-Test Split

Before building forecasting models, the dataset was divided into a training set and a testing set to evaluate the performance of the models on unseen data. The training set is used to fit the models, while the testing set is reserved to assess forecasting accuracy. Since time series models rely heavily on learning temporal patterns, it is important that the training set covers a larger portion of the data. Therefore, my dataset was split in an 80:20 ratio—80% for training the model and 20% for testing its predictive accuracy.

Code for Train-Test Split

```
# Get split index
split_index = int(len(data) * 0.8)

# Split the data
train = data.iloc[:split_index]
test = data.iloc[split_index:]
```

3.7 Forecasting Models

3.7.1 ARIMA Model

ARIMA model stands for Autoregressive Integrated Moving Average Model. It is a forecasting algorithm that only uses the information in the past values of the time series to predict the future values.

Purpose: It works best with univariate time series data, especially when the data is non-seasonal and can be made stationary, and the objective is to model future values based only on past observations and past errors.

Order Parameters:
These models are specified by three order parameters: (p, d, q)
where,

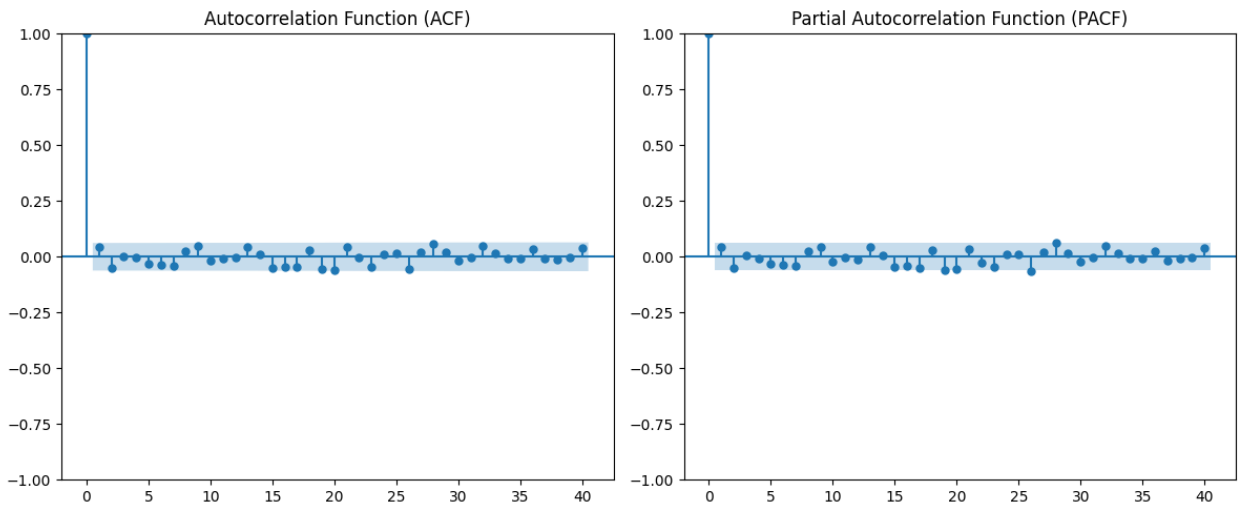
- p is the order of the AR term. It is calculated by PACF plot.
- d is the number of times the observations are differenced to make the time series stationary
- q is the order of the MA term. It is calculated by ACF plot.

ACF (Autocorrelation Function)
ACF measures how correlated the current value of a time series is with its past values (lags). It helps detect repeating patterns, seasonality, and whether the series is stationary. ACF is mainly used to identify the MA (Moving Average) component. Significant spikes indicate lagged error correlations.

PACF (Partial Autocorrelation Function)
PACF measures the correlation between the current value and a lag, after removing the effect of intermediate lags. PACF helps identify the AR (Autoregressive) component. Significant spikes indicate direct dependence on past observations.

Code for plotting ACF and PACF

```
plot_acf(series, lags=40, ax=plt.gca(), alpha=0.05)
plot_pacf(series, lags=40, ax=plt.gca(), alpha=0.05, method='ywm')
```



The ACF and PACF plots for the training data show that after lag 0, all spikes lie within the confidence interval (blue shaded region). This indicates that there is no significant autocorrelation or partial autocorrelation in the data, suggesting that the series is already stationary or behaves close to white noise. Consequently, the AR and MA components are likely very weak or negligible, meaning p and q values should be close to zero. Therefore, the order of ARIMA suggested by the plots is (0,0,0).

Code for finding the Best ARIMA Model via Hit and Trial

```
for p in range(0, 4):
  for q in range(0, 4):
    try:
      model = ARIMA(data_diff, order=(p, 0, q))
      model_fit = model.fit()
      aic = model_fit.aic
      if aic < best_aic:
        best_aic = aic
        best_order = (p, 0, q)
        best_model = model_fit
    except:
      continue
```

Output
Best ARIMA Order: (2, 0, 2)

I used the order of ARIMA as (2, 0, 2) because the code performs an objective and data-driven evaluation, unlike ACF and PACF plots.

Mathematical Equation:
In mathematical terms ARIMA(p,d,q) model can be expressed as:

$$\mathbf{Y_t' = c + \phi_1 Y_{t-1}' + \phi_2 Y_{t-2}' + \ldots\ldots + \phi_p Y_{t-p}' + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots\ldots + \theta_q \epsilon_{t-q}}$$

- Y_t' is the differenced and stationary time series at time t.
- ϕ_i are AR coefficients.
- c is constant or mean of the differenced series.
- θ_1 are MA coefficients.
- ϵ_t is white noise.

AR stands for Autoregression. It is a regression model that signifies the dependence of the current observation on its previous values. The value of p refers to the use of past values in the regression equation.

Mathematical equation:

$$\mathbf{Y_t = c + {}_{j=1}\Sigma^p\ \phi_j Y_{t-j} + \epsilon_t}$$
OR
$$\mathbf{Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \epsilon_t}$$
where,
- **Y_t** is the current observation.
- **Y_{t-j}** are the past values
- **c** is a constant (intercept).
- **ϕ_i** are the autoregressive parameters.
- **ϵ_t** represents the error term at time t (white noise).

Derivation:

Step 1: Assume that the current value of a time series Y_t is a linear function of its past values and a random error term ϵ_t. This leads to the general autoregressive model of order p:

$$Y_t = f(Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}) + \epsilon_t \qquad \dots(i)$$

Step 2: Assume f as a linear combination of past values. A linear combination is a mathematical expression made by multiplying each term by a constant or a coefficient and adding the results and a constant. Therefore,

$$f(Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}) = c + \phi 1Yt-1 + \phi 2Yt-2 + \dots + \phi pYt-p \qquad \dots(ii)$$

Step 3: Substitute the value of (ii) in (i)

$$Yt = c + \phi 1Yt-1 + \phi 2Yt-2 + \dots + \phi pYt-p + \epsilon t$$

I stands for Integration. It indicates the number of times the observations are differenced to transform a non-stationary time series into a stationary one by differencing consecutive observations. Differencing involves the subtraction of the current values of a series with its previous values d number of times.

Mathematical Equation:

$$Yt' = Yt - Yt-1$$

Differencing is done by subtracting the previous value from the current value. If needed, this differencing is repeated d times until the series becomes stationary.

MA stands for Moving Average. It indicates the dependence of the current observation on the previous forecast errors. The value of q refers to the past values of the process that influence the current value.

Mathematical Equation:

$$Yt = c + \epsilon t + \theta 1\epsilon t-1 + \theta 2\epsilon t-2 + \dots + \theta q\epsilon t-q$$

Derivation:

Step 1: Assume that the current value of the series depends on past *error terms* instead of past values of Y.

$$Y_t = \epsilon t + f(\epsilon t-1 + \epsilon t-2 + \dots + \epsilon t-q) \qquad \dots(i)$$

Step 2: Take the linear combination of past forecast errors.

$$f(\epsilon t-1 + \epsilon t-2 + \dots + \epsilon t-q) = \mu + \theta 1\epsilon t-1 + \theta 2\epsilon t-2 + \dots + \theta q\epsilon t-q \qquad \dots(ii)$$

Step 3: Substitute the value of (ii) in (i)

$$Yt = c + \epsilon t + \theta 1\epsilon t-1 + \theta 2\epsilon t-2 + \dots + \theta q\epsilon t-q$$

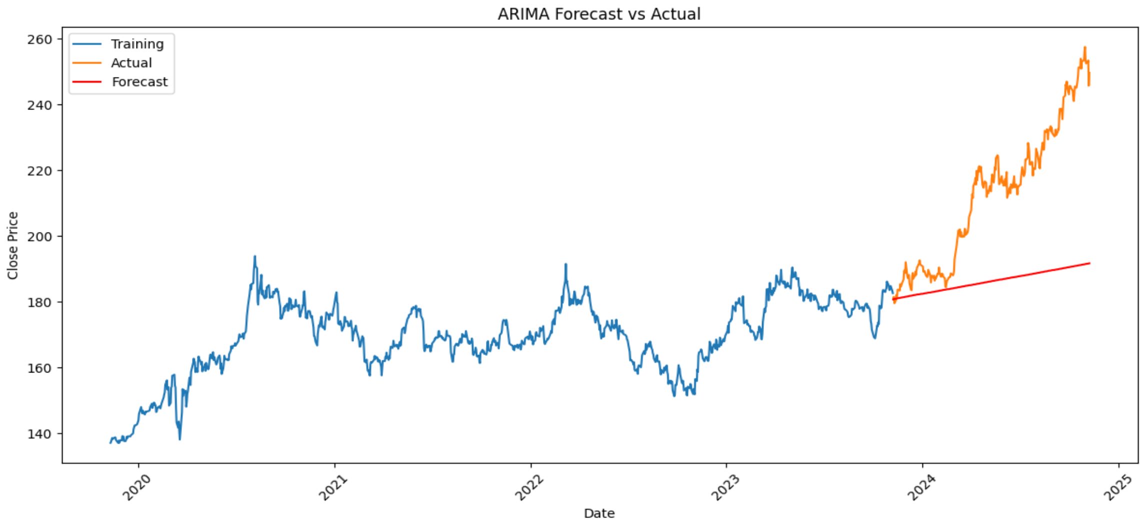
Limitations of ARIMA Model:

- ARIMA does not handle seasonality well for that, SARIMA is used.
- It requires the series to be made stationary, which can sometimes remove meaningful trends.
- There is complex parameter tuning.
- It may underperform on non-linear data as it assumes linear relationships.
- It cannot handle irregular seasonality.
- It is sensitive to outliers.
- It is not suitable for long-term forecasting.

Code for ARIMA Model

```
# Train the model
model = ARIMA(train, order=(2, 0, 2))
model_fit = model.fit()

# Forecast in differenced space
forecast_diff = model_fit.forecast(steps=len(test))
forecast_original.index = date_test
true_test = data['Close'][split_index:split_index + len(forecast_original)]
```



The ARIMA model performed poorly in forecasting due to its sensitivity to outliers, which distorted its ability to capture the underlying patterns in the data.

3.7.2 SARIMA Model

SARIMA Model stands for Seasonal Autoregressive Integrated Moving Average Model. It extends the ARIMA model by incorporating seasonality

Purpose: It is suitable for time series data with repeating seasonal patterns. Its objective is to forecast data which shows predictable seasonal effects that ARIMA alone cannot capture.

Order Parameters:
These models are specified by three order parameters: (P, D, Q, s)

Where,
- P is the order of the SAR term.
- D is the number of seasonal differences needed to make the series stationary.
- Q is the order of the seasonal SMA term.
- s is the order of length of the seasonal cycle.

Code for finding the Best SARIMA Model via Hit and Trial

```
for p in p_values:
  for d in d_values:
      for q in q_values:
          for P in P_values:
              for D in D_values:
                  for Q in Q_values:
                      try:
                          model = SARIMAX(
                              train,
                              order=(p, d, q),
                              seasonal_order=(P, D, Q, s),
                              enforce_stationarity=False,
                              enforce_invertibility=False
                          )
                          results = model.fit(disp=False)
                          if results.aic < best_aic:
                              best_aic = results.aic
                              best_params = ((p, d, q), (P, D, Q, s))
                      except Exception:
                          continue
```

Output
Best SARIMA Order: ((0, 0, 2), (0, 0, 1, 7))

Mathematical Equation:
$$(1 - \phi_1 B)(1 - \Phi_1 B^s)(1 - B)^d(1 - B^s)^D\, y_t = (1 + \theta_1 B)(1 + \Theta_1 B^s)\, \epsilon t$$
where,
- **B** is the backward shift operator, representing the lag operator.
- ϕ_1 is the non seasonal autoregressive coefficient.
- Φ_1 is the seasonal autoregressive coefficient.
- **s** is the seasonal period.
- y_t is the observed time series at time t.
- θ_1 is the non seasonal moving average coefficient.
- Θ_1 is the seasonal moving average coefficient.
- ϵt represents the error term at time t (white noise).

Derivation:
Step 1: Apply differencing to make the data stationary
$$y_t' = (1 - B)^d(1 - B^s)^D\, y_t$$
where,
- $B^k y_t = y_{t-k}$ is the backshift operator which shifts the series back by k periods
- $(1 - B)$ is the non seasonal differencing
- $(1 - B^s)$ is the seasonal differencing

Step 2: Multiply the differencing terms with AR and MA parts.
- Non seasonal AR- $(1 - \phi_1 B)$
- Seasonal AR- $(1 - \Phi_1 B^s)$
- Non seasonal MA- $(1 + \theta_1 B)$
- Seasonal MA- $(1 + \Theta_1 B^s)$

Step 3: Combine all the components.
$$(1 - \phi_1 B)(1 - \Phi_1 B^s)(1 - B)^d(1 - B^s)^D\, y_t = (1 + \theta_1 B)(1 + \Theta_1 B^s)\, \epsilon t$$

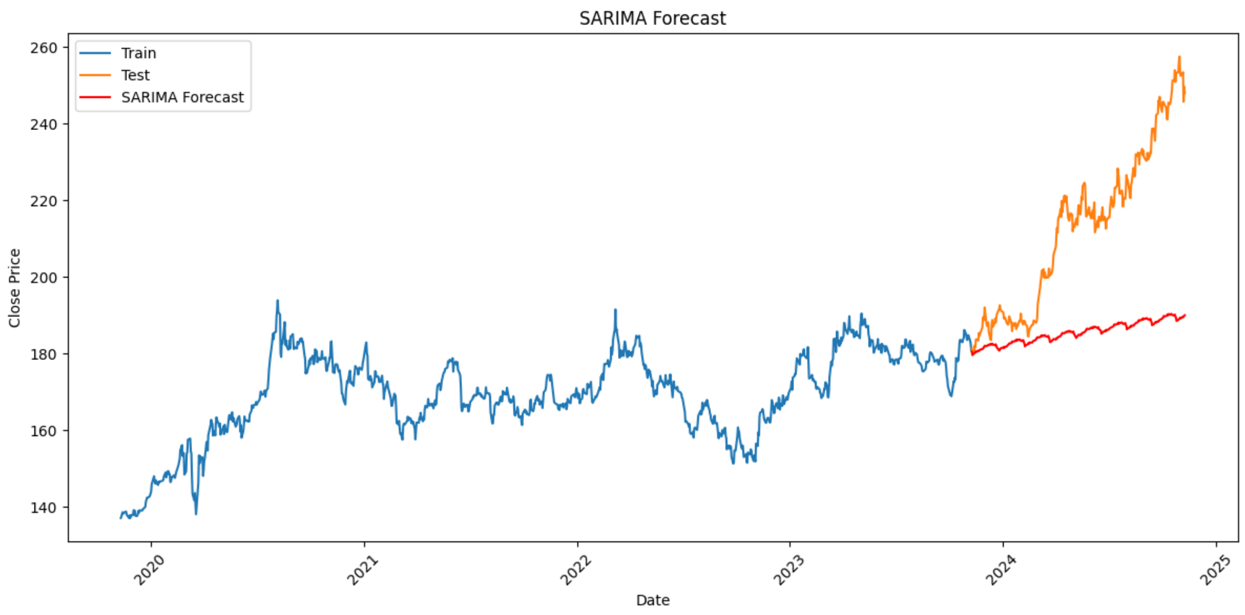
Limitations of SARIMA Model:
- It requires the series to be made stationary, which can sometimes remove meaningful trends.
- There is complex parameter tuning.
- It may underperform on non-linear data as it assumes linear relationships.
- It cannot handle irregular seasonality.
- It is sensitive to outliers.
- It is not suitable for long-term forecasting.

Code for SARIMA Model

```
model = SARIMAX (
  train,
  order=(0, 1, 2),
  seasonal_order=(0, 1, 2, 30),
  enforce_stationarity=False,
  enforce_invertibility=False )

# Fit the model
model_fit = model.fit()

# Forecast
forecast = model_fit.forecast(steps=len(test))
forecast.index = test.index  # To ensure dates match
```



The SARIMA model performed poorly in forecasting due to its sensitivity to outliers, which distorted its ability to capture the underlying patterns in the data.

3.7.3 Exponential smoothing

Exponential smoothing assumes that future patterns will be similar to recent observations. More weight is given to the more recent observations than reduces exponentially as the distance from the observation rises. The more recent the data point, the higher its weight in the forecast. It is especially useful for short-term forecasting when the data shows patterns but no strong seasonality or trend.

Purpose: It is suited for series which are stationary and have random short-term variations but no long-term upward or downward movement. The objective is to smooth out short-term fluctuations in data, capture trends or seasonality, and generate reliable forecasts.

Types: There are three types of exponential smoothing, each suitable for different kinds of time series data.

a. Simple Exponential Smoothing (SES)

This method of forecasting is used when there is no trend or seasonality present in the time series data. In this type of smoothing, the smoothing parameter α (alpha) controls how much weight should be give to each more recent observations as compared to later. The value of α ranges between 0 to 1. It helps balance stability vs sensitivity in forecasts.

- A higher value of α signifies that the gives more weight to recent data. This makes the forecast more sensitive to recent changes, but also more unstable or reactive.
- A lower value of α signifies that more weight is given to past estimates, This makes the forecast more stable, but it reacts slowly to sudden changes.

Mathematical Equation:

$\mathbf{s_t = \alpha x_t + (1 - \alpha)\, s_{t-1}}$

where,

- $\mathbf{s_t}$ is the current smoothed value or the forecast.
- $\mathbf{x_t}$ is the current actual observation.
- $\mathbf{s_{t-1}}$ is the previous smoothed value or the previous forecast.
- $\boldsymbol{\alpha}$ is the smoothing parameter.
- **t** is the time period.

Derivation:

The forecast of the next value is made by combining the current actual observation and the previous smoothed value. A smoothed value is an estimate of the current data point that reduces the effect of noise.

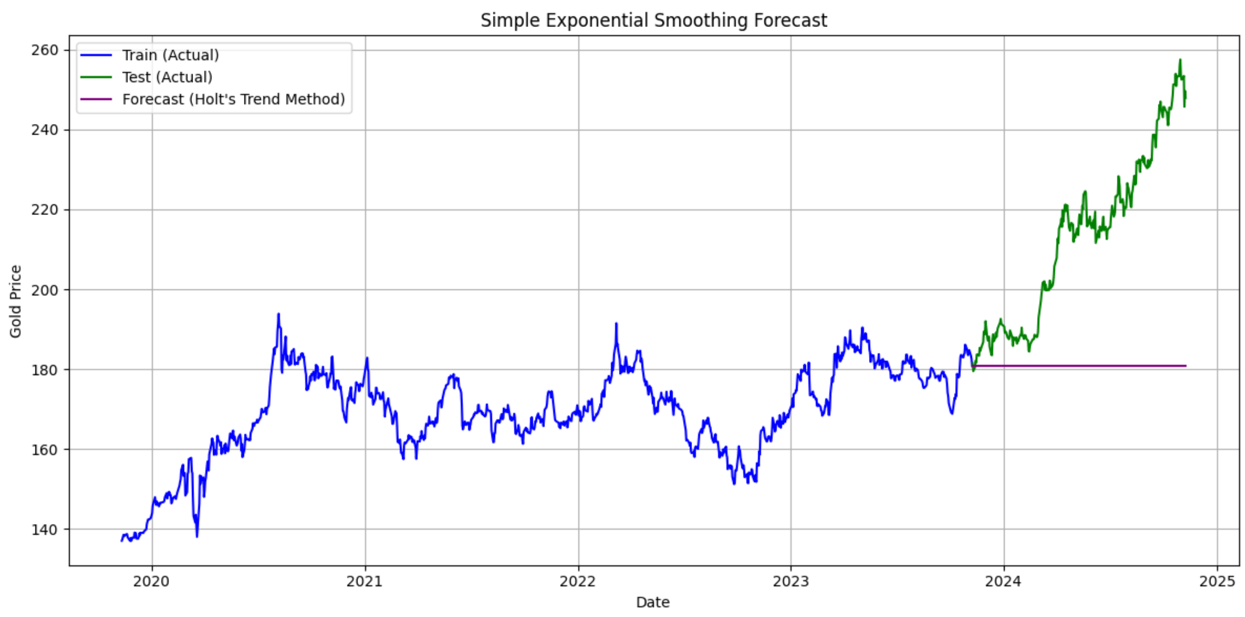
Therefore, by definition:

$\mathbf{s_t = \alpha x_t + (1 - \alpha)\, s_{t-1}}$

This equation shows that current actual observation $\mathbf{x_t}$ is given more weight while the previous forecast $\mathbf{s_{t-1}}$ is given less weight.

Code for Exponential Smoothing: Simple Exponential Smoothing

```
# Fit SES model
model = SimpleExpSmoothing(train, initialization_method='estimated')
model_fit = model.fit(optimized=True)
# Forecast
forecast = model_fit.forecast(steps=len(test))
forecast.index = test.index  # <-- IMPORTANT FIX
```



The SES method assumes that the data has no trend and no seasonality. It focuses only on smoothing past observations to predict a constant average level. In the case of gold price data, which exhibits strong upward trends and fluctuations, SES fails. As shown in the graph, the forecast line remains nearly flat, reflecting the method's limitation in capturing trending or volatile behavior, resulting in poor predictive accuracy**.**

b. Holt's Trend Method

This method of forecasting, also known as double exponential smoothing, is used when there is a trend but no seasonality present in the time series data. It extends single exponential smoothing by also accounting trends in the time series in the data. In this type of smoothing, two parameters α (alpha) and β (beta) are used. β is the trend smoothing parameter that controls how quickly the model responds to recent changes in the trend. A higher value of β gives more weight to recent changes, making the model respond faster to trend shifts. A lower β gives less weight to recent changes, making it smoother and slower to react.

This method can handle both additive and multiplicative trends, i.e., it can handle both linear and exponential increase or decrease in the data.

Mathematical Equations:

i. Level equation

$\mathbf{l_t = \alpha x_t + (1 - \alpha)(l_{t-1} + b_{t-1})}$

Where,

- $\mathbf{l_t}$ is the estimated level at time t.
- α is the smoothing parameter.
- $\mathbf{x_t}$ is the current actual observation.
- $\mathbf{b_{t-1}}$ is the estimated trend at time t-1.

ii. Trend Equation

$\mathbf{b_t = \beta (l_t - l_{t-1}) + (1-\beta) b_{t-1}}$

Where,

- $\mathbf{b_t}$ is the estimated level at time t.
- **β** is the trend smoothing parameter.

iii. Forecast Equation

$\mathbf{\hat{x}_{t+m} = l_t + m b_t}$

Where,

- $\mathbf{\hat{x}_{t+m}}$ is the forecasted value of the time series for time t+m.
- **m** is the number of periods.

Derivation:

Step 1: Assume the forecast is a combination of the current level and trend. Level refers to the estimated base value of the time series at a given time, assuming no trends or seasonality.

$\mathbf{\hat{x}_{t+1} = l_t + b_t}$

Step 2: Combine the current actual observation $\mathbf{x_t}$ with the previous forecasted level $\mathbf{l_{t-1}}$ and trend $\mathbf{b_{t-1}}$. Give more weight to current observations $\mathbf{x_t}$ than past ones $\mathbf{l_{t-1} + b_{t-1}}$.

$$\mathbf{l_t = \alpha x_t + (1 - \alpha)(l_{t-1} + b_{t-1})}$$

Step 3: Compare the current level $\mathbf{l_t}$ with the previous level $\mathbf{l_{t-1}}$ to get an idea of how much the base value is changing.

$$\mathbf{b_t = \beta (l_t - l_{t-1}) + (1-\beta) b_{t-1}}$$

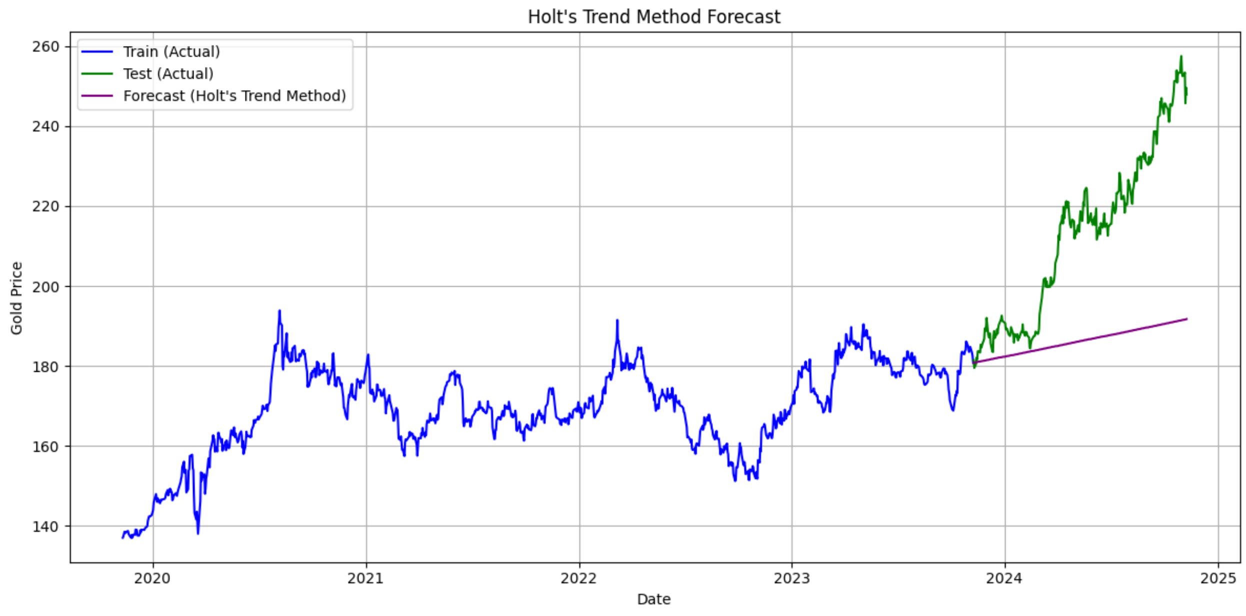
Step 4: For forecasting, the level equation and the trend equation are added.

$$\mathbf{\hat{x}_{t+m} = l_t + m b_t}$$

Code for Exponential Smoothing: Holt's Trend Model

```
#Fit Holt's Trend Method
model = ExponentialSmoothing(
   train,
   trend='add',         # 'add' for additive trend, 'mul' for multiplicative
   seasonal=None,
   initialization_method='estimated'
)
model_fit = model.fit(optimized=True)

# Forecast
forecast = model_fit.forecast(steps=len(test))
forecast.index = test.index  # align forecast with test index
```



The forecast line follows a smooth and gradual upward trend, failing to adapt to the rapid fluctuations in the actual prices. This results in overall poor predictive accuracy for highly volatile market behavior.

c. Holt-Winters Method

This method of forecasting is used when there is trend and seasonality present in the time series data. It extends Holt's trend method by also accounting seasonality in the time series in the data. In this type of smoothing, two parameters α (alpha), β (beta), and γ (gamma) are used. γ is the smoothing parameter for seasonality and controls how quickly the model responds to recent changes in the seasonal pattern. It determines how much weight is given to the most recent seasonal observation. A higher value of γ gives more weight to recent seasonal patterns, making the model respond faster to seasonal adjustment. A lower γ gives less weight to recent seasonal patterns, making it smoother and slower to react.

There are two variations:

- **Additive Seasonality**: used when seasonal fluctuations are approximately constant, regardless of whether the overall level of the series is high or low.
- **Multiplicative Seasonality**: used when seasonal variations grow or shrink proportionally with the level of the series.

Mathematical Equations:

Additive Model:

i. Level equation

$$l_t = \alpha(x_t - s_{t-L}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$

where,

- l_t is the estimated level at time t.
- α is the smoothing parameter.
- x_t is the current actual observation.
- b_{t-1} is the estimated trend at time t-1.
- L is the season length.
- s_{t-L} is the seasonal factor observed exactly one season ago.

ii. Trend Equation

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

Where,

- b_t is the estimated level at time t.
- β is the trend smoothing parameter.

iii. Seasonal Equation

$$s_t = \gamma(x_t - l_t) + (1 - \gamma) s_{t-L}$$

where,

- s_t is the seasonal component at time t.
- γ is the seasonal smoothing parameter.

iv. Forecast Equation

$$\hat{x}_{t+m} = l_t + m b_t + s_{t-L+m}$$

where,

- $\hat{x}_{t+m}$ is the forecasted value of the time series for time t+m.

- **m** is the number of periods.

Derivation:

Step 1: Assume the forecast is a combination of the current level, trend, and seasonal component.

$$\hat{x}_{t+1} = l_t + b_t + s_{t+1-L}$$

Where,
- s_{t+1-L} refers to the same position in the previous seasonal cycle.

Step 2: Combine the de-seasonalized actual observation $x_t - s_{t-L}$ with the previous forecasted level and trend $l_{t-1} + b_{t-1}$. Give more weight to current observations $x_t - s_{t-L}$ than past ones $l_{t-1} + b_{t-1}$.

$$l_t = \alpha(x_t - s_{t-L}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$

Step 3: Compare the current level l_t with the previous level l_{t-1} to get an idea of how much the base value is changing.

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

Step 4: Compare the seasonally adjusted residual $x_t - l_t$ with the previous seasonal estimate s_{t-L}. $x_t - l_t$ is the part of the current observation that is not explained by the current level due to seasonality. Give more weight to current residual $x_t - l_t$ than past seasonal estimate s_{t-L}.

$$s_t = \gamma(x_t - l_t) + (1 - \gamma)s_{t-L}$$

Step 5: Combine the last known level, trend, and the corresponding seasonal component.

$$\hat{x}_{t+m} = l_t + mb_t + s_{t-L+m}$$

Multiplicative Model:

i. Level equation
$l_t = \alpha(x_t / s_{t-L}) + (1 - \alpha)(l_{t-1} + b_{t-1})$
where,
- l_t is the estimated level at time t.
- α is the smoothing parameter.
- x_t is the current actual observation.
- b_{t-1} is the estimated trend at time t-1.
- **L** is the season length.
- s_{t-L} is the seasonal factor observed exactly one season ago.

ii. Trend Equation
$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$
where,
- b_t is the estimated level at time t.
- β is the trend smoothing parameter.

iii. Seasonal Equation
$s_t = \gamma(x_t / l_t) + (1 - \gamma)s_{t-L}$
where,
- s_t is the seasonal component at time t.
- γ is the seasonal smoothing parameter.

iv. Forecast Equation
$\hat{x}_{t+m} = (l_t + mb_t)s_{t-L+m}$
where,
- $\hat{x}_{t+m}$ is the forecasted value of the time series for time t+m.
- **m** is the number of periods.

Derivation:
Step 1: Assume the forecast is a combination of the current level, trend multiplied by the seasonal component.

$$\hat{x}_{t+1} = (l_t + b_t)s_{t+1-L}$$

Where,
- s_{t+1-L} refers to the same position in the previous seasonal cycle.

Step 2: Combine the de-seasonalized actual observation x_t / s_{t-L} with the previous forecasted level and trend $l_{t-1} + b_{t-1}$. Give more weight to current observations $x_t - s_{t-L}$ than past ones $l_{t-1} + b_{t-1}$.

$$l_t = \alpha(x_t / s_{t-L}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$

Step 3: Compare the current level l_t with the previous level l_{t-1} to get an idea of how much the base value is changing.

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

Step 4: Compare the seasonally adjusted residual x_t / l_t with the previous seasonal estimate s_{t-L}. This measures how much of the current observation is due to seasonality. Give more weight to current residual x_t / l_t than past seasonal estimate s_{t-L}.

$$s_t = \gamma(x_t / l_t) + (1 - \gamma)s_{t-L}$$

Step 5: Combine the last known level, trend, and the corresponding seasonal component.

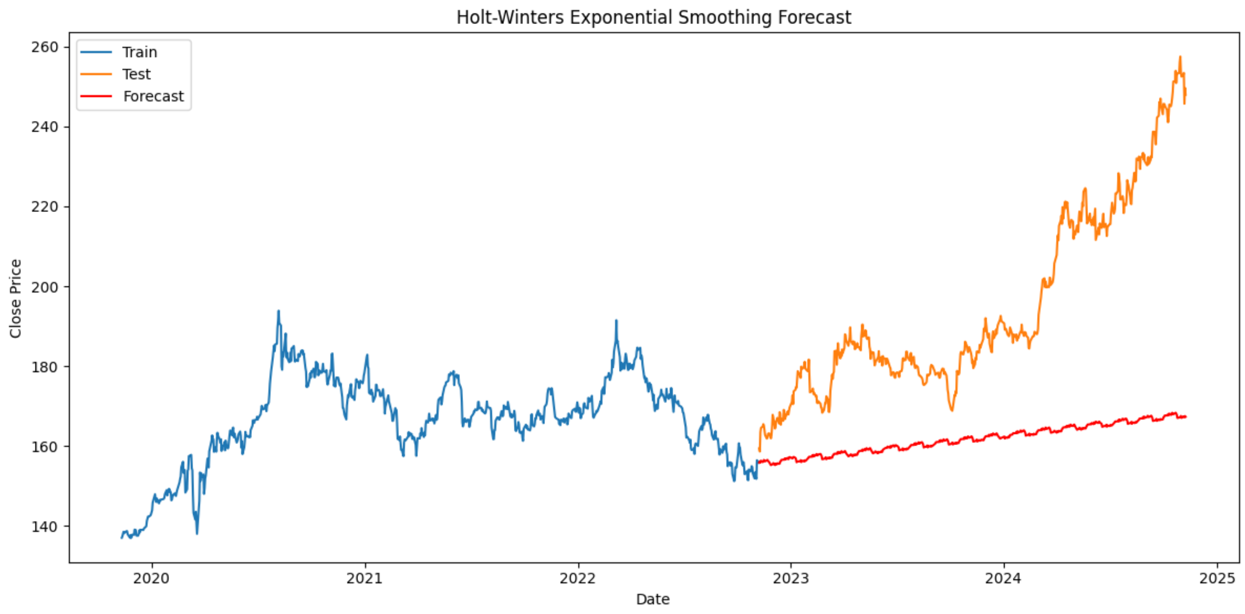
$$\hat{x}_{t+m} = l_t + mb_t + s_{t-L+m}$$

Code for Exponential Smoothing: Holt-Winters Method

```
# Fit Holt-Winters model
seasonal_period = 30
model = ExponentialSmoothing(train, trend="add", seasonal="add", seasonal_periods=seasonal_period,
initialization_method='estimated')

model_fit = model.fit(optimized=True)

# Forecast
forecast = model_fit.forecast(steps=len(test))
```



The Exponential Smoothing: Holt-Winters Method assumes smooth and gradual changes over time. Given the sharp spikes and market volatility in the gold price data, exponential smoothing could not adapt effectively, leading to inaccurate predictions.

Limitations of Exponential Smoothing:

- It assumes linear relationships and may underperform on non-linear data.
- It is sensitive to outliers.
- It gives more weight to recent data, making it less suitable for long-term forecasting.
- It does not consider external or causal variables, i.e., it is purely univariate.
- It cannot handle irregular or changing seasonality well.
- Parameter tuning (α, β, γ) requires careful selection to avoid overfitting or underfitting.

.

3.7.4 Prophet Model

Prophet is a forecasting model, developed by Facebook, that can handle data with strong seasonal effects, missing values, and outliers.

Purpose: This model is best suited for data having clear seasonality. It is very flexible, easy to use, and can handle missing values and outliers.

Mathematical Equations:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Where,

- **y(t)** is the observed value at time t.
- **g(t)** is the trend function.
- **s(t)** is the seasonality function.
- **h(t)** is the holiday effect.
- ϵ_t is the error term.

Derivation:

i. Trend Component

There are two types of trend components:

a. Piecewise Linear Trend

It allows the trend to change at specified points. These points are known as changepoints. It assumes that before a changepoint, the growth rate is constant while after each changepoint, a shift is added in the slope.

Mathematical Equation:

$$g(t) = (k + a(t)^T\delta)\,(t - t_0) + (m + a(t)^T\,\gamma)$$

Where,

- **k** is the slope or the initial growth rate.
- **a(t)** is a vector of length **S** that controls how much each changepoint affects time **t**.
- **S** is the total number of changepoints.
- **δ** is the change in the growth rate (slope) after changepoint.
- $(t - t_0)$ is the time that has passed since the starting point of the series.
- **m** is the intercept or offset term. It shifts the entire trend up or down, without changing its slope.
- **γ** is a vector used to adjust the intercept.

Derivation:

Step 1: Equation of a basic linear trend

$$g(t) = k(t - t_0) + m$$

Step 2: Add an adjustment for each changepoint. After each changepoint t_j, the slope increases by δ_j. If $\delta_j > 0$, the slope increases after t_j whereas if $\delta_j < 0$, the slope decreases after t_j .

$$g(t) = k(t - t_0) + m + \sum_{j=1}^{S} \delta_j \cdot (t - t_j) \cdot 1_{[t > tj]} \quad \ldots(i)$$

Where,

- **S** is the total number of changepoints.
- $(t - t_j)$ is the amount of time that has passed since changepoint t_j.
- $[t > t_j]$ is called an indicator function. If $t > t_j$, meaning the current time **t** is after the changepoint t_j, then $1_{[t > tj]} = 1$ otherwise 0.

Step 3: Define vector **a(t)** of length **S**.

$$a_j(t) = 1_{[t > tj]} \cdot (t - t_j)$$

Therefore, sum becomes the dot product of the two vectors **a(t)** and **δ**.

$\sum_{j=1}^{S} \delta_j \cdot a_j(t) = a(t)^T \delta$ …(ii)

Step 3: Substituting the value of **(ii)** in **(i).**

$g(t) = k(t - t_0) + m + a(t)^T \delta$

Step 4: Grouping the slope-like terms together as both **k** and **a(t)ᵀδ** depend on **(t – t₀)** to get the effective slope. Replace the initial intercept **m** with **m'** to ensure the trend line does not jump at any changepoint and the overall curve remains continuous. Adding $a(t)^T \gamma$ to **m** compensates for the cumulative effect of slope change at each changepoint.

$g(t) = (k + a(t)^T\delta)(t - t_0) + m'$

$m' = m + a(t)^T \gamma$

Step 5: Substituting the value of **m'**.

$g(t) = (k + a(t)^T\delta)(t - t_0) + (m + a(t)^T \gamma)$

b. Logistic Growth Trend

Logistic Growth is a way to model how something grows rapidly at first, then slows down, and eventually levels off.

Mathematical Equation:

$g(t) = C / (1 + \exp(-(k + a(t)^T \delta)(t - t_0)))$

where,

- **C** is the carrying capacity, i.e., the maximum value the trend can reach.
- **exp** is the exponential function, i.e., shorthand for e^x.
- **k** is the growth rate.
- **a(t)** is a vector of length **S** that controls how much each changepoint affects time **t**.
- **S** is the total number of changepoints.
- **δ** is a vector that defines the change in the growth rate (slope) at each changepoint.
- t_0 is the offset time or the midpoint.

Derivation:

Step 1: Equation of standard logistic growth

$g(t) = C / (1 + \exp(-k(t - t_0)))$

Case 1: For early times, $t \ll t_0$

$\exp(-k(t - t_0)) \gg 1$

$\Rightarrow g(t) = C/(\text{large number}) \approx 0$

So the growth starts near zero.

Case 2: when $t = t_0$

$\exp(-k(t - t_0)) = 1$

$\Rightarrow g(t) = C/2$

The growth rate is fastest

Case 3: when $t \gg t_0$

$\exp(-k(t - t_0)) \to 0$

$\Rightarrow g(t) \to C$

The growth rate slows down and eventually stops increasing as it reaches the maximum limit **C**.

Step 2: Adjusting the equation in accordance to changepoints. The slope becomes $k + a(t)^T \delta$.

Therefore, the equation becomes:

$g(t) = C / (1 + \exp(-(k + a(t)^T \delta)(t - t_0)))$

ii. Seasonality Component

Fourier series is used to model the periodic effects in the data. It is a way to represent any repeating (periodic) function as a sum of sines and cosines.

Mathematical Equation:

$s(t) = X(t)^T \beta$

where:

- **X(t)** is a feature vector of seasonal components.
- **β** is a vector of coefficients.

Derivation:

Step 1: Represent seasonality as a fourier series.

$s(t) = \sum_{n=1}^{N} (a_n \cos((2\pi nt)/P) + b_n \sin((2\pi nt)/P))$

Where:

- **P** is the period of seasonality.
- **N** is the order of the fourier series.
- a_n, b_n are the coefficients that are learned from the data. They determine how much each wave contributes.
- **t** is the time in days.

Step 2: Define **X(t)** as a feature vector of seasonal components and **β** as a vector of coefficients.

$X(t) = [\cos((2\pi t)/P), \sin((2\pi t)/P), \ldots, \cos((2\pi Nt)/P), \sin((2\pi Nt)/P)]$

$\beta = [a_1, b_1, \ldots, a_n, b_n]$

Therefore,

$s(t) = X(t)^T \beta$

iii. Holiday Effect

This effect is to model the impact of special events that causes predictable increase or decrease in a time series data.

Mathematical Equation:

$h(t) = Z(t)^T \kappa$

Where,

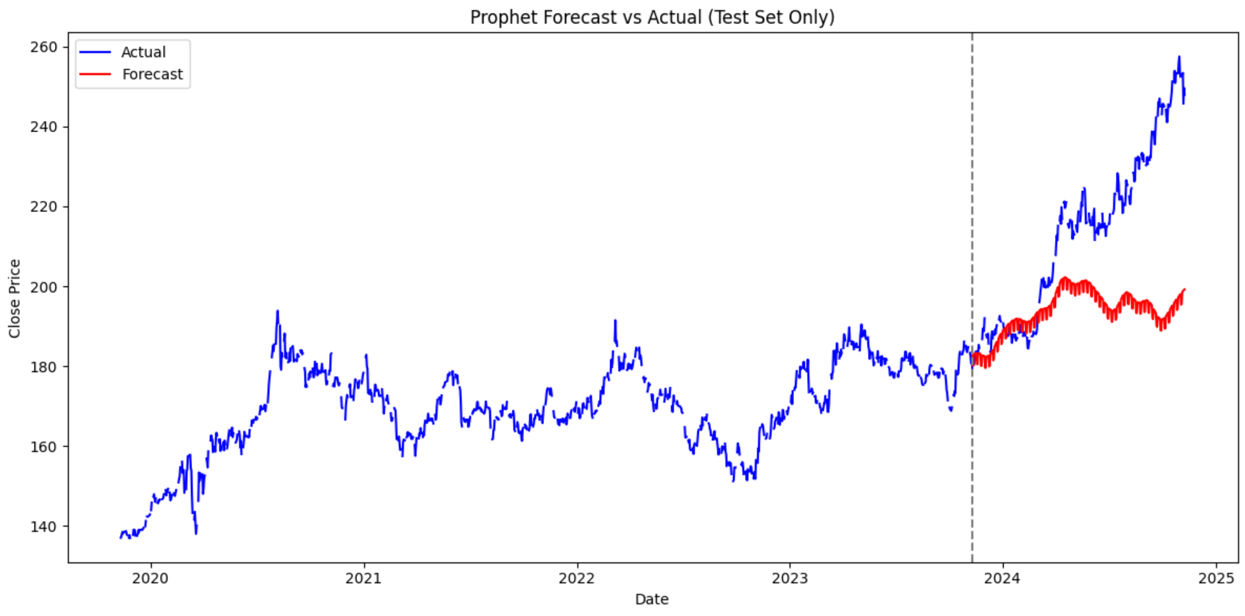
- **Z(t)** is a binary vector indicating whether time **t** is a holiday. Each holiday gets its own binary indicator. The indicator is **1** if time **t** is a holiday otherwise **0**.
- **κ** is a vector of learned parameters that represent the magnitude and direction of the holiday effect. It is automatically calculated during training.

Limitations of Prophet Model:

- It does not work as well for data with no patterns.
- It may underperform on very complex or highly nonlinear data.
- It needs a lot of historical data to learn from.

Code for Prophet Model

```
model = Prophet()
model.fit(train)
future = model.make_future_dataframe(periods=len(test))
forecast = model.predict(future)
```



Prophet outperformed the other models due to its robustness to missing data, outliers, and structural breaks. It captured both the trend and seasonal components effectively and aligned closely with the actual patterns, making it the most reliable model for forecasting gold prices.

3.8 Performance Metrics

3.8.1 MAE (Mean Absolute Error)

It measures the average of the absolute differences between forecasted and true values. It is less sensitive to outliers, however, extreme outliers can still influence it. It is scale-dependent. The value of MAE is always greater than or equal to 0, with 0 representing a perfect prediction. It is generally used when all errors are required to be treated equally and to be in the same units as that of the data. Lower value of MAE indicates a better model.

MAE= (1/n) $_{t=1}\sum^{n}|y_t-\hat{y}_t|$

where,

- **n** is the total number of observations
- y_t actual value at time t
- $\hat{y}_t$ is the predicted value at time t
- $|y_t-\hat{y}_t|$ is the absolute error at time t (distance between actual and predicted)

Code for MAE Calculation

```
mae = mean_absolute_error(test, forecast)
```

3.8.2 RMSE (Root Mean Squared Error)

It measures the square root of the average of the squared differences between the actual values and the predicted values. The error term $|y_t-\hat{y}_t|$ is squared to give higher weight to larger errors, making RMSE sensitive to outliers and then square rooted to bring the result back to the original scale of the data, thus, it has the same unit as the original data. Compared to MAE, RMSE penalizes larger errors more strongly. Lower value of RMSE indicates a better model.

RMSE = $\sqrt{(\,(1/n)\ _{t=1}\sum^{n}|y_t-\hat{y}_t|^2\,)}$

where,

- **n** is the total number of observations
- y_t actual value at time t
- $\hat{y}_t$ is the predicted value at time t
- $|y_t-\hat{y}_t|$ is the absolute error at time t (distance between actual and predicted)

Code for RSME Calculation

```
rmse = np.sqrt(mean_squared_error(test, forecast))
```

3.8.3 MAPE (Mean Absolute Percentage Error)

It expresses the accuracy of the model as a percentage by averaging the absolute percentage errors. It is scale-independent. However, it can be inaccurate if the actual values are very small because division by values close to zero can lead to very large or undefined percentage errors. Lower value of MAPE indicates a better model.

MAPE= (100/n) $_{t=1}\sum^{n}|\,(y_t-\hat{y}_t)/y_t\,|$

where,

- **n** is the total number of observations
- y_t actual value at time t
- $\hat{y}_t$ is the predicted value at time t
- $|y_t-\hat{y}_t|$ is the absolute error at time t (distance between actual and predicted)

Code for MAPE Calculation

```
mape = np.mean(np.abs((test.values - forecast.values) /
test.values))*100
```

4. Results and Prediction

4.1 Model Evaluation and Comparison

I have evaluated three models- ARIMA, SARIMA, Exponential Smoothing (Holt-Winters Method), and Prophet. Due to the presence of significant outliers in the dataset, I could not remove them without losing important information. ARIMA, SARIMA, and Exponential Smoothing are sensitive to outliers, which adversely affects their predictive accuracy. In contrast, Prophet is relatively more robust to outliers. Therefore, Prophet is the most suitable model.

Further, comparing performance metrics such as MAE, RMSE, and MAPE across all models.

MODEL	MAE	RSME	MAPE
ARIMA	26.77	32.36	11.84%
SARIMA	28.22	33.88	12.49%
Simple Exponential Smoothing	32.52	38.84	14.41%
Holt's Trend Model	27.02	37.57	11.95%
Holt Winter's Model	33.53	39.44	16.21%
Prophet	20.27	26.69	8.83%

The table shows that the performance metrics clearly indicate that the model's suitability to data characteristics significantly affect forecasting accuracy. Prophet's comparatively lower MAE (20.27), RMSE (26.69), and MAPE (8.83%) demonstrate its superior ability to capture both trend and seasonal components while being robust to outliers.

ARIMA and Holt's Trend Model perform almost comparably, suggesting that the dataset has a strong trend component but limited complex seasonal patterns. ARIMA's slightly better RMSE (32.36 vs. 37.57) implies that it manages larger deviations.
SARIMA does not outperform ARIMA, despite its seasonal component, which suggests that seasonality may not be strongly pronounced.

Simple Exponential Smoothing and Holt Winter's Model perform poorly, confirming that methods assuming a stable level (SES) or additive/multiplicative seasonal components (Holt-Winter's) are ill-suited for this dataset.

Therefore, Prophet is the best suited model as it has the lowest value of MAE, RMSE, and MAPE out of all six.

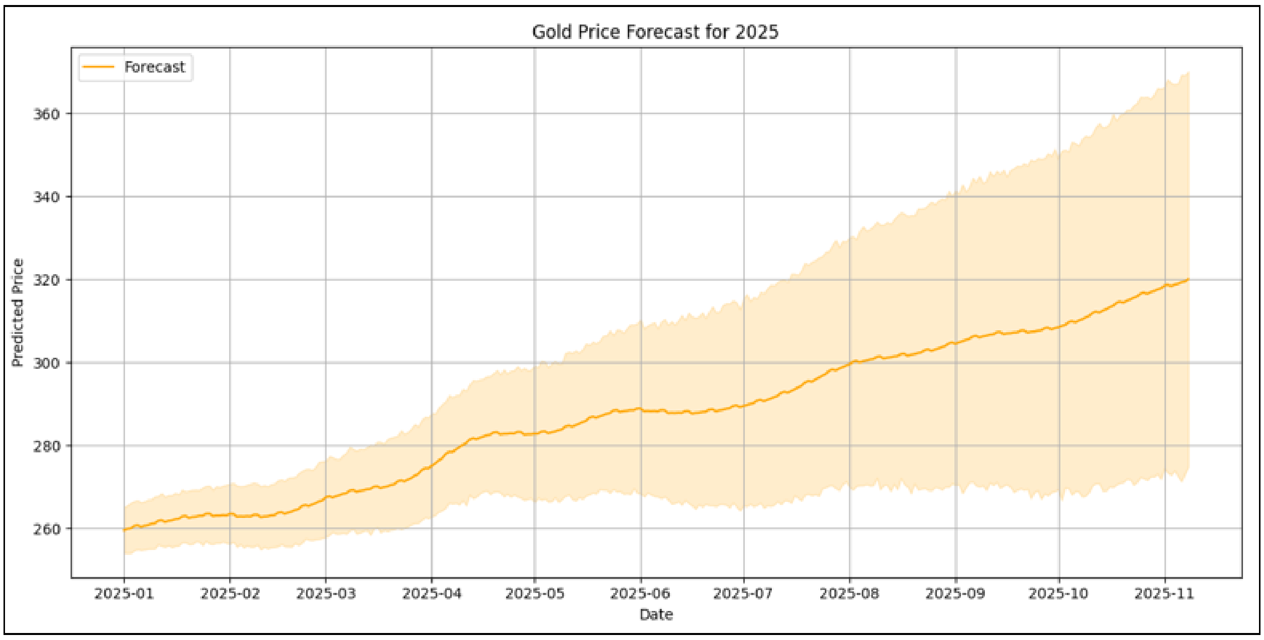
4.2 Forecast for the year 2025

```
# Fit the model
model = Prophet()
model.fit(df)

# Create a future DataFrame until the end of 2025
future = model.make_future_dataframe(periods=365, freq='D')  # 365 for one year if daily data

# Make predictions
forecast = model.predict(future)

# Filter forecast only for 2025
forecast_2025 = forecast[forecast['ds'].dt.year == 2025]
```



The orange line represents the predicted daily prices, while the shaded region indicates the uncertainty interval (upper and lower confidence bounds).

From the forecast:
- Gold prices are expected to continue rising steadily throughout 2025.
- The model captures a moderate upward trend, with predicted close prices starting around 260 and potentially exceeding 320–360 by the end of the year.
- The widening of the confidence interval toward the latter half of the year reflects increased uncertainty in long-range forecasts, which is common in time series models.

4.3 Strengths and Limitations

Prophet demonstrated several advantages that made it particularly well-suited for forecasting gold prices.

- It is robust to outliers, which is essential given the natural volatility and irregularities observed in gold price trends.
- Unlike ARIMA, SARIMA or Exponential Smoothing, Prophet automatically handles holidays and seasonality, and allows for flexible trend adjustments using changepoints. This was especially useful in capturing both long-term trends and recurring seasonal patterns in the data.

However, Prophet has few drawbacks.

- While it handles simple seasonality and changepoints effectively, it may underperform in capturing complex autocorrelation structures that models like SARIMA are explicitly designed for.
- It assumes additive or multiplicative seasonality by default, but this might not always match how the data actually behaves.
- Tuning parameters like the number of changepoints can require some trial-and-error and understanding of the data.
- Although Prophet works well with large datasets, it might follow random noise in the data too closely, especially if there are sudden changes or unexplained fluctuations.

4.4 Interpretation of Forecast Trends

The forecasted increase in gold prices throughout 2025 is correlated to current global and economic developments.

- **Escalating Middle East Tensions:** The ongoing instability involving Iran and Israel is expected to persist into 2025, potentially amplifying global economic uncertainty. This geopolitical stress typically pushes investors toward safer assets like gold, thus increasing its price.
- **Global Economic Slowdown Risks:** Many major economies are slowing down or at risk of recession. As a result, people may lose trust in regular investments like stocks or bonds, and instead invest in gold for safety.
- **Persistent Inflation Pressures:** Even though inflation has eased in some countries, core inflation is still high. This is leading investors and central banks to hold gold as a hedge against future inflation.
- **Increased Central Bank Buying:** Countries like China, India, and Russia have recently accelerated gold accumulation in their reserves. With this trend persisting into mid-2025, it has already contributed to stronger global demand and a steady rise in gold prices.
- **Currency Devaluation Concerns:** As debt levels increase globally, concerns about currency stability may cause more individuals and institutions to turn to gold as a store of value.

Overall, the Prophet model's forecast aligns with real-world expectations: increased volatility, persistent inflation, and geopolitical risks in 2025.

5. Summary and Findings

This study aimed to forecast gold prices using time series models. After evaluating ARIMA, SARIMA, Exponential Smoothing (SES, Holt’s Trend, Holt-Winters), and Prophet, using MAE, RMSE, and MAPE, it was found that Prophet performed best. Its robustness to outliers and ability to handle seasonality and trend shifts, makes it the most suitable time series model for predicting gold prices.

Traditional smoothing methods like SES and Holt-Winters, assumed gradual changes and failed to capture the sharp spikes caused by market volatility. Similarly, ARIMA and SARIMA were less effective due to the data’s non-linear trend and irregular seasonality. Furthermore, these models are sensitive to outliers, unlike Prophet.

The model predicted a continued upward trend in gold prices throughout 2025, supported by recent global economic and geopolitical instability. The dataset exhibited clear trends, seasonal patterns, and cyclic behavior, further validating the use of time series forecasting techniques.

6. Conclusion

This research comprehensively focuses on predicting gold prices by integrating advanced time series models, and explaining the models in detail. Gold, as a safe-haven asset, has consistently responded to macroeconomic fluctuations, inflationary pressures, currency valuation shifts, and geopolitical uncertainties. Historical analysis like the 1970s inflationary period, the 2008 global financial crisis, the 2011 economic concerns, the COVID-19 pandemic, and wars such as the Russia-Ukraine war reaffirmed that gold prices rise during periods of financial instability by influencing investor sentiment. The study also examined the relationship between gold and other asset classes, revealing its generally negative correlation with stock markets and positive correlation with inflation and demand, though exceptions were noted during unusual global events.

This study realises the need for accurate forecasts for investors, policymakers, and researchers. Using historical data spanning 5 years, the research explored 6 forecasting models, including ARIMA, SARIMA, Exponential Smoothing (Simple Exponential Smoothing, Holt’s Linear Trend Model, and Holt Winter’s Model), and Prophet.

The results indicated that traditional statistical models like ARIMA and SARIMA struggled to handle the inherent volatility and outlier-driven nature of gold prices. These models performed poorly due to their sensitivity to sudden price fluctuations and their reliance on linear patterns, which do not fully capture the non-linear and dynamic behavior of gold markets. Exponential Smoothing provided slightly better performance but remained limited in detecting sharp trend shifts.

Prophet, on the other hand, demonstrated better adaptability to trend changes and seasonality, making it more suitable for medium to long-term forecasting. Model evaluation using performance metrics such as Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Root Mean

Square Error (RMSE) confirmed that Prophet provided the most consistent results among the models tested.

The five-year forecast generated through the Prophet model suggests a likely upward trend in gold prices. This projection aligns with current global economic and geopolitical factors. Ongoing high inflation, cautious monetary policies by central banks, and global tensions—like the Russia-Ukraine war and recent issues in the Middle East involving Iran—are likely to keep investors interested in gold as a safe investment during uncertain times. Additionally, fears of a potential global economic slowdown encourage investors to turn to safer assets, further strengthening gold's position in the financial market, leading to a rise in prices.

However, while the predictions offer useful insights, they must be interpreted with caution. Forecasting models, including Prophet, assume that future trends will broadly follow historical patterns, which may not always hold true in times of unprecedented market shocks.

Overall, this study demonstrates how different modeling approaches interpret historical patterns and project future trends. It explains how to carefully select models that can handle seasonality, trend changes, and volatility according to the data. The comparison of ARIMA, SARIMA, Exponential Smoothing, and Prophet shows how modern models like Prophet, which are designed to capture trend shifts and seasonal effects, can outperform traditional linear models in volatile markets. This research also underlines the value of data preprocessing steps such as outlier handling, proper indexing, scaling, and differencing which are essential for improving model accuracy and stability.

References

[1] Kaggle. *Daily Gold Price (2015–2021) Time Series Dataset*.. Retrieved from https://www.kaggle.com/datasets/nisargchodavadiya/daily-gold-price-20152021-time-series

[2] Analytics Vidhya. (n.d.). *Data science, machine learning, and AI tutorials*. Retrieved from https://www.analyticsvidhya.com

[3] ArXiv. (n.d.). *Open-access archive for scholarly articles*. Retrieved from https://arxiv.org

[4] Curieux Academic Journal. (n.d.). *Platform for publishing student research*. Retrieved from https://www.curieuxacademicjournal.com

[5] Emerging Investigators. (n.d.). *Peer-reviewed journal for high school researchers*. Retrieved from https://emerginginvestigators.org

[6] GeeksforGeeks. (n.d.). *Programming, machine learning, and data science tutorials*. Retrieved from https://www.geeksforgeeks.org

[7] Gold.org. (n.d.). *World Gold Council – Gold market insights and data*. Retrieved from https://www.gold.org

[8] IJFMR (International Journal For Multidisciplinary Research). (n.d.). *Research publication for multiple domains*. Retrieved from https://www.ijfmr.com

[9] IJIRT (International Journal of Innovative Research in Technology). (n.d.). *Research publication in technology and science*. Retrieved from https://ijirt.org

[10] IJSART (International Journal for Scientific Research and Technology). (n.d.). *Research articles in technology and science*. Retrieved from https://ijsart.com

[11] Investopedia. (n.d.). *Financial definitions and market analysis*. Retrieved from https://www.investopedia.com

[12] Journal of Emerging Investigators. (n.d.). *Scientific research articles by students*. Retrieved from https://emerginginvestigators.org

[13] Journal of STEM Fellowship. (n.d.). *STEM-focused student research*. Retrieved from https://journal.stemfellowship.org

[14] KDNuggets. (n.d.). *AI, machine learning, and big data resources*. Retrieved from https://www.kdnuggets.com

[15] Kitco. (n.d.). *Gold prices, charts, and market news*. Retrieved from https://www.kitco.com

[16] Lumiere Education. (n.d.). *Research publication opportunities for high school students*. Retrieved from https://www.lumiere-education.com/post/15-journals-to-publish-your-research-in-high-school

[17] Machine Learning Mastery. (n.d.). *Practical guides on machine learning and time series*. Retrieved from https://machinelearningmastery.com

[18] Macrotrends. (n.d.). *Historical gold prices and financial data*. Retrieved from https://www.macrotrends.net

[19] Medium. (n.d.). *Articles on data science and technology*. Retrieved from https://medium.com

[20] Nature. (n.d.). *Scientific research publications*. Retrieved from https://www.nature.com

[21] OTexts. (n.d.). *Forecasting: Principles and Practice (FPP3)*. Retrieved from https://otexts.com/fpp3/

[22] Papers With Code. (n.d.). *Machine learning papers with implementations*. Retrieved from https://paperswithcode.com

[23] PyCaret. (n.d.). *Low-code machine learning library*. Retrieved from https://pycaret.org

[24] Reddit – r/datascience. (n.d.). *Community discussions on data science*. Retrieved from https://www.reddit.com/r/datascience/

[25] ResearchGate. (n.d.). *Scientific papers and research networking*. Retrieved from https://www.researchgate.net

[26] Scikit-learn. (n.d.). *Machine learning library for Python*. Retrieved from https://scikit-learn.org/stable/

[27] Springer. (n.d.). *Scientific journals and books*. Retrieved from https://www.springer.com

[28] Stack Overflow. (n.d.). *Programming Q&A community*. Retrieved from https://stackoverflow.com

[29] Stanford Intersect Journal. (n.d.). *Interdisciplinary science research by students*. Retrieved from https://ojs.stanford.edu/ojs/index.php/intersect

[30] Statista. (n.d.). *Statistics and market data*. Retrieved from https://www.statista.com

[31] Stars – UCF Undergraduate Research Journal. (n.d.). *Undergraduate research papers*. Retrieved from https://stars.library.ucf.edu/urj/

[32] TechCommunity Microsoft. (n.d.). *AI and data science blogs*. Retrieved from https://techcommunity.microsoft.com/t5/ai-cognitive-services-blog/bg-p/Azure-AI

[33] Towards AI. (n.d.). *Machine learning and AI research*. Retrieved from https://towardsai.net
[34] Towards Data Science. (n.d.). *Practical data science tutorials*. Retrieved from https://towardsdatascience.com
[35] Trading Economics. (n.d.). *Global economic indicators and gold price trends*. Retrieved from https://www.tradingeconomics.com
[36] Vanderbilt Young Scientist Journal. (n.d.). *Research journal for young scientists*. Retrieved from https://www.vanderbilt.edu/cseo/services/journals/young-scientist-journal/