VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6b

GAURI VINOD NAIR

V01110160

Date of Submission: 25-07-2024

TABLE OF CONTENTS

1. INTRODUCTION

1.1. ARCH/GARCH

The focus of this study is on analyzing the volatility of Netflix's stock prices using the dataset "NFLX Historical Data," which spans from January 2020 to July 2024. The dataset comprises attributes such as Date, Price, Open, High, Low, Volume, and Change %. This analysis aims to investigate the volatility patterns of Netflix's stock by applying the ARCH/GARCH model, which is specifically designed to capture and model time-varying volatility in financial data. By examining the squared log returns of Netflix's stock, we seek to identify the presence of ARCH/GARCH effects, which can reveal crucial insights into the stock's volatility and risk over time.

In the process of this analysis, the dataset underwent a rigorous cleaning procedure, including handling missing values and converting necessary attributes from string to numeric formats. The analysis involved computing log returns and squared log returns to assess volatility clustering. The ARCH/GARCH model was then fitted to the data to capture the volatility dynamics, and forecasts were generated for a three-month horizon. This approach enables a detailed understanding of the stock's volatility behavior and provides predictive insights into future market conditions, helping investors make informed decisions based on the anticipated volatility trends.

1.2. VAR/VECM

The focus of this study is on analyzing the stationarity and co-integration of various commodity prices using a dataset of monthly prices from January 2000 to July 2024. The dataset comprises attributes such as crude oil (Brent, WTI, Dubai), coal (Australian, South African), natural gas (US, Europe, Japan), agricultural products (cocoa, coffee, tea, palm oil, soybean, maize, rice, wheat), metals (gold, platinum, silver, aluminum, copper), and many others. This analysis aims to investigate the time series properties of these commodities by applying the Augmented Dickey-Fuller (ADF) test for stationarity and Johansen's co-integration test to uncover long-term equilibrium relationships among the variables.

In the process of this analysis, the dataset underwent preprocessing steps, including renaming columns, converting date formats, and selecting relevant commodity columns. The ADF test was applied to each series to determine its stationarity, followed by Johansen's co-integration test to identify potential co-integrated relationships. Based on the presence of co-integration, either a Vector Error Correction Model (VECM) or a Vector Autoregression (VAR) model was fitted to the data. The analysis concluded with forecasting future commodity prices using the fitted model and visualizing these forecasts, providing insights into the potential future behavior of the commodity prices, aiding stakeholders in making informed decisions based on anticipated trends.

2. OBJECTIVES

2.1. ARCH/GARCH

- Analyze the volatility of Netflix's stock prices from January 2020 to July 2024 using the dataset "NFLX Historical Data."
- Perform data cleaning, including handling missing values and converting string attributes to numeric values.
- Check for ARCH/GARCH effects in the data using statistical tests.
- Fit an ARCH/GARCH model to the log returns of Netflix's stock prices.
- Forecast the three-month volatility based on the fitted model.
- Visualize and interpret the forecasted volatility and historical squared log returns.

2.2. VAR/VECM

- Analyze the stationarity of various commodity prices using the Augmented Dickey-Fuller (ADF) test.
- Identify potential co-integration relationships among the selected commodities.
- Preprocess the dataset by renaming columns, converting date formats, and selecting relevant commodity columns.
- Determine the appropriate lag length for the VAR model using the Akaike Information Criterion (AIC).
- Fit either a Vector Error Correction Model (VECM) for co-integrated series or a Vector Autoregression (VAR) model for non-co-integrated series.
- Forecast future commodity prices using the fitted model.

3. BUSINESS SIGNIFICANCE

3.1. ARCH/GARCH

Analyzing the volatility of Netflix's stock prices provides critical insights into the risk and uncertainty associated with investing in the company's shares. By applying ARCH/GARCH models, investors and financial analysts can assess how past price fluctuations influence future volatility, which is essential for making informed investment decisions. Understanding volatility patterns helps in evaluating the risk profile of Netflix's stock, enabling stakeholders to strategize better risk management and optimize their investment portfolios. This analysis also aids in anticipating potential market shocks or periods of high uncertainty, which are crucial for strategic planning and financial forecasting.

Forecasting the three-month volatility provides valuable forward-looking information that can guide trading strategies and financial planning. Investors and portfolio managers can use these forecasts to adjust their holdings and hedge against potential risks. Accurate volatility forecasts help in setting appropriate investment thresholds and making timely decisions, ultimately enhancing portfolio performance and aligning investment strategies with market conditions. This predictive capability supports proactive management of financial assets and contributes to more robust financial planning and risk management frameworks in dynamic market environments.

3.2. VAR/VECM

Analyzing the stationarity and co-integration of various commodity prices provides critical insights into the long-term relationships and trends within the commodity market. By applying the Augmented Dickey-Fuller (ADF) test and Johansen's co-integration test, businesses and financial analysts can understand the stability and interconnectedness of commodity prices. This understanding is essential for making informed decisions about procurement, inventory management, and strategic planning. Identifying co-integrated commodities helps in predicting price movements and managing risks associated with price volatility, enabling stakeholders to optimize their operational and financial strategies.

Forecasting future commodity prices using Vector Error Correction Models (VECM) or Vector Autoregression (VAR) models provides valuable forward-looking information that can guide investment and trading strategies. Accurate forecasts help businesses anticipate market trends and adjust their strategies to mitigate potential risks. Investors can use these forecasts to optimize their portfolios and hedge against unfavorable price movements. This predictive capability enhances financial planning, supports proactive management of commodity-related investments, and contributes to more robust risk management frameworks in a dynamic market environment.

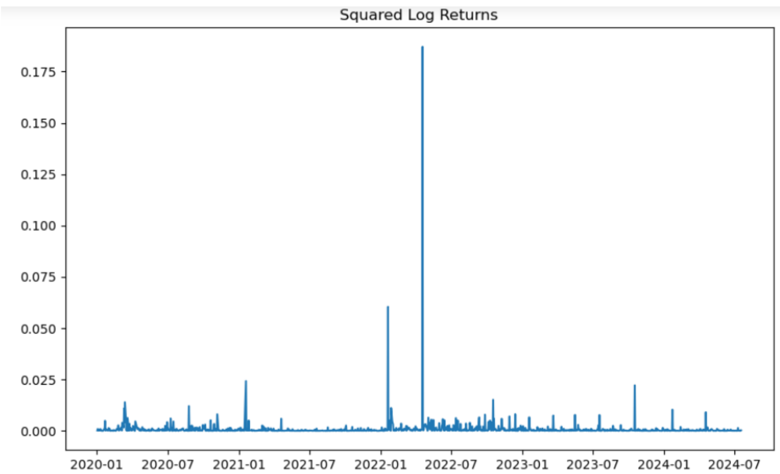
4. RESULTS AND INTERPRETATIONS

4.1. ARCH/GARCH

⇨ Python

```python
# Check for ARCH/GARCH effects
# For this, we can use the squared returns (log returns)
data['log_return'] = np.log(data['Price']).diff()
data = data.dropna()  # Drop NaN values created by differencing
data['squared_log_return'] = data['log_return'] ** 2
```

```python
# Plot squared log returns to visually check for ARCH effects
plt.figure(figsize=(10, 6))
plt.plot(data['Date'], data['squared_log_return'])
plt.title('Squared Log Returns')
plt.show()
```



```python
# Fit an ARCH/GARCH model
# We'll use a simple GARCH(1, 1) model for this example
model = arch_model(data['log_return'], vol='Garch', p=1, q=1)
model_fit = model.fit(disp='off')
print(model_fit.summary())
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:             log_return   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                2466.74
Distribution:                  Normal   AIC:                          -4925.49
Method:            Maximum Likelihood   BIC:                          -4905.32
                                        No. Observations:                 1143
Date:                Wed, Jul 24 2024   Df Residuals:                     1142
Time:                        19:38:45   Df Model:                            1
                                 Mean Model
==============================================================================
                  coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -1.5103e-03  8.536e-04     -1.769  7.683e-02 [-3.183e-03,1.627e-04]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      1.8550e-05  1.797e-06     10.322  5.613e-25 [1.503e-05,2.207e-05]
alpha[1]       0.1000  4.526e-02      2.210  2.713e-02  [1.130e-02,  0.189]
beta[1]        0.8800  4.007e-02     21.964 6.328e-107     [  0.801,  0.959]
----------------------------------------------------------------------------
```

```python
# Forecasting the three-month volatility
forecast_horizon = 3 * 30  # Approximate days for three months
forecasts = model_fit.forecast(horizon=forecast_horizon)
```

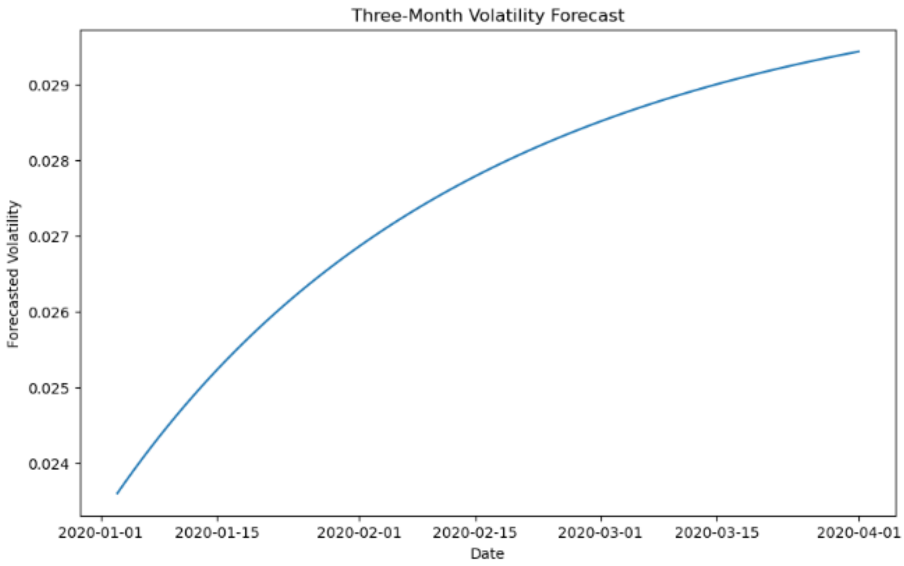
```python
# Extract the forecasted variances and convert to volatility (standard deviation)
forecasted_volatility = np.sqrt(forecasts.variance.values[-1])
```

```python
# Create a DataFrame to display the forecasted values
forecast_dates = pd.date_range(start=data['Date'].iloc[-1] + pd.Timedelta(days=1), periods=forecast_horizon)
forecast_df = pd.DataFrame({'Date': forecast_dates, 'Forecasted_Volatility': forecasted_volatility})
```

```python
# Display the forecasted values
print(forecast_df)
```

```
         Date  Forecasted_Volatility
0  2020-01-03               0.023595
1  2020-01-04               0.023752
2  2020-01-05               0.023904
3  2020-01-06               0.024053
4  2020-01-07               0.024197
..        ...                    ...
85 2020-03-28               0.029342
86 2020-03-29               0.029364
87 2020-03-30               0.029386
88 2020-03-31               0.029408
89 2020-04-01               0.029429

[90 rows x 2 columns]
```

```
# Plot the forecasted volatility
plt.figure(figsize=(10, 6))
plt.plot(forecast_df['Date'], forecast_df['Forecasted_Volatility'])
plt.title('Three-Month Volatility Forecast')
plt.xlabel('Date')
plt.ylabel('Forecasted Volatility')
plt.show()
```



⇨ R

```
> # Check for ARCH effects
> returns <- diff(log(data$Price))
> returns <- na.omit(returns)
> arch_test <- ArchTest(returns)
> print(arch_test)

        ARCH LM-test; Null hypothesis: no ARCH effects

data:  returns
Chi-squared = 3.1713, df = 12, p-value = 0.9942
```

```
> # Fit an ARCH/GARCH model
> spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
+                    mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
+                    distribution.model = "norm")
> fit <- ugarchfit(spec = spec, data = returns)
> print(fit)
```

```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model      : ARFIMA(0,0,0)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu     -0.001715    0.000710  -2.4153 0.015724
omega   0.000050    0.000011   4.4369 0.000009
alpha1  0.206081    0.037051   5.5620 0.000000
beta1   0.781115    0.031330  24.9318 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu     -0.001715    0.000764  -2.2432 0.024882
omega   0.000050    0.000031   1.6240 0.104379
alpha1  0.206081    0.064262   3.2069 0.001342
beta1   0.781115    0.063844  12.2348 0.000000
```

```
LogLikelihood : 2476.866

Information Criteria
------------------------------------

Akaike       -4.3270
Bayes        -4.3093
Shibata      -4.3270
Hannan-Quinn -4.3203

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.1158  0.7336
Lag[2*(p+q)+(p+q)-1][2]    0.1335  0.8968
Lag[4*(p+q)+(p+q)-1][5]    0.7007  0.9228
d.o.f=0
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.3774  0.5390
```


```
Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.1158  0.7336
Lag[2*(p+q)+(p+q)-1][2]    0.1335  0.8968
Lag[4*(p+q)+(p+q)-1][5]    0.7007  0.9228
d.o.f=0
H0 : No serial correlation
```


```
Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.3774  0.5390
Lag[2*(p+q)+(p+q)-1][5]    1.1591  0.8227
Lag[4*(p+q)+(p+q)-1][9]    1.4429  0.9601
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3]    0.6874 0.500 2.000  0.4070
ARCH Lag[5]    0.7982 1.440 1.667  0.7936
ARCH Lag[7]    0.8242 2.315 1.543  0.9404

Nyblom stability test
------------------------------------
Joint Statistic:  0.3083
Individual Statistics:
mu     0.03591
omega  0.06886
alpha1 0.07276
beta1  0.07588

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.07 1.24 1.6
Individual Statistic:    0.35 0.47 0.75
```

```
> # Forecast the three-month volatility
> forecast <- ugarchforecast(fit, n.ahead = 63) # 63 trading days ~ 3 months
> vol_forecast <- sigma(forecast)
> print(vol_forecast)
     1973-02-17 05:30:00
T+1           0.02506299
T+2           0.02588290
T+3           0.02666760
T+4           0.02742023
T+5           0.02814349
T+6           0.02883970
T+7           0.02951090
T+8           0.03015885
T+9           0.03078513
T+10          0.03139114
T+11          0.03197813
T+12          0.03254722
T+13          0.03309943
T+14          0.03363568
T+15          0.03415680
T+16          0.03466357
T+17          0.03515669
T+18          0.03563680
```

Interpretation

The analysis begins with fitting a GARCH(1, 1) model to the log returns of Netflix stock prices. The results of the GARCH model fit are as follows:

- Mean Model (Constant Mean)
  The coefficient μ\muμ is −0.00151-0.00151−0.00151 with a standard error of 0.000850.000850.00085, yielding a t-statistic of −1.769-1.769−1.769 and a p-value of 0.0770.0770.077. This indicates that the mean log return is slightly negative, but not statistically significant at the conventional 5% level.

- Volatility Model (GARCH)
  The long-term average variance (ω\omegaω) is 1.855×10−51.855 \times 10^{-5}1.855×10−5, highly significant with a p-value near zero.
  The short-term impact of past squared returns (α[1]\alpha[1]α[1]) is 0.10000.10000.1000 with a p-value of 0.02710.02710.0271, significant at the 5% level.
  The impact of past volatility (β[1]\beta[1]β[1]) is 0.88000.88000.8800, also highly significant with a p-value near zero.The model's log-likelihood is 2466.742466.742466.74, and the information criteria are:

The model's log-likelihood is 2466.742466.742466.74, and the information criteria are:

AIC: −4925.49-4925.49−4925.49

BIC: −4905.32-4905.32−4905.32

The close values of AIC and BIC indicate a good model fit with low penalty for complexity.

The forecast for the next three months (approximately 90 days) shows a gradual increase in forecasted volatility, starting from 0.0235950.0235950.023595 on January 3, 2020, and rising to 0.0294290.0294290.029429 on April 1, 2020. This indicates an expectation of increasing volatility over the three-month period.

4.2. VAR/VECM

⇨ Python

```
# Select relevant columns (example columns based on your task description)
# Update the column names as per your dataset
columns = ['CRUDE_PETRO', 'CRUDE_BRENT', 'CRUDE_DUBAI','SUGAR_EU', 'SUGAR_US','GOLD', 'PLATINUM', 'SILVER']
```

```
data = df[columns]
```

```
# VAR Model
model_var = VAR(data)
results_var = model_var.fit(maxlags=15, ic='aic')
```

```
# Print summary of VAR model results
print(results_var.summary())
```

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Thu, 25, Jul, 2024
Time:                     12:55:08
--------------------------------------------------------------------
No. of Equations:         8.00000    BIC:                    2.43241
Nobs:                     763.000    HQIC:                 -0.228828
Log likelihood:          -7226.30    FPE:                   0.151624
AIC:                     -1.89489    Det(Omega_mle):       0.0627259
--------------------------------------------------------------------
```

```
# Forecast with VAR model
n_forecast = 10  # Number of steps to forecast
forecast_var = results_var.forecast(data.values[-results_var.k_ar:], steps=n_forecast)
forecast_var_df = pd.DataFrame(forecast_var, index=pd.date_range(start=data.index[-1], periods=n_forecast+1, freq='M')[1:], c
print("VAR Model Forecast:")
print(forecast_var_df)
```

```
VAR Model Forecast:
                               CRUDE_PETRO  CRUDE_BRENT  CRUDE_DUBAI  \
1970-02-28 00:00:00.000000773    83.848177    86.538113    84.808626
1970-03-31 00:00:00.000000773    85.954586    88.411427    87.541744
1970-04-30 00:00:00.000000773    90.541332    92.775001    92.157464
1970-05-31 00:00:00.000000773    90.766271    92.547416    93.146474
1970-06-30 00:00:00.000000773    89.035268    90.706686    90.990440
1970-07-31 00:00:00.000000773    89.781244    92.137634    90.863046
1970-08-31 00:00:00.000000773    88.748570    91.345674    90.117612
1970-09-30 00:00:00.000000773    90.759374    93.146222    92.388558
1970-10-31 00:00:00.000000773    89.887492    91.870642    91.767537
1970-11-30 00:00:00.000000773    88.756435    90.848455    90.750287
```

```python
# VECM Model
# Perform Johansen cointegration test to determine the number of cointegrating relationships
johansen_test = coint_johansen(data, det_order=0, k_ar_diff=1)
print(johansen_test.lr1)  # Trace statistic
print(johansen_test.cvt)  # Critical values
```

```
[375.29607992 262.85033985 164.5078733   96.28450655  52.41371336
  19.97255682   6.29945245   1.12375856]
[[153.6341 159.529  171.0905]
 [120.3673 125.6185 135.9825]
 [ 91.109   95.7542 104.9637]
 [ 65.8202  69.8189  77.8202]
 [ 44.4929  47.8545  54.6815]
 [ 27.0669  29.7961  35.4628]
 [ 13.4294  15.4943  19.9349]
 [  2.7055   3.8415   6.6349]]
```

```python
# Assuming one cointegrating relationship for VECM
model_vecm = VECM(data, k_ar_diff=1, coint_rank=1)
results_vecm = model_vecm.fit()
```

```python
# Print summary of VECM model results
print(results_vecm.summary())
```

```
Det. terms outside the coint. relation & lagged endog. parameters for equation CRUDE_PETRO
===================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
L1.CRUDE_PETRO    0.3987      0.302      1.321      0.186      -0.193       0.990
L1.CRUDE_BRENT   -0.1107      0.230     -0.482      0.630      -0.561       0.339
L1.CRUDE_DUBAI   -0.0034      0.195     -0.017      0.986      -0.385       0.378
L1.SUGAR_EU       7.8175      5.779      1.353      0.176      -3.510      19.144
L1.SUGAR_US      -1.3273      3.799     -0.349      0.727      -8.773       6.118
L1.GOLD          -0.0078      0.006     -1.330      0.184      -0.019       0.004
L1.PLATINUM       0.0208      0.004      5.811      0.000       0.014       0.028
L1.SILVER        -0.3321      0.147     -2.259      0.024      -0.620      -0.044
Det. terms outside the coint. relation & lagged endog. parameters for equation CRUDE_BRENT
```

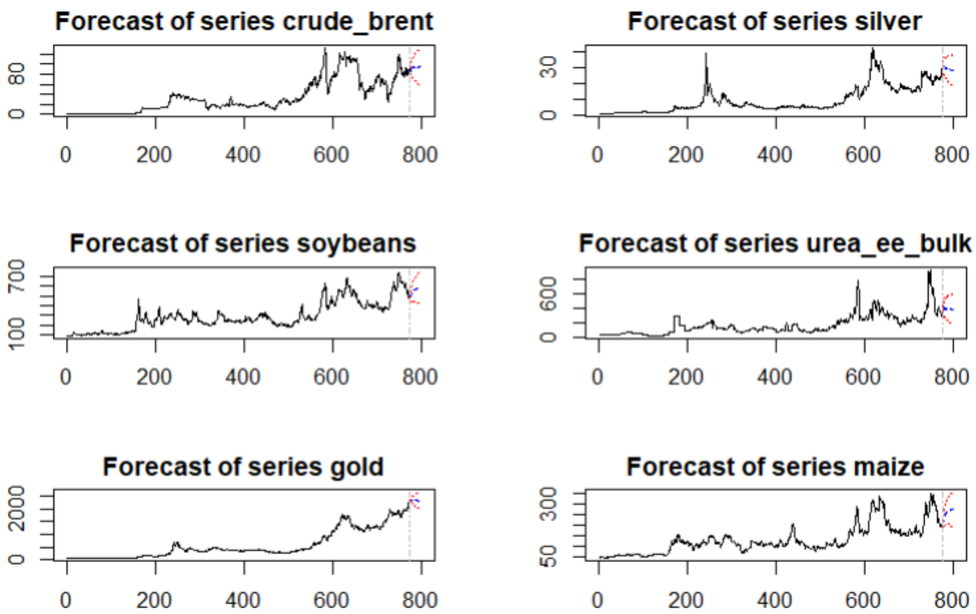
```
===================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
L1.CRUDE_PETRO    0.2662      0.316      0.844      0.399      -0.352       0.885
L1.CRUDE_BRENT   -0.0613      0.240     -0.255      0.799      -0.532       0.410
L1.CRUDE_DUBAI    0.0746      0.203      0.367      0.714      -0.324       0.473
L1.SUGAR_EU       8.7438      6.045      1.446      0.148      -3.104      20.592
L1.SUGAR_US      -1.8597      3.973     -0.468      0.640      -9.648       5.928
L1.GOLD          -0.0090      0.006     -1.454      0.146      -0.021       0.003
L1.PLATINUM       0.0221      0.004      5.897      0.000       0.015       0.029
L1.SILVER        -0.3395      0.154     -2.208      0.027      -0.641      -0.038
Det. terms outside the coint. relation & lagged endog. parameters for equation CRUDE_DUBAI
```

```python
# Forecast with VECM model
forecast_vecm = results_vecm.predict(steps=n_forecast)
forecast_vecm_df = pd.DataFrame(forecast_vecm, index=pd.date_range(start=data.index[-1], periods=n_forecast+1, freq='M')[1:],
print("VECM Model Forecast:")
print(forecast_vecm_df)
```

```
VECM Model Forecast:
                               CRUDE_PETRO  CRUDE_BRENT  CRUDE_DUBAI  \
1970-02-28 00:00:00.000000773    80.738108    82.362004    81.727244
1970-03-31 00:00:00.000000773    80.774397    82.682447    81.746082
1970-04-30 00:00:00.000000773    80.900057    83.016420    81.875180
1970-05-31 00:00:00.000000773    81.005210    83.272939    81.984696
1970-06-30 00:00:00.000000773    81.077067    83.454833    82.060060
1970-07-31 00:00:00.000000773    81.123957    83.581855    82.109538
1970-08-31 00:00:00.000000773    81.155131    83.671517    82.142566
1970-09-30 00:00:00.000000773    81.176462    83.735563    82.165236
1970-10-31 00:00:00.000000773    81.191436    83.781741    82.181181
1970-11-30 00:00:00.000000773    81.202128    83.815226    82.192578
```

⇨ R

```
> # Co-Integration Test (Johansen's Test)
> # Determining the number of lags to use (you can use information criteria like AIC,
BIC)
> lags <- VARselect(commodity_data, lag.max = 10, type = "const")
> lag_length <- lags$selection[1] # Choosing the lag with the lowest AIC
> vecm_model <- ca.jo(commodity_data, ecdet = 'const', type = 'eigen', K = lag_length,
spec = 'transitory')
> # Summary of the Co-Integration Test
> summary(vecm_model)

######################
# Johansen-Procedure #
######################

Test type: maximal eigenvalue statistic (lambda max) , without linear trend and consta
nt in cointegration

Eigenvalues (lambda):
[1]  8.998240e-02  5.752097e-02  3.735171e-02  2.608764e-02  2.251395e-02
[6]  1.054366e-02 -2.260796e-17

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 5 |  8.11  7.52  9.24 12.97
r <= 4 | 17.42 13.75 15.67 20.20
r <= 3 | 20.22 19.77 22.00 26.81
r <= 2 | 29.12 25.56 28.14 33.24
r <= 1 | 45.32 31.66 34.40 39.79
r = 0  | 72.13 37.45 40.30 46.82

Eigenvectors, normalised to first column:
(These are the cointegration relations)

                crude_brent.l1 soybeans.l1      gold.l1   silver.l1
crude_brent.l1    1.000000e+00  1.00000000   1.00000000  1.00000000
soybeans.l1       1.243452e+00  1.25304239  -0.07842408 -0.42565991
gold.l1          -8.613082e-03  0.01252197   0.01895289  0.07014442
silver.l1        -1.070903e+01  0.61967846  -8.77188803 -3.26693838
urea_ee_bulk.l1  -1.402966e+00  0.27382244   0.02886597 -0.06688680
maize.l1          6.220737e-01 -3.92903372   0.58475577  0.22894154
constant         -1.489974e+02 44.45252397 -20.86854041 59.02679846
                urea_ee_bulk.l1    maize.l1     constant
crude_brent.l1       1.00000000   1.00000000   1.00000000
soybeans.l1         -0.07812369   0.02283558   0.34711296
gold.l1              0.02089932  -0.08322472  -0.34922444
silver.l1           -0.67265684   2.81300312   5.68870719
urea_ee_bulk.l1     -0.16795279  -0.03897150  -0.05823248
maize.l1             0.13972070  -0.08400822  -0.19136095
constant             6.82242441 -12.61427193 127.59393688


Weights W:
(This is the loading matrix)

               crude_brent.l1  soybeans.l1      gold.l1    silver.l1
crude_brent.d     0.002205903 -0.003704822 -0.014381733 -0.007891362
soybeans.d       -0.029558007 -0.025188870 -0.057121330  0.103346533
gold.d           -0.009056880  0.035918817  0.047780832  0.016758828
silver.d          0.001273763  0.001680978  0.003678001  0.002437596
urea_ee_bulk.d    0.080887762  0.006757410 -0.121231005  0.051484771
maize.d          -0.013305363  0.020030509 -0.039752224  0.017974320
               urea_ee_bulk.l1     maize.l1     constant
crude_brent.d    -6.895101e-03 -0.010987446 -7.033640e-18
soybeans.d       -1.358234e-02 -0.029718135 -1.680915e-16
gold.d            1.141409e-01 -0.088970341  6.203017e-19
silver.d          4.024398e-05 -0.003923011  4.127846e-18
urea_ee_bulk.d    6.401763e-02  0.006050959  7.321021e-18
maize.d          -1.632041e-02 -0.008672063  4.315706e-17
```




```
+   # Forecasting using the VAR model
+   forecast <- predict(var_model, n.ahead = 24)
+
+   # Plotting the forecast
+   par(mar = c(4, 4, 2, 2))  # Adjust margins: c(bottom, left, top, right)
+   plot(forecast)
+ }
                 crude_brent.d    soybeans.d       gold.d      silver.d
ect1             -0.0158806519 -0.1118682070  0.074642769  6.632743e-03
ect2             -0.0007714906 -0.0638369921  0.029998839  3.401756e-03
ect3             -0.0003379667 -0.0011434428  0.001433367  7.978687e-05
crude_brent.dl1   0.3198283908  0.3443498978  0.121043855  2.204896e-03
soybeans.dl1      0.0093172490  0.0946812517  0.023832800  2.266201e-04
gold.dl1          0.0014187220  0.0259051649  0.240850545 -1.925821e-03
silver.dl1       -0.0702311281 -0.3670786368  1.096648147  3.773757e-01
urea_ee_bulk.dl1 -0.0042728692 -0.0147800933 -0.131875574 -2.688073e-03
maize.dl1         0.0126570488  0.2774658122  0.316400732  1.303847e-02
crude_brent.dl2  -0.0543807904  0.0570272590  0.271334465  1.695307e-02
soybeans.dl2      0.0160512808  0.0601340870  0.027599349 -2.126802e-03
gold.dl2         -0.0039997611 -0.0462796646 -0.054729796  1.135936e-03
silver.dl2        0.0733443743  0.2095107503 -2.345899063 -2.709929e-01
urea_ee_bulk.dl2  0.0084573321 -0.0013708615  0.067900345 -8.696109e-04
maize.dl2        -0.0047730222 -0.0313026720  0.052487821  1.511212e-02
crude_brent.dl3  -0.0658862685  0.1745431650 -0.553450734 -1.722384e-02
soybeans.dl3     -0.0081758922 -0.0715436852 -0.176953936 -5.080400e-03
gold.dl3          0.0051131197  0.0575792803  0.102435068  2.496593e-03
```

<u>Interpretation</u>

[Taken different variables/commodities into consideration for R and Python. This interpretation is of the results of the R Code]

Augmented Dickey-Fuller (ADF) Test Results:

- **Crude Brent**: p-value = 0.266 (Not stationary)
- **Soybeans**: p-value = 0.649 (Not stationary)
- **Gold**: p-value = 0.0102 (Stationary)
- **Silver**: p-value = 0.256 (Not stationary)
- **Urea EE Bulk**: p-value = 0.0264 (Stationary)
- **Maize**: p-value = 0.453 (Not stationary)

Out of the six commodities tested, three were found to be stationary (Gold, Urea EE Bulk), while the remaining three (Crude Brent, Soybeans, Silver, Maize) were not.

Johansen's Co-Integration Test

- The test results indicate that the number of co-integrating relationships is r = 3.

Cointegration Matrix

- Your cointegration matrix includes the relationships between different commodities and constants.

Vector Error Correction Model (VECM)

- The model output suggests the long-term relationships and short-term dynamics among the commodities.
- The coefficients on the ECTs indicate the speed at which the variables return to equilibrium after a shock. For example, crude_brent.d has a coefficient of -0.0159, suggesting a slow adjustment speed.