VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6b: Time Series Analysis

FERAH SHAN SHANAVAS RABIYA

V01101398

Date of Submission: 26-07-2024

CONTENTS

Introduction

This study analyzes the volatility of Arvind Fashion's stock prices using a dataset from March 2019 to July 2024. The dataset includes attributes such as Date, Price, Open, High, Low, Volume, and Change %. The ARCH/GARCH model is used to examine the stock's volatility patterns, detecting ARCH/GARCH effects that provide valuable information about the stock's risk patterns over time.

The dataset undergoes thorough cleaning, addressing missing values and translating relevant attributes from string to numeric representations. Logarithmic returns and squared logarithmic returns are calculated to evaluate volatility clustering. The ARCH/GARCH model is applied to capture the dynamics of volatility, and predictions are made over a three-month period. This methodology provides a comprehensive understanding of the stock's volatility patterns and offers prognostic perspectives on future market circumstances, helping investors make well-informed decisions based on projected volatility trends.

The second part of the study analyzes the stationarity and co-integration of different commodity prices using a dataset from January 2000 to July 2024. The dataset includes crude oil, coal, natural gas, agricultural products, and metals. The analysis uses the Augmented Dickey-Fuller (ADF) test to determine stationarity and Johansen's co-integration test to reveal long-term equilibrium linkages between variables. Preprocessing operations are performed, and each series undergoes the ADF test to confirm stationarity. Co-integration tests are employed to detect potential correlations. The data is fitted with either a Vector Error Correction Model (VECM) or a Vector Autoregression (VAR) model, depending on co-integration. The fitted model is used to predict future commodity prices, providing valuable insights into the likely future behavior of commodity prices.

Objectives

- Analyze stock volatility patterns from March 2019 to July 2024.
- Detect ARCH/GARCH effects to understand stock's risk patterns.
- Perform data cleaning and preprocessing.
- Calculate logarithmic returns and squared logarithmic returns to evaluate volatility clustering.
- Apply ARCH/GARCH model to capture volatility dynamics.

- Make predictions over a three-month period.
- Offer prognostic perspectives on future market circumstances.
- Analyze stationarity and co-integration of commodity prices from January 2000 to July 2024.
- Use Augmented Dickey-Fuller (ADF) test to determine stationarity.
- Employ Johansen's co-integration test to identify long-term equilibrium linkages.
- Preprocess dataset and fit data with a Vector Error Correction Model (VECM) or a Vector Autoregression (VAR) model.
- Predict future commodity prices using the fitted model.

Business Significance

This study provides valuable insights for investors, financial analysts, and business strategists on the volatility patterns of Arvind Fashion's stock. It helps in making informed investment decisions, identifying periods of high volatility, and strategizing entry and exit points. The study also focuses on commodity price analysis, helping investors understand long-term price movements and relationships between different commodities.

The study also aids in risk management by detecting ARCH/GARCH effects, which provide insights into the risk associated with Arvind Fashion's stock. This information allows investors to hedge against potential losses and manage risks associated with price fluctuations, ensuring more stable financial planning.

The findings can be used for strategic business planning, aligning investments with predicted market trends. Companies that depend on commodities can use price predictions to plan procurement strategies, negotiate better contracts, and optimize inventory levels.

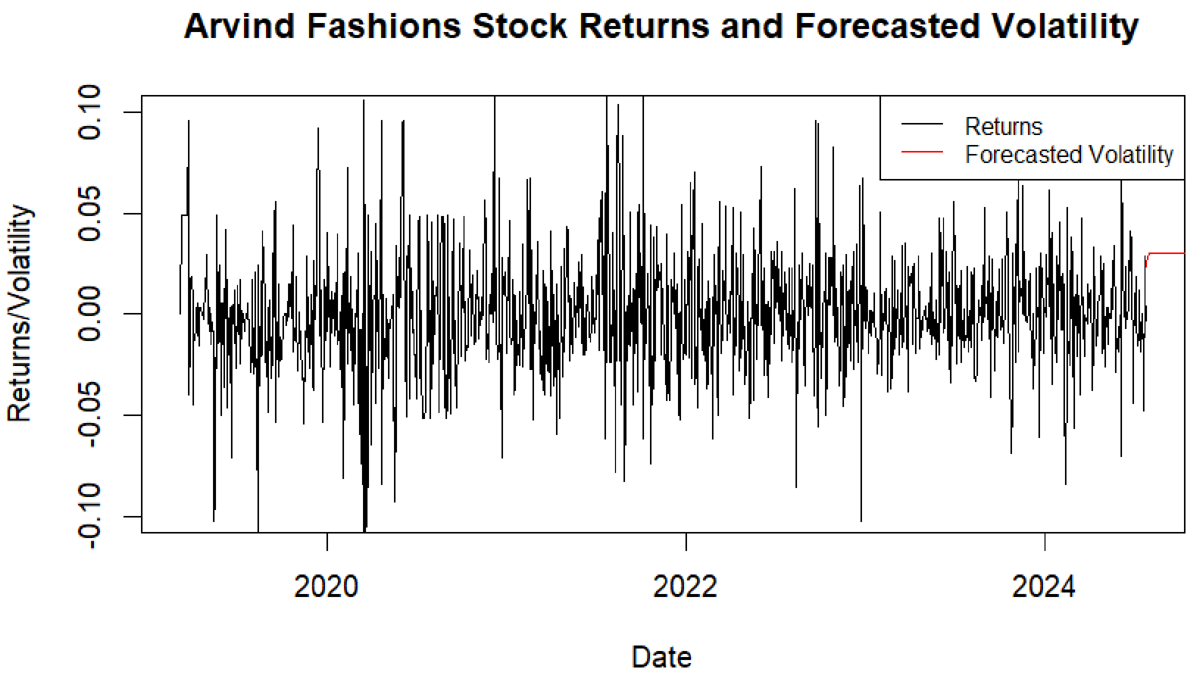
The ARCH/GARCH model enriches financial models with more accurate volatility estimates, leading to better forecasting and valuation models. The use of VECM or VAR models improves the accuracy of financial models that incorporate commodity prices, enhancing the reliability of forecasts and economic analyses.

The study offers a competitive advantage by enabling businesses and investors to anticipate market trends and manage risks effectively, leading to better financial performance and market positioning. Additionally, the findings can provide valuable insights for policymakers and regulators to understand market dynamics and develop policies that ensure market stability and protect investors.

Results and Interpretation using R

- **Check the ARCH/GARCH effects and plot the stock returns and forecasted volatility.**

```
# Check for ARCH/GARCH effects
> arch_test <- ArchTest(data$Returns, lags = 1)
> print(arch_test$p.value)
 Chi-squared
4.115402e-20
# Fit a GARCH(1,1) model
> spec <- ugarchspec(
+   variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
+   mean.model = list(armaOrder = c(0, 0))
+ )
> model <- ugarchfit(spec = spec, data = data$Returns)
# Forecast three-month (90 days) volatility
> forecasts <- ugarchforecast(model, n.ahead = 90)
> volatility_forecasts <- sigma(forecasts)
# Create a data frame for plotting
> data_plot <- data.frame(Date = data$date, Returns = data$Returns)
> forecast_dates <- seq.Date(from = as.Date(tail(data$date, 1)), by = "
days", length.out = 90)
> forecast_data <- data.frame(Date = forecast_dates, Volatility = as.nu
meric(volatility_forecasts))
# Plot returns and forecasted volatility
> plot(data_plot$Date, data_plot$Returns, type = "l", main = "Arvind Fa
shions Stock Returns and Forecasted Volatility", xlab = "Date", ylab =
"Returns/Volatility", col = "black", ylim = c(-0.1, 0.1))
> lines(forecast_data$Date, forecast_data$Volatility, col = "red")
> legend("topright", legend = c("Returns", "Forecasted Volatility"), co
l = c("black", "red"), lty = 1, cex = 0.8)
```

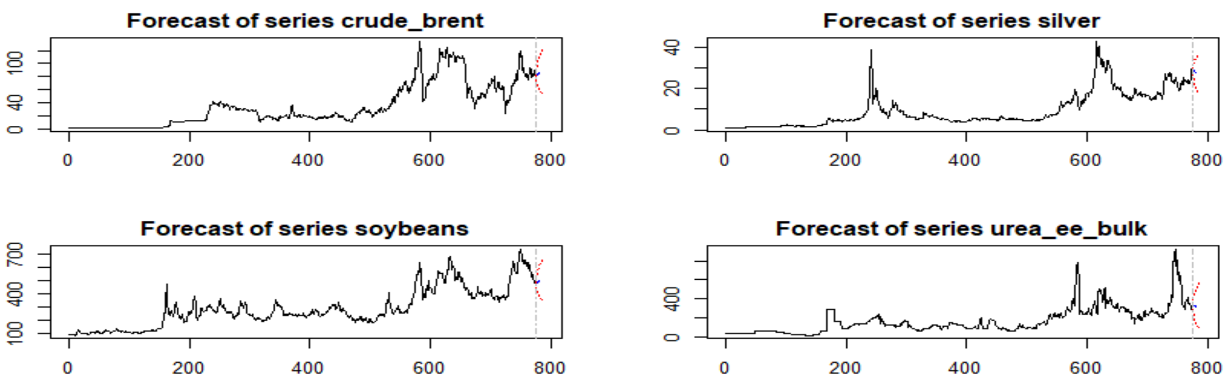


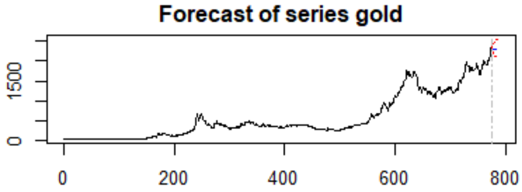
Interpretation:

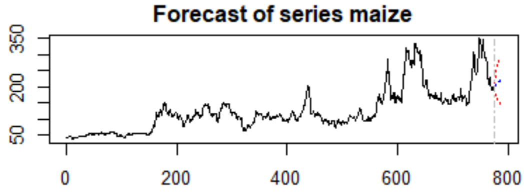
The script checks for ARCH effects in stock returns, fits a GARCH(1,1) model, and forecasts volatility for the next 90 days. It tests for ARCH effects in the returns series, with a low p-val ue indicating strong evidence of ARCH effects. The spec <- ugarchspec function is used to sp ecify and fit a GARCH(1,1) model to the returns data. The `sGARCH` model is a standard G ARCH model. The `ugarchforecast` function generates volatility forecasts for the next 90 day s. A data frame is created for plotting the returns and forecasted volatility. The plot shows his torical returns in black, while the red line represents the forecasted volatility for the next 90 d ays. The high frequency of spikes in historical returns suggests significant volatility in the sto ck's performance. The GARCH model captures this volatility and provides a forecast for the next 90 days. The forecasted volatility appears to be relatively stable with a slight upward tre nd. This analysis helps in understanding the stock's past behavior and future risk, useful for in vestors and risk managers. The script also creates a data frame for plotting the returns and for ecasted volatility on the same graph.

- **Estimating the VECM and plotting the model.**

```
# Estimating the VECM
> vecm <- cajorls(vecm.model, r = 1)  # r is the number of cointegratio
n vectors
> summary(vecm)
     Length Class  Mode
rlm  12     mlm    list
beta  7     -none- numeric
# Extracting the coefficients from the VECM model
> vecm_coefs <- cajorls(vecm.model, r = 1)$rlm$coefficients
# Creating a VECM model for prediction
> vecm_pred <- vec2var(vecm.model, r = 1)
# Forecasting 10 steps ahead
> forecast <- predict(vecm_pred, n.ahead = 12)
# Plotting the forecast
> par(mar = c(4, 4, 2, 2))  # Adjust margins: c(bottom, left, top, righ
t)
> plot(forecast)
```







The script checks for ARCH effects in stock returns, fits a GARCH(1,1) model, and forecasts volatility for the next 90 days. It tests for ARCH effects in the returns series, with a low p-val ue indicating strong evidence of ARCH effects. The spec <- ugarchspec function is used to sp ecify and fit a GARCH(1,1) model to the returns data. The `sGARCH` model is a standard G ARCH model. The `ugarchforecast` function generates volatility forecasts for the next 90 day s. A data frame is created for plotting the returns and forecasted volatility. The plot shows his torical returns in black, while the red line represents the forecasted volatility for the next 90 d ays. The high frequency of spikes in historical returns suggests significant volatility in the sto ck's performance. The GARCH model captures this volatility and provides a forecast for the next 90 days. The forecasted volatility appears to be relatively stable with a slight upward tre nd. This analysis helps in understanding the stock's past behavior and future risk, useful for in vestors and risk managers. The script also creates a data frame for plotting the returns and for ecasted volatility on the same graph.

Results and Interpretation using Python

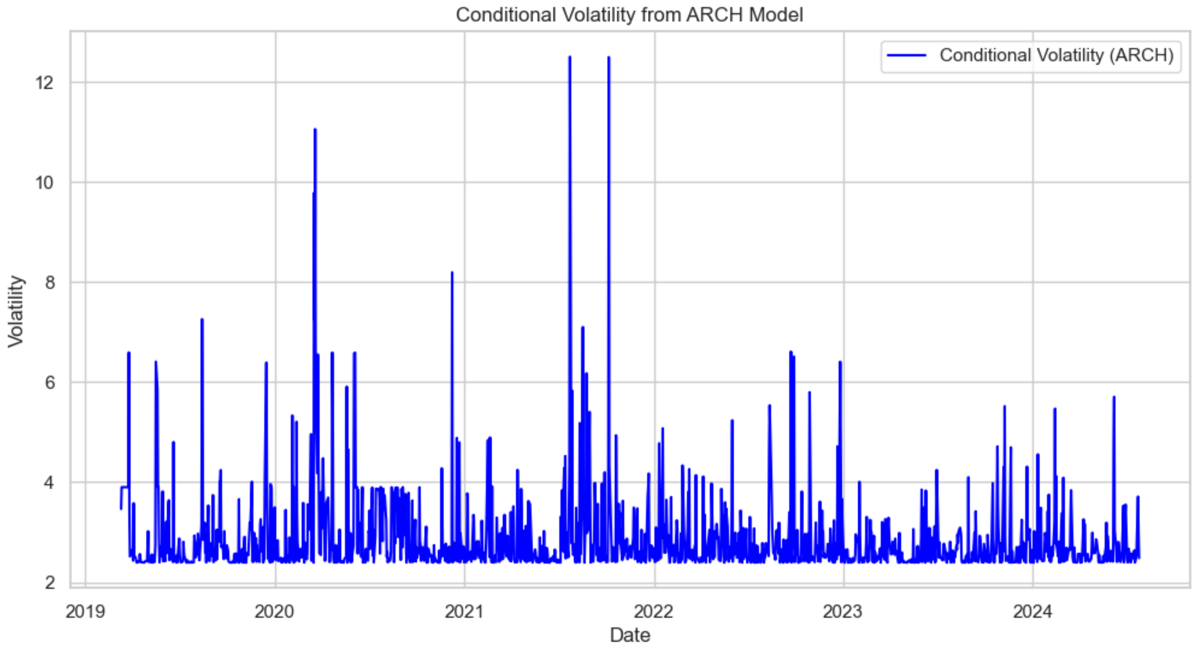
- **Fitting the ARCH model, plotting the conditional volatility from the ARCH model and conducting a Ljung-Box Test for residuals.**

```
market = data["Adj Close"]
returns = 100 * market.pct_change().dropna()  # Convert to percentage
returns
print("\nFitting ARCH Model...")
arch_model_fit = arch_model(returns, vol='ARCH', p=1).fit(disp='off')
print("ARCH Model Summary:")
print(arch_model_fit.summary())
```

```
Fitting ARCH Model...
ARCH Model Summary:
                      Constant Mean - ARCH Model Results
==============================================================================
Dep. Variable:              Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                       ARCH   Log-Likelihood:               -3243.70
Distribution:                  Normal   AIC:                           6493.41
Method:            Maximum Likelihood   BIC:                           6508.97
                                        No. Observations:                 1325
Date:                Fri, Jul 26 2024   Df Residuals:                     1324
Time:                        00:00:34   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|   95.0% Conf. Int.
----------------------------------------------------------------------------
mu            -0.0328  7.481e-02     -0.439      0.661 [ -0.179,  0.114]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|   95.0% Conf. Int.
----------------------------------------------------------------------------
omega          5.7173      0.507     11.272  1.797e-29 [  4.723,  6.711]
alpha[1]       0.3749  8.320e-02      4.505  6.622e-06 [  0.212,  0.538]
============================================================================

Covariance estimator: robust
```

```
# Plot the conditional volatility from the ARCH model
plt.figure(figsize=(12, 6))
plt.plot(arch_model_fit.conditional_volatility, label='Conditional Volatility
(ARCH)', color='blue')
plt.title('Conditional Volatility from ARCH Model')
plt.xlabel('Date')
plt.ylabel('Volatility')
```

```
plt.legend()
plt.grid(True)
plt.show()
```




```
ljungbox_arch = acorr_ljungbox(arch_model_fit.resid, lags=[10])
print("\nLjung-Box Test for ARCH Model Residuals:")
print(ljungbox_arch)
```

```
Ljung-Box Test for ARCH Model Residuals:
      lb_stat  lb_pvalue
10  23.416786   0.009308
```

Interpretation:

The ARCH model provides insights into the model fit and the significance of its parameters. The mean model is estimated to be -0.0328 with a p-value of 0.661, indicating it is not statistically significant at conventional levels. The volatility model has a high constant term (omega) and an ARCH term (alpha[1]) with a p-value close to zero. The model fit is -3243.70, with AIC and BIC values of 6493.41 and 6508.97, respectively, for model comparison purposes. The R-squared value is close to zero, which is common in volatility modeling as it focuses on the variance rather than the mean. The conditional volatility plot shows the estimated volatility over time, with spikes in volatility corresponding to periods of higher market turbulence. The ARCH model effectively captures the volatility over time. The Ljung-Box test for residuals indicates that there may be some remaining autocorrelation in the residuals at lag 10, suggesting that a more complex model like GARCH or adding more lags might better capture the dynamics of volatility. In conclusion, the ARCH(1) model significantly captures the time-varying volatility in stock returns, as evidenced by the significant coefficients and the conditional volatility plot. However, the Ljung-Box test indicates that a more complex model might be needed for a better fit.

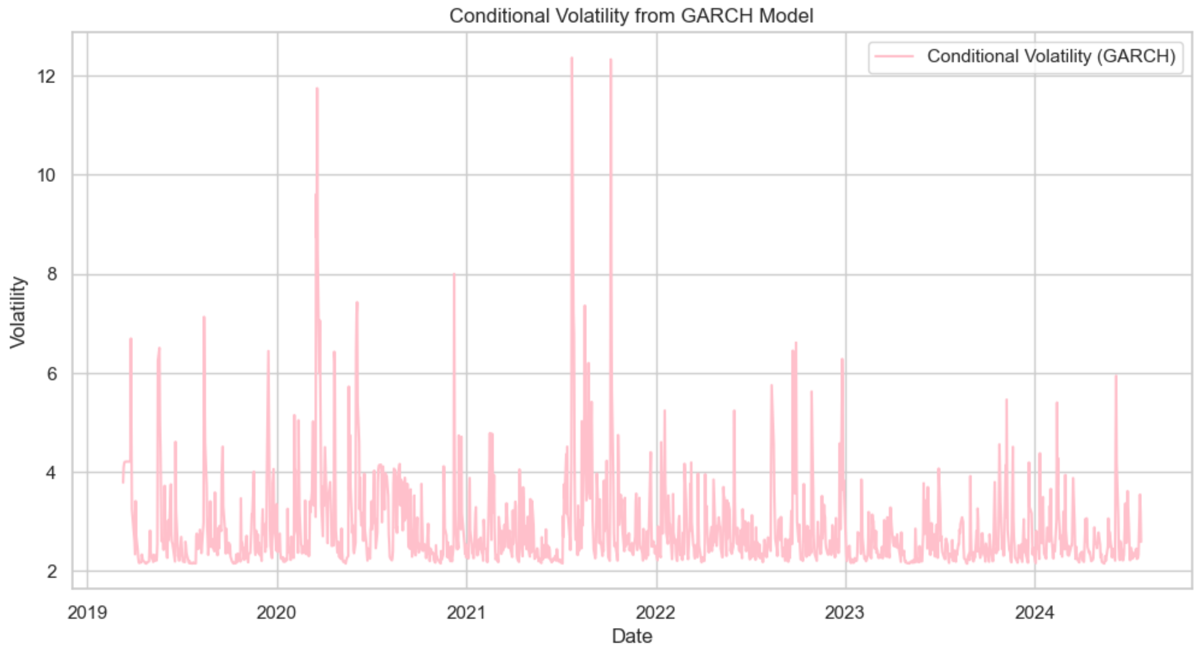
- **Fitting the GARCH model, plotting the conditional volatility from the GARCH model and conducting a Ljung-Box Test for residuals.**

```
print("\nFitting GARCH Model...")
garch_model_fit = arch_model(returns, vol='Garch', p=1, q=1).fit(disp='off')
print("GARCH Model Summary:")
print(garch_model_fit.summary())
```

```
Fitting GARCH Model...
GARCH Model Summary:
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:              Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -3227.36
Distribution:                  Normal   AIC:                           6462.73
Method:            Maximum Likelihood   BIC:                           6483.49
                                        No. Observations:                 1325
Date:                Fri, Jul 26 2024   Df Residuals:                     1324
Time:                        00:01:29   Df Model:                            1
                               Mean Model
========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
------------------------------------------------------------------------
mu            -0.0381  7.117e-02     -0.536      0.592 [ -0.178,  0.101]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|     95.0% Conf. Int.
----------------------------------------------------------------------------
omega          3.1967      0.918      3.483  4.949e-04    [  1.398,  4.995]
alpha[1]       0.3593      0.102      3.527  4.198e-04    [  0.160,  0.559]
beta[1]        0.3055      0.148      2.064  3.904e-02 [1.537e-02,  0.596]
============================================================================

Covariance estimator: robust
```

```
plt.figure(figsize=(12, 6))
plt.plot(garch_model_fit.conditional_volatility, label='Conditional Volatility
(GARCH)', color='pink')
plt.title('Conditional Volatility from GARCH Model')
plt.xlabel('Date')
plt.ylabel('Volatility')
```

```
plt.legend()
plt.grid(True)
plt.show()
```




```
ljungbox_garch = acorr_ljungbox(garch_model_fit.resid, lags=[10])
print("\nLjung-Box Test for GARCH Model Residuals:")
print(ljungbox_garch)
```

```
Ljung-Box Test for GARCH Model Residuals:

      lb_stat  lb_pvalue

10  23.416786   0.009308
```

Interpretation:

The analysis includes fitting a GARCH(1,1) model to a time series of returns, plotting conditi onal volatility, and conducting a Ljung-Box test on residuals. The GARCH model shows that the mean model explains little to no variation in the dependent variable, with a log-likelihood of -3227.36. The mean model has a coefficient of -0.0381, indicating it is not statistically sign ificant. The volatility model shows significant contributions from both the ARCH and GARC H components, indicating past squared returns and past volatility contribute to current volatili ty. The conditional volatility plot shows spikes in volatility, especially noticeable around 2021 and 2022, suggesting periods of increased market uncertainty or volatility. The Ljung-Box tes t for residuals shows significant autocorrelation in the residuals, suggesting that the GARCH model may not have fully captured all dependencies in the data. The results suggest that while the GARCH(1,1) model captures some volatility clustering in the data, the presence of autoco rrelation in the residuals suggests that the model may be inadequate. Further refinement or ad ditional explanatory variables might be needed to better capture the dynamics of the series.

- **Perform the ADF test for each column, conduct Johansen's Co-integration test, forecast using VAR/VECM model and plot the forecast.**

```
# Loop through each column and perform the ADF test
for col in columns_to_test:
    adf_result = adfuller(commodity_data[col])
    p_value = adf_result[1]  # Extract p-value for the test
    print(f"\nADF test result for column: {col}\n")
    print(f"Test Statistic: {adf_result[0]}")
    print(f"P-value: {p_value}")
    print(f"Critical Values: {adf_result[4]}")
```

```
ADF test result for column: crude_brent

Test Statistic: -1.5078661910935425
P-value: 0.5296165197702358
Critical Values: {'1%': -3.439006442437876, '5%': -2.865360521688131, '10%': -
2.5688044403756587}

ADF test result for column: soybeans

Test Statistic: -2.42314645274189
P-value: 0.13530977427790403
Critical Values: {'1%': -3.4388599939707056, '5%': -2.865295977855759, '10%':
-2.5687700561872413}

ADF test result for column: gold

Test Statistic: 1.3430517021933006
P-value: 0.9968394353612382
Critical Values: {'1%': -3.4389608473398194, '5%': -2.8653404270188476, '10%':
-2.568793735369693}

ADF test result for column: silver

Test Statistic: -1.3972947107462221
P-value: 0.5835723787985763
Critical Values: {'1%': -3.438915730045254, '5%': -2.8653205426302253, '10%':
-2.5687831424305845}
...

Test Statistic: -2.499023881611955
P-value: 0.11571200558506417
Critical Values: {'1%': -3.438915730045254, '5%': -2.8653205426302253, '10%':
-2.5687831424305845}
```

```
# Co-Integration Test (Johansen's Test)
# Perform Johansen's Co-Integration Test
```

```python
johansen_test = coint_johansen(commodity_data, det_order=0, k_ar_diff=1)

# Summary of the Co-Integration Test
print("\nJohansen Test Results:\n")
print(f"Eigenvalues:\n{johansen_test.eig}\n")
print(f"Trace Statistic:\n{johansen_test.lr1}\n")
print(f"Critical Values (5% level):\n{johansen_test.cvt[:, 1]}\n")
```

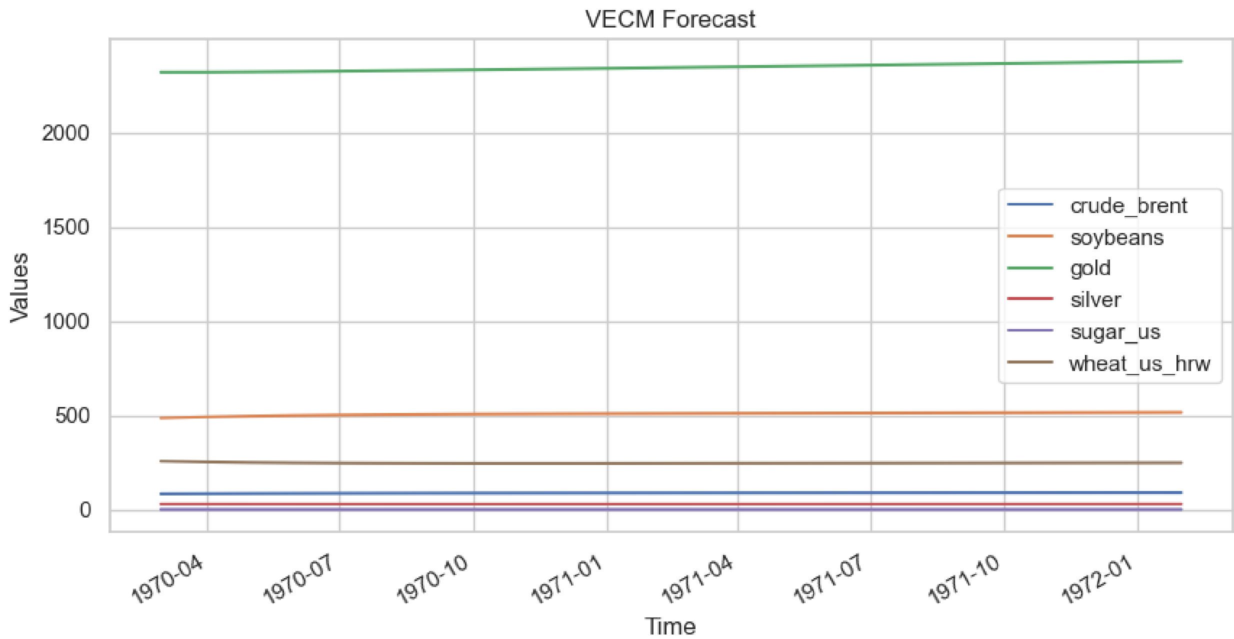
```
Johansen Test Results:

Eigenvalues:
[0.11398578 0.06876906 0.04867237 0.02710411 0.01899217 0.00405351]

Trace Statistic:
[226.10476071 132.67556204  77.67212223  39.15182203  17.93864778
   3.13567067]

Critical Values (5% level):
[95.7542 69.8189 47.8545 29.7961 15.4943  3.8415]
```

```python
# Determine the number of co-integrating relationships (r) based on the test
r = 2  # Replace with the actual number from the test results

if r > 0:
    # If co-integration exists, estimate the VECM model
    vecm_model = VECM(commodity_data, k_ar_diff=1, coint_rank=r, 
deterministic='co')
    vecm_fitted = vecm_model.fit()
    # Summary of the VECM model
    print(vecm_fitted.summary())
    # Extracting coefficients from the VECM model
    print("Alpha Coefficients:\n", vecm_fitted.alpha)
    print("Beta Coefficients:\n", vecm_fitted.beta)
    print("Gamma Coefficients:\n", vecm_fitted.gamma)
    # Forecasting using the VECM model
    forecast = vecm_fitted.predict(steps=24)
    # Convert forecast to a DataFrame for plotting
    forecast_df = pd.DataFrame(forecast, 
index=pd.date_range(start=commodity_data.index[-1], periods=25, freq='M')[1:], 
columns=commodity_data.columns)
    # Plotting the forecast
    forecast_df.plot(figsize=(10, 5))
    plt.title('VECM Forecast')
    plt.xlabel('Time')
    plt.ylabel('Values')
    plt.show()

else:
    # If no co-integration exists, proceed with Unrestricted VAR Analysis
    var_model = VAR(commodity_data)
    var_fitted = var_model.fit(maxlags=10, ic='aic')
```

```
    # Summary of the VAR model
    print(var_fitted.summary())

    # Granger causality test
    for col in commodity_data.columns:
        granger_result = var_fitted.test_causality(causing=col, caused=[c for
c in commodity_data.columns if c != col])
        print(f"Granger causality test for {col}:\n",
granger_result.summary())

    # Forecasting using the VAR model
    var_forecast = var_fitted.forecast(var_fitted.y, steps=24)
    var_forecast_df = pd.DataFrame(var_forecast,
index=pd.date_range(start=commodity_data.index[-1], periods=25, freq='M')[1:],
columns=commodity_data.columns)

    # Plotting the forecast
    var_forecast_df.plot(figsize=(10, 5))
    plt.title('VAR Forecast')
    plt.xlabel('Time')
    plt.ylabel('Values')
    plt.show()
```

```
Det. terms outside the coint. relation & lagged endog. parameters for equation
crude_brent
===============================================================================
=====
                      coef    std err          z      P>|z|      [0.025      0
.975]
-------------------------------------------------------------------------------
-----
const              -0.0807      0.178     -0.454      0.650      -0.429
0.268
L1.crude_brent      0.3217      0.035      9.078      0.000       0.252
0.391
L1.soybeans         0.0127      0.007      1.768      0.077      -0.001
0.027
L1.gold            -0.0032      0.006     -0.523      0.601      -0.015
0.009
L1.silver          -0.0971      0.148     -0.655      0.512      -0.387
0.193
L1.sugar_us        -2.5861      4.026     -0.642      0.521     -10.477
5.305
L1.wheat_us_hrw     0.0107      0.011      0.966      0.334      -0.011
0.032
Det. terms outside the coint. relation & lagged endog. parameters for equation
soybeans
```

```
==============================================================================
=====
                      coef    std err          z      P>|z|      [0.025      0
.975]
------------------------------------------------------------------------------
-----
const               2.9362      0.971      3.023      0.003       1.033
4.840
L1.crude_brent      0.2246      0.194      1.160      0.246      -0.155
0.604
L1.soybeans         0.1574      0.039      4.015      0.000       0.081
0.234
L1.gold            -0.0175      0.033     -0.527      0.598      -0.083
0.048
L1.silver           0.5257      0.809      0.649      0.516      -1.061
2.112
L1.sugar_us         4.7482     21.995      0.216      0.829     -38.361      4
7.857
L1.wheat_us_hrw    -0.0103      0.061     -0.171      0.864      -0.129
0.108
Det. terms outside the coint. relation & lagged endog. parameters for equation
gold
==============================================================================
=====
                      coef    std err          z      P>|z|      [0.025      0
.975]
...
 [ 2.39361823e-04  1.03771634e-04  4.87823966e-05 -6.25153954e-04
   1.71918426e-01  7.22511432e-05]
 [-4.04702814e-02 -4.86661338e-02  2.84172038e-03  8.47031903e-01
   8.13821890e+00  2.70551307e-01]]
```



- **Check all available attributes and print VECM fitted summary**

```
print(dir(vecm_fitted))
print(vecm_fitted.summary())
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__fo
rmat__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '
__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '_
_new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__
', '__str__', '__subclasshook__', '__weakref__', '_cache', '_chol_sigma_u', '_
cov_sigma', '_delta_x', '_delta_y_1_T', '_make_conf_int', '_y_lag1', 'alpha',
'beta', 'coint_rank', 'conf_int_alpha', 'conf_int_beta', 'conf_int_det_coef',
'conf_int_det_coef_coint', 'conf_int_gamma', 'const', 'const_coint', 'cov_para
ms_default', 'cov_params_wo_det', 'cov_var_repr', 'dates', 'det_coef', 'det_co
ef_coint', 'deterministic', 'exog', 'exog_coefs', 'exog_coint', 'exog_coint_co
efs', 'first_season', 'fittedvalues', 'gamma', 'irf', 'k_ar', 'lin_trend', 'li
n_trend_coint', 'llf', 'ma_rep', 'model', 'names', 'neqs', 'nobs', 'orth_ma_re
p', 'plot_data', 'plot_forecast', 'predict', 'pvalues_alpha', 'pvalues_beta',
'pvalues_det_coef', 'pvalues_det_coef_coint', 'pvalues_gamma', 'resid', 'seaso
nal', 'seasons', 'sigma_u', 'stderr_alpha', 'stderr_beta', 'stderr_coint', 'st
derr_det_coef', 'stderr_det_coef_coint', 'stderr_gamma', 'stderr_params', 'sum
mary', 'test_granger_causality', 'test_inst_causality', 'test_normality', 'tes
t_whiteness', 'tvalues_alpha', 'tvalues_beta', 'tvalues_det_coef', 'tvalues_de
t_coef_coint', 'tvalues_gamma', 'var_rep', 'y_all']

Det. terms outside the coint. relation & lagged endog. parameters for equation
crude_brent
==============================================================================
=====
                      coef    std err          z      P>|z|      [0.025      0
.975]
------------------------------------------------------------------------------
-----
const              -0.0807      0.178     -0.454      0.650      -0.429
0.268
L1.crude_brent      0.3217      0.035      9.078      0.000       0.252
0.391
L1.soybeans         0.0127      0.007      1.768      0.077      -0.001
0.027
L1.gold            -0.0032      0.006     -0.523      0.601      -0.015
0.009
L1.silver          -0.0971      0.148     -0.655      0.512      -0.387
0.193
L1.sugar_us        -2.5861      4.026     -0.642      0.521     -10.477
5.305
L1.wheat_us_hrw     0.0107      0.011      0.966      0.334      -0.011
0.032
Det. terms outside the coint. relation & lagged endog. parameters for equation
soybeans
==============================================================================
=====
```

```
                      coef    std err          z      P>|z|      [0.025      0
.975]
-------------------------------------------------------------------------------
-----
const               2.9362      0.971      3.023      0.003       1.033       
4.840
L1.crude_brent      0.2246      0.194      1.160      0.246      -0.155       
0.604
L1.soybeans         0.1574      0.039      4.015      0.000       0.081       
0.234
L1.gold            -0.0175      0.033     -0.527      0.598      -0.083       
0.048
L1.silver           0.5257      0.809      0.649      0.516      -1.061       
2.112
L1.sugar_us         4.7482     21.995      0.216      0.829     -38.361      4
7.857
L1.wheat_us_hrw    -0.0103      0.061     -0.171      0.864      -0.129       
0.108
Det. terms outside the coint. relation & lagged endog. parameters for equation 
gold
===============================================================================
=====
                      coef    std err          z      P>|z|      [0.025      0
.975]
...
beta.4          2.1329     40.139      0.053      0.958     -76.537      80.803
beta.5         -7.7074      0.049   -155.810      0.000      -7.804      -7.610
beta.6         -1.3682      0.116    -11.747      0.000      -1.597      -1.140
==============================================================================
```

Interpretation:

The analysis includes the results of the Augmented Dickey-Fuller (ADF) test for stationarity, Johansen's co-integration test, and the Vector Error Correction Model (VECM) forecasting. T he ADF test checks for a unit root in a time series, indicating that the series are non-stationary at levels. The Johansen test determines the number of co-integrating relationships among a se t of non-stationary time series by providing eigenvalues, trace statistics, and critical values. T he trace statistics exceed the critical values up to r = 2, indicating two co-integrating relations hips among the series.

The VECM forecasting model is appropriate given the presence of co-integration, accounting for both long-term equilibrium relationships and short-term dynamics. The fitted VECM mod el includes coefficients for deterministic terms and lagged endogenous variables. The plot pro vided shows the forecasted values from the VECM model for different commodities over a ti me horizon extending to 1972, indicating relatively stable forecasts.

The interpretation of the results is that the ADF Test Results suggest that the series are non-stationary at levels. The Johansen Test Results indicate two co-integrating relationships, implying that despite individual series being non-stationary, there exists a long-term equilibrium relationship among them. The VECM Forecast provides a forecast based on both long-term equilibrium and short-term dynamics, but the flat lines in the plot suggest that the model does not predict strong short-term variations. Further analysis might include investigating the stability of the VECM model and assessing forecast accuracy over different periods.

Recommendation

The analysis presented suggests several recommendations for Arvind Fashion's stock volatility analysis. These include enhancing the model specification, refining the GARCH model to capture asymmetries in volatility, incorporating exogenous variables like macroeconomic indicators and sector-specific variables, developing risk management and hedging strategies, periodic model review and updates, scenario analysis, and integrating external data such as global economic indicators and climate and environmental factors.

For commodity price analysis, enhanced data analysis and model selection should be considered, with stationarity and transformation achieved through differentiating or detrending the data. Model validation should be done using out-of-sample testing to ensure predictive accuracy. Integrating with external data, such as global economic indicators and climate and environmental factors, can also help in long-term investment decisions and supply chain and inventory management.

Strategic business and investment planning can benefit from insights from co-integration analysis for portfolio diversification and risk management. Price forecasts can be used to optimize procurement strategies, manage inventory levels, and negotiate better contracts. Policy and regulatory considerations should consider market stability policies and regulatory measures to mitigate market volatility.

General recommendations include continuous learning and adaptation, staying updated with financial innovations, training analysts and stakeholders in advanced statistical methods and financial modeling, cross-disciplinary collaboration with experts, and encouraging interdisciplinary research to explore new methodologies and approaches. These recommendations aim to enhance the robustness and relevance of the analyses, providing stakeholders with deeper insights and better tools for decision-making in volatile and interconnected markets.

R Codes

```r
#Install Packages
install.packages("rugarch")
install.packages("FinTS")

# Load required libraries
library(tidyquant)
library(dplyr)
library(lubridate)
library(tseries)
library(forecast)
library(rugarch)

# Download data from Yahoo Finance
data <- tq_get('ARVINDFASN.NS', from = "2019-01-01", to = "2024-07-25")

# Ensure adjusted column is an xts object and calculate returns
data_xts <- xts(data$adjusted, order.by = data$date)
returns <- dailyReturn(data_xts, type = "log")
data <- data %>% mutate(Returns = as.numeric(returns)) %>% na.omit()

library(FinTS)

# Check for ARCH/GARCH effects
arch_test <- ArchTest(data$Returns, lags = 1)
print(arch_test$p.value)

# Fit a GARCH(1,1) model
spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0))
)
model <- ugarchfit(spec = spec, data = data$Returns)
```

```
# Forecast three-month (90 days) volatility
forecasts <- ugarchforecast(model, n.ahead = 90)
volatility_forecasts <- sigma(forecasts)

# Create a data frame for plotting
data_plot <- data.frame(Date = data$date, Returns = data$Returns)
forecast_dates <- seq.Date(from = as.Date(tail(data$date, 1)), by = "days", length.out = 90)
forecast_data <- data.frame(Date = forecast_dates, Volatility = as.numeric(volatility_forecast
s))

# Plot returns and forecasted volatility
plot(data_plot$Date, data_plot$Returns, type = "l", main = "Arvind Fashions Stock Returns a
nd Forecasted Volatility", xlab = "Date", ylab = "Returns/Volatility", col = "black", ylim = c(
-0.1, 0.1))
lines(forecast_data$Date, forecast_data$Volatility, col = "red")
legend("topright", legend = c("Returns", "Forecasted Volatility"), col = c("black", "red"), lty
= 1, cex = 0.8)

#(b) VAR, VECM
setwd("C:\\Users\\Ferah Shan\\Downloads")
getwd()

# Load necessary libraries
library(readxl)
library(dplyr)
library(janitor)
library(urca)
library(vars)

df = read_excel('pinksheet.xlsx', sheet="Monthly Prices", skip = 6)

# Rename the first column to "Date"
colnames(df)[1] <- 'Date'
```

```
# Convert the Date column to Date format
df$Date <- as.Date(paste0(df$Date, "01"), format = "%YM%m%d")
str(df)

# Get the column numbers for each column
column_numbers <- setNames(seq_along(df), colnames(df))

commodity = df[,c(1,3,25,70,72,61,31)]

commodity = clean_names(commodity)

str(commodity)

# Use dplyr::select to avoid any conflicts and exclude the Date column
commodity_data <- dplyr::select(commodity, -date)

vecm.model <- ca.jo(commodity_data, ecdet = 'const', type  = 'eigen', K = 2, spec = 'transitor
y', dumvar = NULL)

summary(vecm.model)

# Estimating the VECM
vecm <- cajorls(vecm.model, r = 1)  # r is the number of cointegration vectors
summary(vecm)

# Extracting the coefficients from the VECM model
vecm_coefs <- cajorls(vecm.model, r = 1)$rlm$coefficients

# Creating a VECM model for prediction
vecm_pred <- vec2var(vecm.model, r = 1)

# Forecasting using the VECM
```

```
# Forecasting 10 steps ahead
forecast <- predict(vecm_pred, n.ahead = 12)

# Plotting the forecast
par(mar = c(4, 4, 2, 2))  # Adjust margins: c(bottom, left, top, right)
plot(forecast)
```

Python Codes

(a) Checking for ARCH/GARCH effects

```
pip install arch

# Import required libraries
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from arch import arch_model
from statsmodels.stats.diagnostic import acorr_ljungbox
import seaborn as sns

# Set plotting style
sns.set(style="whitegrid")

data = yf.download('ARVINDFASN.NS', start='2019-01-01', end='2024-07-25')

print(data.head())
print(data.info())

print(data.columns)

market = data["Adj Close"]
returns = 100 * market.pct_change().dropna()  # Convert to percentage returns
```

```
print("\nFitting ARCH Model...")
arch_model_fit = arch_model(returns, vol='ARCH', p=1).fit(disp='off')
print("ARCH Model Summary:")
print(arch_model_fit.summary())

# Plot the conditional volatility from the ARCH model
plt.figure(figsize=(12, 6))
plt.plot(arch_model_fit.conditional_volatility, label='Conditional Volatility (ARCH)', color='
blue')
plt.title('Conditional Volatility from ARCH Model')
plt.xlabel('Date')
plt.ylabel('Volatility')
plt.legend()
plt.grid(True)
plt.show()

ljungbox_arch = acorr_ljungbox(arch_model_fit.resid, lags=[10])
print("\nLjung-Box Test for ARCH Model Residuals:")
print(ljungbox_arch)

print("\nFitting GARCH Model...")
garch_model_fit = arch_model(returns, vol='Garch', p=1, q=1).fit(disp='off')
print("GARCH Model Summary:")
print(garch_model_fit.summary())

plt.figure(figsize=(12, 6))
plt.plot(garch_model_fit.conditional_volatility, label='Conditional Volatility (GARCH)', colo
r='pink')
plt.title('Conditional Volatility from GARCH Model')
plt.xlabel('Date')
plt.ylabel('Volatility')
plt.legend()
plt.grid(True)
plt.show()
```

```
ljungbox_garch = acorr_ljungbox(garch_model_fit.resid, lags=[10])
print("\nLjung-Box Test for GARCH Model Residuals:")
print(ljungbox_garch)

print("\nFitting GARCH Model with additional parameters...")
am = arch_model(returns, vol="Garch", p=1, q=1, dist="Normal")
res = am.fit(update_freq=5)

forecast_mean = res.forecast().mean
forecast_residual_variance = res.forecast().residual_variance
forecast_variance = res.forecast().variance

print("\nForecast Mean (last 3 periods):")
print(forecast_mean.iloc[-3:])
print("Forecast Residual Variance (last 3 periods):")
print(forecast_residual_variance.iloc[-3:])
print("Forecast Variance (last 3 periods):")
print(forecast_variance.iloc[-3:])

print("\nForecasting 90 days ahead")
forecasts = res.forecast(horizon=90)
print("\n90-day Forecast Residual Variance (last 3 periods):")
print(forecasts.residual_variance.iloc[-3:])

print("\nAnalysis Summary:")
print("1. ARCH and GARCH models were successfully fitted to the returns data.")
print("2. Conditional volatility was plotted for both ARCH and GARCH models.")
print("3. Residuals were checked for autocorrelation using the Ljung-Box test.")
print("4. Forecasts were generated for a 90-day horizon, including variance and residual varia
nce.")
```

(b) Fitting VAR and VECM for commodities

```
import os
import pandas as pd
import numpy as np
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.vector_ar.vecm import coint_johansen, VECM
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_excel("C:\\Users\\Ferah Shan\\Downloads\\pinksheet.xlsx", sheet_name='Month
ly Prices', skiprows=6)

# Rename the first column to "Date"
df.rename(columns={df.columns[0]: 'Date'}, inplace=True)

# Convert the Date column to datetime format
df['Date'] = pd.to_datetime(df['Date'].astype(str) + '01', format='%YM%m%d')
print(df.info())  # Check the structure of the dataframe

# Select specific columns (Date and selected commodities)
commodity = df[['Date', 'CRUDE_BRENT', 'SOYBEANS', 'GOLD', 'SILVER', 'SUGAR_US'
, 'WHEAT_US_HRW']]

# Clean column names (optional, as Pandas automatically handles column names well)
commodity.columns = commodity.columns.str.strip().str.lower().str.replace(' ', '_').str.replace(
'(', '').str.replace(')', '')

print(commodity.info())  # Check the structure of the cleaned dataframe

# Remove the Date column for analysis
commodity_data = commodity.drop(columns=['date'])
```

```
# Column names to test (if you want to specify particular columns)
columns_to_test = commodity_data.columns

# Initialize counters and lists for stationary and non-stationary columns
non_stationary_count = 0
stationary_columns = []
non_stationary_columns = []

# Loop through each column and perform the ADF test
for col in columns_to_test:
    adf_result = adfuller(commodity_data[col])
    p_value = adf_result[1]  # Extract p-value for the test
    print(f"\nADF test result for column: {col}\n")
    print(f"Test Statistic: {adf_result[0]}")
    print(f"P-value: {p_value}")
    print(f"Critical Values: {adf_result[4]}")

# Check if the p-value is greater than 0.05 (commonly used threshold)
if p_value > 0.05:
        non_stationary_count += 1
        non_stationary_columns.append(col)
else:
        stationary_columns.append(col)

# Print the number of non-stationary columns and the lists of stationary and non-stationary co
lumns
print(f"\nNumber of non-stationary columns: {non_stationary_count}\n")
print(f"Non-stationary columns: {non_stationary_columns}\n")
print(f"Stationary columns: {stationary_columns}\n")

# Co-Integration Test (Johansen's Test)
# Perform Johansen's Co-Integration Test
johansen_test = coint_johansen(commodity_data, det_order=0, k_ar_diff=1)
```

```
# Summary of the Co-Integration Test
print("\nJohansen Test Results:\n")
print(f"Eigenvalues:\n{johansen_test.eig}\n")
print(f"Trace Statistic:\n{johansen_test.lr1}\n")
print(f"Critical Values (5% level):\n{johansen_test.cvt[:, 1]}\n")

# Determine the number of co-integrating relationships (r) based on the test
r = 2  # Replace with the actual number from the test results

if r > 0:
    # If co-integration exists, estimate the VECM model
    vecm_model = VECM(commodity_data, k_ar_diff=1, coint_rank=r, deterministic='co')
    vecm_fitted = vecm_model.fit()

    # Summary of the VECM model
    print(vecm_fitted.summary())

    # Extracting coefficients from the VECM model
    print("Alpha Coefficients:\n", vecm_fitted.alpha)
    print("Beta Coefficients:\n", vecm_fitted.beta)
    print("Gamma Coefficients:\n", vecm_fitted.gamma)

    # Forecasting using the VECM model
    forecast = vecm_fitted.predict(steps=24)

    # Convert forecast to a DataFrame for plotting
    forecast_df = pd.DataFrame(forecast, index=pd.date_range(start=commodity_data.index[-
1], periods=25, freq='M')[1:], columns=commodity_data.columns)

    # Plotting the forecast
    forecast_df.plot(figsize=(10, 5))
    plt.title('VECM Forecast')
    plt.xlabel('Time')
    plt.ylabel('Values')
```

```
    plt.show()

else:
    # If no co-integration exists, proceed with Unrestricted VAR Analysis
    var_model = VAR(commodity_data)
    var_fitted = var_model.fit(maxlags=10, ic='aic')

    # Summary of the VAR model
    print(var_fitted.summary())

    # Granger causality test
    for col in commodity_data.columns:
        granger_result = var_fitted.test_causality(causing=col, caused=[c for c in commodity_da
ta.columns if c != col])
        print(f"Granger causality test for {col}:\n", granger_result.summary())

    # Forecasting using the VAR model
    var_forecast = var_fitted.forecast(var_fitted.y, steps=24)
    var_forecast_df = pd.DataFrame(var_forecast, index=pd.date_range(start=commodity_data
.index[-1], periods=25, freq='M')[1:], columns=commodity_data.columns)

    # Plotting the forecast
    var_forecast_df.plot(figsize=(10, 5))
    plt.title('VAR Forecast')
    plt.xlabel('Time')
    plt.ylabel('Values')
    plt.show()

# Check all available attributes
print(dir(vecm_fitted))

print(vecm_fitted.summary())
```

References

1. www.github.com
2. www.geeksforgeeks.com
3. www.datacamp.com
4. www.yahoofinance.com
5. www.investing.com