VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6b- Time Series Analysis

(Part – A)

JYOTHIS KANIYAMPARAMBIL THANKACHAN

V01110144

Date of Submission: 25-07-2024

CONTENTS

INTRODUCTION

Investors, risk managers, and lawmakers all need to understand how volatile asset returns are in order to make smart decisions. Volatility, which is the change in the prices of assets, shows market risks and how stable investments are. The goal of this task is to use advanced econometric methods, especially the ARCH (Autoregressive Conditional Heteroskedasticity) and GARCH (Generalised Autoregressive Conditional Heteroskedasticity) models, to look at how volatile TESLA Corporation (TSLA) stock is.

TESLA is a major player in the tech industry and is known for its progress in artificial intelligence (AI) and graphics processing units (GPUs). The price of the company has gone up and down a lot, which makes it a great example of volatility research. This study looks at how volatile TSLA stock is to get a better idea of how its price changes and to guess what the future risk levels will be.

The results of this study will help us figure out how risky TESLA's stock is and can help us make business decisions and plan how to handle risk. The study's results will be shown on different graphs, like conditional volatility and predicted variance, which will give a full picture of TSLA's financial volatility.

OBJECTIVES

The main goals of this task are to use advanced economic methods to look at how volatile TESLA Corporation (TSLA) stock is and to make predictions about how volatile it will be in the future. In particular, the goals are to look for ARCH/GARCH effects, fit an ARCH/GARCH model, and guess the three-month volatility. Through detailed econometric analysis and projections, this organised method aims to give a full picture of how volatile TESLA's stock is and help people make smart decisions.

BUSINESS SIGNIFICANCE

To make smart business choices and handle financial risks, you need to understand and analyse stock volatility. In conclusion, being able to correctly predict and understand stock volatility gives you useful information that can help you make better investment choices, handle financial risks better, and back up smart business actions.

Result in Python Language

- **Fit an ARCH Model**

```
[57]: # Step 3: Fit an ARCH Model
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())
```

Result

```
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:               -2023.33
Distribution:                  Normal   AIC:                           4052.66
Method:            Maximum Likelihood   BIC:                           4066.53
                                        No. Observations:                  752
Date:                Thu, Jul 25 2024   Df Residuals:                      751
Time:                        09:07:37   Df Model:                            1
                                 Mean Model
=========================================================================
                 coef    std err          t      P>|t|   95.0% Conf. Int.
-------------------------------------------------------------------------
mu             0.0337      0.132      0.255      0.799 [ -0.225,  0.293]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega         12.7219      0.929     13.695  1.081e-42     [ 10.901, 14.543]
alpha[1]   1.8604e-13  3.534e-02  5.265e-12      1.000 [-6.926e-02,6.926e-02]
============================================================================

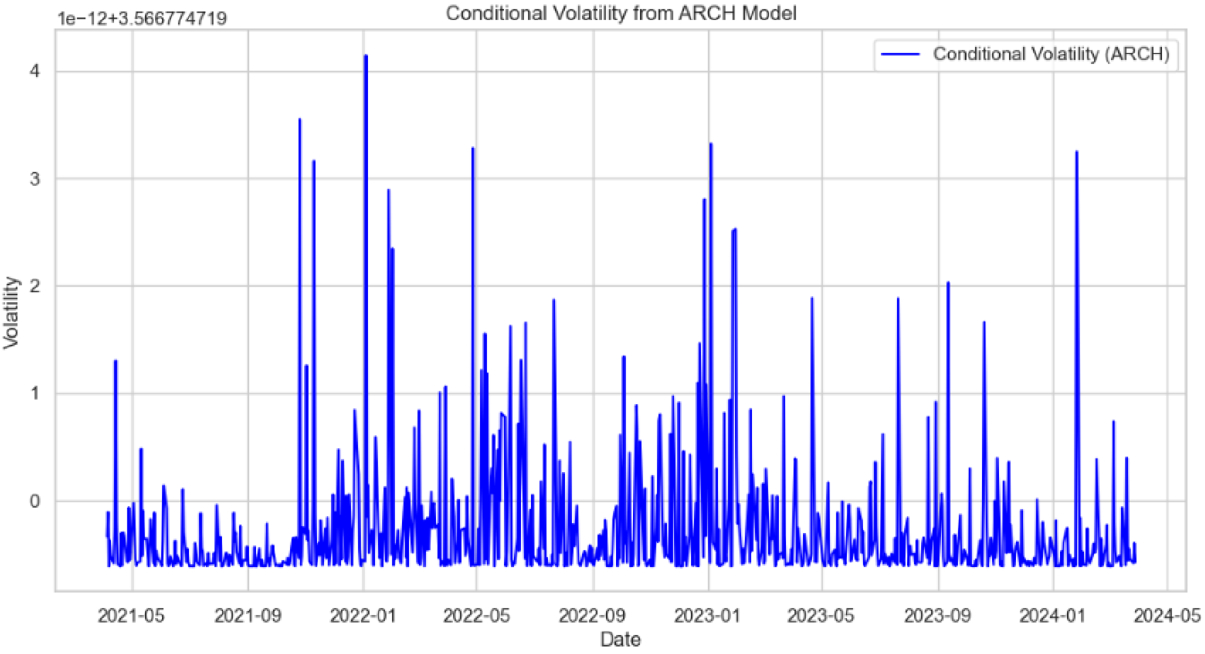
Covariance estimator: robust
```

Interpretation: The ARCH model helps to identify periods of high and low volatility.

- **Plot Conditional Volatility**:

```
[58]: # 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()
```

Result



Interpretation: Helps in understanding the variability and pattern of volatility over time.

- **Check Residuals for Autocorrelation**:

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

Result

```
Ljung-Box Test for GARCH Model Residuals:
     lb_stat  lb_pvalue
10  21.55807   0.017521
```

Interpretation: Significant p-values indicate autocorrelation, suggesting that the model may not fully capture the volatility dynamics.

- **Fit a GARCH Model**

```
# Step 4: Fit a GARCH Model
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())
```

Result

```
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:               -2004.58
Distribution:                Normal   AIC:                           4017.15
Method:          Maximum Likelihood   BIC:                           4035.64
                                      No. Observations:                  752
Date:              Thu, Jul 25 2024   Df Residuals:                      751
Time:                      09:07:44   Df Model:                            1
                               Mean Model
========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
------------------------------------------------------------------------
mu             0.0666      0.119      0.560      0.576 [ -0.167,  0.300]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.2196      0.163      1.350      0.177  [-9.914e-02,  0.538]
alpha[1]       0.0331  1.102e-02      3.001  2.690e-03 [1.147e-02,5.466e-02]
beta[1]        0.9495  1.536e-02     61.820      0.000      [  0.919,  0.980]
============================================================================

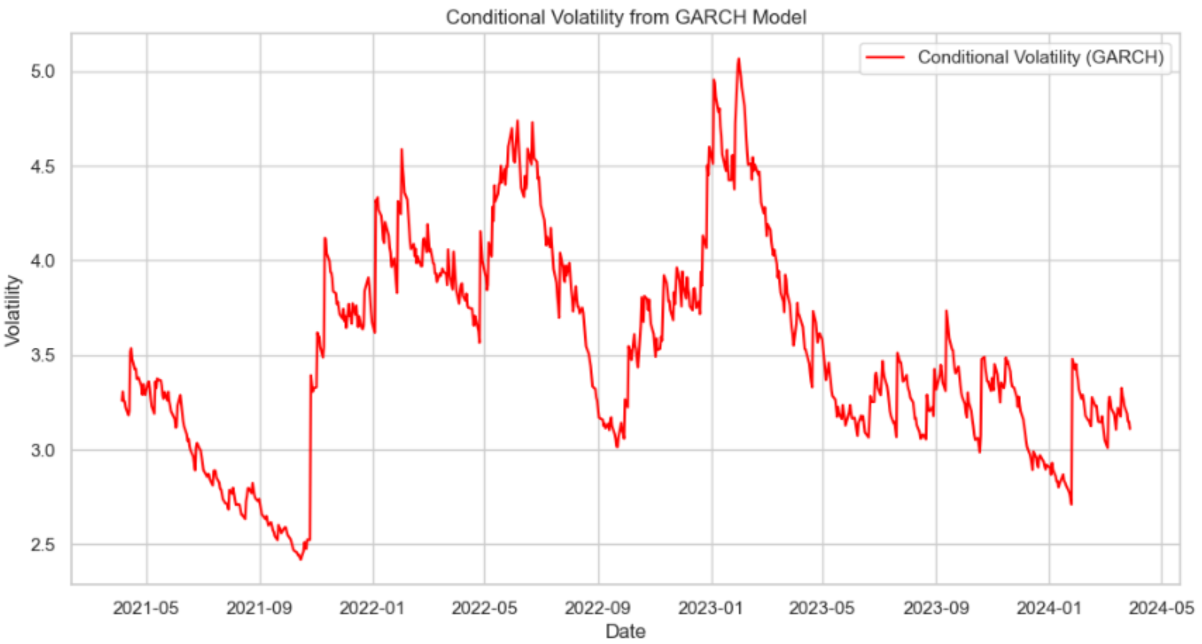
Covariance estimator: robust
```

Interpretation: The GARCH model accounts for more complex volatility patterns compared to the ARCH model.

- **Plot Conditional Volatility**:

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

Result



Interpretation: Provides insights into how the GARCH model captures volatility patterns over time.

- **Check Residuals for Autocorrelation**:

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

Interpretation: Like the ARCH model, significant p-values indicate residual autocorrelation, suggesting the need for further model refinement.

```
Ljung-Box Test for GARCH Model Residuals:
      lb_stat  lb_pvalue
10   21.55807   0.017521
```

- **Fit GARCH Model with Additional Parameters**

```
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)
```

Result

```
Fitting GARCH Model with additional parameters...
Iteration:      5,   Func. Count:     34,   Neg. LLF: 2006.0309912410062
Iteration:     10,   Func. Count:     63,   Neg. LLF: 2004.5755308148487
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2004.575528033089
            Iterations: 12
            Function evaluations: 72
            Gradient evaluations: 12
```

Interpretation: The inclusion of additional parameters and distribution assumptions helps in refining the forecast.

- **Print Forecast Details**:

```
# Print forecast details
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:])
```

Result

```
Forecast Mean (last 3 periods):
                 h.1
Date
2024-03-28  0.066586
Forecast Residual Variance (last 3 periods):
                 h.1
Date
2024-03-28  9.582995
Forecast Variance (last 3 periods):
                 h.1
Date
2024-03-28  9.582995
```

Interpretation: Indicates the expected future behavior of the stock's volatility.

- **Forecasting with a Horizon of 90 Days**

```
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:])
```

Result

```
Forecasting 90 days ahead...
```

```
90-day Forecast Residual Variance (last 3 periods):
                h.01      h.02      h.03      h.04      h.05      h.06  \
Date
2024-03-28  9.582995  9.635704  9.687495  9.738383  9.788385  9.837516

                h.07      h.08      h.09       h.10  ...       h.81      h.82  \
Date                                                 ...
2024-03-28  9.885792  9.933226  9.979834  10.025629  ...  11.866989  11.87991

                 h.83       h.84      h.85       h.86       h.87       h.88  \
Date
2024-03-28  11.892607  11.905082  11.91734  11.929384  11.941219  11.952848

                 h.89     h.90
Date
2024-03-28  11.964273  11.9755

[1 rows x 90 columns]
```

Interpretation: Forecasted residual variance provides insights into expected future volatility levels.

R Language

- Fit an ARCH Model

```
print("\nFitting ARCH Model...")
arch_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),
                        mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
                        distribution.model = "norm")

arch_fit <- ugarchfit(spec = arch_spec, data = returns)
print("ARCH Model Summary:")
print(arch_fit)
```

Result

```
[1] "ARCH Model Summary:"
> print(arch_fit)

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

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

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
omega     12.806    0.733284 17.463686        0
alpha1     0.000    0.030071  0.000001        1

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
omega     12.806    1.193164   10.733        0
alpha1     0.000    0.042758    0.000        1

LogLikelihood : -2026.039

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

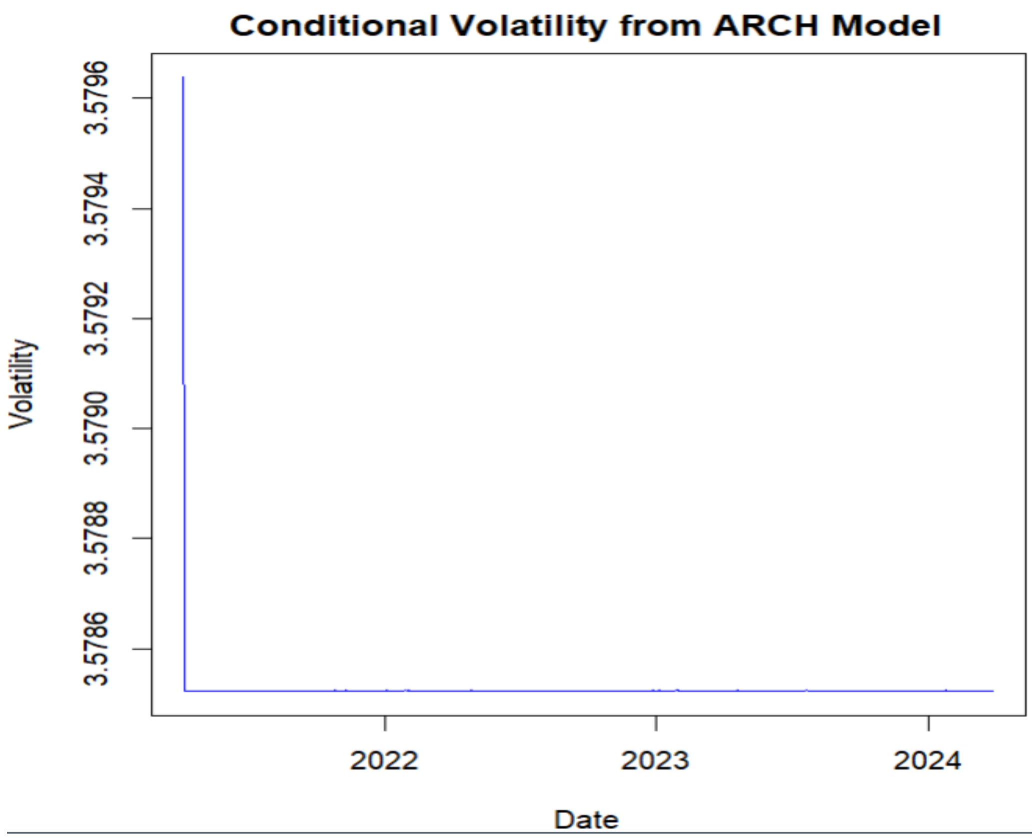
Akaike       5.3937
Bayes        5.4060
Shibata      5.3937
Hannan-Quinn 5.3985
```

- **Time series plot**

```
# Plot the conditional volatility from the ARCH model
## Extract conditional volatility
cond_volatility <- sigma(arch_fit)

# Create a time series plot for conditional volatility
# Use the index of the returns, which is aligned with the conditional volatility
plot(index(returns), cond_volatility, type = 'l',
     main = 'Conditional Volatility from ARCH Model',
     xlab = 'Date', ylab = 'Volatility', col = 'blue')
grid()
```

Result



- ```
  # Check residuals for autocorrelation
  arch_residuals <- residuals(arch_fit)
  arch_ljung_box <- Box.test(arch_residuals, lag = 10, type = "Ljung-Box")
  print("\nLjung-Box Test for ARCH Model Residuals:")
  print(arch_ljung_box)
  ```

Result

```
> print("\nLjung-Box Test for ARCH Model Residuals:
[1] "\nLjung-Box Test for ARCH Model Residuals:"
> print(arch_ljung_box)

	Box-Ljung test

data:  arch_residuals
X-squared = 22.586, df = 10, p-value = 0.01238

>
```
```

- **Fit a GARCH Model**

```
print("\nFitting GARCH Model...")
garch_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                         mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
                         distribution.model = "norm")

garch_fit <- ugarchfit(spec = garch_spec, data = returns)
print("GARCH Model Summary:")
print(garch_fit)
```

Result

```
*---------------------------------*
*          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|)
omega   0.231268    0.100104   2.3103 0.020873
alpha1  0.034232    0.009067   3.7756 0.000160
beta1   0.947552    0.013310  71.1903 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
omega   0.231268    0.153474   1.5069 0.131840
alpha1  0.034232    0.010528   3.2515 0.001148
beta1   0.947552    0.012434  76.2040 0.000000

LogLikelihood : -2007.246

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

Akaike       5.3464
Bayes        5.3648
Shibata      5.3464
Hannan-Quinn 5.3535

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.0297  0.8632
Lag[2*(p+q)+(p+q)-1][2]    0.3593  0.7636
Lag[4*(p+q)+(p+q)-1][5]    3.4842  0.3257
d.o.f=0
H0 : No serial correlation
```

```
Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                      3.210 0.07318
Lag[2*(p+q)+(p+q)-1][5]     6.027 0.08865
Lag[4*(p+q)+(p+q)-1][9]     7.199 0.18307
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3]     3.693 0.500 2.000 0.05463
ARCH Lag[5]     4.217 1.440 1.667 0.15530
ARCH Lag[7]     4.615 2.315 1.543 0.26640

Nyblom stability test
------------------------------------
Joint Statistic:  0.6541
Individual Statistics:
omega  0.09128
alpha1 0.15749
beta1  0.09929

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         0.846 1.01 1.35
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value    prob sig
Sign Bias           0.4004 0.68895
Negative Sign Bias  1.7379 0.08265   *
Positive Sign Bias  1.0243 0.30603
Joint Effect        4.0694 0.25407
```

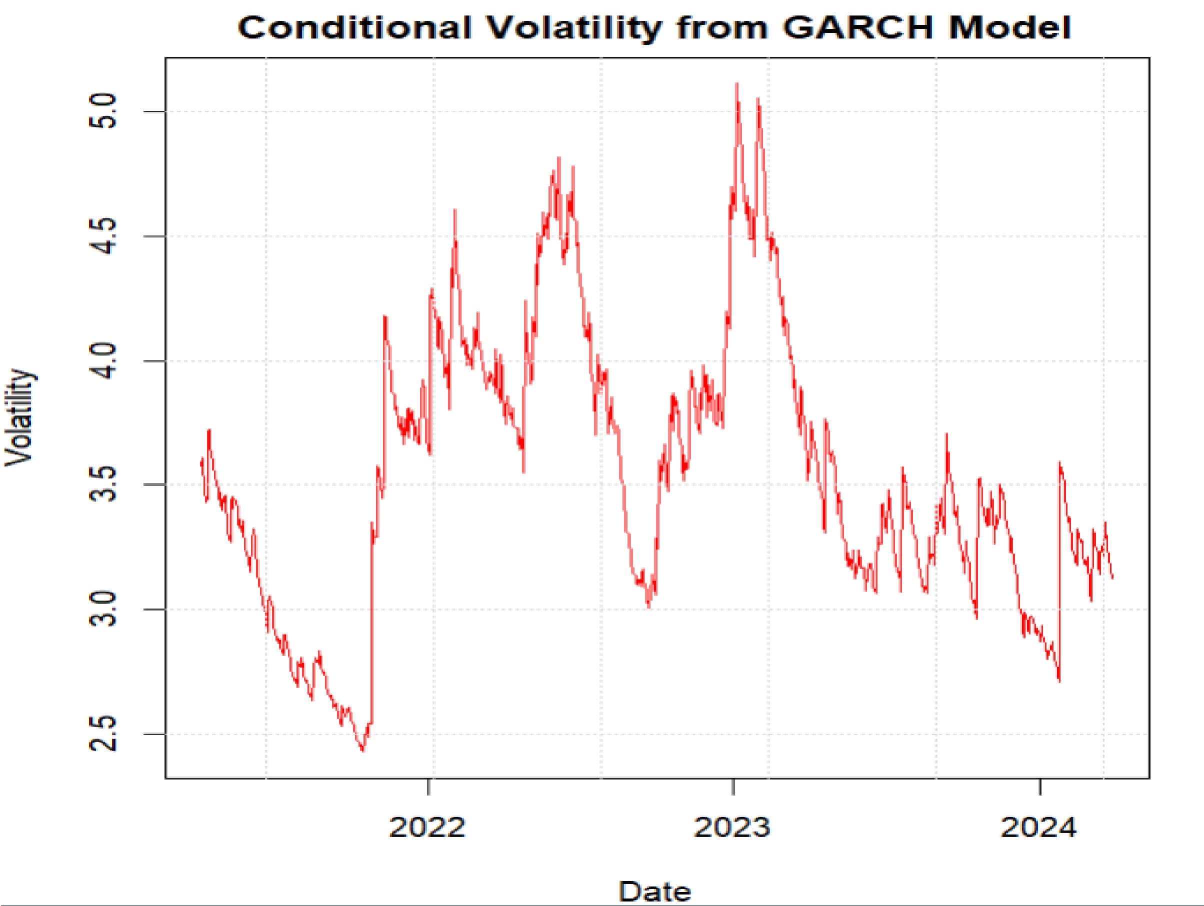
```
Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     39.86    0.0034119
2    30     53.05    0.0041501
3    40     72.36    0.0009244
4    50     70.47    0.0238554


Elapsed time : 0.05567789
```

```r
# Plot the conditional volatility from the GARCH model
# Extract conditional volatility from the fitted model
cond_volatility <- sigma(garch_fit)

# Plot the conditional volatility from the fitted GARCH model
plot(index(returns), cond_volatility, type = 'l',
     main = 'Conditional Volatility from GARCH Model',
     xlab = 'Date', ylab = 'Volatility', col = 'red')
grid()
```

Result



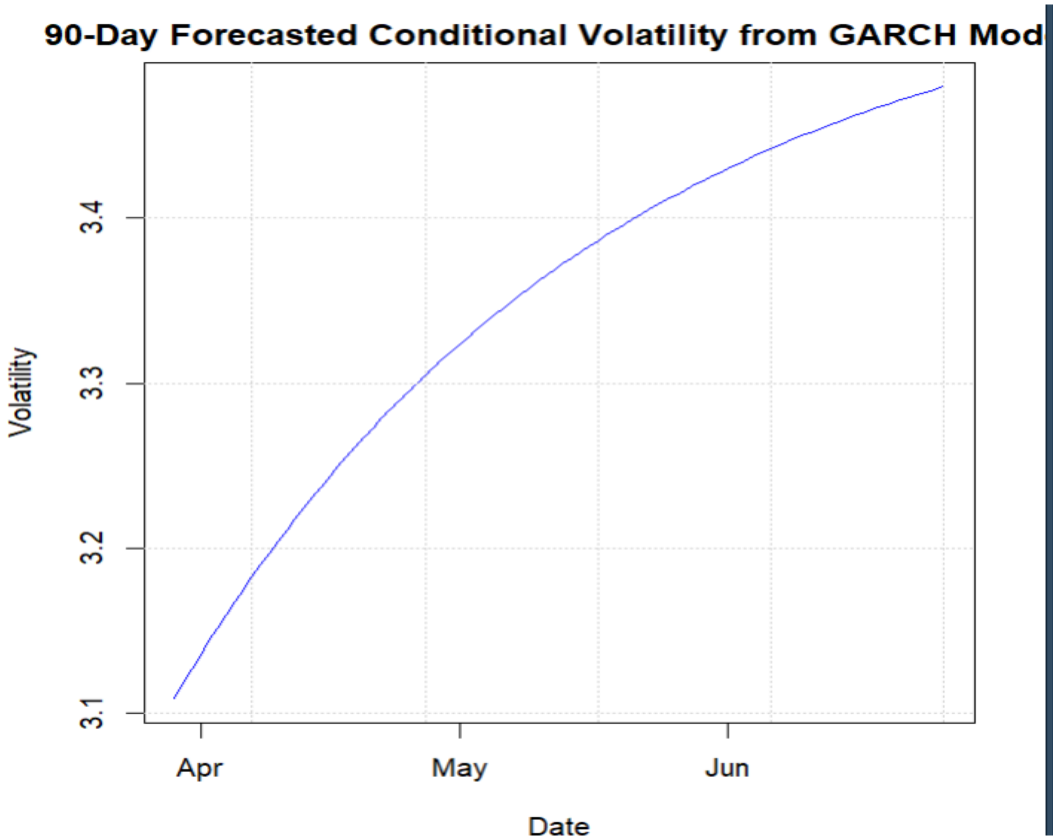
```r
garch_forecast <- ugarchforecast(garch_fit, n.ahead = 90)

# Extract forecasted conditional volatility
forecast_volatility <- sigma(garch_forecast)

# Create a time series for forecast dates
forecast_dates <- seq(from = as.Date(tail(index(returns), 1)) + 1,
                      by = "days", length.out = length(forecast_volatility))

# Plot the forecasted conditional volatility
plot(forecast_dates, forecast_volatility, type = 'l',
     main = '90-Day Forecasted Conditional Volatility from GARCH Model',
     xlab = 'Date', ylab = 'Volatility', col = 'blue')
grid()
```

Result



```
garch_residuals <- residuals(garch_fit)
garch_ljung_box <- Box.test(garch_residuals, lag = 10, type = "Ljung-Box")
print("\nLjung-Box Test for GARCH Model Residuals:")
print(garch_ljung_box)
```

Result

```
[1] "\nLjung-Box Test for GARCH Model Residuals:"
> print(garch_ljung_box)

        Box-Ljung test

data:  garch_residuals
X-squared = 22.586, df = 10, p-value = 0.01238
```

- **Fit GARCH Model with Additional Parameters**

```
print("\nFitting GARCH Model with additional parameters...")

# Specify GARCH model with normal distribution
garch_spec_additional <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                                    mean.model = list(armaOrder = c(0, 0)),
                                    distribution.model = "norm")

# Fit the model
garch_fit_additional <- ugarchfit(spec = garch_spec_additional, data = returns)

# Forecast details
garch_forecast_additional <- ugarchforecast(garch_fit_additional, n.ahead = 1)

# Extract forecast details
forecast_mean <- as.numeric(ugarchforecast(garch_fit_additional, n.ahead = 1)@forecast$seriesFor)
forecast_residual_variance <- as.numeric(ugarchforecast(garch_fit_additional, n.ahead = 1)@forecast$sigmaFor)
forecast_variance <- forecast_residual_variance^2

# Print forecast details for the last 3 periods
print("\nForecast Mean (last 3 periods):")
print(tail(forecast_mean, 3))
print("Forecast Residual Variance (last 3 periods):")
print(tail(forecast_residual_variance, 3))
print("Forecast Variance (last 3 periods):")
print(tail(forecast_variance, 3))
```

Result

```
> print("\nForecast Mean (last 3 periods):")
[1] "\nForecast Mean (last 3 periods):"
> print(tail(forecast_mean, 3))
[1] 0.008485198
> print("Forecast Residual Variance (last 3 periods):")
[1] "Forecast Residual Variance (last 3 periods):"
> print(tail(forecast_residual_variance, 3))
[1] 3.109972
> print("Forecast Variance (last 3 periods):")
[1] "Forecast Variance (last 3 periods):"
> print(tail(forecast_variance, 3))
[1] 9.671928
> 
```

```
# Forecasting with a horizon of 90 days
print("\nForecasting 90 days ahead...")
forecasts <- ugarchforecast(garch_fit_additional, n.ahead = 90)

# Extract forecast residual variance and variance
forecast_residual_variance_90 <- as.numeric(forecasts@forecast$sigmaFor)
forecast_variance_90 <- forecast_residual_variance_90^2

# Create a sequence of dates for plotting the 90-day forecast
forecast_dates <- seq(from = as.Date(tail(index(returns), 1)) + 1,
                      by = "days", length.out = 90)

# Print forecast residual variance for the 90-day horizon
print("\n90-day Forecast Residual Variance (last 3 periods):")
print(tail(forecast_residual_variance_90^2, 3))
```

Result

```
> print("\nForecasting 90 days ahead...")
[1] "\nForecasting 90 days ahead..."
> forecasts <- ugarchforecast(garch_fit_additional, n.ahead = 90)
>
> # Extract forecast residual variance and variance
> forecast_residual_variance_90 <- as.numeric(forecasts@forecast$sigmaFor)
> forecast_variance_90 <- forecast_residual_variance_90^2
>
> # Create a sequence of dates for plotting the 90-day forecast
> forecast_dates <- seq(from = as.Date(tail(index(returns), 1)) + 1,
+                       by = "days", length.out = 90)
>
> # Print forecast residual variance for the 90-day horizon
> print("\n90-day Forecast Residual Variance (last 3 periods):")
[1] "\n90-day Forecast Residual Variance (last 3 periods):"
> print(tail(forecast_residual_variance_90^2, 3))
[1] 12.08633 12.09746 12.10839
>
```

- **Plot Forecasts**

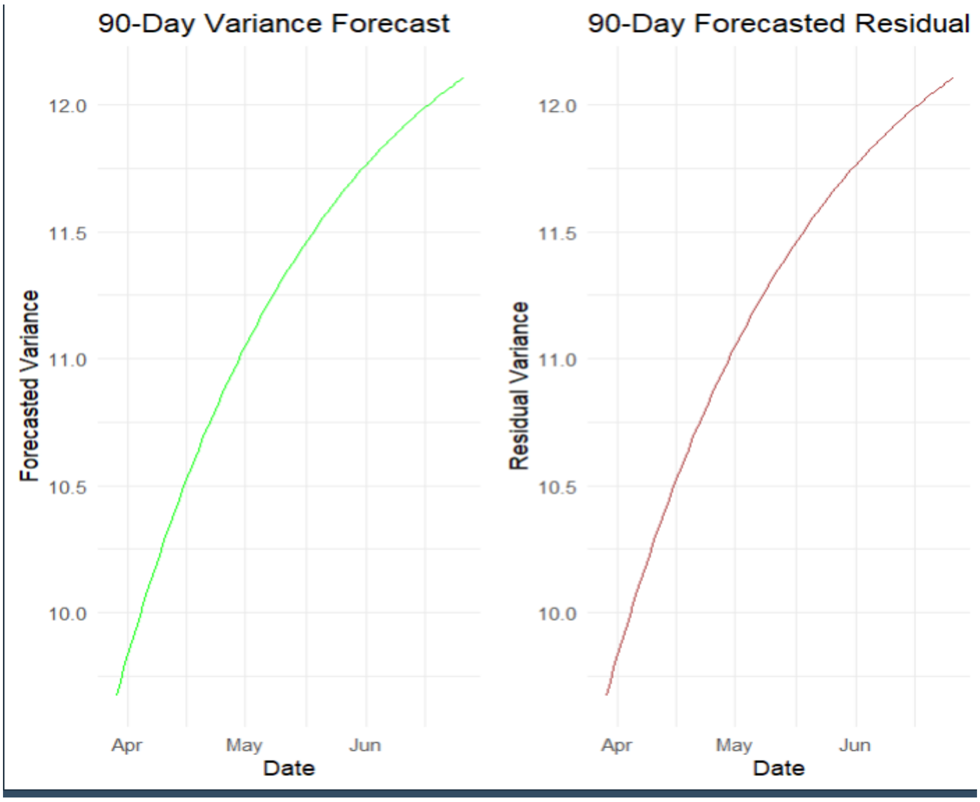
```
# Plot the 90-day variance forecast
forecast_variance_plot <- ggplot(data = data.frame(Date = forecast_dates,
                                                   Variance = forecast_variance_90),
                                 aes(x = Date, y = Variance)) +
  geom_line(color = 'green') +
  ggtitle('90-Day Variance Forecast') +
  xlab('Date') +
  ylab('Forecasted Variance') +
  theme_minimal()

# Display the plot
print(forecast_variance_plot)

# Plot the 90-day forecasted residual variance
forecast_residual_variance_plot <- ggplot(data = data.frame(Date = forecast_dates,
                                                   ResidualVariance = forecast_residual_variance_90^2),
                                 aes(x = Date, y = ResidualVariance)) +
  geom_line(color = 'brown') +
  ggtitle('90-Day Forecasted Residual Variance') +
  xlab('Date') +
  ylab('Residual Variance') +
  theme_minimal()

# Arrange and display both plots side by side
grid.arrange(forecast_variance_plot, forecast_residual_variance_plot, ncol = 2)
```

Result



RECOMMENDATIONS

Based on the analysis using ARCH and GARCH models, the following recommendations are provided for effectively utilizing volatility forecasts and improving financial strategies. To use the information from the ARCH and GARCH models, you need to think strategically about risk management, making business decisions, and following the rules. Firms and investors can better handle uncertain markets, make their financial plans work better, and improve total financial security.

CODES

R

```r
# Install required libraries if not already installed

required_packages <- c("quantmod", "rugarch", "ggplot2", "tseries",
"gridExtra")


new_packages <- required_packages[!(required_packages %in%
installed.packages()[,"Package"])]


if(length(new_packages)) install.packages(new_packages)


# Load required libraries

library(quantmod)

library(rugarch)

library(ggplot2)

library(tseries)

library(gridExtra)


# Step 1: Download Historical Data of Tesla

ticker <- "TSLA"

getSymbols(ticker, src = "yahoo", from = "2021-04-01", to = "2024-03-31")
```

```
# Extract adjusted close price and calculate returns
data <- Ad(get(ticker))
returns <- 100 * diff(log(data))
returns <- na.omit(returns)

# Check data structure
print(head(TSLA))
print(str(TSLA))

# Step 2: Calculate Returns
market <- Cl(TSLA)  # Adjusted Close prices
returns <- 100 * diff(log(market))  # Convert to percentage returns
returns <- na.omit(returns)

# Step 3: Fit an ARCH Model
print("\nFitting ARCH Model...")
arch_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),
                        mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
                        distribution.model = "norm")

arch_fit <- ugarchfit(spec = arch_spec, data = returns)
```

```
print("ARCH Model Summary:")
print(arch_fit)

# Plot the conditional volatility from the ARCH model
## Extract conditional volatility
cond_volatility <- sigma(arch_fit)

# Create a time series plot for conditional volatility
# Use the index of the returns, which is aligned with the conditional volatility
plot(index(returns), cond_volatility, type = 'l',
     main = 'Conditional Volatility from ARCH Model',
     xlab = 'Date', ylab = 'Volatility', col = 'blue')
grid()

# Check residuals for autocorrelation
arch_residuals <- residuals(arch_fit)
arch_ljung_box <- Box.test(arch_residuals, lag = 10, type = "Ljung-Box")
print("\nLjung-Box Test for ARCH Model Residuals:")
print(arch_ljung_box)

data <- Ad(get(ticker))
```

```
returns <- 100 * diff(log(data))

returns <- na.omit(returns)

# Step 4: Fit a GARCH Model

print("\nFitting GARCH Model...")

garch_spec <- ugarchspec(variance.model = list(model = "sGARCH",
garchOrder = c(1, 1)),

                mean.model = list(armaOrder = c(0, 0), include.mean =
FALSE),

                distribution.model = "norm")


garch_fit <- ugarchfit(spec = garch_spec, data = returns)

print("GARCH Model Summary:")

print(garch_fit)


# Plot the conditional volatility from the GARCH model

# Extract conditional volatility from the fitted model

cond_volatility <- sigma(garch_fit)


# Plot the conditional volatility from the fitted GARCH model

plot(index(returns), cond_volatility, type = 'l',

    main = 'Conditional Volatility from GARCH Model',

    xlab = 'Date', ylab = 'Volatility', col = 'red')
```

```
grid()

garch_forecast <- ugarchforecast(garch_fit, n.ahead = 90)

# Extract forecasted conditional volatility
forecast_volatility <- sigma(garch_forecast)

# Create a time series for forecast dates
forecast_dates <- seq(from = as.Date(tail(index(returns), 1)) + 1,
                      by = "days", length.out = length(forecast_volatility))

# Plot the forecasted conditional volatility
plot(forecast_dates, forecast_volatility, type = 'l',
     main = '90-Day Forecasted Conditional Volatility from GARCH Model',
     xlab = 'Date', ylab = 'Volatility', col = 'blue')
grid()
# Check residuals for autocorrelation
garch_residuals <- residuals(garch_fit)
garch_ljung_box <- Box.test(garch_residuals, lag = 10, type = "Ljung-Box")
print("\nLjung-Box Test for GARCH Model Residuals:")
print(garch_ljung_box)
```

```
# Step 5: Fit GARCH Model with Additional Parameters
print("\nFitting GARCH Model with additional parameters...")

# Specify GARCH model with normal distribution
garch_spec_additional <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                    mean.model = list(armaOrder = c(0, 0)),
                    distribution.model = "norm")

# Fit the model
garch_fit_additional <- ugarchfit(spec = garch_spec_additional, data = returns)

# Forecast details
garch_forecast_additional <- ugarchforecast(garch_fit_additional, n.ahead = 1)

# Extract forecast details
forecast_mean <- as.numeric(ugarchforecast(garch_fit_additional, n.ahead = 1)@forecast$seriesFor)
forecast_residual_variance <- as.numeric(ugarchforecast(garch_fit_additional, n.ahead = 1)@forecast$sigmaFor)
forecast_variance <- forecast_residual_variance^2

# Print forecast details for the last 3 periods
```

```
print("\nForecast Mean (last 3 periods):")
print(tail(forecast_mean, 3))
print("Forecast Residual Variance (last 3 periods):")
print(tail(forecast_residual_variance, 3))
print("Forecast Variance (last 3 periods):")
print(tail(forecast_variance, 3))

# Forecasting with a horizon of 90 days
print("\nForecasting 90 days ahead...")
forecasts <- ugarchforecast(garch_fit_additional, n.ahead = 90)

# Extract forecast residual variance and variance
forecast_residual_variance_90 <- as.numeric(forecasts@forecast$sigmaFor)
forecast_variance_90 <- forecast_residual_variance_90^2

# Create a sequence of dates for plotting the 90-day forecast
forecast_dates <- seq(from = as.Date(tail(index(returns), 1)) + 1,
                      by = "days", length.out = 90)

# Print forecast residual variance for the 90-day horizon
print("\n90-day Forecast Residual Variance (last 3 periods):")
```

```
print(tail(forecast_residual_variance_90^2, 3))


# Step 6: Plot Forecasts

# Plot the 90-day variance forecast

forecast_variance_plot <- ggplot(data = data.frame(Date = forecast_dates,

                                 Variance = forecast_variance_90),

                     aes(x = Date, y = Variance)) +

  geom_line(color = 'green') +

  ggtitle('90-Day Variance Forecast') +

  xlab('Date') +

  ylab('Forecasted Variance') +

  theme_minimal()


# Display the plot

print(forecast_variance_plot)


# Plot the 90-day forecasted residual variance

forecast_residual_variance_plot <- ggplot(data = data.frame(Date =
forecast_dates,

                                          ResidualVariance =
forecast_residual_variance_90^2),

                              aes(x = Date, y = ResidualVariance)) +
```

```
  geom_line(color = 'brown') +
  ggtitle('90-Day Forecasted Residual Variance') +
  xlab('Date') +
  ylab('Residual Variance') +
  theme_minimal()

# Arrange and display both plots side by side
grid.arrange(forecast_variance_plot, forecast_residual_variance_plot, ncol = 2)
```

REFERENCES

1. https://www.w3schools.com/