In [1]:
```
# Objective: We will implement ARCH, GARCH, & EWMA Model in Python
```

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
# Data 1: Yahoo Finance
# Data 2: Alpha Vantage
```

In [4]:
```
# Download the data from Alpha Vantage
import requests
from io import StringIO

api_key = 'SLK1N6T6LSBSR8NK'
symbol = 'JPM'

# url to get daily stock data of JPM
url = f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&sym

# Fetch the data
response = requests.get(url)

# Convert the csv text into a dataframe
df = pd.read_csv(StringIO(response.text))
df.index = df['timestamp']
df = df.loc[:'2015-01-01']
df = df['close']
df
```

Out[4]:
```
timestamp
2025-05-30    264.00
2025-05-29    264.37
2025-05-28    263.49
2025-05-27    265.29
2025-05-23    260.71
               ...
2015-01-08     60.39
2015-01-07     59.07
2015-01-06     58.98
2015-01-05     60.55
2015-01-02     62.49
Name: close, Length: 2618, dtype: float64
```

In [5]:
```
# Import data from Yahoo Finance
import yfinance as yf

symbol = 'JPM'
df = yf.download(symbol, start = '2015-01-01', end = '2025-01-01')
df = df['Close']
df
```

```
YF.download() has changed argument auto_adjust default to True

[*********************100%***********************]  1 of 1 completed
```

Out[5]:

Ticker	JPM
Date	
2015-01-02	47.174248
2015-01-05	45.709736
2015-01-06	44.524536
2015-01-07	44.592491
2015-01-08	45.588947
...	...
2024-12-24	239.589218
2024-12-26	240.409912
2024-12-27	238.462036
2024-12-30	236.632812
2024-12-31	237.018433

2516 rows × 1 columns

ARCH Model

In [6]:
```
# !pip install arch

from arch import arch_model
```

In [7]:
```python
# Step 1: Get the data from Yahoo Finance
df = yf.download('JPM', start = '2022-01-01', end = '2025-01-01')

# Step 2: Calculate daily returns
df['returns'] = df['Close'].pct_change()*100 # daily % return

# Step 3: Drop missing value (first value is NaN)
returns = df['returns'].dropna()

# Step 4: Create and fit ARCH(1) model
model = arch_model(returns, vol = 'ARCH', p = 1) # Call the model
results = model.fit(disp = 'off') # Train the model

# Step 5: Show Summary
results.summary()
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[7]:

Constant Mean - ARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-1400.95
Distribution:	Normal	**AIC:**	2807.90
Method:	Maximum Likelihood	**BIC:**	2821.76
		No. Observations:	752
Date:	Sat, May 31 2025	**Df Residuals:**	751
Time:	02:57:29	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.0978	5.518e-02	1.772	7.646e-02	[-1.039e-02, 0.206]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	2.1877	0.332	6.584	4.592e-11	[1.536, 2.839]
alpha[1]	0.1224	7.467e-02	1.639	0.101	[-2.394e-02, 0.269]

Covariance estimator: robust

In [8]:
```
# Analysis of the ARCH Model
# mu    0.1260: The model estimates that the average daily return is 0.1
# omega     1.6656: The long run base level of variance
# alpha[1]  0.4211: How much yesterday's squared shock impacts today's v
```

In [9]:
```
# Step 6: Forecast 5 days ahead
forecast = results.forecast(horizon = 5)
predicted_variance = forecast.variance

# Volatility = square root (Variance)
predicted_volatility = predicted_variance ** 0.5
predicted_volatility
```

Out[9]:

	h.1	h.2	h.3	h.4	h.5
Date					
2024-12-31	1.479267	1.567023	1.577429	1.578698	1.578853

In [10]:
```
# Calculate Avg of predicted volatility

# predicted_volatility = [1.290793, 1.538573, 1.631696, 1.669358, 1.6849
predicted_volatility = [1.479269, 1.567023, 1.577429, 1.578698, 1.578853
predicted_avg_vol = sum(predicted_volatility)/len(predicted_volatility)
predicted_avg_vol
```

Out[10]: 1.5562543999999998

In [11]:
```
# Step 7: Get the data and calculate realized volatility

start_date = pd.to_datetime('2024-12-31')
end_date = pd.to_datetime('2025-01-09') #  buffer for weekends and holid

real_df = yf.download('JPM', start_date, end_date)
real_df['returns'] = real_df['Close'].pct_change()*100
real_df = real_df.dropna()

realized_vol = real_df['returns'].std()*np.sqrt(5) # Predicting the vol
realized_vol
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[11]: 1.6919435641135898

In [12]:
```
print("ARCH Model Predicted Volatility:", predicted_avg_vol)
print("ARCH Model Actual Volatility:", realized_vol)
```

```
ARCH Model Predicted Volatility: 1.5562543999999998
ARCH Model Actual Volatility: 1.6919435641135898
```

In [13]:
```
# 2015 - 2024
# ARCH Model Predicted Volatility: 1.5630772
# ARCH Model Actual Volatility: 0.7566628401922761

# 2022 - 2024
# ARCH Model Predicted Volatility: 1.5562543999999998
# ARCH Model Actual Volatility: 0.7566628401922761
```

GARCH Model

In [14]:
```
from arch import arch_model
```

In [15]:
```
# Step 1: Get the data from Yahoo Finance
df = yf.download('JPM', start = '2022-01-01', end = '2025-01-01')

# Step 2: Calculate daily returns
df['returns'] = df['Close'].pct_change()*100 # daily % return

# Step 3: Drop missing value (first value is NaN)
returns = df['returns'].dropna()

# Step 4: Create and fit ARCH(1) model
model = arch_model(returns, vol = 'GARCH', p = 1, q = 1) # Call the mode
results = model.fit(disp = 'off') # Train the model

# Step 5: Show Summary
results.summary()
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[15]:

Constant Mean - GARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	-1387.47
Distribution:	Normal	**AIC:**	2782.94
Method:	Maximum Likelihood	**BIC:**	2801.43
		No. Observations:	752
Date:	Sat, May 31 2025	**Df Residuals:**	751
Time:	02:57:30	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.1146	5.754e-02	1.991	4.648e-02	[1.786e-03, 0.227]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0220	1.702e-02	1.294	0.196	[-1.134e-02,5.537e-02]
alpha[1]	0.0165	1.112e-02	1.481	0.139	[-5.326e-03,3.827e-02]
beta[1]	0.9737	1.217e-02	79.973	0.000	[0.950, 0.998]

Covariance estimator: robust

In [16]:
```
# Analysis of the GARCH Model
# mu    0.1146: The model estimates that the average daily return is 0.1
# omega     0.0220: The long run base level of variance
# alpha[1]  0.0165: How much yesterday's squared shock impacts today's v
# beta[1]   0.9737: How much yesterday's variance impacts today's varian
```

In [17]:
```
# Step 6: Forecast 5 days ahead
forecast = results.forecast(horizon = 5)
predicted_variance = forecast.variance

# Volatility = square root (Variance)
predicted_volatility = predicted_variance ** 0.5
predicted_volatility
```

Out[17]:

	h.1	h.2	h.3	h.4	h.5
Date					
2024-12-31	1.623483	1.622252	1.621033	1.619824	1.618627

In [18]:
```
# Calculate Avg of predicted volatility

predicted_volatility = [1.623483, 1.622252, 1.621032, 1.619824, 1.618626
predicted_avg_vol = sum(predicted_volatility)/len(predicted_volatility)
predicted_avg_vol
```

Out[18]: 1.6210434

In [19]:
```
# Step 7: Get the data and calculate realized volatility

start_date = pd.to_datetime('2024-12-31')
end_date = pd.to_datetime('2025-01-09') #  buffer for weekends and holid

real_df = yf.download('JPM', start_date, end_date)
real_df['returns'] = real_df['Close'].pct_change()*100
real_df = real_df.dropna()

realized_vol = real_df['returns'].std()*np.sqrt(5)
realized_vol
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[19]: 1.6919435641135898

In [20]:
```
print("GARCH Model Predicted Volatility:", predicted_avg_vol)
print("GARCH Model Actual Volatility:", realized_vol)
```

```
GARCH Model Predicted Volatility: 1.6210434
GARCH Model Actual Volatility: 1.6919435641135898
```

EWMA Model

In [21]:
```
# Step 1: Download the data
df = yf.download('JPM', start = '2022-01-01', end = '2025-01-01')

# Step 2: Calculate daily returns
df['returns'] = df['Close'].pct_change()
df = df.dropna()

# Step 3: Step lamda value for EWMA Model
lamda = 0.94

# Step 4: Initialize variance and calculate EWMA
ewma_var = []
var_t = df['returns'].var()

for ret in df['returns']:
    variance_tplus1 = lamda*var_t + (1-lamda)* (ret**2)
    ewma_var.append(variance_tplus1)

# Volatility = sqrt(variance)
df['ewma_vol'] = np.sqrt(ewma_var)

# Step 5: Predicted Volatility
latest_daily_vol = df['ewma_vol'].iloc[-1] # predicted volatility
latest_daily_vol

# EWMA can only be used for 1 day prediction
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[21]:
```
0.0152686564627976
```

In [22]:
```
# Step 6: Get the data and calculate realized volatility

start_date = pd.to_datetime('2024-12-31')
end_date = pd.to_datetime('2025-01-03') #  buffer for weekends and holid
```
```
real_df = yf.download('JPM', start_date, end_date)
real_df['returns'] = real_df['Close'].pct_change()*100
real_df = real_df.dropna()

realized_vol = real_df['returns']
realized_vol
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[22]:
```
Date
2025-01-02    0.120973
Name: returns, dtype: float64
```

In [23]:
```
# EWMA Results
# Model = 0.14
# Realized. = 0.12
```

THANK YOU