```
# 📦 Import libraries
!pip install arch
```

```
Collecting arch
  Downloading arch-7.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: numpy>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from arch) (2.0.2)
Requirement already satisfied: scipy>=1.8 in /usr/local/lib/python3.11/dist-packages (from arch) (1.16.0)
Requirement already satisfied: pandas>=1.4 in /usr/local/lib/python3.11/dist-packages (from arch) (2.2.2)
Requirement already satisfied: statsmodels>=0.12 in /usr/local/lib/python3.11/dist-packages (from arch) (0.14.5)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.4->arch) (2.9.0.pos
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.4->arch) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.4->arch) (2025.2)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels>=0.12->arch) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels>=0.12->arch) (25.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.4->arch)
Downloading arch-7.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (985 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 985.3/985.3 kB 22.3 MB/s eta 0:00:00
Installing collected packages: arch
Successfully installed arch-7.2.0
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from arch import arch_model
```

```
# Load your data
df = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/ARIMA_data.csv', index_col=0, parse_dates=True)
```

```
# Step 1: Calculate Log Returns
df['log_return'] = np.log(df['Price'] / df['Price'].shift(1))
df = df.dropna()
```

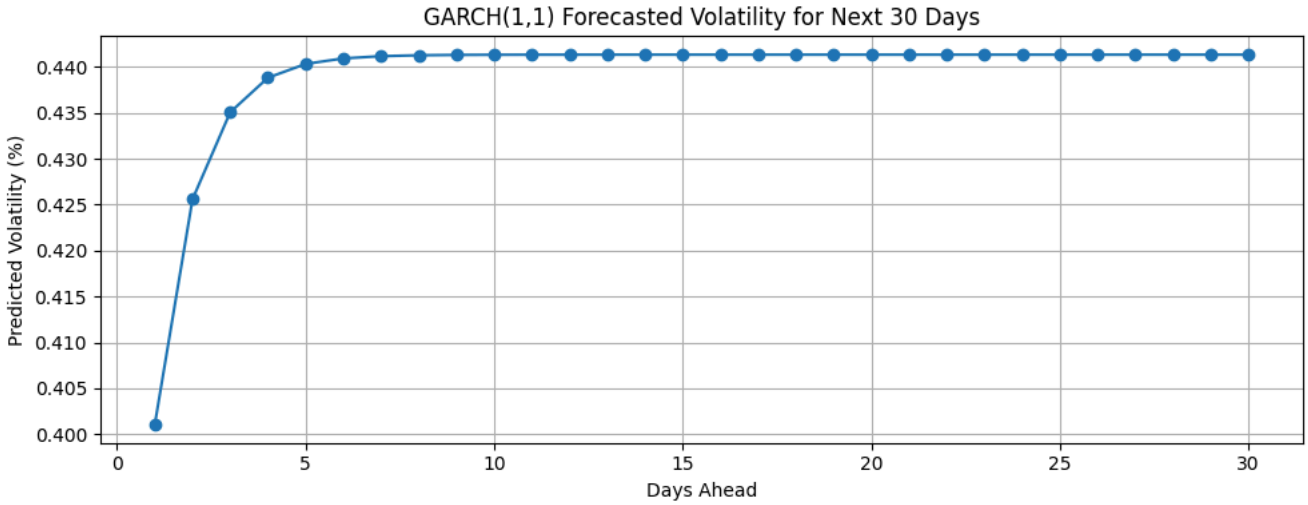
```
# Scale returns for better numerical behavior
returns = df['log_return'] * 100  # Multiply by 100 to work in % scale
```

```
# Step 2: Fit GARCH(1,1)
model = arch_model(returns, vol='GARCH', p=1, q=1, mean='Constant', dist='normal')
model_fit = model.fit(disp='off')
```

```
# Step 3: Forecast next 30 days of volatility
forecast_horizon = 30
forecast = model_fit.forecast(horizon=forecast_horizon)
```

```
# Extract predicted volatility (standard deviation, not variance)
volatility_forecast = np.sqrt(forecast.variance.values[-1, :])
```

```
# Step 4: Plot
plt.figure(figsize=(10, 4))
plt.plot(range(1, forecast_horizon + 1), volatility_forecast, marker='o')
plt.title("GARCH(1,1) Forecasted Volatility for Next 30 Days")
plt.xlabel("Days Ahead")
plt.ylabel("Predicted Volatility (%)")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
1 # Optional: View model summary
2 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:               -226.586
Distribution:                  Normal   AIC:                           461.171
Method:            Maximum Likelihood   BIC:                           477.858
                                        No. Observations:                  479
Date:                Tue, Aug 05 2025   Df Residuals:                      478
Time:                        14:21:56   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0189  3.159e-02      0.597      0.551 [-4.307e-02,8.078e-02]
                          Volatility Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.1164  2.929e-02      3.974  7.082e-05 [5.897e-02,  0.174]
alpha[1]       0.4024      0.371      1.085      0.278   [ -0.325,  1.129]
beta[1]        0.0000  9.959e-02      0.000      1.000   [ -0.195,  0.195]
============================================================================

Covariance estimator: robust
```

```
1 Start coding or generate with AI.
```