Multivariate GARCH (MGARCH) under Dynamic Conditional Correlation (DCC) specification in Python.

AARON DE LA ROSA

MGARCH (Multivariate GARCH): The Multivariate Generalized AutoRegressive Conditional Heteroskedasticity (MGARCH) is an extension of the GARCH model that allows for more than one time series. It allows the conditional-on-past-history covariance matrix of the dependent variables to follow a flexible dynamic structure.

This means that **MGARCH models can capture the time-varying volatility and co-movements in multiple time series. Financial institutions typically use them to estimate the volatility of returns for stocks, bonds, and market indices.**

DCC (Dynamic Conditional Correlation): **DCC is a specific type of MGARCH model. It allows the correlation structure to vary over time, which is particularly useful when dealing with financial data where correlations between assets can change.**

In DCC models, the correlation matrix at each time is modeled as a weighted average of its own past and recent shocks. This makes DCC models very flexible and capable of capturing complex dynamics in the correlations of multivariate time series.

The Dynamic Conditional Correlation (DCC) model is typically used when you need to estimate time-varying correlations between multiple time series. DCC models can be particularly useful for:

Financial Risk Management: DCC models are often used in financial risk management to estimate the correlations between different assets in a portfolio. These correlations can change over time due to various market conditions, and understanding these dynamics can help in managing portfolio risk.

Economic Research: In economic research, DCC models can be used to study the co-movements of different economic variables over time. For example, you might want to understand how the GDP growth rates of different countries are correlated.

Asset Pricing: DCC models can also be used in asset pricing to understand how the returns of different assets move together. This can help in constructing diversified portfolios and in pricing derivatives.

DCC models can be very useful, but they also require a lot of data and can be computationally intensive.

Therefore, they are typically used in situations where it's important to capture the dynamic correlations between multiple time series.

Limitation of GARCH is the restriction to a single dimensional time-series. However, we are typically dealing with multiple time-series.

We'll use Yahoo Finance to get data for DAX and S&P 500 for 5 years.

MGARCH_DCC1.ipynb

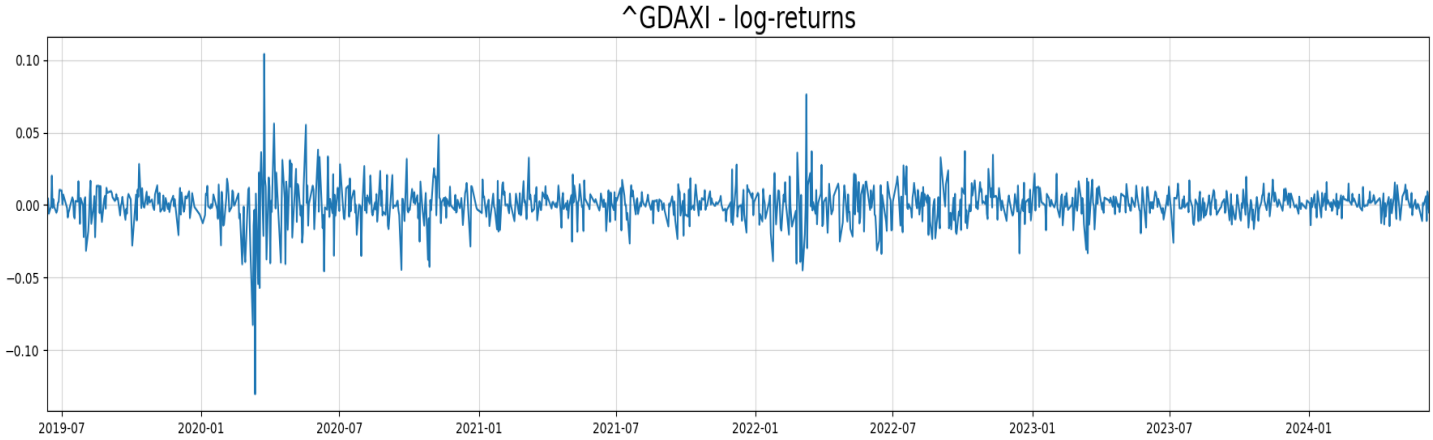
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se guardó por última vez: 11 de junio

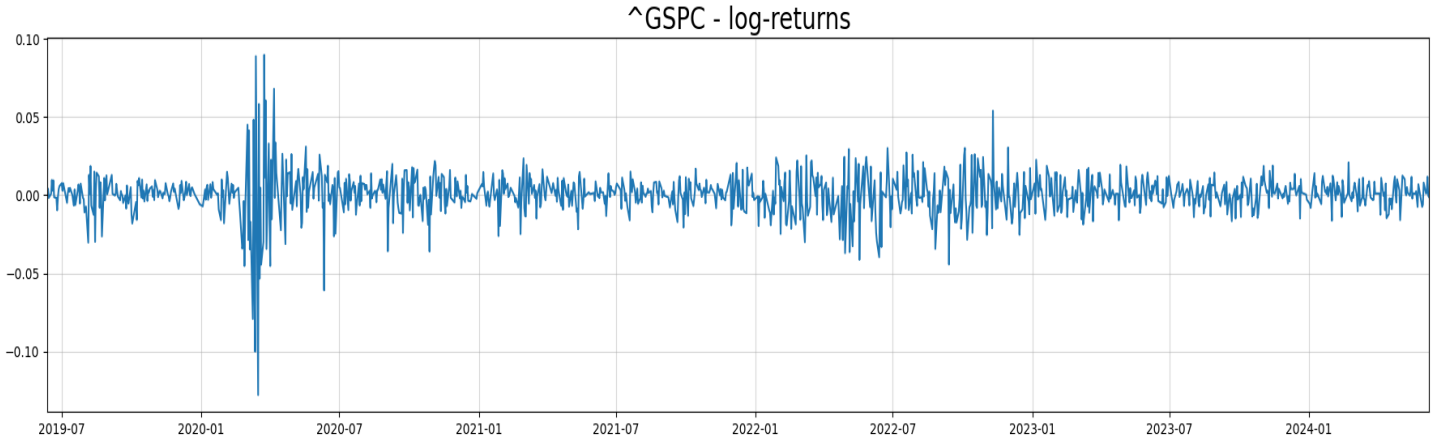
Comentar Compartir

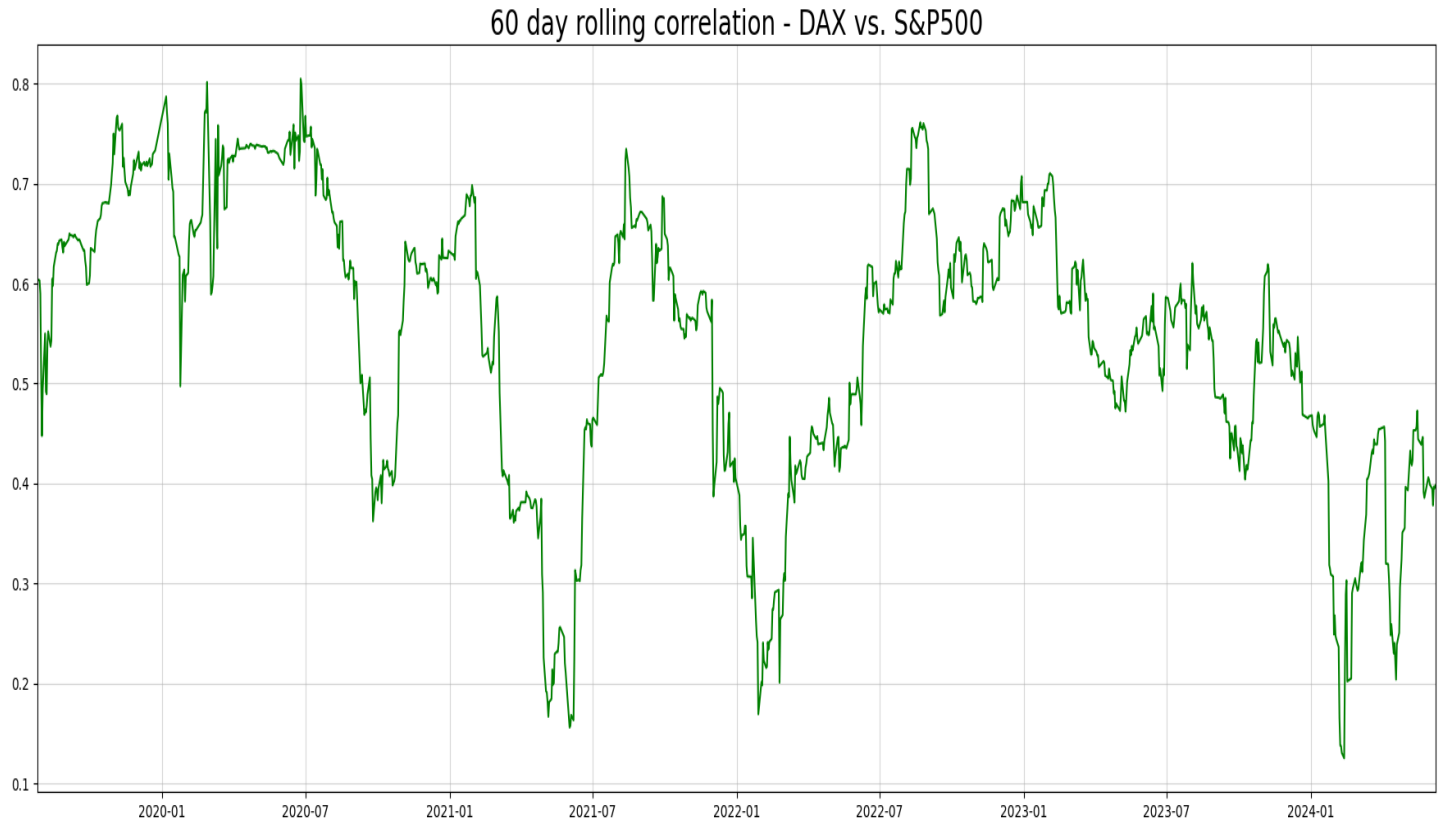
+ Código + Texto Conectar Gemini

```
[ ]  1 import pandas as np
     2 import numpy as np
     3 import tensorflow as tf
     4 import tensorflow_probability as tfp
     5 import matplotlib.pyplot as plt
     6 from datetime import timedelta
     7 import pandas as pd
```

```
[ ]  1 import yfinance as yf
     2 data = yf.download("^GDAXI ^GSPC", start="2019-06-10", end="2024-06-10", interval="1d")
     3
     4 close = data["Close"]
     5 returns = np.log(close).diff().dropna()
     6
     7
     8 fig, axs = plt.subplots(2,1, figsize = (22,5*2))
     9
    10 for i in range(2):
    11     axs[i].plot(returns.iloc[:,i])
    12     axs[i].grid(alpha=0.5)
    13     axs[i].margins(x=0)
    14     axs[i].set_title("{} - log-returns".format(returns.columns[i]),size=20)
```







MGARCH_DCC1.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Comentar Compartir

+ Código + Texto

Conectar Gemini

```python
class MGARCH_DCC(tf.keras.Model):


    def __init__(self, y):
        """
        Args:
            y: NxM numpy.array of N observations of M correlated time-series
        """
        super().__init__()
        n_dims = y.shape[1]
        self.n_dims = n_dims

        self.MU = tf.Variable(np.mean(y,0)) #use a mean variable

        self.sigma0 = tf.Variable(np.std(y,0)) #initial standard deviations at t=0

        #we initialize all restricted parameters to lie inside the desired range
        #by keeping the learning rate low, this should result in admissible results
        #for more complex models, this might not suffice
        self.alpha0 = tf.Variable(np.std(y,0))
        self.alpha = tf.Variable(tf.zeros(shape=(n_dims,))+0.25)
        self.beta = tf.Variable(tf.zeros(shape=(n_dims,))+0.25)

        self.L0 = tf.Variable(np.float32(np.linalg.cholesky(np.corrcoef(y.T)))) #decomposition of A_0
        self.A = tf.Variable(tf.zeros(shape=(1,))+0.9)
```

MGARCH_DCC1.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se guardaron todos los cambios

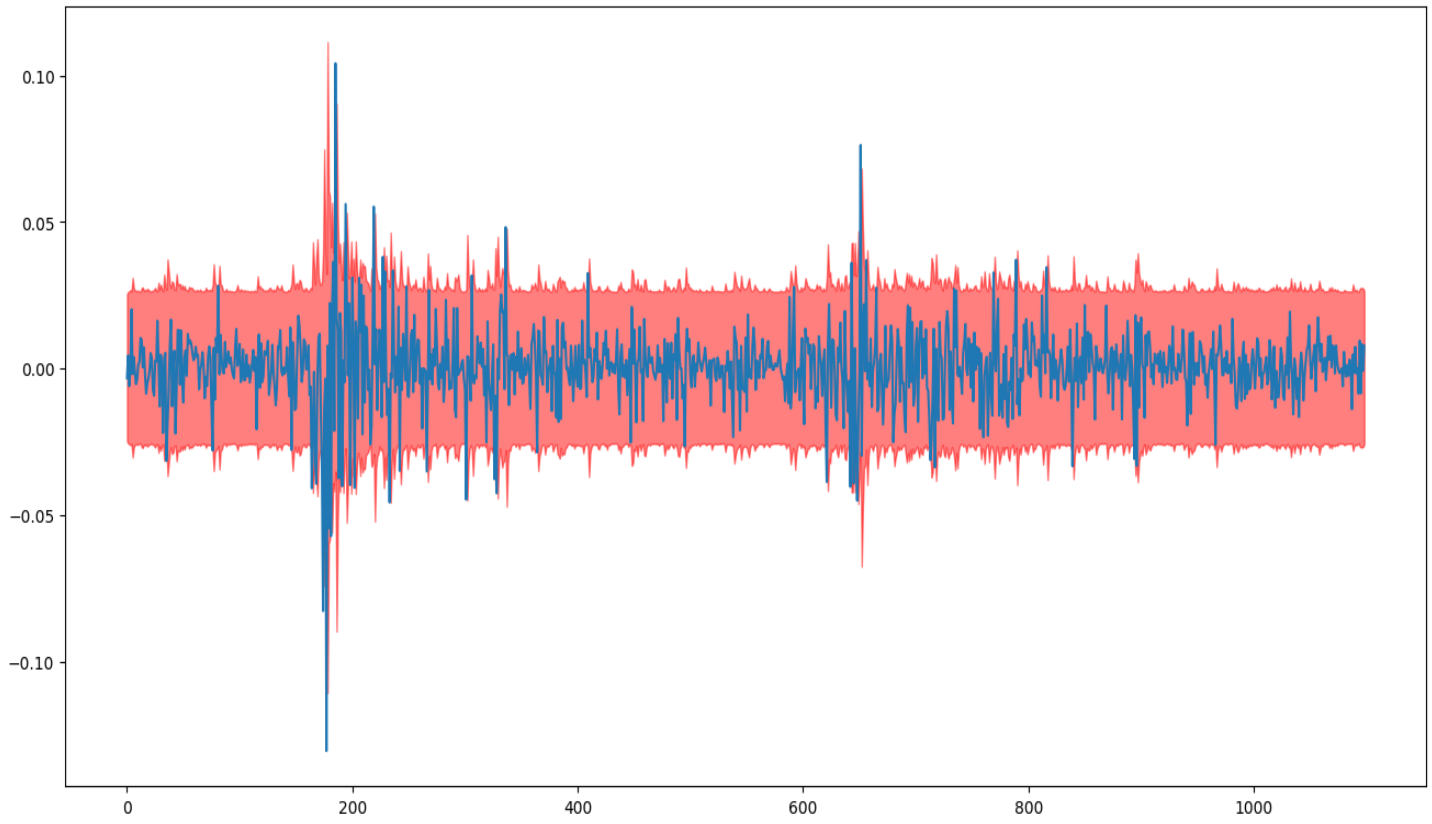
Comentar Compartir

+ Código + Texto

Conectar Gemini

```
[ ]  1 train = np.float32(returns)[:-90,:]
     2 test = np.float32(returns)[-90:,:]
```

```
[ ]  1 model = MGARCH_DCC(train)
     2
     3
     4 from scipy.stats import norm
     5 out = model(train)
     6
     7 means = out.mean().numpy()
     8 stds = out.stddev().numpy()
     9
    10 lowers = norm(means, stds).ppf(0.05)
    11 uppers = norm(means, stds).ppf(0.95)
    12
    13 plt.figure(figsize = (16,6))
    14
    15 i = 0
    16
    17 plt.figure(figsize = (16,8))
    18 plt.plot(train[:,i])
    19 plt.fill_between(np.arange(len(train)),lowers[:,i],uppers[:,i],color="red",alpha=0.5)
```



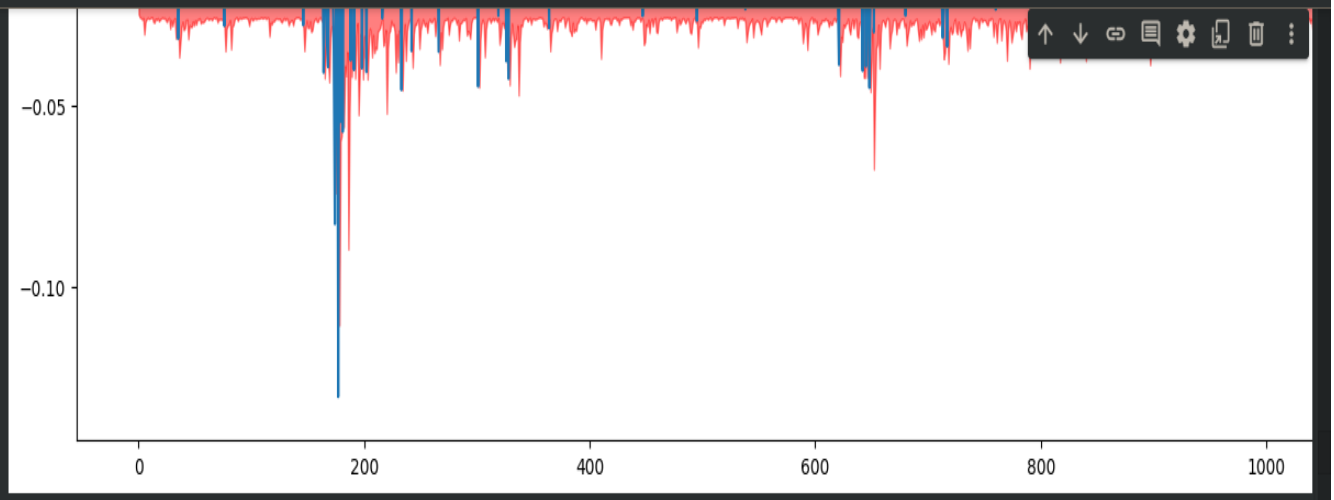
MGARCH_DCC1.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se guardaron todos los cambios

Comentar Compartir

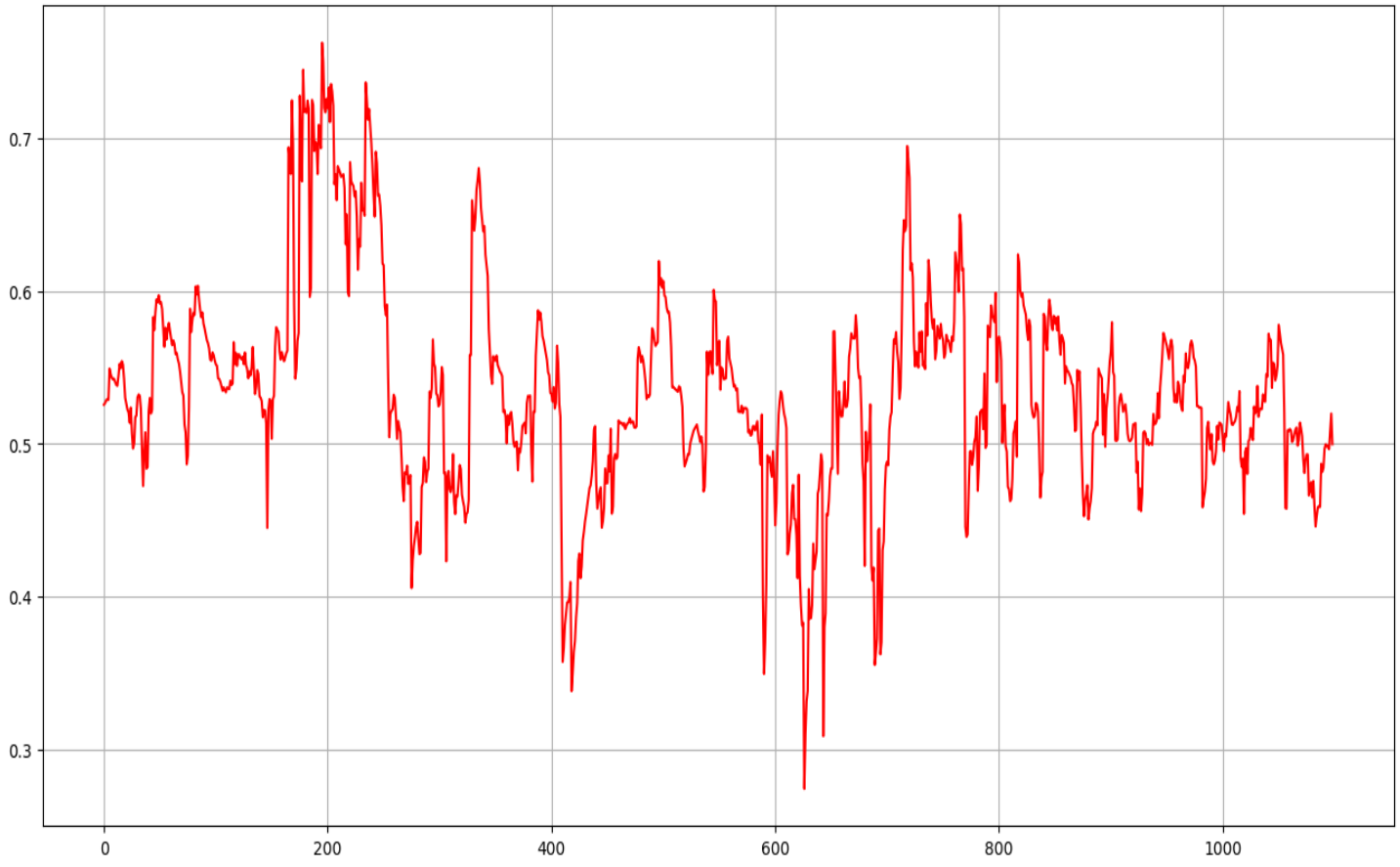
+ Código + Texto

Conectar Gemini




```
[ ]  1 plt.figure(figsize = (16,8))
     2
     3 corr12 = [model.cov_to_corr(out.covariance()[i,:,:])[0,1].numpy() for i in range(len(train))]
     4 plt.grid()
     5 plt.plot(corr12,c="red")
```

[<matplotlib.lines.Line2D at 0x787ef5de5ea0>]



MGARCH_DCC1.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se guardaron todos los cambios

Comentar Compartir

+ Código + Texto

Conectar Gemini

0 200 400 600 800 1000

```
[ ]  1 model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-2))
```

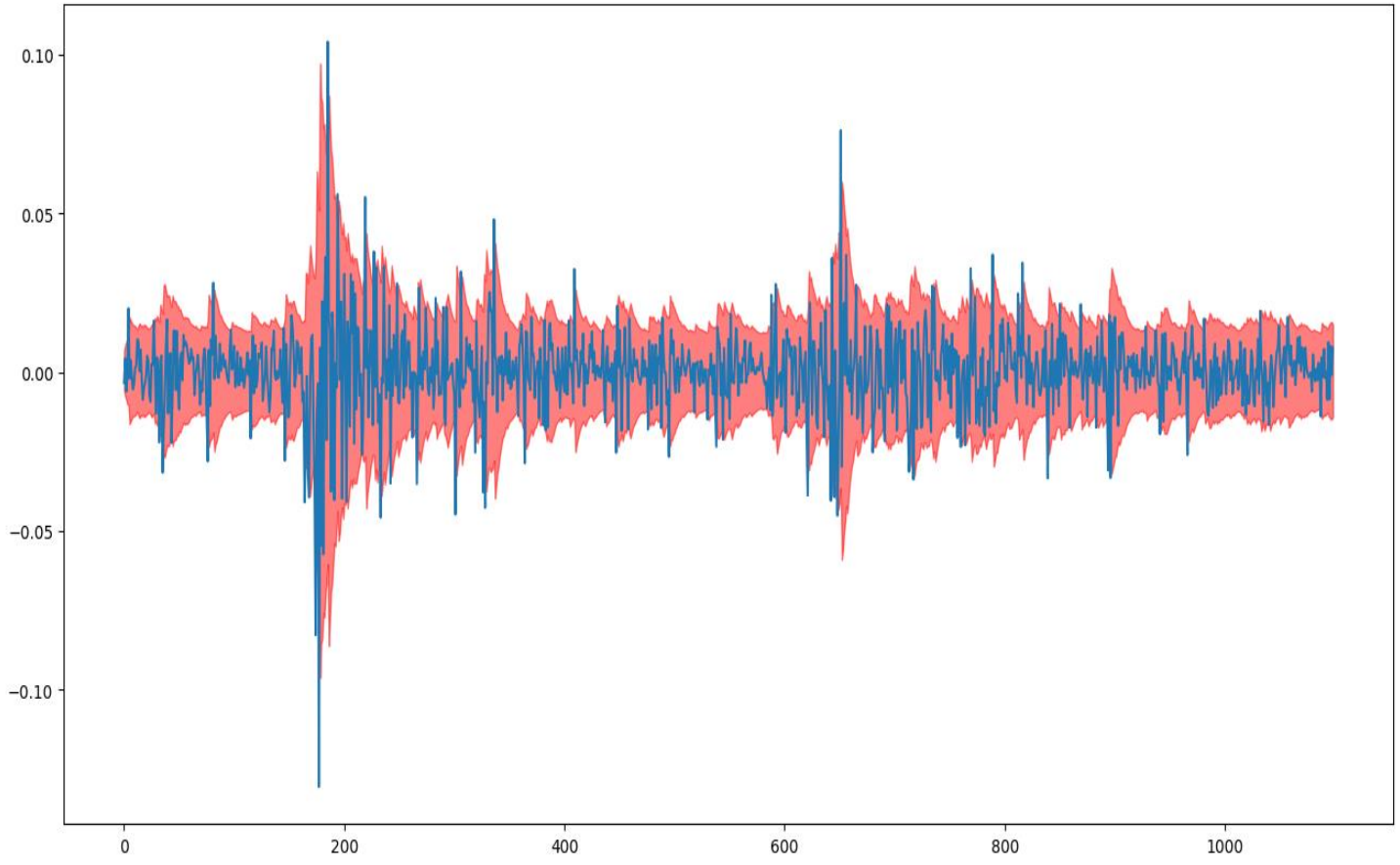
```
[ ]  1 model.fit(train, train, batch_size=len(train), shuffle=False, epochs = 300, verbose=False)
```

```
<keras.src.callbacks.History at 0x787ef53c7700>
```

```
[ ]  1 out = model(train)
     2
     3 means = out.mean().numpy()
     4 stds = out.stddev().numpy()
     5
     6 lowers = norm(means, stds).ppf(0.05)
     7 uppers = norm(means, stds).ppf(0.95)
```

```
1 i = 0
2
3 plt.figure(figsize = (16,8))
4 plt.plot(train[:,i])
5 plt.fill_between(np.arange(len(train)),lowers[:,i],uppers[:,i],color="red",alpha=0.5)
```

```
<matplotlib.collections.PolyCollection at 0x787f04fab4f0>
```



MGARCH_DCC1.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se guardaron todos los cambios

Comentar Compartir

+ Código + Texto Conectar Gemini

```
[ ]  1 corrs = fcast[1][:,:,0,1]
     2 corr_means = np.mean(corrs,0)
     3 corr_lowers = np.quantile(corrs,0.05,0)
     4 corr_uppers = np.quantile(corrs,0.95,0)
     5
     6
     7 conditional_dists = model(np.float32(returns.values))
     8
     9 conditional_correlations = [model.cov_to_corr(conditional_dists.covariance()[i,:,:])[0,1].numpy() for i in range(len(returns))]
    10
    11
    12 idx_train = returns[:-90].index
    13 idx_test = pd.date_range(returns[:-90].index[-1] + timedelta(days=1), returns[:-90].index[-1] + timedelta(days=90))
    14
    15 fig, axs = plt.subplots(2,1,figsize=(20,12), gridspec_kw={'height_ratios': [2, 1]})
    16
    17 axs[0].set_title("Dynamic Conditional Correlation (DCC) - DAX, S&P500", size=20)
    18
    19 axs[0].axhline(np.corrcoef(returns.T)[0,1], c="green",alpha=0.75,ls="dashed",lw=2, label="Unconditional sample correlation")
    20
    21 axs[0].plot(idx_train[30:],conditional_correlations[30:-90],c="red", label="MGARCH in-sample conditional correlation")
    22 axs[0].plot(idx_test,conditional_correlations[-90:],c="red",ls="dotted",lw=3, label="MGARCH out-of-sample DCC")
    23
    24 axs[0].plot(idx_test, corr_means,color="blue",lw=3, alpha=0.9, label="MGARCH correlation mean forecast")
```