Модель связи валютных курсов и цен на нефть.
GARCH model

Максимова Анна Михайловна

5 декабря 2023 г.

Аннотация

Значение нефти как энергетического ресурса и влияние в мировой экономической системе неуклонно возрастают. В частности, растущая торговля нефтью оказала давление на текущий платежный баланс и привела к колебаниям обменного курса. Поскольку цена на нефть является широко проверенным источником для изменения обменных курсов, исследование основных механизмов изменения цены на нефть на биржи предоставляет полезную информацию рыночным инвесторам и имеет важные последствия для политиков и центральных банков. И эта связь между ценой на нефть и обменным курсом доллара США, которую можно наблюдать с 1990-х годов, привлекает интерес многих экономистов.

1 План

1. Введение
2. Теоретическая часть
3. Практическая часть
4. Результаты, анализ
5. Литература

2 Теоретическая часть

Волатильность – это статистическая мера разброса данных или отклонения от среднего значения за определенный период времени.

В финансах это относится к тому, насколько цена меняется между периодами. Например, если она высока, цена может сильно увеличиваться или уменьшаться от одного дня к другому. Высокая волатильность очень рискованна, она может быть выгодна, если вы хотите быстро получить большую прибыль, но также может привести к потере больших денег за короткий период времени.

2.1 Авторегрессионные модели

Авторегрессионная модель (AR) прогнозирует будущее поведение на основе данных о поведении в прошлом. Этот тип анализа используется, когда существует корреляция между значениями временного ряда и их предыдущими и последующими значениями.

Авторегрессионное моделирование использует только прошлые данные для прогнозирования будущего поведения. Линейная регрессия выполняется для данных из текущего ряда на основе одного или нескольких прошлых значений того же ряда.

AR-модели - это модели линейной регрессии, в которых переменная результата (Y) в некоторый момент времени напрямую связана с предикторной переменной (X). В AR-моделях Y зависит от X и предыдущих значений для Y, что отличается от простой линейной регрессии.

Модель AR(1) выглядит следующим образом:

$$X_t = \beta + \phi_1 X_{t-1} + \epsilon_t$$

, где X_{t-1} - значение за прошлый период, ϕ_1 - коэффициент, представляющий будущую часть предыдущего значения, ϵ_t - остаточное значение, представляющее разницу между прогнозом периода t и правильным значением ($\epsilon_t = y_t - \hat{y}_t$). В этой модели учитывается результат только за один прошлый период

Соответственно AR(p) модель имеет следующее представление:

$$\beta + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

2.2 Модель скользящего среднего

Модель скользящего среднего очень похожа на модель AR, однако разница здесь заключается в том, что модель скользящего среднего (MA) предполагает, что данный временной шаг линейно зависит не от соответствующих прошлых предсказанных значений X_t, а от соответствующих прошлых зна-

чений ошибок, поэтому математически модель MA(p) может быть выражена как:

$$\beta + \sum_{i=1}^{p} \phi_i \epsilon_{t-i} + \epsilon_t$$

2.3 ARMA модель

Модель ARMA представляет собой просто комбинацию моделей AR(p) и MA(q), поэтому математически можно выразить модель ARMA(p, q) следующим образом:

$$\beta + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

2.4 ARCH модель

Авторегрессионная условная гетероскедастичность (ARCH — AutoRegressive Conditional Heteroscedasticity) — применяемая в эконометрике модель для анализа временных рядов, у которых условная (по прошлым значениям ряда) дисперсия ряда зависит от прошлых значений ряда, прошлых значений этих дисперсий и иных факторов. Данные модели предназначены для «объяснения» кластеризации волатильности на финансовых рынках, когда периоды высокой волатильности длятся некоторое время, сменяясь затем периодами низкой волатильности, причём среднюю (долгосрочную, безусловную) волатильность можно считать относительно стабильной.

Модели ARCH впервые были предложены Робертом Энглом в 1982 году. Уже в 1986 году Боллерслев предложил обобщение этих моделей (GARCH). В дальнейшем различные авторы предложили и иные варианты моделей данного типа, учитывающих те или иные особенности.

Процессы с условной гетероскедастичностью можно представить следующим образом:

$$u_t = \sigma_t \epsilon_t$$

, где ϵ_t имеет, например, стандартное нормальное распределение, а $\sigma_t = f(u_{t-1}, u_{t-2}, ...)$ - то есть является случайной величиной зависящей от прошлых значений u_t. Соответственно, условная дисперсия такого процесса (при зафиксированном прошлом процесса) будет равна σ_t:

$$V(u_t | u_{t-1}, u_{t-2}, ...) = \sigma_t^2$$

При этом безусловная дисперсия σ^2 предполагается постоянной, поэтому в принципе оценку параметров моделей с условной гетероскедастичностью в случайных ошибках можно осуществить посредством обычного метода наименьших квадратов, который и в данном случае будет наиболее эффективной из линейных оценок.

Модели различаются, в первую очередь, функциональными формами для условной дисперсии, а также базовыми распределениями для ϵ_t (по

умолчанию нормальное распределение, однако применяются и другие распределения с более тяжелыми хвостами)

ARCH-моделью порядка q (обозначают ARCH(q)) называют временной ряд u_t с функцией условной дисперсии следующего вида:

$$\sigma_t^2 = \beta + \sum_{i=1}^{q} \alpha_i u_{t-i}^2$$

Для недопущения отрицательных значений дисперсии предполагается, что все коэффициенты модели неотрицательны, причём константа строго положительна. Если данный процесс стационарный, то безусловная дисперсия постоянна. Необходимое условие стационарности — сумма коэффициентов модели (без константы) строго меньше единицы. Если сумма коэффициентов равна единице, имеем интегрированный ARCH (нестационарный). ARCH-процессы характеризуются положительным эксцессом («толстые хвосты»).

ARCH-модель предполагает зависимость условной дисперсии только от квадратов прошлых значений временного ряда. Обобщить данную модель можно предположив, что условная дисперсия зависит также от прошлых значений самой условной дисперсии. Это так называемый обобщённый ARCH (Generalized ARCH — GARCH). В этом случае **GARCH(p, q) модель** (где p — порядок GARCH-членов σ^2 и q — порядок ARCH-членов u^2) описывается следующим образом:

$$\sigma_t^2 = \beta + \sum_{i=1}^{q} \alpha_i u_{t-i}^2 + \sum_{i=1}^{p} \theta_i \sigma_{t-i}^2$$

Необходимое условие: $\sum_{i=1}^{q} \alpha_i + \sum_{i=1}^{p} \theta_i < 1$. Безусловная дисперсия стационарного GARCH(p, q)-процесса будет постоянна. Если сумма коэффициентов равна единице, то имеем интегрированный GARCH — IGARCH, безусловная дисперсия которого бесконечна.

Существует множества эвристик GARCH модели: GARCH-в-среднем (GARCH-in-Mean, GARCH-M), асимметричные модели GARCH (EGARCH, AGARCH, TGARCH и GJR-GARCH)

В GARCH-моделях используются различные распределения для лучшего соответствия эмпирическим особенностям финансовых рядов. Уже использование нормального распределения объясняет в значительной степени «толстые хвосты» в распределении доходностей. Тем не менее, этого оказывается недостаточно. Часто полезным оказывается использование распределение Стьюдента с малым числом степеней свободы, которое само имеет более толстые хвосты по сравнению с нормальным распределением. Такие модели иногда обозначают GARCH-t. В целях учёта асимметрии используют также специальное скошенное распределение Стьюдента (t-распределение Хансена). Такие модели иногда обозначают GARCH-HT

2.5 Копулы

Пусть X_1X_2 - случайные величины, функции распределения вероятности которых определены на множествах A и B, соответственно. Будем обозначать i-ую реализацию j-ой случайной величины как $x_j(i)$.

Вначале будем называть функцию $C(X_1, X_2)$ возрастающей по каждой переменной X_1X_2, если для нее выполняется следующее условие:

$$C\left(x_1(2), x_2(2)\right)+C\left(x_1(1), x_2(1)\right)-C\left(x_1(2), x_2(1)\right)-C\left(x_1(1), x_2(2)\right) \geq 0 \text{ при } x_j(1) \leq x_j(2);$$

Определим подкопулу $C\left(X_1, X_2\right)$ как двумерную функцию двух переменных X_1 и X_2, определенную на таком множестве $A \cdot B$, что $A \in [0;1]$ и $B \in [0;1]$, с областью значений $[0;1]$ и удовлетворяющую следующим условиям: (1) Ограничение снизу, т.е. $C\left(X_1, X_2\right) = 0$, если $\exists i : X_i = 0$; (2) $C\left(X_1, X_2\right) = X_i$, если $\forall j \neq i : X_j = 1$; (3) Возрастание по каждой переменной;

Копула - это подкопула в случае, когда $A = [0;1]$ и $B = [0;1]$. Именно на данном этапе возможно применить копулы к моделированию совместных вероятностных распределений, поскольку вероятность любой случайной величины также принадлежит отрезку от нуля до единицы. Для соответствующего перехода воспользуемся теоремой Шкляра.

Теорема Шкляра: Пусть H - совместная функция распределения двух случайных величин (x, y), которые имеют частные функции распределения F и G, соответственно. Тогда существует такая копула C, что для любого x, y, т.е.

$$\exists C : H(x, y) = C[F(x); G(y)], \forall x, y \in (-\infty; +\infty)$$

Причем если функции F и G непрерывны, то копула C единственна; в противном случае, копула C может быть всегда определена на области значений F и G. Наоборот, если C - копула, а F и G - частные (маргинальные) функции распределения, то функция H, определенная выше, является функцией совместного распределения с аргументами F и G.

Таким образом, копула является вероятностью наступления совместного события для случайных величин, т.е. $C\left(X_1, X_2\right) = P\left(X_1 \leq x_1; X_2 \leq x_2\right)$.

Свойства копул - Ограниченность

$$0 \leq C\left(x_1, \ldots, x_n\right) \leq 1$$

- Любая копула лежит в границах Фреше-Хефдинга (Frechet-Hoeffding)

$$\text{Max}(0, u + v - 1) \leq C(u, v) \leq \text{Min}(u, v)$$

- Упорядоченность (доминирование). Копула C_2 доминирует над копулой C_1, или $C_1 \prec C_2$, в случае, когда для $\forall x_1, \ldots, x_n$ верно $C_1\left(x_1, \ldots, x_n\right) \leq C_2\left(x_1, \ldots, x_n\right)$;

N-мерная копула - это функция, определенная на n-мерном единичном кубе $[0;1]^n$ с областью значений $[0;1]$ и удовлетворяющая следующим условиям: (1) Ограничение снизу, т.е. $C\left(x_1, \ldots, x_n\right) = 0$, если $\exists i : x_i = 0$;

(2) $C(x_1, \ldots, x_n) = x_i$, если $\forall j \neq i : x_j = 1$; (3) Возрастание по каждой переменной;

Существует много видов копул, одни из них: Двумерная гауссова копула определяется как

$$C(u,v) = \int_{-\infty}^{\phi^{-1}(u)} \int_{-\infty}^{\phi^{-1}(v)} \left(1/2\pi|R|^{1/2}\right) \exp\left\{-(u,v)'R^{-1}(u,v)/2\right\} dudv$$

где ϕ представляет собой одномерную стандартную нормальную функцию распределения u и v и R ссылается на корреляционную матрицу. Двумерная Стьюдент-t копула определяется как

$$C_t(u,v;R,n) = \int_{-\infty}^{t_n^{-1}(u)} \int_{-\infty}^{t_n^{-1}} \left(\Gamma(n+2)/2|R|^{-1/2}\right) / (\Gamma(n/2)(n\pi)) \left(1 + 1/n(u,v)'R^{-1}(u,v)\right)^{-(n+2)/2} dudv$$

где $t_n^{-1}(u)$ обратная функция распределения стандартного одномерного Стьюдент-t распределения с v степени свободы. R - корреляционная матрица.

2.6 Copula-GARCH модель

Говорят, что n-мерный временной ряд $\boldsymbol{r}_t^T = (r_{1t}, r_{2t}, \ldots, r_{nt})$ следует копула-GARCH процессу, если его многомерное условное распределение имеет вид (76):

$$F\left(\boldsymbol{r}_t \mid \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2\right) = C\left(F\left(r_{1t} \mid \mu_{1t}, \sigma_t^2\right), F\left(r_{2t} \mid \mu_{2t}, \sigma_t^2\right), \ldots, F\left(r_{nt} \mid \mu_{nt}, \sigma_t^2\right)\right),$$
$$\text{где } \boldsymbol{\mu}_t = (\mu_{1t}, \mu_{2t}, \ldots, \mu_{nt}), \boldsymbol{\sigma}_t^2 = (\sigma_{1t}, \sigma_{2t}, \ldots, \sigma_{nt}).$$

Основная идея использования копула-GARCH модели заключается в том, чтобы при помощи соответствующей динамической модели моделировать каждый одномерный временной ряд независимо, а затем моделировать зависимость данных временных рядов при помощи копулы. Преимущество данного подхода заключается в том, что он не требует предположения о нормальном распределении вектора доходностей активов и позволяет конструировать гибкие многомерные распределения через копулы.

3 Правктическая часть

3.1 Сбор данных

В наших данных мы не будем напрямую использовать цены на нефть и курс доллара к рублю, а поступим следующим образом: 1) Отнормируем цены на нефть в долларах:

$$X = \frac{oil\_price}{PPI_{USA}}$$

2) Отнормируем курс доллара к рублю:

$$Y = 100 * (F_{USD/RUB} * \frac{CPI_{USA}}{CPI_{RUS}})^{-1}$$

, где $F_{USD/RUB}$ - обменный курс доллара к рублю Для сбора данных использовались следующие ресурсы (период 2016-01-01 - 2022-12-01):

- oil prices `https://www.kaggle.com/datasets/sc231997/crude-oil-price`
- PPI `https://data.oecd.org/price/producer-price-indices-ppi.htm`
- курс доллара к рублю `https://ru.investing.com/currencies/usd-rub-historical-data`
- CPI_{RUS} `https://www.econdb.com/series/CPIRU/russian-federation-consumer-price-index/?from=2018-12-04&to=2023-12-04`
- CPI_{USA} `https://fred.stlouisfed.org/series/CPIAUCSL`

3.2 Обучение модели

In [341…
```
import numpy as np
import pandas as pd
from datetime import datetime

date_parse = '%Y-%m-%d'
```

In [342…
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn

seaborn.set_style("darkgrid")
plt.rc("figure", figsize=(16, 6))
plt.rc("savefig", dpi=90)
plt.rc("font", family="sans-serif")
plt.rc("font", size=14)
```

Сбор данных

Обучение модели будет происходить на данных с использованием цен на сырую нефть (USD) и курса доллара к рублю:

In [343…
```
ppi_monthly_usa = pd.read_csv("ppi.csv")
ppi_monthly_usa = ppi_monthly_usa[['Value', 'TIME']][1848:1932]
ppi_monthly_usa = ppi_monthly_usa.reset_index()[['Value', 'TIME']]
ppi_monthly_usa
```

Out[343…

	Value	TIME
0	96.84721	2016-01
1	96.30827	2016-02
2	96.63164	2016-03
3	97.27836	2016-04
4	98.03288	2016-05
...	...	...
79	135.53970	2022-08
80	135.82860	2022-09
81	136.88980	2022-10
82	136.27220	2022-11
83	131.74990	2022-12

84 rows × 2 columns

In [344…
```
crude_oil = pd.read_csv("crude-oil-price.csv")
crude_oil = crude_oil[394:478][['date', 'price']]
crude_oil = crude_oil.reset_index()[['date', 'price']]
crude_oil
```

Out[344…

	date	price
0	2016-01-01T00:00:00	33.62
1	2016-02-01T00:00:00	33.75
2	2016-03-01T00:00:00	38.34
3	2016-04-01T00:00:00	45.92
4	2016-05-02T00:00:00	49.10
...	...	...
79	2022-08-01T00:00:00	89.03
80	2022-09-01T00:00:00	78.72
81	2022-10-03T00:00:00	85.40
82	2022-11-01T00:00:00	80.66
83	2022-12-01T00:00:00	80.45

84 rows × 2 columns

In [345…
```
usd_rub = pd.read_csv("usd_rub.csv")
usd_rub = usd_rub[["Дата", "Цена"]].iloc[::-1]
```

```
usd_rub = usd_rub.reset_index()[["Дата", "Цена"]]
usd_rub
```

Out[345…

	Дата	Цена
0	01.01.2016	75.4644
1	01.02.2016	75.1682
2	01.03.2016	67.0288
3	01.04.2016	64.6542
4	01.05.2016	66.6968
...	...	...
79	01.08.2022	60.2300
80	01.09.2022	58.4500
81	01.10.2022	61.4775
82	01.11.2022	60.9900
83	01.12.2022	69.9000

84 rows × 2 columns

In [346…
```
us_cpi = pd.read_csv("CPIAUCSL.csv")
us_cpi
```

Out[346…

	DATE	CPIAUCSL
0	2016-01-01	237.652
1	2016-02-01	237.336
2	2016-03-01	238.080
3	2016-04-01	238.992
4	2016-05-01	239.557
...	...	...
79	2022-08-01	295.320
80	2022-09-01	296.539
81	2022-10-01	297.987
82	2022-11-01	298.598
83	2022-12-01	298.990

84 rows × 2 columns

In [347…
```
cpi_rus = pd.read_csv("cpi_rus.csv")[300:384]
cpi_rus = cpi_rus.reset_index()[["Date", "CPIRU"]]
cpi_rus
```

Out[347…

	Date	CPIRU
0	2016-01-01	158.5
1	2016-02-01	159.5
2	2016-03-01	160.2
3	2016-04-01	160.9
4	2016-05-01	161.6
...	...	...
79	2022-08-01	228.7
80	2022-09-01	228.8
81	2022-10-01	229.3
82	2022-11-01	230.1
83	2022-12-01	231.9

84 rows × 2 columns

In [348…
```
oil_normalized = pd.DataFrame()
oil_normalized["price"] = crude_oil["price"] / ppi_monthly_usa["Value"]
oil_normalized["Date"] = cpi_rus["Date"]
oil_normalized["Date"] = oil_normalized["Date"].apply(lambda x: datetime.strptime(x, date_parse))
```

```
oil_normalized = oil_normalized.set_index("Date")["price"]
oil_normalized
```

Out[348…
```
Date
2016-01-01    0.347145
2016-02-01    0.350437
2016-03-01    0.396764
2016-04-01    0.472047
2016-05-01    0.500852
                ...
2022-08-01    0.656856
2022-09-01    0.579554
2022-10-01    0.623859
2022-11-01    0.591904
2022-12-01    0.610627
Name: price, Length: 84, dtype: float64
```

In [349…
```
exchange_normalized = pd.DataFrame()
exchange_normalized["price"] = 100 / (usd_rub["Цена"] * us_cpi["CPIAUCSL"] / cpi_rus["CPIRU"])
exchange_normalized["Date"] = cpi_rus["Date"]
exchange_normalized["Date"] = exchange_normalized["Date"].apply(lambda x: datetime.strptime(x, date_parse))
exchange_normalized = exchange_normalized.set_index("Date")["price"]
exchange_normalized
```

Out[349…
```
Date
2016-01-01    0.883783
2016-02-01    0.894052
2016-03-01    1.003872
2016-04-01    1.041300
2016-05-01    1.011411
                ...
2022-08-01    1.285762
2022-09-01    1.320048
2022-10-01    1.251672
2022-11-01    1.263488
2022-12-01    1.109601
Name: price, Length: 84, dtype: float64
```

Обучение copula-based GARCH модели

Для того чтобы обучить copula-based GARCH модель нужно действовать по следующему плану:

1. Обучить GARCH модель
2. Получить стандартизированные остатки
3. Применить интегральное преобразование вероятности к остаткам
4. На данных из п.3 обучить copula

In [350…
```
import arch
from scipy.stats import norm
from scipy.optimize import minimize
```

Будем предсказывать последние пол года

In [351…
```
split_date = datetime(2022, 7, 1)
```

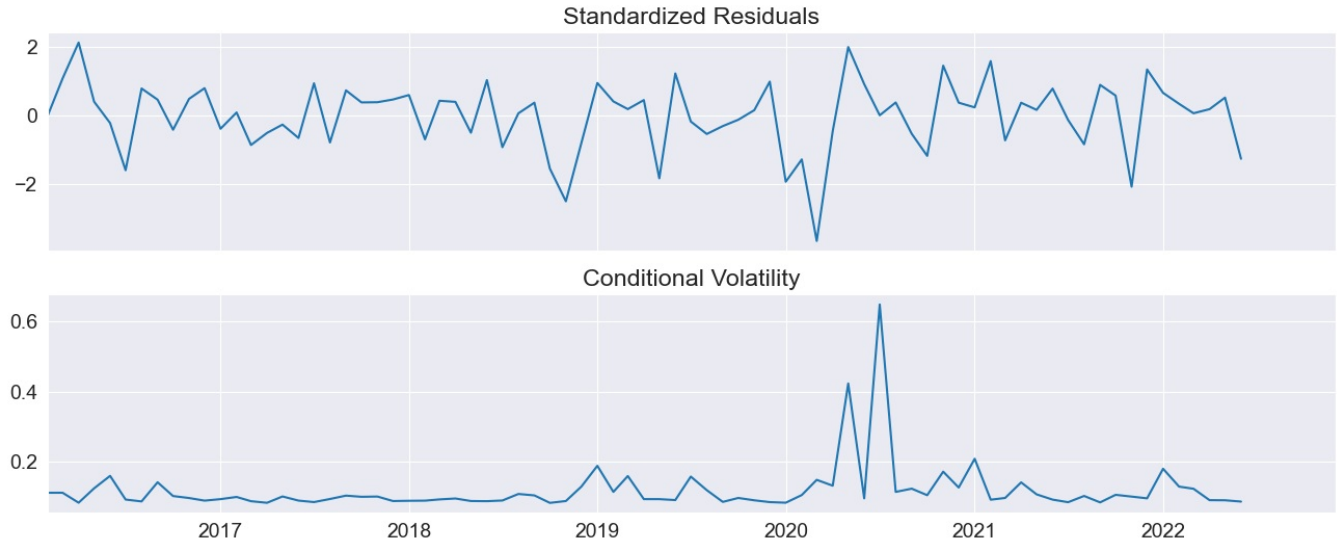
In [352…
```
# Step a) Fit ARMA-GARCH model and extract standardized residuals
data = oil_normalized.pct_change().dropna()
arma_model = arch.arch_model(data, vol='Garch', p=5, q=5)
result = arma_model.fit(disp="off", last_obs=split_date)
standardized_residuals = result.resid / result.conditional_volatility
standardized_residuals_oil = standardized_residuals
print(result.summary())
```

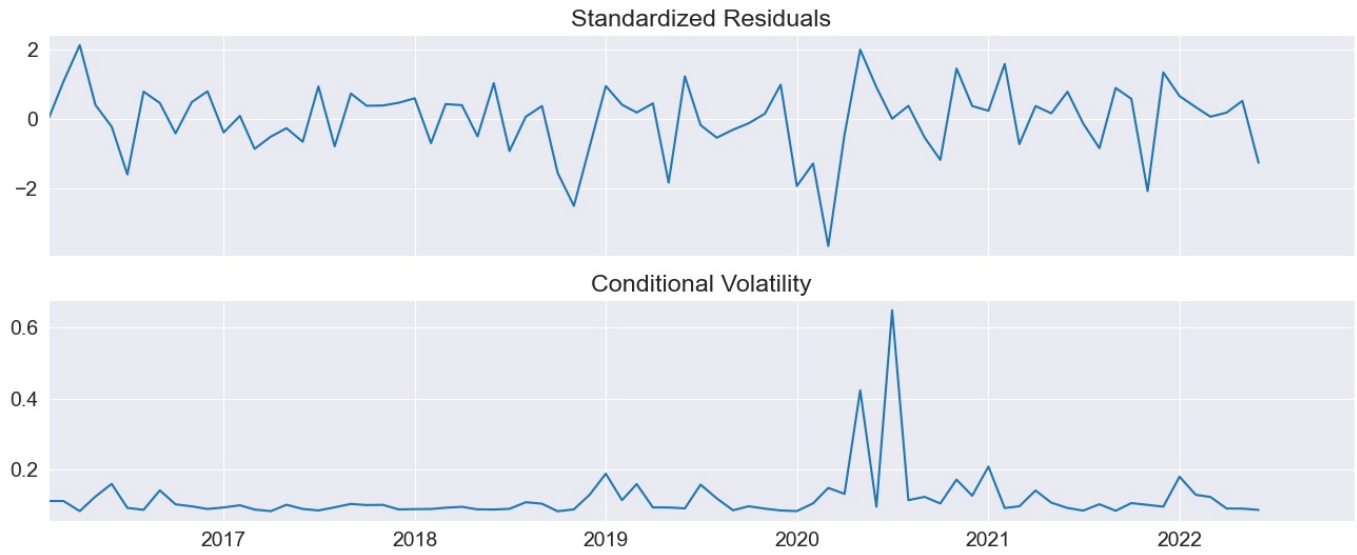
```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                60.5328
Distribution:                  Normal   AIC:                           -97.0655
Method:            Maximum Likelihood   BIC:                           -68.9399
                                        No. Observations:                   77
Date:                Tue, Dec 05 2023   Df Residuals:                       76
Time:                        13:30:08   Df Model:                            1
                                Mean Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu             0.0105  1.154e-02      0.911      0.362 [-1.210e-02,3.314e-02]
                              Volatility Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
omega      6.5396e-03  2.891e-03      2.262  2.370e-02  [8.730e-04,1.221e-02]
alpha[1]   4.9381e-10  1.296e-02  3.811e-08      1.000 [-2.539e-02,2.539e-02]
alpha[2]       0.5765  2.806e-02     20.549  7.834e-94      [  0.522,  0.632]
alpha[3]   5.3905e-10      0.190  2.843e-09      1.000      [ -0.372,  0.372]
alpha[4]   5.2248e-10      0.148  3.541e-09      1.000      [ -0.289,  0.289]
alpha[5]       0.0000      0.139      0.000      1.000      [ -0.273,  0.273]
beta[1]    7.6323e-10      0.332  2.299e-09      1.000      [ -0.651,  0.651]
beta[2]    7.5466e-10      0.229  3.291e-09      1.000      [ -0.449,  0.449]
beta[3]    7.2427e-03      0.240  3.024e-02      0.976      [ -0.462,  0.477]
beta[4]        0.0491  4.459e-02      1.102      0.271   [-3.827e-02,  0.137]
beta[5]    7.4403e-10  3.326e-02  2.237e-08      1.000 [-6.518e-02,6.518e-02]
==============================================================================

Covariance estimator: robust
```

```
/Users/anymax/Documents/financials/venv/lib/python3.8/site-packages/arch/univariate/base.py:310: DataScaleWarnin
g: y is poorly scaled, which may affect convergence of the optimizer when
estimating the model parameters. The scale of y is 0.02109. Parameter
estimation work better when this value is between 1 and 1000. The recommended
rescaling is 10 * y.

This warning can be disabled by either rescaling y before initializing the
model or by setting rescale=False.

  warnings.warn(
```

In [353…
```
result.plot()
```

Out[353…

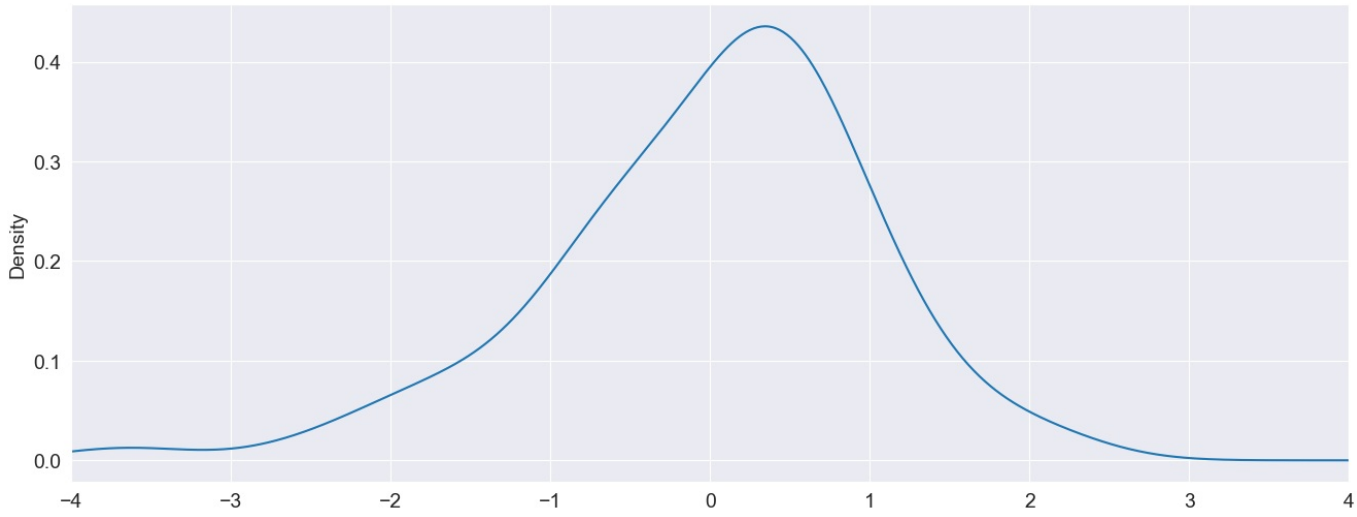




Проверим, что действительно получили нормальное распределение

In [354…
```
standardized_residuals.plot(kind="kde", xlim=(-4, 4))
```

Out[354…
```
<Axes: ylabel='Density'>
```



In [355…
```
standardized_residuals = standardized_residuals[:-6]
```

Теперь получим из standardized_residuals остатки из распределения Uniform(0, 1)

In [356…
```
# Step b) Apply PIT to standardized residuals
pit_residuals = norm.cdf(standardized_residuals)
```

In [357…
```
from copulas.univariate import GaussianUnivariate

# Step c) Fit desired copula to PIT residuals
copula = GaussianUnivariate()
copula.fit(pit_residuals)
```

Применение модели

Генерация последнего полугода

1. Генерация копул на нужный период
2. Получение смоделированных стандартизированных остатков, используя обратную функцию
3. Получение среднего и дисперсии из обученной GARCH модели
4. Результат = среднее + дисперсия * остатки из п.2

In [358…
```
N = 1
# Step d) Run simulation for n times using copula coefficients
def simulate_copula(copula, n):
    simulated_pit_residuals = copula.sample(n)
    return simulated_pit_residuals

# Step e) Obtain simulated standardized residuals using inverse function
def invert_pit(simulated_pit_residuals):
    simulated_standardized_residuals = norm.ppf(simulated_pit_residuals)
    return simulated_standardized_residuals

simulated_asset_res = []
for i in range(N):
    simulated_pit_residuals = simulate_copula(copula, 6)
    print("sample", simulated_pit_residuals)
    simulated_standardized_residuals = invert_pit(simulated_pit_residuals)
    print("copulas", simulated_standardized_residuals)
    forecast_mean = result.forecast(horizon=5, start=split_date).mean['h.1'].iloc[-1]
    forecast_variance = result.forecast(horizon=5, start=split_date).variance['h.1'].iloc[-1]
    simulated_asset = forecast_mean + (forecast_variance) * simulated_standardized_residuals
    for i in range(6):
        if i == 0:
            simulated_asset[i] = oil_normalized[-1] + oil_normalized[-1] * simulated_asset[i]
        else:
            simulated_asset[i] = simulated_asset[i - 1] + simulated_asset[i - 1] * simulated_asset[i]
    simulated_asset_res.append(simulated_asset)

simulated_asset_res
```

```
sample [0.62729829 0.85240566 0.81023971 0.50118572 0.24946012 0.49589218]
copulas [ 0.32470622  1.04680684  0.87877998  0.00297217 -0.67618966 -0.01029695]
/Users/anymax/Documents/financials/venv/lib/python3.8/site-packages/arch/__future__/_utility.py:11: FutureWarnin
g:
The default for reindex is True. After September 2021 this will change to
False. Set reindex to True or False to silence this message. Alternatively,
you can use the import comment

from arch.__future__ import reindexing

to globally set reindex to True and silence this warning.

  warnings.warn(
/Users/anymax/Documents/financials/venv/lib/python3.8/site-packages/arch/__future__/_utility.py:11: FutureWarnin
g:
The default for reindex is True. After September 2021 this will change to
False. Set reindex to True or False to silence this message. Alternatively,
you can use the import comment

from arch.__future__ import reindexing

to globally set reindex to True and silence this warning.

  warnings.warn(
```

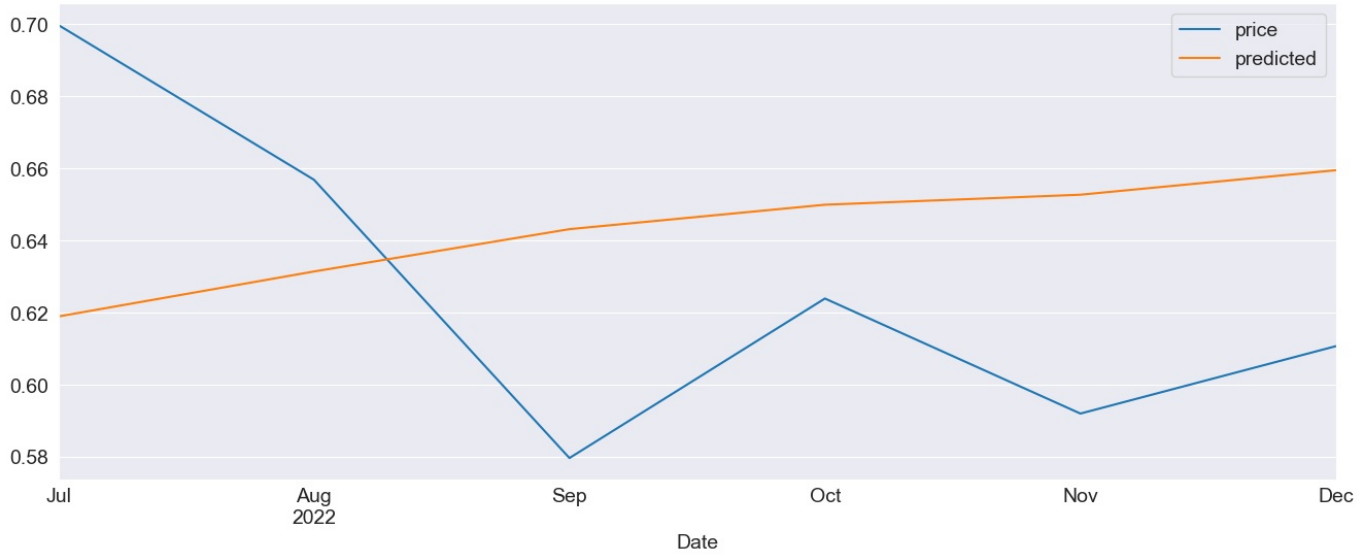
Out[358…
```
[array([0.61887906, 0.63136599, 0.64312613, 0.64990876, 0.65269067,
        0.65949428])]
```

In [359…
```
simulated_asset_res_oil = simulated_asset_res
```

Сравним с исходными данными

In [360…
```
test = oil_normalized[-6:]
df = pd.DataFrame(test)
df["predicted"] = simulated_asset_res[0]
df.plot()
```

Out[360…
```
<Axes: xlabel='Date'>
```



Проделаем то же самое с exchange_normalized

In [361…

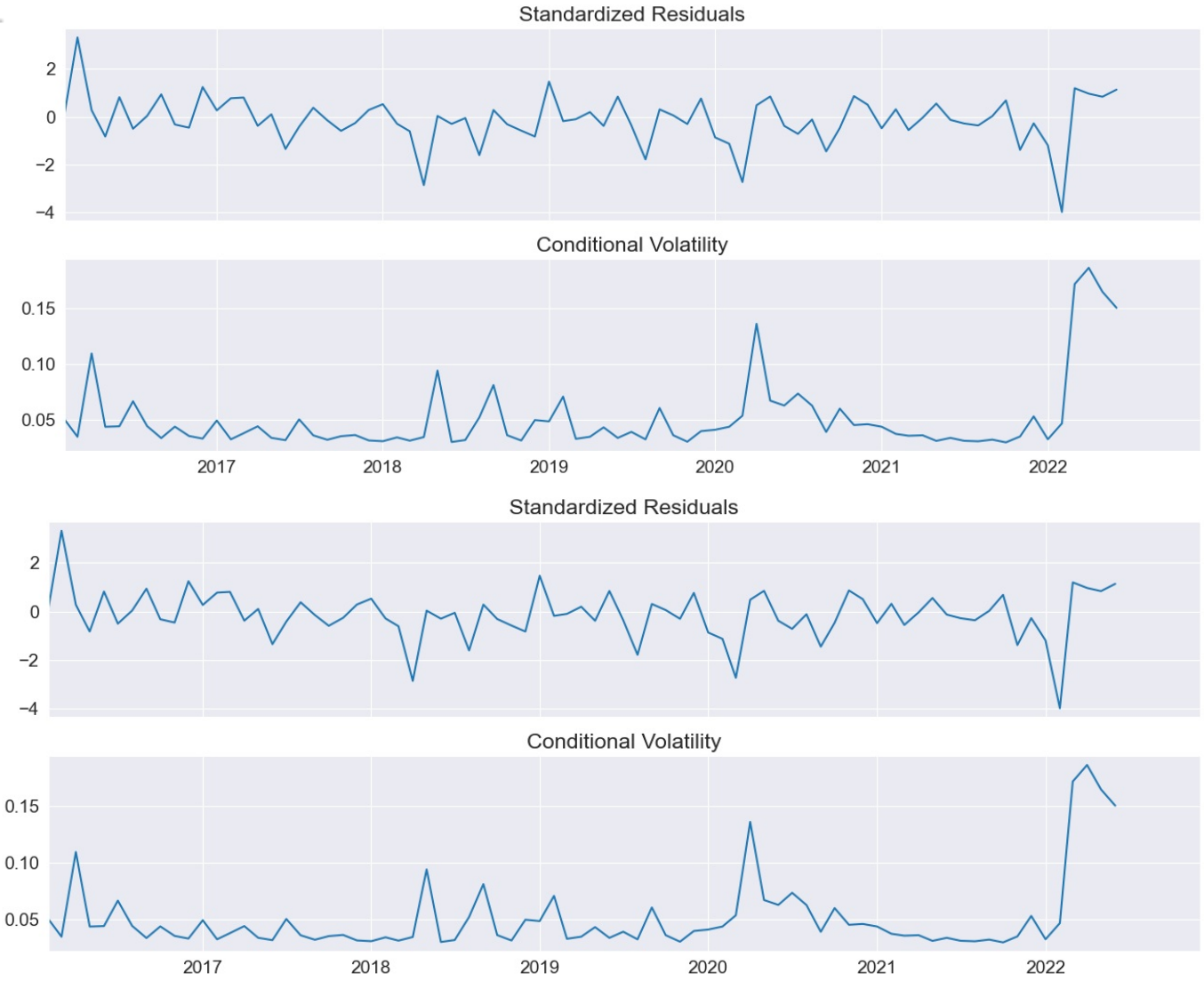
```
# Step a) Fit ARMA-GARCH model and extract standardized residuals
data = exchange_normalized.pct_change().dropna()
arma_model = arch.arch_model(data, vol='Garch', p=5, q=5)
result = arma_model.fit(disp="off", last_obs=split_date)
standardized_residuals = result.resid / result.conditional_volatility
standardized_residuals_exchange = standardized_residuals
print(result.summary())
```

```
/Users/anymax/Documents/financials/venv/lib/python3.8/site-packages/arch/univariate/base.py:310: DataScaleWarnin
g: y is poorly scaled, which may affect convergence of the optimizer when
estimating the model parameters. The scale of y is 0.003394. Parameter
estimation work better when this value is between 1 and 1000. The recommended
rescaling is 10 * y.

This warning can be disabled by either rescaling y before initializing the
model or by setting rescale=False.

  warnings.warn(
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                127.890
Distribution:                  Normal   AIC:                           -231.780
Method:            Maximum Likelihood   BIC:                           -203.655
                                        No. Observations:                   77
Date:                Tue, Dec 05 2023   Df Residuals:                       76
Time:                        13:30:09   Df Model:                           1
                                Mean Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
mu         7.5568e-03  3.325e-03      2.272  2.306e-02 [1.039e-03,1.407e-02]
                              Volatility Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
omega      8.8644e-04  8.001e-04      1.108      0.268 [-6.818e-04,2.455e-03]
alpha[1]       0.8116      0.402      2.018  4.359e-02    [2.333e-02,  1.600]
alpha[2]       0.0000      0.447      0.000      1.000      [ -0.875,  0.875]
alpha[3]       0.0000      0.433      0.000      1.000      [ -0.849,  0.849]
alpha[4]       0.1884      0.560      0.336      0.737      [ -0.910,  1.287]
alpha[5]       0.0000      0.741      0.000      1.000      [ -1.452,  1.452]
beta[1]        0.0000      0.253      0.000      1.000      [ -0.496,  0.496]
beta[2]        0.0000      0.471      0.000      1.000      [ -0.922,  0.922]
beta[3]        0.0000      0.426      0.000      1.000      [ -0.834,  0.834]
beta[4]        0.0000      0.433      0.000      1.000      [ -0.849,  0.849]
beta[5]        0.0000      0.169      0.000      1.000      [ -0.332,  0.332]
==============================================================================

Covariance estimator: robust
```

In [362…
```
result.plot()
```

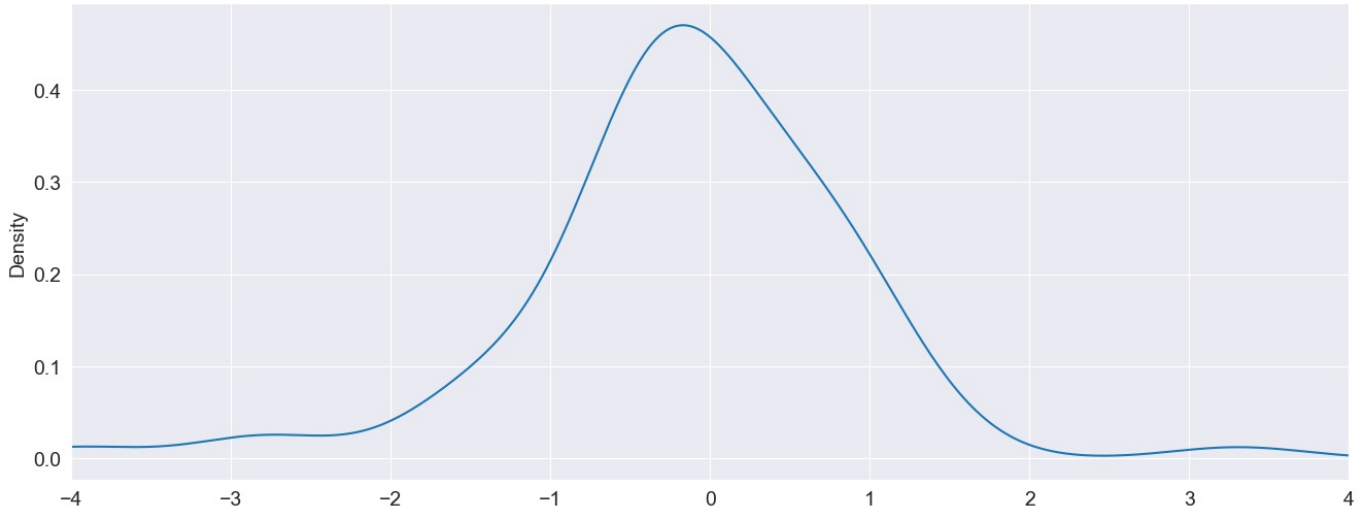
Out[362…



Проверим, что действительно получили нормальное распределение

In [363…
```
standardized_residuals.plot(kind="kde", xlim=(-4, 4))
```

Out[363… <Axes: ylabel='Density'>



In [364…
```
standardized_residuals = standardized_residuals[:-6]
```

Теперь получим из standardized_residuals остатки из распределения Uniform(0, 1)

In [365…
```
# Step b) Apply PIT to standardized residuals
pit_residuals = norm.cdf(standardized_residuals)
```

In [366…
```
from copulas.univariate import GaussianUnivariate

# Step c) Fit desired copula to PIT residuals
copula = GaussianUnivariate()
copula.fit(pit_residuals)
```

Применение модели

Генерация последнего полугода

1. Генерация копул на нужный период
2. Получение смоделированных стандартизированных остатков, используя обратную функцию
3. Получение среднего и дисперсии из обученной GARCH модели
4. Результат = среднее + дисперсия * остатки из п.2

In [372…
```
N = 1
# Step d) Run simulation for n times using copula coefficients
def simulate_copula(copula, n):
    simulated_pit_residuals = copula.sample(n)
    return simulated_pit_residuals

# Step e) Obtain simulated standardized residuals using inverse function
def invert_pit(simulated_pit_residuals):
    simulated_standardized_residuals = norm.ppf(simulated_pit_residuals)
    return simulated_standardized_residuals

simulated_asset_res = []
for i in range(N):
    simulated_pit_residuals = simulate_copula(copula, 6)
    print("sample", simulated_pit_residuals)
    simulated_standardized_residuals = invert_pit(simulated_pit_residuals)
    print("copulas", simulated_standardized_residuals)
    forecast_mean = result.forecast(horizon=5, start=split_date).mean['h.1'].iloc[-1]
    forecast_variance = result.forecast(horizon=5, start=split_date).variance['h.1'].iloc[-1]
    simulated_asset = forecast_mean + (forecast_variance) * simulated_standardized_residuals
    for i in range(6):
        if i == 0:
            simulated_asset[i] = exchange_normalized[-1] + exchange_normalized[-1] * simulated_asset[i]
        else:
            simulated_asset[i] = simulated_asset[i - 1] + simulated_asset[i - 1] * simulated_asset[i]
    simulated_asset_res.append(simulated_asset)

simulated_asset_res
```

```
sample [0.92272235 0.24831425 0.90717934 0.67008497 0.52124455 0.62374669]
copulas [ 1.42362416 -0.67980411  1.32358378  0.44014781  0.05327739  0.31533592]
```

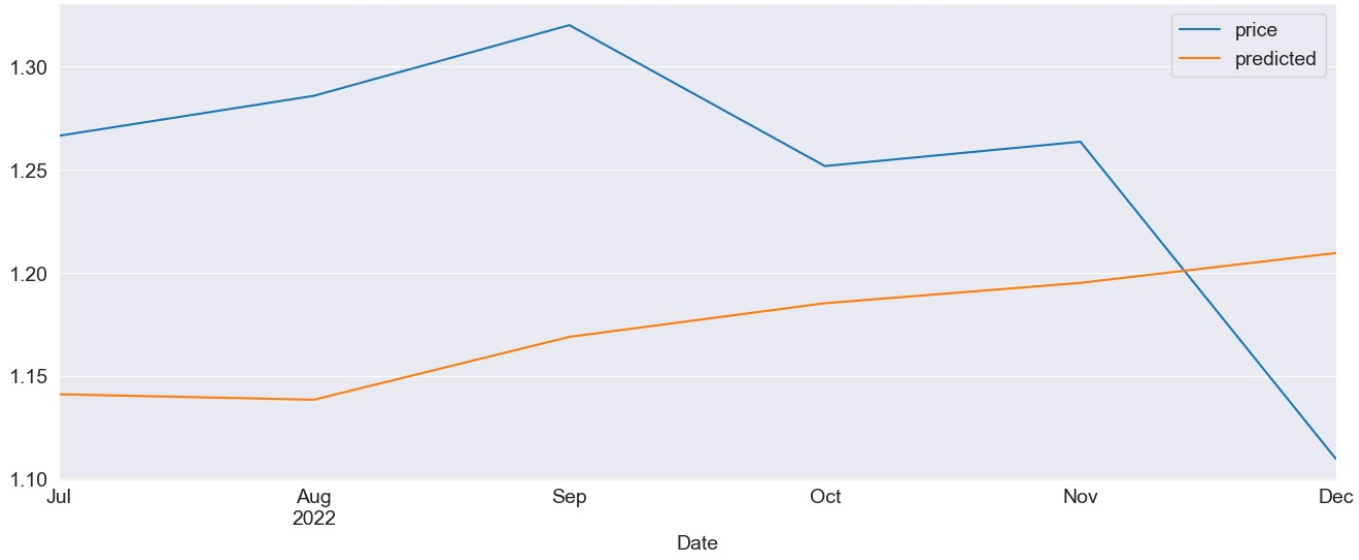
```
/Users/anymax/Documents/financials/venv/lib/python3.8/site-packages/arch/__future__/_utility.py:11: FutureWarnin
g:
The default for reindex is True. After September 2021 this will change to
False. Set reindex to True or False to silence this message. Alternatively,
you can use the import comment

from arch.__future__ import reindexing

to globally set reindex to True and silence this warning.

  warnings.warn(
/Users/anymax/Documents/financials/venv/lib/python3.8/site-packages/arch/__future__/_utility.py:11: FutureWarnin
g:
The default for reindex is True. After September 2021 this will change to
False. Set reindex to True or False to silence this message. Alternatively,
you can use the import comment

from arch.__future__ import reindexing

to globally set reindex to True and silence this warning.

  warnings.warn(
```

Out[372… `[array([1.14094743, 1.13829525, 1.16879686, 1.18510697, 1.19498037, 1.2094879 ])]`

In [373…
```
simulated_asset_res_exchange = simulated_asset_res
```

In [374…
```
test = exchange_normalized[-6:]
df = pd.DataFrame(test)
df["predicted"] = simulated_asset_res[0]
df.plot()
```

Out[374… `<Axes: xlabel='Date'>`

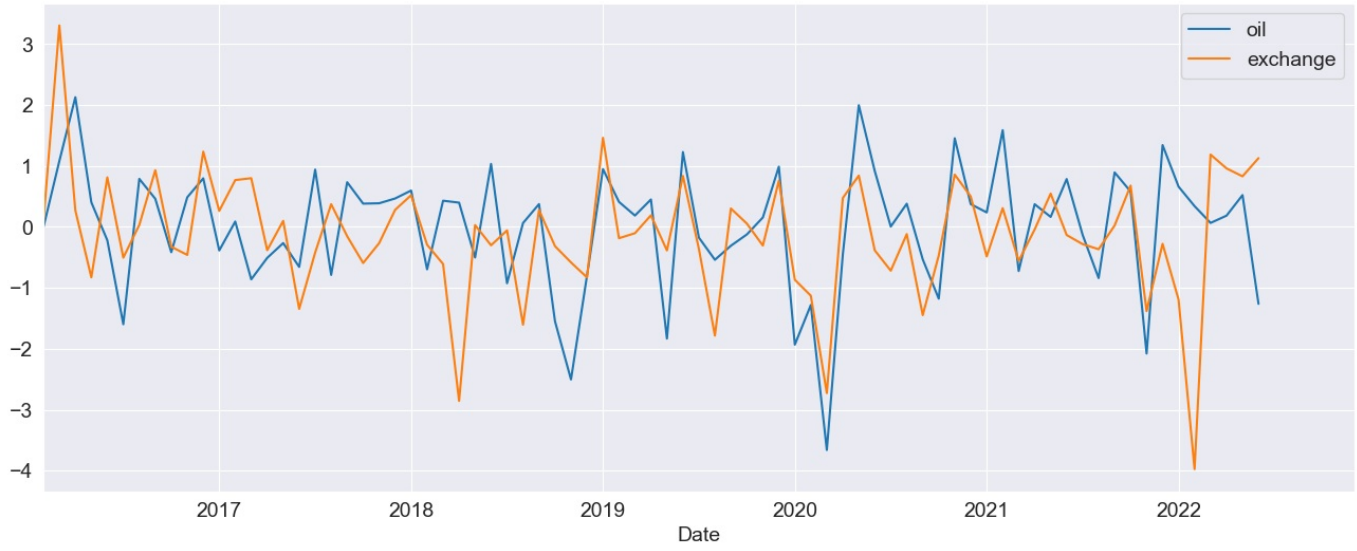


Анализ

Посмотрим на взаимосвязь стандартизированных остатков По графику видно, что данные коррелируют между собой, отличие сильное лишь в 2022 году, это может быть связано с началом специальной операции между Россией и Украиной, вследствии чего рубль сильно упал

In [370…
```
df = pd.DataFrame(standardized_residuals_oil)
df.rename(columns = {0 : 'oil'}, inplace=True )
df_2 = pd.DataFrame(standardized_residuals_exchange)
df_2.rename(columns = {0 : 'exchange'}, inplace=True )
df["exchange"] = df_2["exchange"]
df.plot()
```

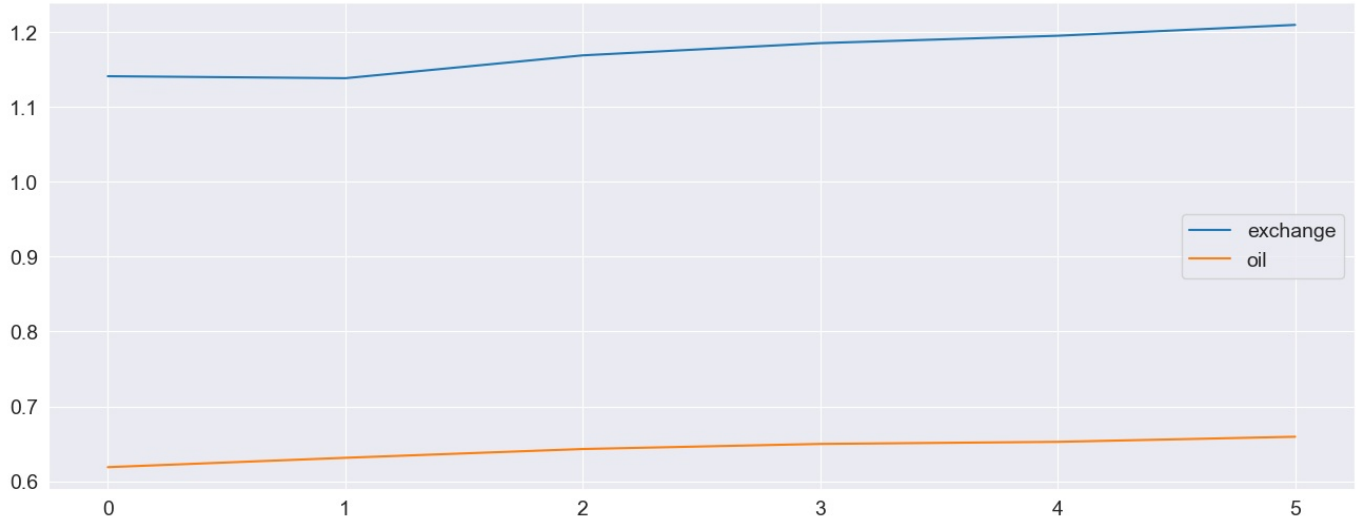
Out[370… `<Axes: xlabel='Date'>`



Посмотрим на то, как коррелируют между собой результаты полученные из двух моделей: По графику как сильно похожи предсказания для exchange и oil prices, значит copula GARCH модель хорошо описывает связь между ними

In [378…
```
plt.plot(np.arange(6), simulated_asset_res_exchange[0], label='exchange')
plt.plot(np.arange(6), simulated_asset_res_oil[0], label='oil')
plt.legend()
```

Out[378… `<matplotlib.legend.Legend at 0x2a43f5670>`



4 Литература

Список литературы

[1] GARCH model: структура зависимости между ценами на сырую нефть и тремя обменными курсами доллара США (Китай, Индия и Южная Корея) URL: https://www.aimspress.com/article/10.3934/energy.2019.4.465

[2] Исследование трех GARCH моделей (GARCH, GJR-GARCH and CGARCH) при выявлении взаимосвязи между ценой на нефть и обменным курсом. URL: https://www.ir.nctu.edu.tw/bitstream/11536/15733/1/000300753300028.pdf

[3] Волатильность между ценой на сырую нефть и обменным курсом Нигерии с использованием многомерной модели GARCH. URL: https://seahipaj.org/journals-ci/june-2020/IJIMSEP/full/IJIMSEP-J-1-2020.pdf

[4] О взаимосвязи волатильности обменного курса нефти и доллара США: внутридневной анализ. URL: https://hal.science/hal-04141662/document

[5] Дипломная работа: ПОСТРОЕНИЕ ИНВЕСТИЦИОННОГО ПОРТФЕЛЯ С МИНИМАЛЬНЫМ РИСКОМ. URL: https://nauchkor.ru/uploads/documents/60d96e56e4dde5000108d094.pdf

[6] Применение модели copula-GARCH к временным рядам цен на товары URL: https://github.com/Ocalak/Application-of-the-copula-GARCH-model-to-commodity-price-time-series/blob/main/report.pdf