

```
In [7]: # 2022-10-17 10:30 Понед Фр
# Анализ и прогнозирование временного ряда стоимости угля.
# На основе Project_01_Commodity/TS_commodity_14.ipynb. Пересчет в золото и нефть, Корреляция. Шкалирование.
# Логарифм.изм. дневных стоймостей угля в разн ед.изм.
# ARIMA. Прогноз на основе ARIMA. Прогоноз получился, но очень примитивный. TS_Coal_01.ipynb

# 2022-10-18 10:25 Фр TS_Coal_02.ipynb. Изменил деление при расчете в золоте и нефти
# Добавляю сглажив и экспоненц сглаж.
# Начал sktime

# 2022-10-19 10:20 Фр TS_Coal_03.ipynb. Добавил скользящ средн., закоммент лишнее, по чистил прогноз ARIMA, VAR

# 2022-10-20 10:45 Фр TS_Coal_04.ipynb. Добавить корректн временн индексы в прогноз знач. Продолжить прогноз в золо

# 2022-10-21 15:50 пятн Нвг TS_Coal_05.ipynb. Убираю лишнее.
# Также делаю проверку прогноза на фактических данных.

# 2022-10-22 11:20 субб Огн TS_Coal_07.ipynb. (В варианте Об намудрил с автозаменой).
# Добавить полосы Боллинджера. См. библиотеку TA-Lib (не смог установить на Linux).
# Пересечен скользь средних. VAR прогноз по оценке всех активов в золоте и нефти
# 2022-10-22 15:15 субб Огн TS_Coal_08.ipynb.

# 2022-10-23 11:25 вскр Огн TS_Coal_09.ipynb. Добавляю библ визуал граф mplcyberpunk

# 2022-10-23 16:25 вскр Огн TS_Coal_10.ipynb

# 2022-10-24 10:20 понед Фр TS_Coal_11.ipynb. Пробую Darts - удалил не пошло
# надо пробовать алгоритм из: 01a_forecasting_sklearn_1.ipynb.

# 2022-10-25 10:15 вторн Фр TS_Coal_11.ipynb. Também взять в работу скрипт "LSTM_CodeTrading_2.ipynb"
# Гипотеза. Еще раз взглянуть на скользящие средние на шкалир данным. LSTM убрал в отдельный файл

# 2022-10-25 15:50 вторн Фр TS_Coal_12.ipynb.# Еще смотрел tutorial

# 2022-10-27 11:10 четв Фр TS_Coal_13.ipynb.
# Добавил в папку файл Coal_BollingerBand_1.ipynb - данные технич. анализа по углю на основе pandas_ta и интреакт в
# 16:00 Делал пробный доклад Евгению. Надо укоротить скрипт, добавить тестовое осмысленное изложение, добавить или у

# 2022-10-28 11:40 пятн Нвг TS_Coal_14.ipynb. Вернул S&P500, NASDAQ
```

```
# 2022-10-30 14:20 вскр Огн TS_Coal_15.ipynb. Делаю ссылки как стандартах html. Из вар 15 убрал расчеты по statsmod
# создал TS_Coal_PCA_1.ipynb для тестирования

# 2022-10-31 10:00 пнд Фр TS_Coal_16.ipynb
# добавил в sktime прогноз VARMAX, Prophet. Еще научится добавлять параметры

# 2022-11-01 10:40 пнд Фр TS_Coal_17.ipynb
# делаю различн методы прогнозиров. Потом проверка на фактич пяти - семи прошл периодов и прогноз на 2-3 периода вп
# еще ни как не решил задачу с прогнозом по дневн знач методом VAR, только по месячному df получается.

# 2022-11-02 10:20 вторн Фр Фр TS_Coal_18.ipynb
# Собираю данные акций что бы на них запустить sktime VAR для угля. Создал папку Moex: moex_2.ipynb
# работал с файлом TS_coal_comm_moex_1.ipynb, moex_2.ipynb изучал точность прогноза для факт периода при разных сос

# 2022-11-04 09:30 пятн (5 нояб продолжил в этом же варианте) Огн TS_Coal_19.ipynb. Автоматизация загрузки. Еще над

# 2022-11-06 10:20 вскр Огн TS_Coal_20.ipynb

# 2023-03-15 15:00 срд Фр вар 21. Читал книгу агоритм торг.. и вернулся к этому файлу когда уткнулся в ARIMA
# сейчас пошли сложности с pandas_datareader : TypeError: string indices must be integers
# pip install pandas_datareader==0.9.0
# пытаюсь перейти я yfinance. Переустановил yfinance - данные стали загружаться

# 2023-06-09 14:00 птн TS_Coal_22.ipynb. Пробую для публикации на https://hub.mos.ru/
# вар 23 Пошли ошибки с библ Prophet. убрал

# 2023-06-23 11:00 птн Фр. Думаю о публикации на Хабр. Добавляю Разложение временного ряда на компоненты statsmodel

# 2023-08-16 17:00 срд Фр. вар 25. Опять вернулся к этим вариантам. Повторяю и смотрю что с прогнозам

# 2023-08-23 12:40 Огн. вар 26. Добавил график корр из seaborn более нагл

# 2024-05-19 09:20 вскр Огн. вар 27. 19 апр купил книгу Груздева Прогнозирование временных рядов...". Повторяю свой
# на DELL UBUNTU 24.04 заново устанавливал pandas_ta mplcyberpunk sktime prophet. Причем prophet на удивление усано
```

Start

Оглавление / Краткое содержание

Часть первая

01. Загрузка библиотек Python

02. Загрузка данных

Уголь

Нефть

Золото

Серебро

Платина

Алюминий

Медь

Commodity Index Future

03. Соединение все представленных временных рядов в один dataframe

03.1 Корреляция временных рядов

03.2 Ежедневная волатильность временных рядов

04. Шкалирование временных рядов

05. Переход к измерению в унциях золота

05.1 Корреляция временных рядов выраженная в унциях золота

06. Переход к измерению в баррелях нефти

06.1 Корреляция временных рядов выраженная в баррелях нефти

00. Краткое содержание

07. Представление шкалированной стоимости угля в разных ед. измерения

Цель работы: провести анализ стоимости угля, проанализировать коррелирующие показатели, возможные предикторы и построить прогноз стоимости угля.

07.1 Корреляция стоимости угля выраженная в разных единицах измерения

В **первой части** данной работы представлен анализ временных рядов стоимости основных биржевых товаров, в том числе с **Частью второй** применением показателей из технического анализа. Произведен переход к единицам измерения такие как: унции золота, и тонны нефти. В этом же единицах приведен анализ представленных временных рядов.

08. Построение прогнозов стоимости угля на основании использования библиотеки SKTIME

Во **второй части** произведено прогнозирование стоимости угля с использованием статистического анализа, глубокого обучения в

08.2 Метод прогнозирования SKTIME VAR в доллах

Disclaimer: Принимая во внимание, происходящие события, стирающие понятие рынка, свободного ценообразования, котировок и т.д. все нежеизложенные рассуждения стали носить умозрительный характер, которые могли бы иметь основания в парадигмах, действовавших до 2022 года.

08.3 Метод прогнозирования SKTIME VAR в унциях золота

08.4 Метод прогнозирования SKTIME VAR в баррелях нефти

Введение

Окончание

Динамика биржевых котировок товаров группы commodities подвержены значительной волатильности. Само изменения графиков, а также совокупность влияющих факторов (предикторов) на такие изменения стоимостей изучаются и анализируются как биржевыми спекулянтами так и учёными. Есть обоснованные предположения, что «Случайное блуждание — хорошая стартовая модель для описания финансовых временных рядов». Модель случайного блуждания предполагает, что движение элемента (или группы элементов) не предсказуемо. Поэтому попробуем для оценки тенденций применить методы технического анализа, которые используют биржевые спекулянты.

Следующий возникающий вопрос заключается, а в чем мы производим измерения и почему именно в долл. или фунтах. Однако, согласитесь, что трудно что-либо «измерять» если твоя «лнейка» постоянно увеличивается и причем в некоторых случаях в геометрической прогрессии путём денежной эмиссии в долларах США (см. показатель M2 с сайта Федеральной резервной системы США (<https://fred.stlouisfed.org/categories/29>)). В как же получить "абсолютный" эквивалент истинной стоимости. Ответить на данный вопрос достаточно затруднительно. Отсылаю к экономической теории и теории эффективного фондового рынка.

В таких условияхлагаю рассмотреть возможность и "вернуться" к так называемому «золотому стандарту», добавить к нему ещё и «нефтяной стандарт», заподчикивать биржевые товары в унциях золота и баррелях нефти. Исходя из этого в данных расчётах

и «нефтяной стандарт», далее оценивать биржевые товары в единицах золота и баррелей нефти. Исходя из этого в дальнейших расчетах использование «измеритель» в долл.США будем условно - только для пересчета одних биржевых товаров в тройские унции и баррели.

Многофакторность влияющих предпосылок на волатильность и динамику цен приводит к признанию случайного блуждания, как основной гипотезы. Однако наличие причинно-следственных связей заметны даже не всегда осведомленным обывателям.

Во второй части:

Производится прогнозирование по временным рядам (с акцентом на прогноз стоимости тонны угля) с помощью пакета [sktime](#). (Также ссылка на [GitHub sktime](#))

Пакет Sktime является гибкой модульной платформой с открытым исходным кодом для задач, связанных с исследованиями временных рядов.

как многомерные воспользуемся моделью VAR - векторная авторегрессия. (https://ru.wikipedia.org/wiki/Векторная_авторегрессия) и построим прогноз стоимости активов на ближайшее будущее в разных единицах измерения. Ключом к прогнозированию многомерных макроэкономических временных рядов является отображение как временных вариаций для каждого признака, так и возможных корреляций между ними в каждом периоде. Модель VAR служит таким целям, используя матрицы в качестве коэффициентов для отображения линейных зависимостей между периодами и между объектами. Он предполагает случайный процесс, который генерирует данные, и пытается оценить параметры модели, которые можно рассматривать как плавное приближение к структуре, которая сгенерировала данные.

Часть первая

01. Загрузка библиотек

[К оглавлению](#)

In [8]: # Загрузка необходимых библиотек Python

```
import numpy as np
import pandas as pd

import pandas_datareader as pdr

import yfinance as yf

import matplotlib.pyplot as plt
from matplotlib import pyplot

import mplcyberpunk

from pandas.plotting import scatter_matrix

import seaborn as sns

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import normalize

#for ARIMA
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

import statsmodels.api as sm

# Для разлож временного ряда на компоненты
import pandas_datareader.data as web
import statsmodels.tsa.api as tsa
```

```
import statsmodels.tsa.api as tsa

from pylab import rcParams

import itertools

import warnings

from scipy.stats import pearsonr, spearmanr

# sktime

from sktime.registry import all_estimators

from sktime.forecasting.base import ForecastingHorizon
from sktime.forecasting.arima import ARIMA
from sktime.forecasting.arima import AutoARIMA
from sktime.forecasting.var import VAR
from sktime.forecasting.varmax import VARMAX
from sktime.forecasting.ets import AutoETS
from sktime.forecasting.fbprophet import Prophet
from sktime.forecasting.naive import NaiveForecaster
from sktime.forecasting.bats import BATS
from sktime.forecasting.tbats import TBATS
from sktime.forecasting.theta import ThetaForecaster

from sktime.forecasting.model_evaluation import evaluate
from sktime.forecasting.model_selection import ExpandingWindowSplitter
from sktime.forecasting.model_selection import temporal_train_test_split

from sktime.forecasting.model_selection import (
    ForecastingGridSearchCV,
    ExpandingWindowSplitter)
from sktime.forecasting.compose import MultiplexForecaster
from sktime.forecasting.naive import NaiveForecaster
from sktime.forecasting.theta import ThetaForecaster
from sktime.forecasting.model_evaluation import evaluate

from sktime.utils import plotting
from sktime.utils.plotting import plot_series
```

```
from sktime.performance_metrics.forecasting import mean_absolute_percentage_error
from sktime.performance_metrics.forecasting import MeanAbsolutePercentageError

import pandas_ta as ta

import datetime as dt

import prophet
```

In []:

```
# Задание параметров вывода
# Формат чисел
# pd.option_context('display.float_format', '{:0.20f}'.format)
pd.set_option("display.precision", 5)

pd.set_option('display.max_columns', 25)
# pd.set_option('display.max_rows', 15)

# настройка графиков
plt.style.use("cyberpunk")
```

In [10]: # Сегодня

```
today = pd.Timestamp.now().normalize()
# today = pd.Timestamp.now().to_period('D')

today
```

Out[10]: Timestamp('2024-05-19 00:00:00')

02. Загрузка данных

[К оглавлению](#)

Загружаем данные с <https://finance.yahoo.com/>, сразу получаем представление в графике и строим datagrame по цене закрытия.

Дату начала анализа временных рядов принял за 1 января 2015 года.

```
In [11]: # Задание дата начала выборки данных.
```

```
start='2008-01-01'
```

```
start
```

```
Out[11]: '2008-01-01'
```

```
In [12]: # yfinance
```

```
start_date = dt.datetime(2020,1,1)
end_date = dt.datetime(2024,5,18)
test = yf.download(tickers = "GOOGL", start = start_date, end = end_date)
```

```
test
```

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

Out[12]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2020-01-02	67.42050	68.43400	67.32450	68.43400	68.43400	27278000
2020-01-03	67.40000	68.68750	67.36600	68.07600	68.07600	23408000
2020-01-06	67.58150	69.91600	67.55000	69.89050	69.89050	46768000
2020-01-07	70.02300	70.17500	69.57800	69.75550	69.75550	34330000
2020-01-08	69.74100	70.59250	69.63150	70.25200	70.25200	35314000
...
2024-05-13	164.25999	169.28000	164.00000	169.14000	169.14000	31327600
2024-05-14	169.77000	171.25000	168.80000	170.34000	170.34000	25127100
2024-05-15	170.63000	172.64999	170.50999	172.50999	172.50999	26948400
2024-05-16	173.28999	175.12000	172.69000	174.17999	174.17999	27867900
2024-05-17	174.17999	176.27000	173.69000	176.06000	176.06000	22235500

1102 rows × 6 columns

Загрузка сразу нескольких тикеров с yahoo finance

```
In [13]: ### array(['^IXIC', '^GSPC', 'ZC=F', 'HG=F', 'ALI=F', 'NG=F', 'PL=F', 'SI=F', 'GC=F', 'BZ=F', 'MTF=F'], dtype=object)
tickers = ['^IXIC', '^GSPC', 'ZC=F', 'HG=F', 'ALI=F', 'NG=F', 'PL=F', 'SI=F', 'GC=F', 'BZ=F', 'MTF=F']
df0 = pd.DataFrame()

for tic in tickers:
    df_temp = yf.download(tic, start=start)
    df_temp['Ticker'] = tic
    df0 = pd.concat([df_temp, df0])
```

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

In [14]: df0

Out[14]:

	Open	High	Low	Close	Adj Close	Volume	Ticker
Date							
2010-12-17	115.50000	115.50000	115.50000	122.50000	122.50000	2	MTF=F
2010-12-20	122.50000	122.50000	122.50000	122.50000	122.50000	2	MTF=F
2010-12-21	123.25000	123.25000	123.25000	123.25000	123.25000	2	MTF=F
2010-12-22	125.50000	125.50000	125.50000	125.50000	125.50000	2	MTF=F
2010-12-23	125.25000	125.25000	125.25000	125.25000	125.25000	2	MTF=F
...
2024-05-13	16400.31055	16407.05078	16334.86035	16388.24023	16388.24023	4452750000	^IXIC
2024-05-14	16391.16016	16526.26953	16386.42969	16511.17969	16511.17969	7270240000	^IXIC
2024-05-15	16601.14062	16749.74023	16544.08984	16742.39062	16742.39062	8538990000	^IXIC
2024-05-16	16738.10938	16797.83008	16693.44922	16698.32031	16698.32031	11932600000	^IXIC
2024-05-17	16708.49023	16726.41016	16613.83984	16685.97070	16685.97070	9587280000	^IXIC

42488 rows × 7 columns

```
In [15]: df0['Ticker'].unique()
```

```
Out[15]: array(['MTF=F', 'BZ=F', 'GC=F', 'SI=F', 'PL=F', 'NG=F', 'ALI=F', 'HG=F',
   'ZC=F', '^GSPC', '^IXIC'], dtype=object)
```

```
In [ ]:
```

```
In [16]: # переименование тикеров в норм значения по заданному словарю
```

```
dic = {'MTF=F': 'coal', 'BZ=F': 'brent', 'GC=F': 'gold', 'SI=F': 'silver', 'PL=F': 'platinum', 'NG=F': 'gaz',
       'ALI=F': 'aluminum', 'HG=F': 'copper', 'ZC=F': 'corn', '^GSPC': 'SP', '^IXIC': 'NASDAQ'}
```

```
df0['Ticker'] = df0['Ticker'].replace(dic)
```

```
df0
```

```
Out[16]:
```

	Open	High	Low	Close	Adj Close	Volume	Ticker
Date							
2010-12-17	115.50000	115.50000	115.50000	122.50000	122.50000	2	coal
2010-12-20	122.50000	122.50000	122.50000	122.50000	122.50000	2	coal
2010-12-21	123.25000	123.25000	123.25000	123.25000	123.25000	2	coal
2010-12-22	125.50000	125.50000	125.50000	125.50000	125.50000	2	coal
2010-12-23	125.25000	125.25000	125.25000	125.25000	125.25000	2	coal
...
2024-05-13	16400.31055	16407.05078	16334.86035	16388.24023	16388.24023	4452750000	NASDAQ
2024-05-14	16391.16016	16526.26953	16386.42969	16511.17969	16511.17969	7270240000	NASDAQ
2024-05-15	16601.14062	16749.74023	16544.08984	16742.39062	16742.39062	8538990000	NASDAQ
2024-05-16	16738.10938	16797.83008	16693.44922	16698.32031	16698.32031	11932600000	NASDAQ
2024-05-17	16708.49023	16726.41016	16613.83984	16685.97070	16685.97070	9587280000	NASDAQ

42488 rows × 7 columns

```
In [17]: # беру только столбец с ценой и тикер
```

```
df01 = df0[['Close', 'Ticker']]
```

```
df01
```

Out[17]:

	Close	Ticker
Date		
2010-12-17	122.50000	coal
2010-12-20	122.50000	coal
2010-12-21	123.25000	coal
2010-12-22	125.50000	coal
2010-12-23	125.25000	coal
...
2024-05-13	16388.24023	NASDAQ
2024-05-14	16511.17969	NASDAQ
2024-05-15	16742.39062	NASDAQ
2024-05-16	16698.32031	NASDAQ
2024-05-17	16685.97070	NASDAQ

42488 rows × 2 columns

In [18]:

```
# сводная табл
df02 = pd.pivot_table(df01, index=df01.index, values=['Close'] , columns=['Ticker'])
df02
```

Out[18]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver	Close
Date												
2008-01-02	2609.62988	1447.16003		NaN	97.84	NaN	462.50	3.0505	7.850	857.00000	1547.00000	15.167
2008-01-03	2602.67993	1447.16003		NaN	97.60	NaN	466.00	3.1730	7.674	866.40002	1541.80005	15.382
2008-01-04	2504.64990	1411.63000		NaN	96.79	NaN	466.75	3.1415	7.841	863.09998	1539.09998	15.346
2008-01-07	2499.45996	1416.18005		NaN	94.39	NaN	466.25	3.1250	7.879	859.59998	1524.19995	15.180
2008-01-08	2440.51001	1390.18994		NaN	95.54	NaN	478.75	3.2735	7.967	878.00000	1553.59998	15.707
...
2024-05-13	16388.24023	5221.41992	2493.75	83.36	105.75	458.50	4.8045	2.381	2336.10010	1005.29999	28.221	
2024-05-14	16511.17969	5246.68018	2491.50	82.38	105.25	453.75	4.9535	2.344	2353.39990	1039.30005	28.485	
2024-05-15	16742.39062	5308.14990	2533.50	82.75	105.50	462.50	4.9695	2.416	2388.69995	1063.30005	29.514	
2024-05-16	16698.32031	5297.10010	2529.00	83.27	106.40	457.00	4.8920	2.495	2380.00000	1065.40002	29.665	
2024-05-17	16685.97070	5303.27002	2602.50	83.96	111.00	452.75	5.0825	2.638	2419.80005	1094.69995	31.775	

4124 rows × 11 columns

In [19]: # убираю мильтиндекс из сводн табл

df02 = df02['Close']

df02

Out[19]:	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
	Date											
	2008-01-02	2609.62988	1447.16003	NaN	97.84	NaN	462.50	3.0505	7.850	857.00000	1547.00000	15.167
	2008-01-03	2602.67993	1447.16003	NaN	97.60	NaN	466.00	3.1730	7.674	866.40002	1541.80005	15.382
	2008-01-04	2504.64990	1411.63000	NaN	96.79	NaN	466.75	3.1415	7.841	863.09998	1539.09998	15.346
	2008-01-07	2499.45996	1416.18005	NaN	94.39	NaN	466.25	3.1250	7.879	859.59998	1524.19995	15.180
	2008-01-08	2440.51001	1390.18994	NaN	95.54	NaN	478.75	3.2735	7.967	878.00000	1553.59998	15.707

	2024-05-13	16388.24023	5221.41992	2493.75	83.36	105.75	458.50	4.8045	2.381	2336.10010	1005.29999	28.221
	2024-05-14	16511.17969	5246.68018	2491.50	82.38	105.25	453.75	4.9535	2.344	2353.39990	1039.30005	28.485
	2024-05-15	16742.39062	5308.14990	2533.50	82.75	105.50	462.50	4.9695	2.416	2388.69995	1063.30005	29.514
	2024-05-16	16698.32031	5297.10010	2529.00	83.27	106.40	457.00	4.8920	2.495	2380.00000	1065.40002	29.665
	2024-05-17	16685.97070	5303.27002	2602.50	83.96	111.00	452.75	5.0825	2.638	2419.80005	1094.69995	31.775

4124 rows × 11 columns

In [20]: `df02.info()`

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4124 entries, 2008-01-02 to 2024-05-17
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   NASDAQ    4123 non-null   float64
 1   SP         4123 non-null   float64
 2   aluminum   2489 non-null   float64
 3   brent      4064 non-null   float64
 4   coal       3265 non-null   float64
 5   corn        4119 non-null   float64
 6   copper     4121 non-null   float64
 7   gaz         4122 non-null   float64
 8   gold        4121 non-null   float64
 9   platinum   3821 non-null   float64
 10  silver     4120 non-null   float64
dtypes: float64(11)
memory usage: 386.6 KB
```

```
In [21]: # создаю переменную с названием столбцов
col = df02.columns

col
```

```
Out[21]: Index(['NASDAQ', 'SP', 'aluminum', 'brent', 'coal', 'corn', 'copper', 'gaz',
       'gold', 'platinum', 'silver'],
              dtype='object', name='Ticker')
```

```
In [22]: df02.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4124 entries, 2008-01-02 to 2024-05-17
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   NASDAQ      4123 non-null    float64
 1   SP          4123 non-null    float64
 2   aluminum    2489 non-null    float64
 3   brent       4064 non-null    float64
 4   coal        3265 non-null    float64
 5   corn         4119 non-null    float64
 6   copper      4121 non-null    float64
 7   gaz          4122 non-null    float64
 8   gold         4121 non-null    float64
 9   platinum    3821 non-null    float64
 10  silver      4120 non-null    float64
dtypes: float64(11)
memory usage: 386.6 KB
```

In []:

Уголь

Уголь \$ за тонну

MTFF

Coal (API2) CIF ARA (ARGUS-McCl (MTF=F)

С 1 ноября 2022 использую: Coal (API2) CIF ARA (ARGUS-McCl (MTFZ22.NYM)

2023-06-09: MTF=F

<https://finance.yahoo.com/quote/MTFZ22.NYM?p=MTFZ22.NYM>

<https://finance.yahoo.com/quote/MTF=F?p=MTF=F>

[Ссылка на оглавление](#)

[К оглавлению](#)

In [23]: # что-то серия выходит, а нужен df с названиями столбцов

```
# уголь

# coal = df03[['Date', 'coal']]

# coal.rename(columns={'': 'coal'}, inplace=True)

# coal.rename(columns={'':'coal'}, inplace=True)

# coal = df02[['gaz', 'coal']]

coal = df02[['coal']].copy()

coal
```

Out[23]: Ticker coal

Date	
2008-01-02	NaN
2008-01-03	NaN
2008-01-04	NaN
2008-01-07	NaN
2008-01-08	NaN
...	...
2024-05-13	105.75
2024-05-14	105.25
2024-05-15	105.50
2024-05-16	106.40
2024-05-17	111.00

4124 rows × 1 columns

In [24]: # coal.info()
coal.index

Out[24]: DatetimeIndex(['2008-01-02', '2008-01-03', '2008-01-04', '2008-01-07',
'2008-01-08', '2008-01-09', '2008-01-10', '2008-01-11',
'2008-01-14', '2008-01-15',
'...',
'2024-05-06', '2024-05-07', '2024-05-08', '2024-05-09',
'2024-05-10', '2024-05-13', '2024-05-14', '2024-05-15',
'2024-05-16', '2024-05-17'],
dtype='datetime64[ns]', name='Date', length=4124, freq=None)

```
In [25]: coal.dropna(how='any', inplace=True)  
coal
```

```
Out[25]:    Ticker    coal
```

Date	coal
2010-12-17	122.50
2010-12-20	122.50
2010-12-21	123.25
2010-12-22	125.50
2010-12-23	125.25
...	...
2024-05-13	105.75
2024-05-14	105.25
2024-05-15	105.50
2024-05-16	106.40
2024-05-17	111.00

3265 rows × 1 columns

```
In [26]: # график за ближайшие дни  
fig = plt.figure(figsize=(12, 4))
```

```
plt.grid(2)  
plt.title('Стоимость тонны угля в $ за предыдущие дни')  
plt.plot(coal['coal']['2023-05-01':])  
# plt.plot(coal)
```

```
plt.xlabel("Период", fontsize = 10)  
plt.ylabel("$", fontsize = 10)
```

```
plt.show()
```



```
In [ ]:
```

```
In [27]: # скользящие средние разных периодов
```

```
coal_rol180 = coal.rolling(180).mean()
coal_rol90 = coal.rolling(90).mean()
coal_rol30 = coal.rolling(30).mean()

# график скользящих средних

fig, ax = plt.subplots(figsize = (12,4), dpi = 100)

# plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(coal_rol180, label='Скользящая средняя 180 дней')
plt.plot(coal_rol90, label='Скользящая средняя 90 дней')
plt.plot(coal_rol30, label='Скользящая средняя 30 дней')
plt.title('Coal. Уголь в $ за тонну. Скользящие средние')

plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)

plt.legend()
plt.show()
```



На вышепредставленном графике на основании рекомендаций из технического анализа следует обратить внимание на пересечение короткой и длинной скользящих средних. Особенность пересечения короткой скользящей средней с верху или снизу показывает изменение тренда.

In [28]: # Уголь в \$. Полосы Боллинджера с заданной даты

```
n = 30
date = '2021-01-01'

coal_d = coal[date:]
coal_ma = coal_d.rolling(window=n).mean()
coal_sd = coal_d.rolling(window=n).std()

coal_line1 = coal_ma + (2 * coal_sd)
coal_line2 = coal_ma - (2 * coal_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(coal_d, label='Биржевая котировка')
plt.plot(coal_line1, label='Верхняя полоса')
plt.plot(coal_line2, label='Нижняя полоса')
plt.title(f'Coal. Уголь в $ за тонну. Полосы Боллинджера с {date} по {n}-дневному периоду')

plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)

plt.legend()
plt.show()

# На графике полос Боллинджера смотрим на ... дописать
```



```
In [29]: ##### разложение временного ряда на компоненты. Тест. 2023-06-23
# Надо решить: You must specify a period or x must be a pandas object with a PeriodIndex or a DatetimeIndex with a
# после as.freq('D) ошибка: ValueError: This function does not handle missing values
```

```
In [30]: # привожу к месячным значениям
coal2 = coal.resample("M").mean().squeeze().dropna()
```

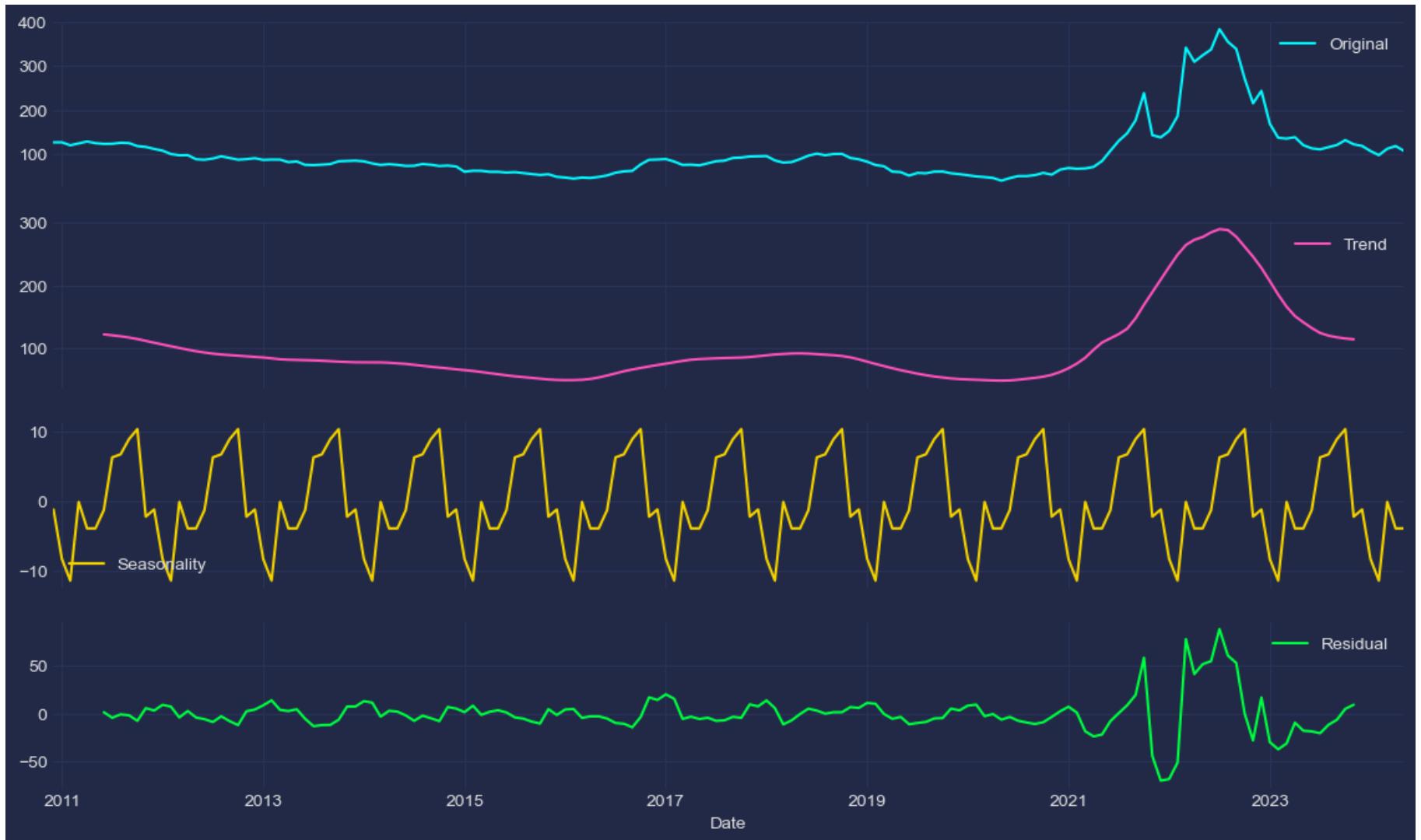
```
coal2
```

```
Out[30]: Date
2010-12-31    126.40000
2011-01-31    126.44250
2011-02-28    119.64211
2011-03-31    123.92391
2011-04-30    128.21000
...
2024-01-31    106.62000
2024-02-29    97.22600
2024-03-31    111.73200
2024-04-30    117.84682
2024-05-31    106.95000
Freq: M, Name: coal, Length: 162, dtype: float64
```

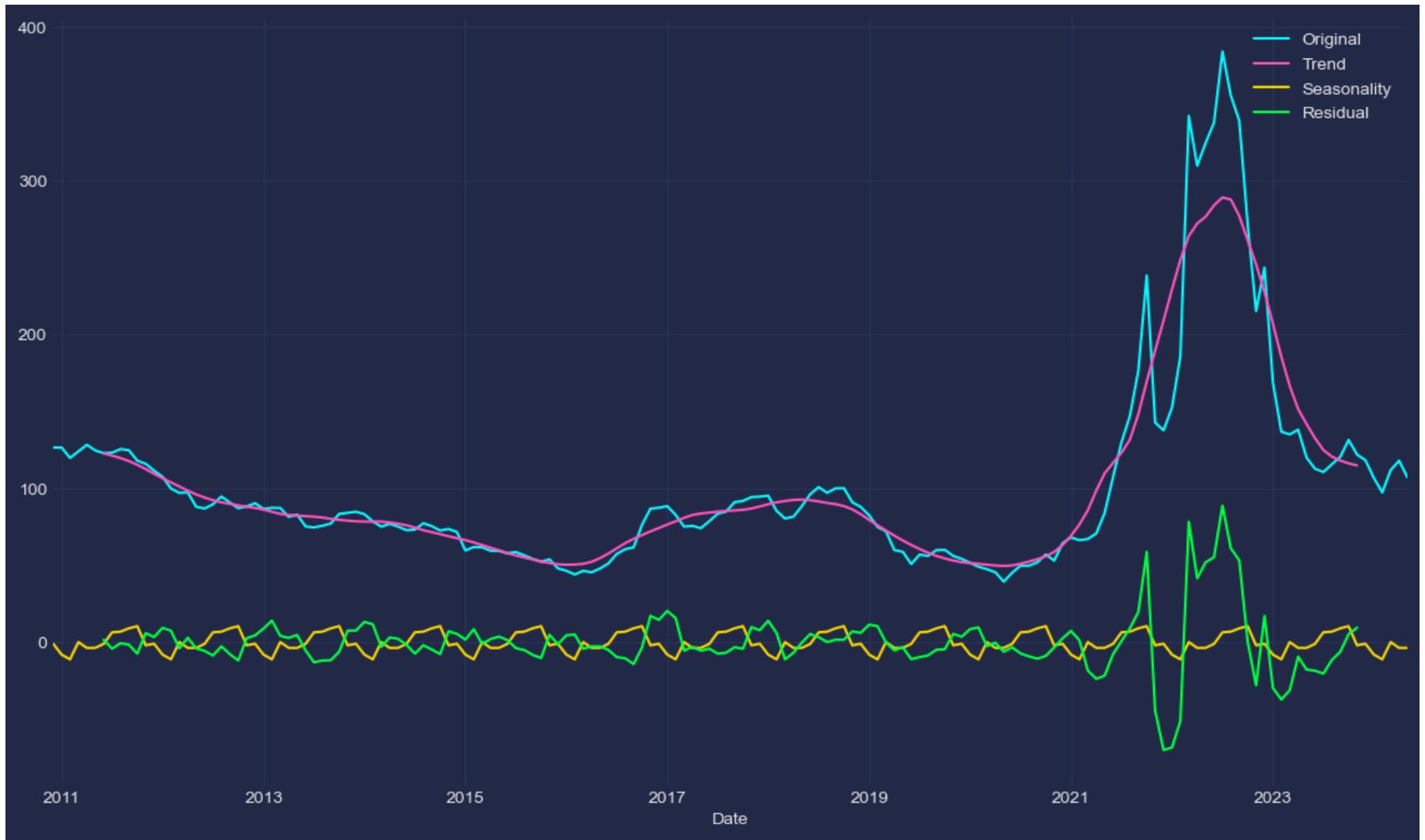
```
In [31]: components = tsa.seasonal_decompose(coal2, model='additive')

ts = (coal2.to_frame('Original')
      .assign(Trend=components.trend)
      .assign(Seasonality=components.seasonal)
      .assign(Residual=components.resid))

ts.plot(subplots=True, figsize=(14, 8));
```



```
In [32]: # 2024-05-19 на одном графике  
ts.plot(subplots=False, figsize=(14, 8));
```



In [33]: ts

Out[33]:

	Original	Trend	Seasonality	Residual
Date				
2010-12-31	126.40000	NaN	-1.17733	NaN
2011-01-31	126.44250	NaN	-8.22219	NaN
2011-02-28	119.64211	NaN	-11.31256	NaN
2011-03-31	123.92391	NaN	-0.09261	NaN
2011-04-30	128.21000	NaN	-3.87917	NaN
...
2024-01-31	106.62000	NaN	-8.22219	NaN
2024-02-29	97.22600	NaN	-11.31256	NaN
2024-03-31	111.73200	NaN	-0.09261	NaN
2024-04-30	117.84682	NaN	-3.87917	NaN
2024-05-31	106.95000	NaN	-3.87196	NaN

162 rows × 4 columns

In [34]:

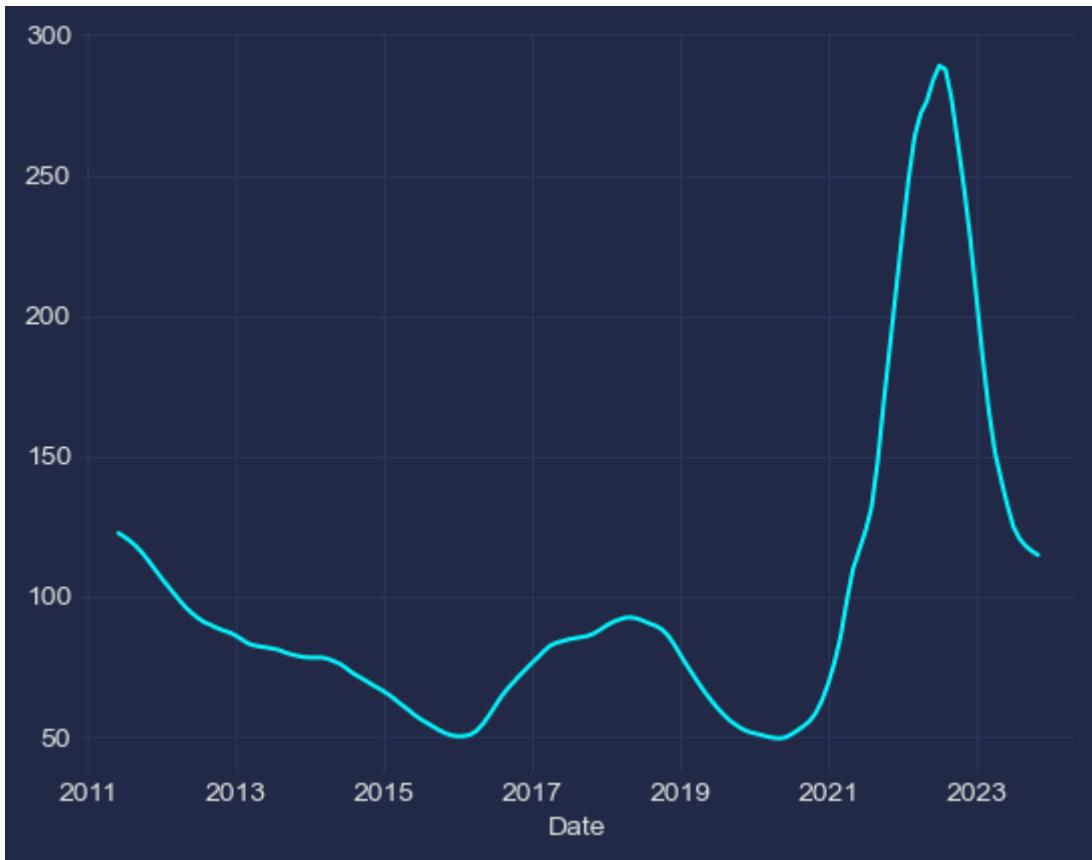
```
coal_trend = ts['Trend']
coal_seas = ts['Seasonality']

# coal_trend
coal_seas
```

```
Out[34]: Date
2010-12-31    -1.17733
2011-01-31    -8.22219
2011-02-28    -11.31256
2011-03-31    -0.09261
2011-04-30    -3.87917
...
2024-01-31    -8.22219
2024-02-29    -11.31256
2024-03-31    -0.09261
2024-04-30    -3.87917
2024-05-31    -3.87196
Freq: M, Name: Seasonality, Length: 162, dtype: float64
```

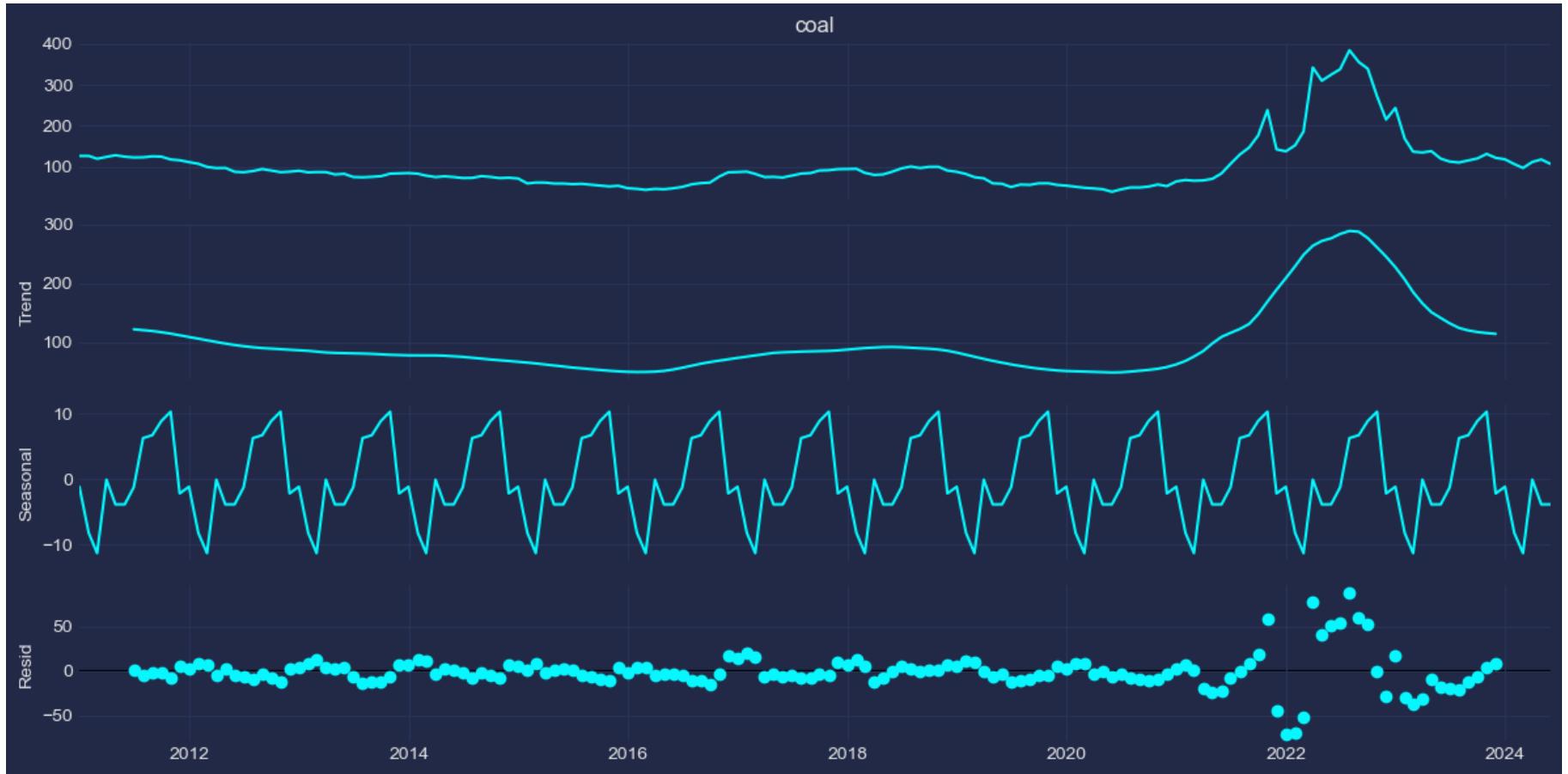
```
In [35]: ts['Trend'].plot()
```

```
Out[35]: <AxesSubplot: xlabel='Date'>
```



In [36]: # тоже декомпозиция

```
from pylab import rcParams
rcParams['figure.figsize'] = 12, 6
decomposition = sm.tsa.seasonal_decompose(coal2, model='additive')
fig = decomposition.plot()
plt.show()
```



In []:

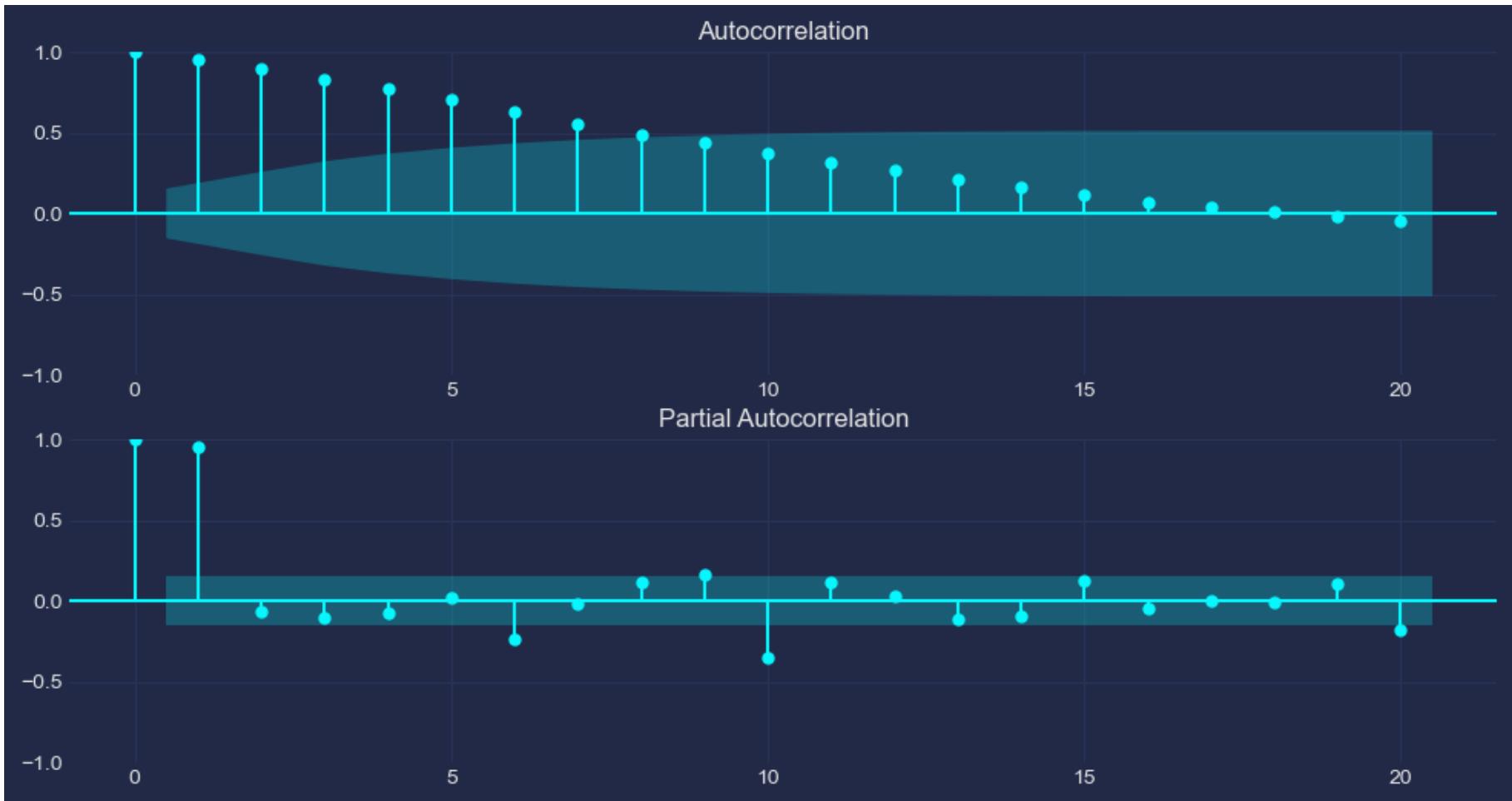
In []:

In [37]: # Графики автокорреляции и частичной автокорреляции месячных значений стоимости угля. Метод 'ywmlle'

```
plt.figure(figsize=(12, 6))
plt.subplot(211)
plot_acf(coal2, lags=20, ax=plt.gca())

plt.subplot(212)
plot_pacf(coal2, method='ywmlle', lags=20, ax=plt.gca())

plt.show()
```

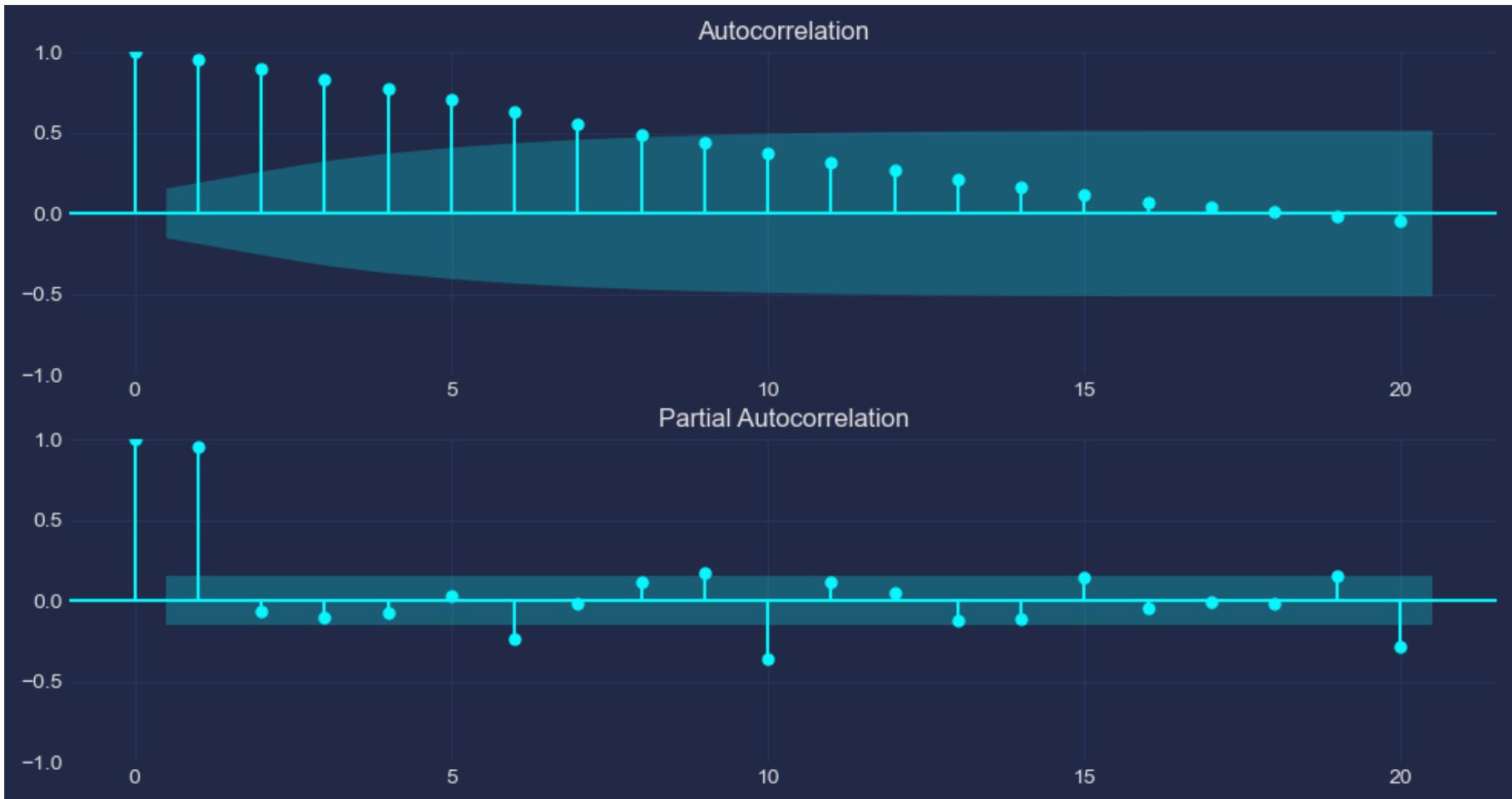


In [38]: # Графики автокорреляции и частичной автокорреляции месячных значений стоимости угля. Метод 'ols'

```
plt.figure(figsize=(12, 6))
plt.subplot(211)
plot_acf(coal2, lags=20, ax=plt.gca())

plt.subplot(212)
plot_pacf(coal2, method='ols', lags=20, ax=plt.gca())

plt.show()
```



```
In [39]: # модель ARIMA (1, 0, 0)
model = sm.tsa.ARIMA(coal2, order=(1, 0, 0))
model_fit = model.fit()
model_fit.summary()

# критерий качества модели (самый популярный) AIC - информ критерий Акаике. Чем меньше тем лучше.
# Также если разница мменьше 10 то нет различий в моделях
# BIC - Байесовский информационный критерий
```

Out[39]:

SARIMAX Results

Dep. Variable: coal **No. Observations:** 162
Model: ARIMA(1, 0, 0) **Log Likelihood:** -714.198
Date: Sun, 19 May 2024 **AIC:** 1434.397
Time: 09:38:00 **BIC:** 1443.660
Sample: 12-31-2010 **HQIC:** 1438.158
- 05-31-2024

Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
const	104.0135	56.700	1.834	0.067	-7.117	215.144
ar.L1	0.9455	0.023	41.586	0.000	0.901	0.990
sigma2	389.7173	13.123	29.696	0.000	363.996	415.439

Ljung-Box (L1) (Q): 0.75 **Jarque-Bera (JB):** 6133.62

Prob(Q): 0.39 **Prob(JB):** 0.00

Heteroskedasticity (H): 61.39 **Skew:** 2.86

Prob(H) (two-sided): 0.00 **Kurtosis:** 32.60

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

```
In [40]: # остатки модели
```

```
residuals = pd.DataFrame(model_fit.resid)

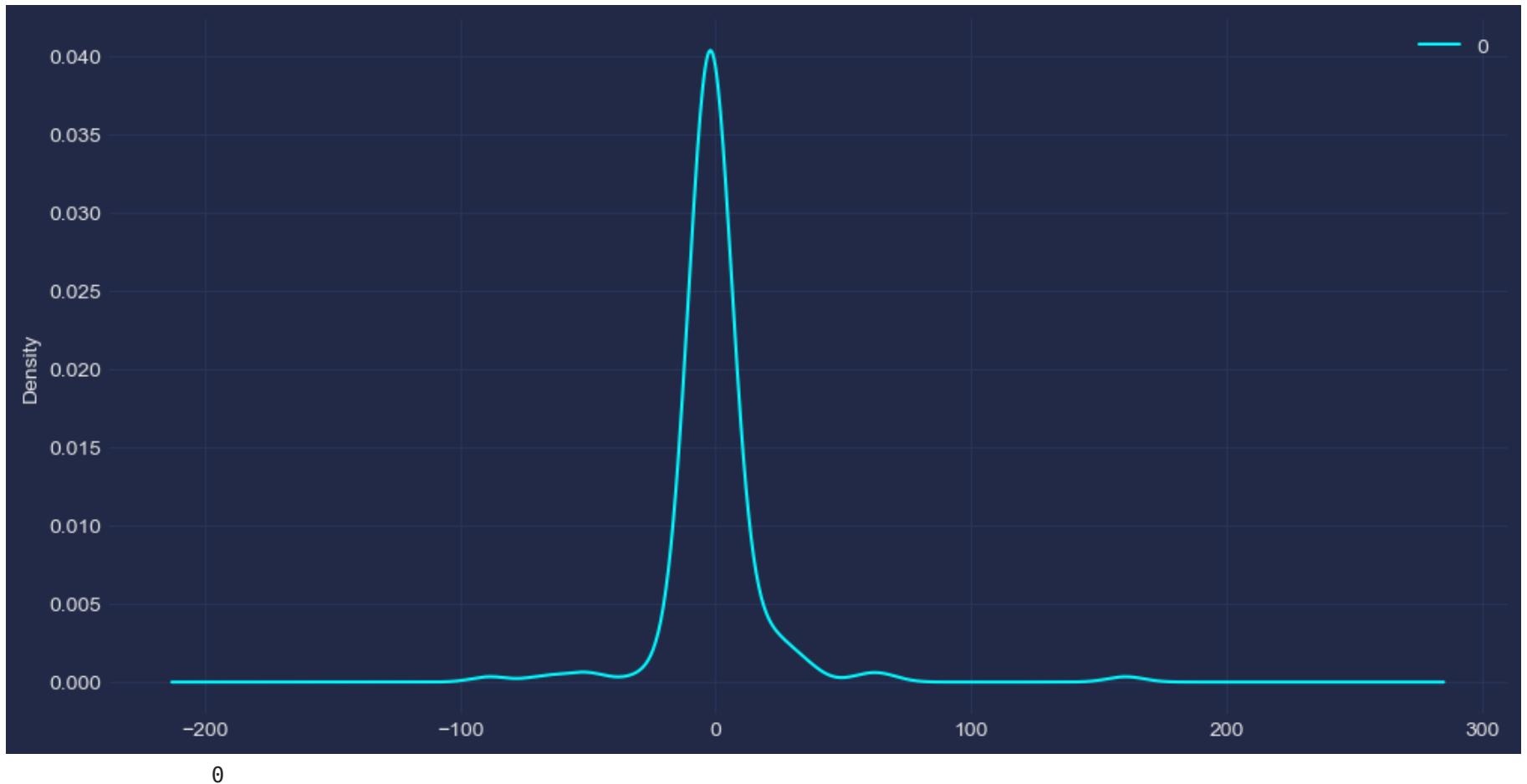
residuals.plot()
plt.show()
print(residuals.describe())
```



```
          0
count    162.00000
mean     -0.13375
std      19.87327
min     -88.32836
25%     -4.63367
50%     -1.22995
75%      1.80942
max     160.64814
```

```
In [41]: # ядерная оценка плотности остатков модели

residuals.plot(kind='kde')
plt.show()
print(residuals.describe())
```

 θ

	θ
count	162.00000
mean	-0.13375
std	19.87327
min	-88.32836
25%	-4.63367
50%	-1.22995
75%	1.80942
max	160.64814

```
In [42]: # строю прогноз на последние наблюдения  
  
# coal2_fit = model_fit.predict(0, 20)  
  
coal2_fit = model_fit.predict(-10, -1)  
  
coal2_fit
```

```
Out[42]: Date  
2023-08-31    110.08751  
2023-09-30    114.83760  
2023-10-31    119.42544  
2023-11-30    129.86409  
2023-12-31    120.88949  
2024-01-31    117.41054  
2024-02-29    106.47799  
2024-03-31    97.59579  
2024-04-30    111.31148  
2024-05-31    117.09315  
Freq: M, Name: predicted_mean, dtype: float64
```

```
In [43]: coal2.tail(10).plot()  
coal2_fit.tail(10).plot()
```

```
Out[43]: <AxesSubplot: xlabel='Date'>
```



In []:

In []:

In []:

Нефть

Нефть \$ за баррель

BZ=F

<https://finance.yahoo.com/quote/BZ=F?p=BZ=F>

[К оглавлению](#)

```
In [44]: #  
brent = df02[['brent']].copy()  
  
brent.dropna(how='any', inplace=True)  
  
brent
```

Out[44]: Ticker brent

Date
2008-01-02 97.84
2008-01-03 97.60
2008-01-04 96.79
2008-01-07 94.39
2008-01-08 95.54
...
2024-05-13 83.36
2024-05-14 82.38
2024-05-15 82.75
2024-05-16 83.27
2024-05-17 83.96

4064 rows × 1 columns

In []:

In [45]: # нефть

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(brent, label='Биржевая котировка')
plt.plot(brent.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(brent.ewm(span=90).mean(), label='Экспоненциальная скользящая средняя')
plt.plot(brent.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.title('Brent. Нефть в $ за баррель')

plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)

plt.legend()
plt.show()
```



```
In [46]: # скользящие средние разных периодов
```

```
brent_rol180 = brent.rolling(180).mean()
brent_rol90 = brent.rolling(90).mean()
brent_rol30 = brent.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(brent_rol180, label='Скользящая средняя 180 дней')
plt.plot(brent_rol90, label='Скользящая средняя 90 дней')
plt.plot(brent_rol30, label='Скользящая средняя 30 дней')
plt.title('brent. Нефть в $ за баррель. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In [47]: # полосы Боллинджера с заданной даты

```
n = 30
date = '2021-01-01'

brent_d = brent[date:]
brent_ma = brent_d.rolling(window=n).mean()
brent_sd = brent_d.rolling(window=n).std()

brent_line1 = brent_ma + (2 * brent_sd)
brent_line2 = brent_ma - (2 * brent_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(brent_d, label='Биржевая котировка')
plt.plot(brent_line1, label='Верхняя граница')
plt.plot(brent_line2, label='Нижняя граница')
plt.title(f'Brent. Нефть в $ за баррель. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



Золото

Золото \$ за тройскую унцию

GC=F

<https://finance.yahoo.com/quote/GC=F?p=GC=F>

[К оглавлению](#)

```
In [48]: gold = df02[['gold']].copy()  
  
gold.dropna(how='any', inplace=True)  
  
gold
```

Out[48]: **Ticker** gold

	Date
2008-01-02	857.00000
2008-01-03	866.40002
2008-01-04	863.09998
2008-01-07	859.59998
2008-01-08	878.00000
...	...
2024-05-13	2336.10010
2024-05-14	2353.39990
2024-05-15	2388.69995
2024-05-16	2380.00000
2024-05-17	2419.80005

4121 rows × 1 columns

In []:

In [49]: # золото

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(gold, label='Биржевая котировка')
plt.plot(gold.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(gold.ewm(span=90).mean(), label='Экспоненциальная скользяща средняя')
plt.plot(gold.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.title('Gold. Золото в $ за тройскую унцию')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In []:

In [50]: # скользящие средние разных периодов

```
gold_rol180 = gold.rolling(180).mean()
gold_rol90 = gold.rolling(90).mean()
gold_rol30 = gold.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(gold_rol180, label='Скользящая средняя 180 дней')
plt.plot(gold_rol90, label='Скользящая средняя 90 дней')
plt.plot(gold_rol30, label='Скользящая средняя 30 дней')
plt.title('Gold. Золото в $ за тройскую унцию. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [51]: # полосы Боллинджера с заданной даты
```

```
n = 30
date = '2021-01-01'

gold_d = gold[date:]
gold_ma = gold_d.rolling(window=n).mean()
gold_sd = gold_d.rolling(window=n).std()

gold_line1 = gold_ma + (2 * gold_sd)
gold_line2 = gold_ma - (2 * gold_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(gold_d, label='Биржевая котировка')
plt.plot(gold_line1, label='Верхняя граница')
plt.plot(gold_line2, label='Нижняя граница')
plt.title(f'gold. Золото в $ за тройскую унцию. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In []:

In [52]: # макс значение за заданный период

gold['2020-01-01': '2020-05-31'].max()

Out[52]: Ticker

gold 1756.69995
dtype: float64

Серебро

Серебро \$ за тройскую унцию

SI=F

<https://finance.yahoo.com/quote/SI=F?p=SI=F>

[К оглавлению](#)

```
In [53]: silver = df02[['silver']].copy()  
  
silver.dropna(how='any', inplace=True)  
  
silver
```

Out[53]: Ticker silver

Date	
2008-01-02	15.167
2008-01-03	15.382
2008-01-04	15.346
2008-01-07	15.180
2008-01-08	15.707
...	...
2024-05-13	28.221
2024-05-14	28.485
2024-05-15	29.514
2024-05-16	29.665
2024-05-17	31.775

4120 rows × 1 columns

In [54]: # серебро

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(silver, label='Биржевая котировка')
plt.plot(silver.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(silver.ewm(span=90).mean(), label='Экспоненциальная скользящая средняя')
plt.plot(silver.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.title('Silver. Серебро в $ за тройскую унцию')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [55]: # скользящие средние разных периодов

silver_rol180 = silver.rolling(180).mean()
silver_rol90 = silver.rolling(90).mean()
silver_rol30 = silver.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(silver_rol180, label='Скользящая средняя 180 дней')
plt.plot(silver_rol90, label='Скользящая средняя 90 дней')
plt.plot(silver_rol30, label='Скользящая средняя 30 дней')
plt.title('Silver. Серебро в $ за тройскую унцию. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [56]: # полосы Боллинджера с заданной даты
```

```
n = 30
date = '2021-01-01'

silver_d = silver[date:]
silver_ma = silver_d.rolling(window=n).mean()
silver_sd = silver_d.rolling(window=n).std()

silver_line1 = silver_ma + (2 * silver_sd)
silver_line2 = silver_ma - (2 * silver_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(silver_d, label='Биржевая котировка')
plt.plot(silver_line1, label='Верхняя граница')
plt.plot(silver_line2, label='Нижняя граница')
plt.title(f'Silver. Серебро в $ за тройскую унцию. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



Платина

Платина \$ за тройскую унцию

PL=F

<https://finance.yahoo.com/quote/PL=F?p=PL=F>

[К оглавлению](#)

```
In [57]: platinum = df02[['platinum']].copy()  
platinum.dropna(how='any', inplace=True)  
platinum
```

Out[57]: Ticker platinum

Date
2008-01-02 1547.00000
2008-01-03 1541.80005
2008-01-04 1539.09998
2008-01-07 1524.19995
2008-01-08 1553.59998
...
2024-05-13 1005.29999
2024-05-14 1039.30005
2024-05-15 1063.30005
2024-05-16 1065.40002
2024-05-17 1094.69995

3821 rows × 1 columns

In []:

In [58]: # Platinum

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(platinum, label='Биржевая котировка')
plt.plot(platinum.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(platinum.ewm(span=90).mean(), label='Экспоненциальная скользяща средняя')
plt.plot(platinum.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.title('Platinum. Платина в $ за тройскую унцию')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In [59]: # скользящие средние разных периодов

```
platinum_rol180 = platinum.rolling(180).mean()
platinum_rol90 = platinum.rolling(90).mean()
platinum_rol30 = platinum.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(platinum_rol180, label='Скользящая средняя 180 дней')
plt.plot(platinum_rol90, label='Скользящая средняя 90 дней')
plt.plot(platinum_rol30, label='Скользящая средняя 30 дней')
plt.title('Platinum. Платина в $ за тройскую унцию. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [60]: # полосы Боллинджера с заданной даты
```

```
n = 30
date = '2021-01-01'

platinum_d = platinum[date:]
platinum_ma = platinum_d.rolling(window=n).mean()
platinum_sd = platinum_d.rolling(window=n).std()

platinum_line1 = platinum_ma + (2 * platinum_sd)
platinum_line2 = platinum_ma - (2 * platinum_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(platinum_d, label='Биржевая котировка')
plt.plot(platinum_line1, label='Верхняя граница')
plt.plot(platinum_line2, label='Нижняя граница')
plt.title(f'Platinum. Платина в $ за тройскую унцию. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



Натуральный газ

Натуральный газ (в долларах (\$) на единицу ММБtu(1 млн БТЕ) (1 Btu ≈ 1054,615 Дж)

NG=F

<https://finance.yahoo.com/quote/NG=F?p=NG=F>

[К оглавлению](#)

```
In [61]: gaz = df02[['gaz']].copy()  
  
gaz.dropna(how='any', inplace=True)  
  
gaz
```

Out[61]: Ticker gaz

Date	
2008-01-02	7.850
2008-01-03	7.674
2008-01-04	7.841
2008-01-07	7.879
2008-01-08	7.967
...	...
2024-05-13	2.381
2024-05-14	2.344
2024-05-15	2.416
2024-05-16	2.495
2024-05-17	2.638

4122 rows × 1 columns

In [62]: # газ

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(gaz, label='Биржевая котировка')
plt.plot(gaz.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(gaz.ewm(span=90).mean(), label='Экспоненциальная скользящая средняя')
plt.plot(gaz.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.title('GAZ. Натуральный газ в $ за единицу ММБтн')
plt.legend()
plt.show()
```



```
In [63]: # график за ближайшие дни
fig = plt.figure(figsize=(12, 4))

plt.grid(2)
plt.title('Натуральный газ в $ за единицу MMBtu')
plt.plot(gaz['2023-02-01':])

plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)

plt.show()
```



```
In [64]: # скользящие средние разных периодов

gaz_rol180 = gaz.rolling(180).mean()
gaz_rol90 = gaz.rolling(90).mean()
gaz_rol30 = gaz.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(True)
plt.plot(gaz_rol180, label='Скользящая средняя 180 дней')
plt.plot(gaz_rol90, label='Скользящая средняя 90 дней')
plt.plot(gaz_rol30, label='Скользящая средняя 30 дней')
plt.title('GAZ. Натуральный газ в $ за единицу ММБт. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [65]: # полосы Боллинджера с заданной даты
```

```
n = 30
date = '2021-01-01'

gaz_d = gaz[date:]
gaz_ma = gaz_d.rolling(window=n).mean()
gaz_sd = gaz_d.rolling(window=n).std()

gaz_line1 = gaz_ma + (2 * gaz_sd)
gaz_line2 = gaz_ma - (2 * gaz_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(gaz_d, label='Биржевая котировка')
plt.plot(gaz_line1, label='Верхняя граница')
plt.plot(gaz_line2, label='Нижняя граница')
plt.title(f'GAZ. Натуральный газ в $ за единицу ММВт. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



Алюминий

Алюминий \$ за тонну

ALI=F

<https://finance.yahoo.com/quote/ALI=F?p=ALI=F>

[К оглавлению](#)

```
In [66]: aluminum = df02[['aluminum']].copy()  
aluminum.dropna(how='any', inplace=True)  
aluminum
```

Out[66]: Ticker aluminum

	Date
2014-05-06	2172.75
2014-05-07	2149.00
2014-05-08	2141.75
2014-05-09	2107.25
2014-05-12	2088.25
...	...
2024-05-13	2493.75
2024-05-14	2491.50
2024-05-15	2533.50
2024-05-16	2529.00
2024-05-17	2602.50

2489 rows × 1 columns

In [67]: # Aluminum

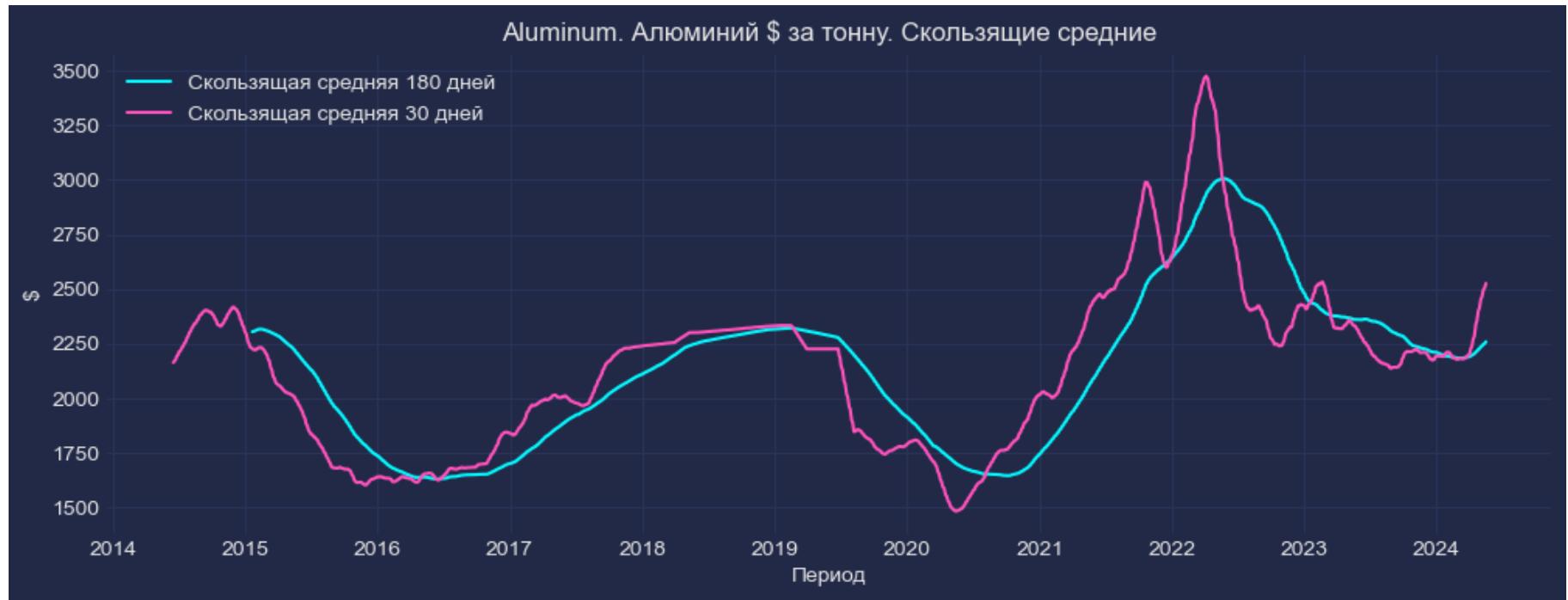
```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(aluminum, label='Биржевая котировка')
plt.plot(aluminum.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(aluminum.ewm(span=90).mean(), label='Экспоненциальная скользящая средняя')
plt.plot(aluminum.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.title('Aluminum. Алюминий $ за тонну')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In [68]: # скользящие средние разных периодов

```
aluminum_rol180 = aluminum.rolling(180).mean()
aluminum_rol90 = aluminum.rolling(90).mean()
aluminum_rol30 = aluminum.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(aluminum_rol180, label='Скользящая средняя 180 дней')
# plt.plot(aluminum_rol90, label='Скользящая средняя 90 дней')
plt.plot(aluminum_rol30, label='Скользящая средняя 30 дней')
plt.title('Aluminum. Алюминий $ за тонну. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In [69]: # полосы Боллинджера с заданной даты

```
n = 30
date = '2021-01-01'

aluminum_d = gaz[date:]
aluminum_ma = aluminum_d.rolling(window=n).mean()
aluminum_sd = aluminum_d.rolling(window=n).std()

aluminum_line1 = aluminum_ma + (2 * aluminum_sd)
aluminum_line2 = aluminum_ma - (2 * aluminum_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(aluminum_d, label='Биржевая котировка')
plt.plot(aluminum_line1, label='Верхняя граница')
plt.plot(aluminum_line2, label='Нижняя граница')
plt.title(f'Aluminum. Алюминий $ за тонну. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In []:

Медь

Медь \$ за фунт

HG=

<https://finance.yahoo.com/quote/HG%3DF?p=HG%3DF>

[К оглавлению](#)

```
In [70]: copper = df02[['cupper']].copy()
copper.dropna(how='any', inplace=True)
copper
```

Out[70]: Ticker copper

Date	copper
2008-01-02	3.0505
2008-01-03	3.1730
2008-01-04	3.1415
2008-01-07	3.1250
2008-01-08	3.2735
...	...
2024-05-13	4.8045
2024-05-14	4.9535
2024-05-15	4.9695
2024-05-16	4.8920
2024-05-17	5.0825

4121 rows × 1 columns

In [71]: # Copper

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(copper, label='Биржевая котировка')
plt.plot(copper.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(copper.ewm(span=90).mean(), label='Экспоненциальная скользяща средняя')
plt.plot(copper.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.title('Copper. Медь $ за фунт')
plt.legend()
plt.show()
```



In [72]: # скользящие средние разных периодов

```
cupper_rol180 = copper.rolling(180).mean()
cupper_rol90 = copper.rolling(90).mean()
cupper_rol30 = copper.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(cupper_rol180, label='Скользящая средняя 180 дней')
plt.plot(cupper_rol90, label='Скользящая средняя 90 дней')
plt.plot(cupper_rol30, label='Скользящая средняя 30 дней')
plt.title('Copper. Медь $ за фунт. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



In [73]: # полосы Боллинджера с заданной даты

```
n = 30
date = '2021-01-01'

cupper_d = gaz[date:]
cupper_ma = copper_d.rolling(window=n).mean()
cupper_sd = copper_d.rolling(window=n).std()

cupper_line1 = copper_ma + (2 * copper_sd)
cupper_line2 = copper_ma - (2 * copper_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(copper_d, label='Биржевая котировка')
plt.plot(cupper_line1, label='Верхняя граница')
plt.plot(cupper_line2, label='Нижняя граница')
plt.title(f'Copper. Медь $ за фунт. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



Кукуруза

Кукуруза \$ за 1 тонну

ZC=F

<https://finance.yahoo.com/quote/ZC=F?p=ZC=F>

[К оглавлению](#)

```
In [74]: corn = df02[['cupper']].copy()  
corn.dropna(how='any', inplace=True)  
corn
```

Out[74]: Ticker copper

Date	
2008-01-02	3.0505
2008-01-03	3.1730
2008-01-04	3.1415
2008-01-07	3.1250
2008-01-08	3.2735
...	...
2024-05-13	4.8045
2024-05-14	4.9535
2024-05-15	4.9695
2024-05-16	4.8920
2024-05-17	5.0825

4121 rows × 1 columns

In [75]: # Corn

```
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(corn, label='Биржевая котировка')
plt.plot(corn.rolling(90, min_periods=30).mean(), label='Скользящая средняя')
plt.plot(corn.ewm(span=90).mean(), label='Экспоненциальная скользящая средняя')
plt.plot(corn.expanding(30).mean(), label='Скользящая средняя по расширяющемуся окну')
plt.title('Corn. Кукуруза в $ за тонну')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [76]: # скользящие средние разных периодов

corn_rol180 = corn.rolling(180).mean()
corn_rol90 = corn.rolling(90).mean()
corn_rol30 = corn.rolling(30).mean()

# график скользящих средних
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(corn_rol180, label='Скользящая средняя 180 дней')
plt.plot(corn_rol90, label='Скользящая средняя 90 дней')
plt.plot(corn_rol30, label='Скользящая средняя 30 дней')
plt.title('Corn. Кукуруза в $ за тонну. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



```
In [77]: # полосы Боллинджера с заданной даты
```

```
n = 30
date = '2021-01-01'

corn_d = corn[date:]
corn_ma = corn_d.rolling(window=n).mean()
corn_sd = corn_d.rolling(window=n).std()

corn_line1 = corn_ma + (2 * corn_sd)
corn_line2 = corn_ma - (2 * corn_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(corn_d, label='Биржевая котировка')
plt.plot(corn_line1, label='Верхняя граница')
plt.plot(corn_line2, label='Нижняя граница')
plt.title(f'Corn. Кукуруза в $ за тонну. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("$", fontsize = 10)
plt.legend()
plt.show()
```



S&P500

[^]GSPC

<https://finance.yahoo.com/quote/%5EGSPC?p=%5EGSPC>

[К оглавлению](#)

```
In [78]: SP = df02[['SP']].copy()  
SP.dropna(how='any', inplace=True)  
SP
```

```
Out[78]:   Ticker      SP
              Date
2008-01-02  1447.16003
2008-01-03  1447.16003
2008-01-04  1411.63000
2008-01-07  1416.18005
2008-01-08  1390.18994
...
2024-05-13  5221.41992
2024-05-14  5246.68018
2024-05-15  5308.14990
2024-05-16  5297.10010
2024-05-17  5303.27002
4123 rows × 1 columns
```

```
In [79]: # S&P 500 (^GSPC)

SP.plot(figsize=(10,4), grid=True, title='Индекс S&P500')
```

```
Out[79]: <AxesSubplot: title={'center': 'Индекс S&P500'}, xlabel='Date'>
```



NASDAQ

[^]IXIC

<https://finance.yahoo.com/quote/%5EIXIC?p=%5EIXIC&.tsrc=fin-srch>

[К оглавлению](#)

In [80]: NASDAQ = df02['NASDAQ']

```
NASDAQ.dropna(how='any', inplace=True)
```

```
NASDAQ
```

```
/tmp/ipykernel_85268/3362688422.py:3: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
    NASDAQ.dropna(how='any', inplace=True)
```

```
Out[80]: Date
```

```
2008-01-02    2609.62988  
2008-01-03    2602.67993  
2008-01-04    2504.64990  
2008-01-07    2499.45996  
2008-01-08    2440.51001  
...  
2024-05-13    16388.24023  
2024-05-14    16511.17969  
2024-05-15    16742.39062  
2024-05-16    16698.32031  
2024-05-17    16685.97070  
Name: NASDAQ, Length: 4123, dtype: float64
```

```
In [81]: # NASDAQ Composite (^IXIC)
```

```
NASDAQ.plot(figsize=(10,4), grid=True, title='Индекс NASDAQ')
```

```
Out[81]: <AxesSubplot: title={'center': 'Индекс NASDAQ'}, xlabel='Date'>
```



03. Анализ общего dataframe

df

[К оглавлению](#)

```
In [82]: # делаю прореживание данных на среднемесячные значения  
# Считаю, что нужно работать не с дневными значениями, т.к. корреляция с предыдущим днем почти равна ед.  
  
df_month = df02.resample('M').mean()  
  
# и делаю для ежедневн значений  
df_day = df02.resample('D').mean()  
  
df_month
```

Out[82]:	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
	Date											
2008-01-31	2418.09380	1378.76381		NaN	92.11238	NaN	488.57143	3.20171	7.99148	891.35713	1587.18947	16.05210
2008-02-29	2325.82554	1354.87251		NaN	94.65250	NaN	516.07500	3.58955	8.64225	925.11000	2135.56668	17.65610
2008-03-31	2254.81804	1316.94299		NaN	102.79900	NaN	547.68750	3.81107	9.62430	962.93000	2030.57501	19.15790
2008-04-30	2368.09955	1370.46909		NaN	110.44182	NaN	593.36364	3.93939	10.28809	909.66364	2012.18502	17.47827
2008-05-31	2483.24097	1403.21762		NaN	124.38095	NaN	597.85714	3.77421	11.38162	888.20476	NaN	17.01200
...
2024-01-31	15081.38816	4804.49151	2190.63095	79.19714	106.62000	451.84524	3.81207	2.71500	2028.04285	920.87619	22.86643	
2024-02-29	15808.93501	5011.96143	2172.20000	81.62400	97.22600	423.37500	3.79663	1.79550	2025.21499	891.93000	22.66120	
2024-03-31	16216.29551	5170.57249	2216.12500	84.66550	111.73200	429.50000	3.97643	1.74730	2161.87000	911.55000	24.52955	
2024-04-30	15950.86355	5112.49274	2489.32955	89.00000	117.84682	434.62500	4.36091	1.79123	2332.59090	941.63636	27.49568	
2024-05-31	16330.81318	5198.45162	2521.21154	83.27154	106.95000	452.42308	4.72654	2.27115	2340.45386	1002.95385	28.12438	

197 rows × 11 columns

```
In [83]: # df_day.isna().sum()
```

```
In [84]: # сохранил в excel что бы было.
# df02.to_excel('ts_base_coal_2024-05-19.xlsx')
```

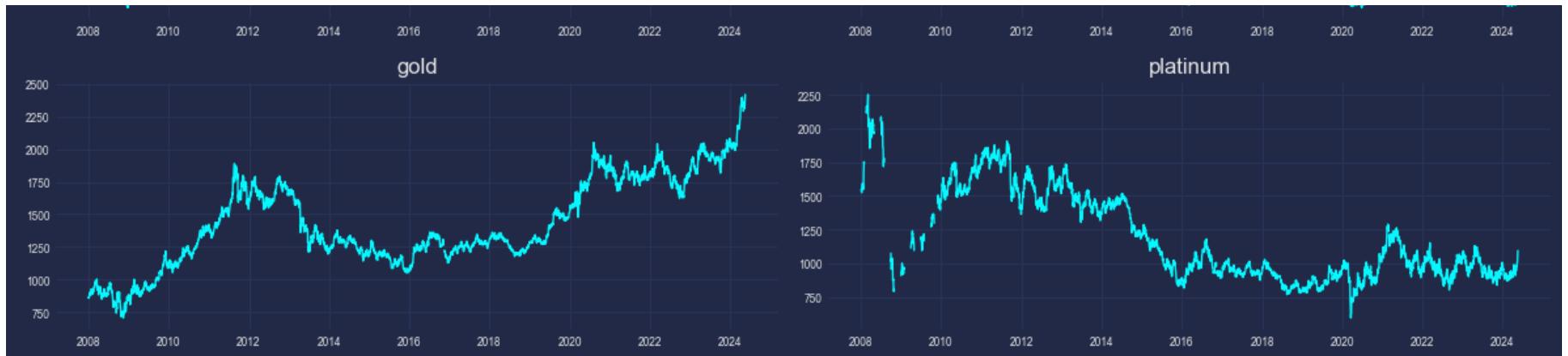
```
In [85]: # сводные статистики
df02.describe()
```

Out[85]: Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
count	4123.00000	4123.00000	2489.00000	4064.00000	3265.00000	4119.00000	4121.00000	4122.00000	4121.00000	3821.00000	4120.00000
mean	6544.75870	2415.90400	2148.50793	78.59565	102.32427	482.95418	3.21215	3.78138	1431.62604	1183.59422	20.82082
std	4253.65406	1172.82945	380.16036	25.04750	65.04839	139.55350	0.71354	1.86892	333.87183	324.13028	6.31274
min	1268.64001	676.53003	1452.00000	19.33000	38.58000	293.50000	1.24750	1.48200	704.90002	595.90002	8.79000
25%	2840.79004	1361.22498	1824.00000	58.41500	62.70000	366.75000	2.69300	2.66400	1211.69995	927.40002	16.39650
50%	5036.37012	2099.12012	2208.00000	76.46000	85.80000	423.00000	3.18500	3.23800	1330.09998	1031.19995	18.73150
75%	9255.01514	3141.30493	2328.25000	102.57500	116.40000	604.25000	3.74300	4.26550	1727.00000	1464.00000	24.18000
max	16742.39062	5308.14990	3873.00000	146.08000	438.35001	831.25000	5.08250	13.57700	2419.80005	2251.10010	48.58400

```
In [86]: # графики для общего представления временных рядов
fig, axes = plt.subplots(nrows=5, ncols=2, dpi=100, figsize=(12,12))
for i, ax in enumerate(axes.flatten()):
    data_temp = df02[df02.columns[i]]
    ax.plot(data_temp, linewidth=1) # , color='blue'
    # Decorations
    ax.set_title(df02.columns[i])
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)
    ax.grid(1)

plt.tight_layout();
```





03.1 Корреляция временных рядов

corr

[К оглавлению](#)

```
In [87]: # Корреляция
corr = df02.corr()
corr
```

Out[87]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Ticker											
NASDAQ	1.00000	0.99211	0.51677	-0.19071	0.38303	0.09022	0.41301	-0.20704	0.75163	-0.63158	0.09456
SP	0.99211	1.00000	0.55262	-0.17530	0.40576	0.08324	0.39685	-0.20074	0.73369	-0.65849	0.06280
aluminum	0.51677	0.55262	1.00000	0.75906	0.66418	0.65563	0.80629	0.58262	0.38433	0.19346	0.44851
brent	-0.19071	-0.17530	0.75906	1.00000	0.44516	0.72189	0.65336	0.49205	0.15442	0.68360	0.56907
coal	0.38303	0.40576	0.66418	0.44516	1.00000	0.57117	0.55852	0.80126	0.41844	-0.03262	0.21105
corn	0.09022	0.08324	0.65563	0.72189	0.57117	1.00000	0.72717	0.34955	0.45848	0.47009	0.72463
cupper	0.41301	0.39685	0.80629	0.65336	0.55852	0.72717	1.00000	0.24532	0.62295	0.39317	0.70729
gaz	-0.20704	-0.20074	0.58262	0.49205	0.80126	0.34955	0.24532	1.00000	-0.31501	0.38822	-0.08234
gold	0.75163	0.73369	0.38433	0.15442	0.41844	0.45848	0.62295	-0.31501	1.00000	-0.16719	0.64021
platinum	-0.63158	-0.65849	0.19346	0.68360	-0.03262	0.47009	0.39317	0.38822	-0.16719	1.00000	0.56808
silver	0.09456	0.06280	0.44851	0.56907	0.21105	0.72463	0.70729	-0.08234	0.64021	0.56808	1.00000

In [89]: # Корреляция график heatmap

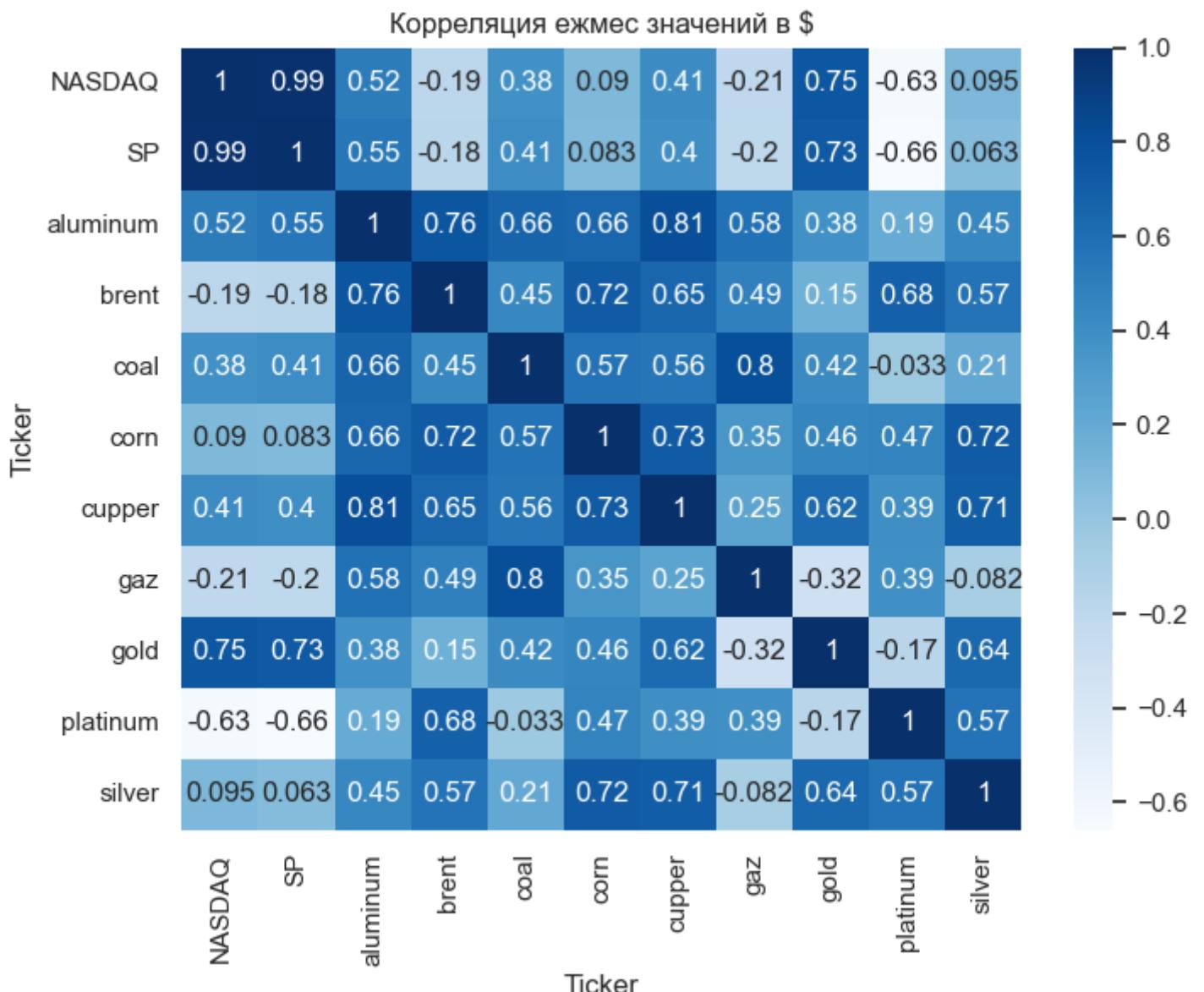
```

sns.set(rc = {'figure.figsize':(8, 6)})

sns.heatmap(corr, cmap='Blues', annot=True).set_title('Корреляция ежмес значений в $')

```

Out[89]: Text(0.5, 1.0, 'Корреляция ежмес значений в \$')



Октябрь 2022:

Обратите внимание на пары товаров с высокой корреляцией (выше 0,75) такие как:

Уголь - Газ 0,90

Золото - Серебро 0,85

Уголь - Зерно 0,83

Медь - Серебро 0,79

Нефть - Алюминий 0,76

Газ - Нефть 0,75

Газ - Зерно 0,75

Также следует заметить показатели корреляции платины, которые ведут себя практически независимо от других биржевых товаров.

Смотрим на показатель:

Платина - Серебро 0,36 - это пригодиться для последующего сравнения

июнь 2023:

Уголь - Газ 0,83

Уголь - Зерно 0,58

май 2024:

Уголь - Газ 0,8

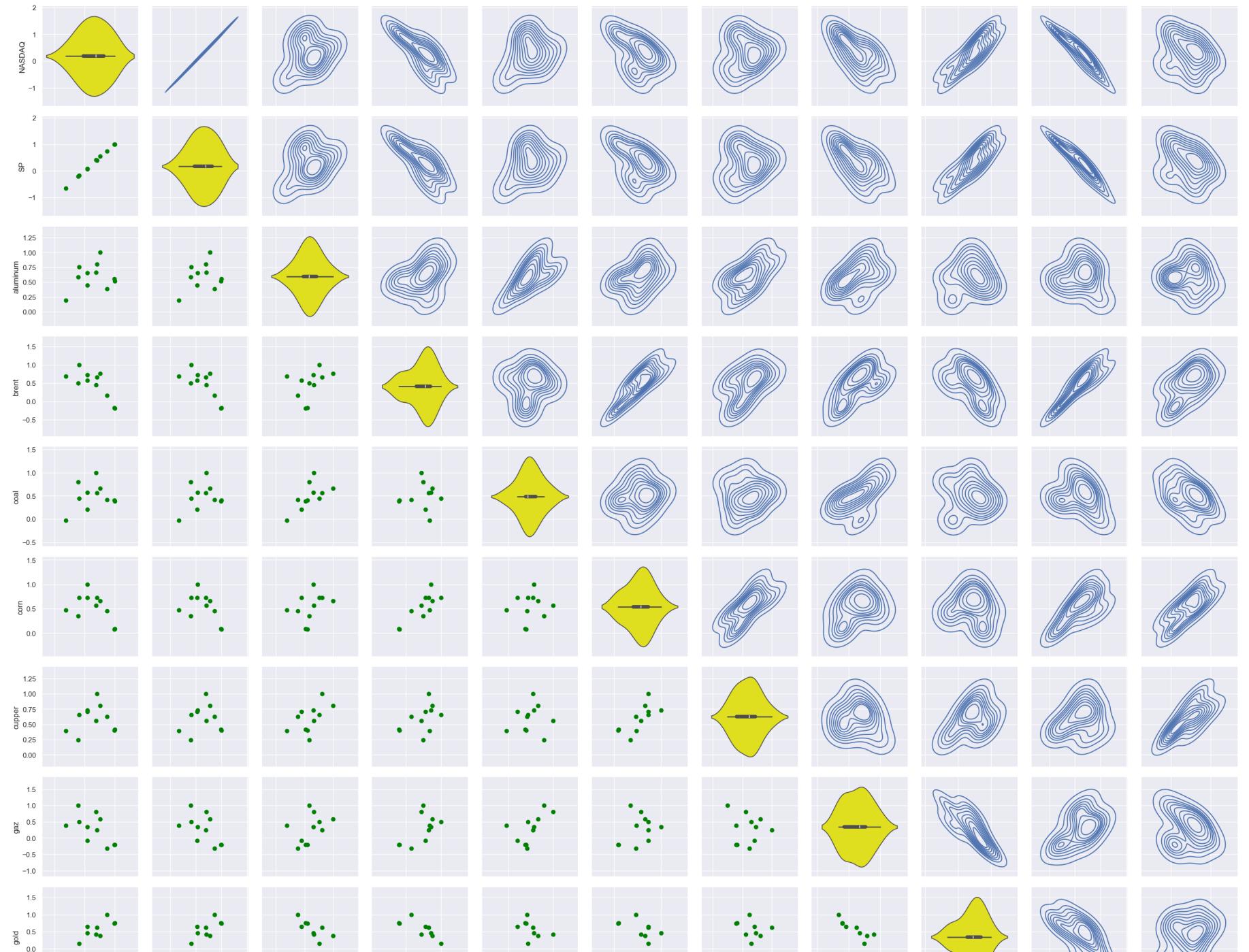
Уголь - Зерно 0,57

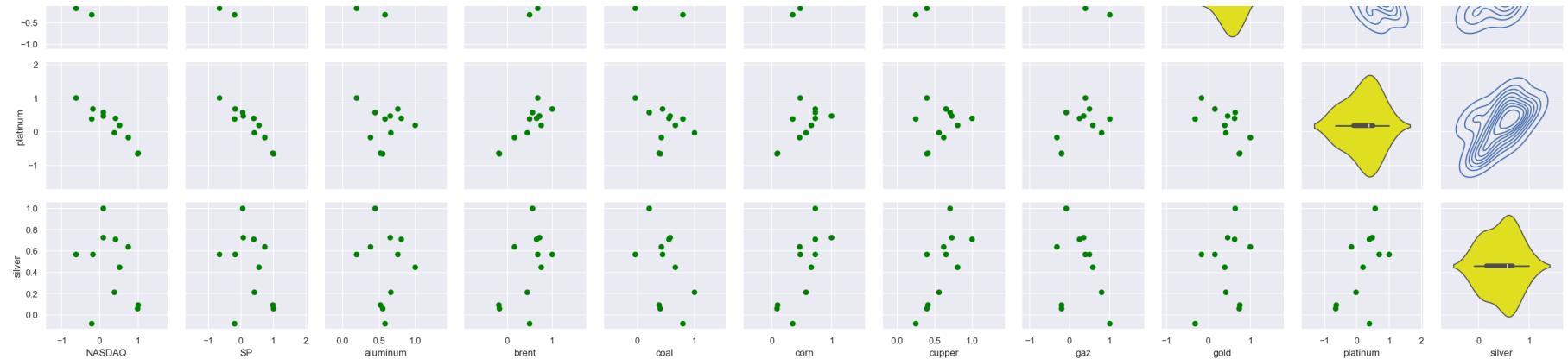
```
In [90]: ### # 2023-08-23 12:40 Огн. вар 26
```

```
g = sns.PairGrid(corr)
g.map_diag(sns.violinplot, color = 'Yellow')
g.map_lower(plt.scatter, color = 'green')
# g.map_upper(sns.kdeplot, cmap = 'Reds' , hue = 'kidder')
g.map_upper(sns.kdeplot, col='diet', kind='kde')
```



```
ontour: 'col', 'kind'  
cset = contour_func()  
Out[90]: <seaborn.axisgrid.PairGrid at 0x74e100167bf0>
```





In [91]:

```
# Определяем корреляцию Пирсона
# Коэффициент корреляции Пирсона применяется для исследования взаимосвязи двух переменных,
# измеренных в метрических шкалах на одной и той же выборке.
# Он позволяет определить, насколько пропорциональна изменчивость двух переменных.

# pearsonr(df['Coal'], df['Gold'])

# pearsonr(df['Brent'], df['CIF'])

# Коэфт корреляции Спирмена
# Коэффициент корреляции Спирмена определяется как коэффициент корреляции Пирсона между ранговыми переменными.

# spearmanr(df['Brent'], df['Coal'])

# spearmanr(df['Brent'], df['CIF'])
```

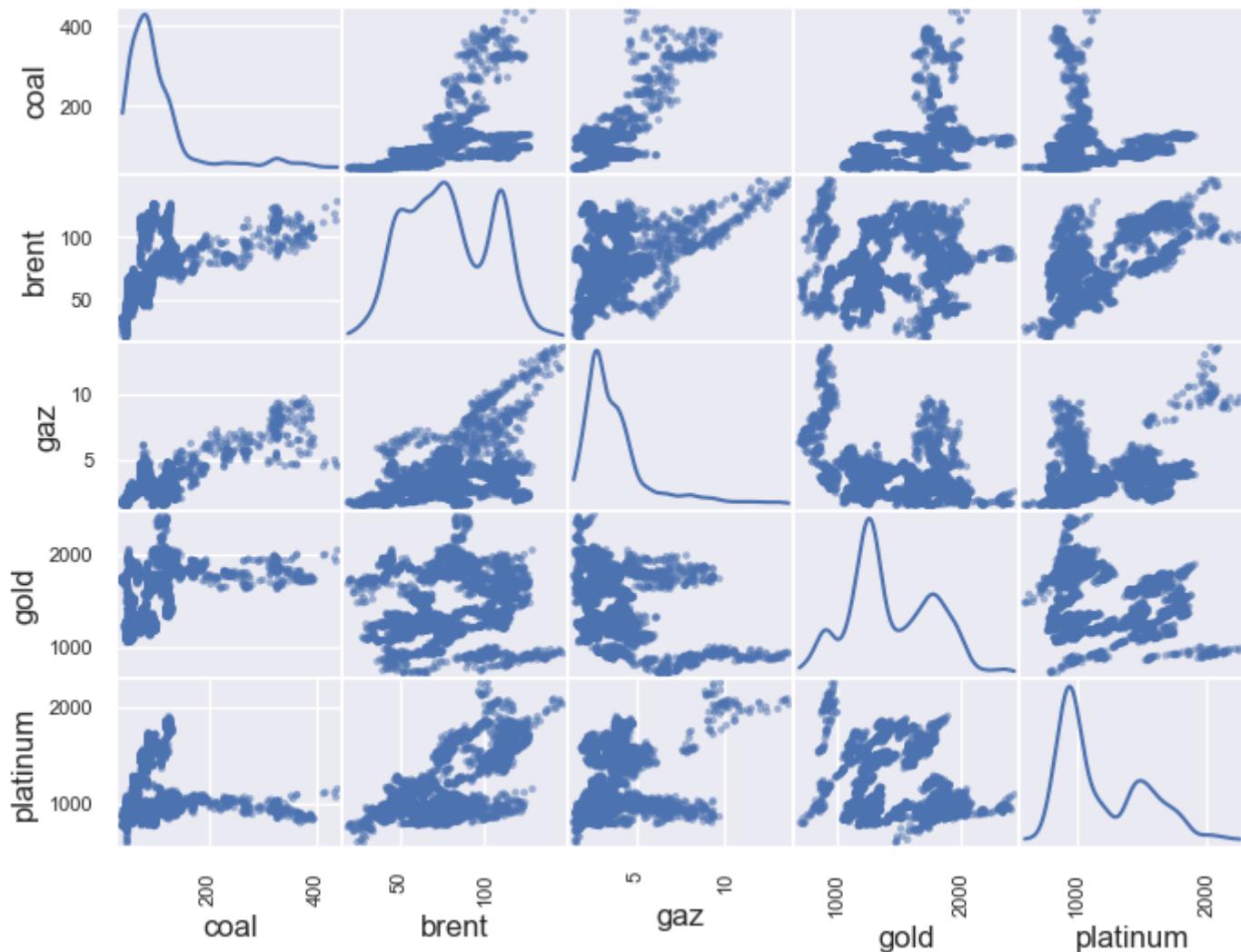
In [92]:

```
# Корреляция. Более наглядный график "scatter_matrix" по отдельным временным рядам
```

```
df1 = df02[["coal", 'brent', "gaz", 'gold', 'platinum']]

print(df1.corr())
pd.plotting.scatter_matrix(df1, diagonal='kde'); # , diagonal='kde'
```

Ticker	coal	brent	gaz	gold	platinum
Ticker					
coal	1.00000	0.44516	0.80126	0.41844	-0.03262
brent	0.44516	1.00000	0.49205	0.15442	0.68360
gaz	0.80126	0.49205	1.00000	-0.31501	0.38822
gold	0.41844	0.15442	-0.31501	1.00000	-0.16719
platinum	-0.03262	0.68360	0.38822	-0.16719	1.00000



In []:

03.2 Ежедневная волатильность временных рядов

df_corr

[К оглавлению](#)

In [93]: # Ежедн изм

```
df_diff = df02 - df02.shift(1)
df_diff.dropna(inplace=True)
df_diff
```

Out[93]:

	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
			Date									

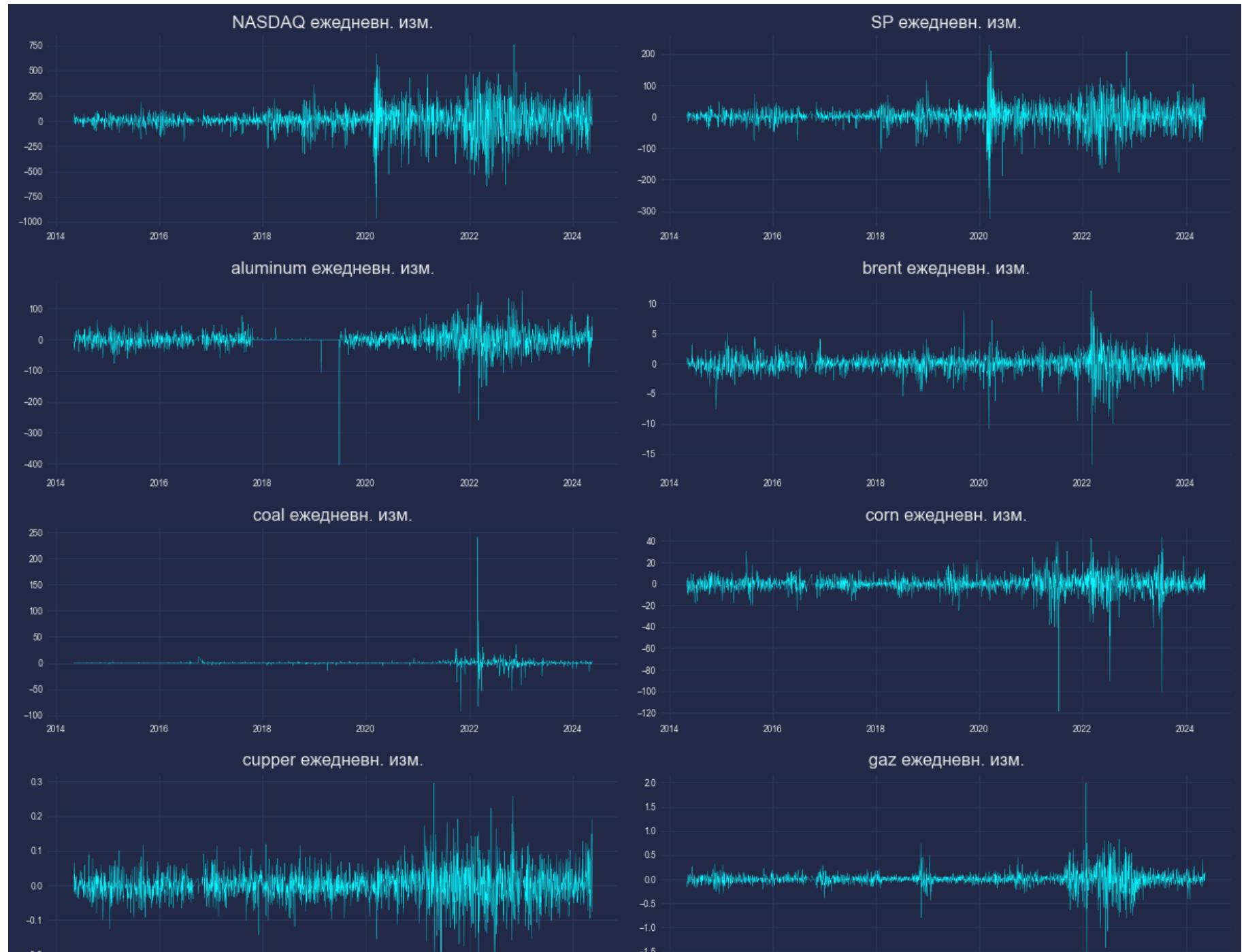
2014-05-07	-13.09009	10.48999	-23.75	1.07000	-0.30	-2.75	-0.0245	-0.059	-19.70007	-23.29993	-0.300
2014-05-08	-16.16992	-2.57996	-7.25	-0.09000	0.30	3.00	0.0320	-0.168	-1.19995	3.29993	-0.204
2014-05-09	20.37012	2.84998	-34.50	-0.15000	0.00	-8.25	0.0180	-0.041	-0.09998	-8.19995	-0.017
2014-05-12	71.98975	18.17004	-19.00	0.52000	-0.05	-7.50	0.0660	-0.097	8.29993	12.00000	0.423
2014-05-13	-13.68994	0.79993	-3.00	0.82999	0.00	5.75	-0.0170	-0.076	-1.00000	13.79993	0.004
...
2024-05-13	47.37012	-1.26025	5.50	0.57000	-1.15	2.75	0.1110	0.129	-31.19995	4.20001	-0.054
2024-05-14	122.93945	25.26025	-2.25	-0.98000	-0.50	-4.75	0.1490	-0.037	17.29980	34.00006	0.264
2024-05-15	231.21094	61.46973	42.00	0.37000	0.25	8.75	0.0160	0.072	35.30005	24.00000	1.029
2024-05-16	-44.07031	-11.04980	-4.50	0.52000	0.90	-5.50	-0.0775	0.079	-8.69995	2.09998	0.151
2024-05-17	-12.34961	6.16992	73.50	0.69000	4.60	-4.25	0.1905	0.143	39.80005	29.29993	2.110

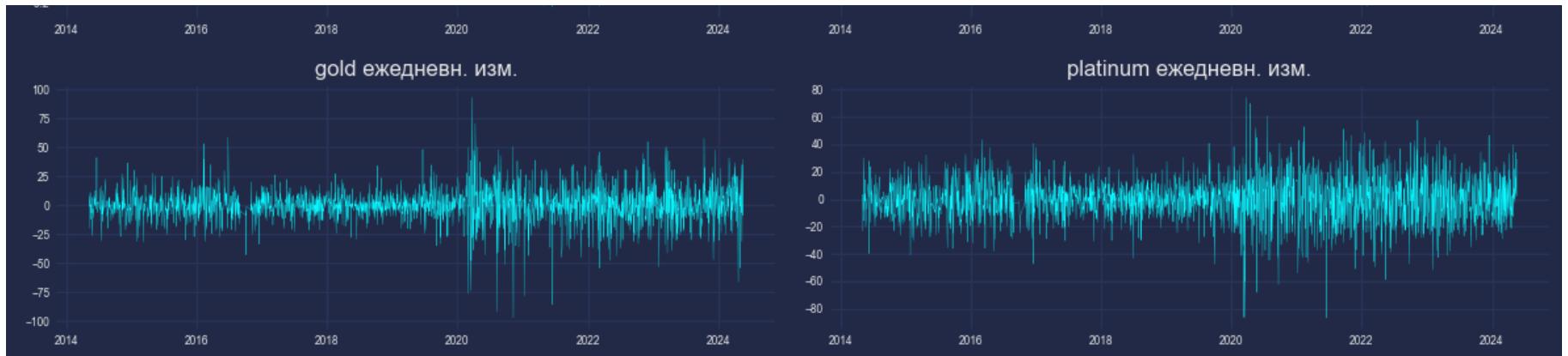
2354 rows × 11 columns

```
In [94]: # настройка графиков
plt.style.use("cyberpunk")

# графики ежедн изм
fig, axes = plt.subplots(nrows=5, ncols=2, dpi=100, figsize=(12,12))
for i, ax in enumerate(axes.flatten()):
    data_temp = df_diff[df_diff.columns[i]]
    ax.plot(data_temp, linewidth=0.2) # , color='black'
    # Decorations
    ax.set_title(df_diff.columns[i] + ' ежедневн. изм.')
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)
    ax.grid(1)

plt.tight_layout();
```





04. Шкалирование временных рядов

df1_scaler

В данном случае шкалирование произвожу методом MinMaxScaler в диапазоне (0, 1)

[К оглавлению](#)

In [95]: # Шкалирование

```
scalerMMS = preprocessing.MinMaxScaler()
```

In [96]: # Шкалирование получение Series

```
series_scaler = scalerMMS.fit_transform(df02)  
series_scaler
```

```
Out[96]: array([[0.08666224, 0.16638455,          nan, ..., 0.0886932 , 0.5746133 ,
   0.16025029],
 [0.08621309, 0.16638455,          nan, ..., 0.09417459, 0.57147171,
  0.16565312],
 [0.07987785, 0.15871337,          nan, ..., 0.09225025, 0.56984045,
  0.16474845],
 ...,
 [1.        , 1.        , 0.44671623, ..., 0.98186478, 0.2823828 ,
  0.52078203],
 [0.99715193, 0.99761427, 0.4448575 , ..., 0.97679162, 0.28365151,
  0.5245766 ],
 [0.99635383, 0.9989464 , 0.47521685, ..., 1.        , 0.30135325,
  0.57759963]])
```

```
In [97]: # DataFrame ежедневных данных из series с нормализованными данными,
```

```
df_scaler = pd.DataFrame(series_scaler, index=df02.index)

df_scaler.columns = col

df_scaler
```

Out[97]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Date											
2008-01-02	0.08666	0.16638	NaN	0.61941	NaN	0.31427	0.47014	0.52650	0.08869	0.57461	0.16025
2008-01-03	0.08621	0.16638	NaN	0.61751	NaN	0.32078	0.50209	0.51195	0.09417	0.57147	0.16565
2008-01-04	0.07988	0.15871	NaN	0.61112	NaN	0.32218	0.49387	0.52575	0.09225	0.56984	0.16475
2008-01-07	0.07954	0.15970	NaN	0.59219	NaN	0.32125	0.48957	0.52890	0.09021	0.56084	0.16058
2008-01-08	0.07573	0.15408	NaN	0.60126	NaN	0.34449	0.52829	0.53617	0.10094	0.57860	0.17382
...
2024-05-13	0.97711	0.98127	0.43030	0.50517	0.16802	0.30683	0.92751	0.07433	0.95119	0.24734	0.48829
2024-05-14	0.98506	0.98673	0.42937	0.49744	0.16677	0.29800	0.96636	0.07127	0.96128	0.26788	0.49492
2024-05-15	1.00000	1.00000	0.44672	0.50036	0.16740	0.31427	0.97053	0.07722	0.98186	0.28238	0.52078
2024-05-16	0.99715	0.99761	0.44486	0.50446	0.16965	0.30404	0.95033	0.08375	0.97679	0.28365	0.52458
2024-05-17	0.99635	0.99895	0.47522	0.50990	0.18115	0.29614	1.00000	0.09558	1.00000	0.30135	0.57760

4124 rows × 11 columns

In [98]:

```
# настройка графиков
plt.style.use("cyberpunk")

# График шкалированных временных рядов для наглядности
# В целом вид очень зашумленного и переполненного графика

df_scaler.plot(figsize=(14, 6), grid=True, title='Шкалированные значения временных рядов изначально выраженных в $')
```



Вышепредставленный график шкалированных в диапазоне 0 - 1 временных рядов стоимости commodity выглядит очень зашумлено. Однако характерно видно начало COVID-19.

```
In [99]: # Добавляю скользящие средние шкалированных значений
day = 30

df_scaler_roll_mean = df_scaler.rolling(day).mean() # (90, min_periods=30)

df_scaler_roll_mean
```

Out[99]:

	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
	Date											
2008-01-02		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2008-01-03		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2008-01-04		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2008-01-07		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2008-01-08		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
2024-05-13	0.95242	0.95978	0.43419	0.53671	0.18992	0.27130	0.83456	0.03512	0.94889	0.21576	0.47112	
2024-05-14	0.95300	0.96007	0.43646	0.53499	0.18855	0.27299	0.84224	0.03644	0.95069	0.21812	0.47336	
2024-05-15	0.95401	0.96077	0.43869	0.53326	0.18743	0.27490	0.84899	0.03803	0.95252	0.22072	0.47551	
2024-05-16	0.95540	0.96185	0.44064	0.53132	0.18661	0.27624	0.85460	0.04002	0.95429	0.22325	0.47763	
2024-05-17	0.95635	0.96256	0.44350	0.52942	0.18603	0.27739	0.86193	0.04237	0.95612	0.22660	0.48129	

4124 rows × 11 columns

In [100...]

```
# График скользящих средних шкалированных значений
df_scaler_roll_mean.plot(figsize=(14, 6), grid=True, title=f'Скользящие средние шкалированных значений по периоду в
# пока отключил т.к. сложно интерпретируемо
```

Out[100]: <AxesSubplot: title={'center': 'Скользящие средние шкалированных значений по периоду в 30 дней'}, xlabel='Date'>



При переходе к скользящим средним шкалированных значений уже становится видно влияние начала событий 2022 года.

05. Переход к измерению в унциях золота

`df_gold`

Таким образом произвожу дефляцию (деление временного ряда на другой ряд) (стр. 279 книги Стефана Янсена) с целью конвертирования в реальную меру.

[К оглавлению](#)

```
In [101]: # Прересчет в унции золота
```

```
df_gold = pd.DataFrame()

for column in df02.columns:
    df_gold[column] = df02[column] / df02['gold']

df_gold
```

```
Out[101]:
```

	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Date											
2008-01-02	3.04508	1.68863	NaN	0.11417	NaN	0.53967	0.00356	0.00916	1.0	1.80513	0.01770
2008-01-03	3.00402	1.67031	NaN	0.11265	NaN	0.53786	0.00366	0.00886	1.0	1.77955	0.01775
2008-01-04	2.90192	1.63553	NaN	0.11214	NaN	0.54078	0.00364	0.00908	1.0	1.78322	0.01778
2008-01-07	2.90770	1.64749	NaN	0.10981	NaN	0.54240	0.00364	0.00917	1.0	1.77315	0.01766
2008-01-08	2.77962	1.58336	NaN	0.10882	NaN	0.54527	0.00373	0.00907	1.0	1.76948	0.01789
...
2024-05-13	7.01521	2.23510	1.06748	0.03568	0.04527	0.19627	0.00206	0.00102	1.0	0.43033	0.01208
2024-05-14	7.01588	2.22940	1.05868	0.03500	0.04472	0.19281	0.00210	0.00100	1.0	0.44162	0.01210
2024-05-15	7.00900	2.22219	1.06062	0.03464	0.04417	0.19362	0.00208	0.00101	1.0	0.44514	0.01236
2024-05-16	7.01610	2.22567	1.06261	0.03499	0.04471	0.19202	0.00206	0.00105	1.0	0.44765	0.01246
2024-05-17	6.89560	2.19161	1.07550	0.03470	0.04587	0.18710	0.00210	0.00109	1.0	0.45239	0.01313

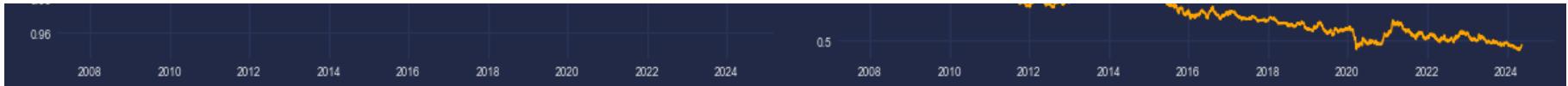
4124 rows × 11 columns

In [102...]

```
# графики стоимости выраженных в унциях золота
fig, axes = plt.subplots(nrows=5, ncols=2, dpi=100, figsize=(12,10))
for i, ax in enumerate(axes.flatten()):
    data_temp = df_gold[df_gold.columns[i]]
    ax.plot(data_temp, color='orange', linewidth=1)
    # Decorations
    ax.set_title(df_gold.columns[i] + ' in ounces of gold')
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)
    ax.grid(1)

plt.tight_layout();
```





In [104]: # удаляю столбцы с постоянными значениями

```
df1_gold = df_gold.drop('gold', axis=1)

# делаю прореживание данных на среднемесячные значения
# так же делая для измерения в золоте и нефти
df_gold_month = df1_gold.resample('M').mean()
df_gold_day = df1_gold.resample('D').mean()

df_gold_month
```

Out[104]:

	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	platinum	silver
Date										
2008-01-31	2.71576	1.54817	NaN	0.10343	NaN	0.54812	0.00359	0.00897	1.78656	0.01801
2008-02-29	2.51588	1.46540	NaN	0.10229	NaN	0.55787	0.00388	0.00934	2.25349	0.01907
2008-03-31	2.34426	1.36909	NaN	0.10681	NaN	0.56912	0.00396	0.01000	2.10902	0.01988
2008-04-30	2.60550	1.50771	NaN	0.12148	NaN	0.65257	0.00433	0.01132	2.20222	0.01921
2008-05-31	2.79762	1.58094	NaN	0.13999	NaN	0.67354	0.00425	0.01282	NaN	0.01915
...
2024-01-31	7.43724	2.36921	1.08014	0.03905	0.05257	0.22280	0.00188	0.00134	0.45402	0.01127
2024-02-29	7.80660	2.47492	1.07266	0.04031	0.04800	0.20906	0.00187	0.00089	0.44043	0.01119
2024-03-31	7.50198	2.39192	1.02513	0.03916	0.05169	0.19867	0.00184	0.00081	0.42167	0.01135
2024-04-30	6.84126	2.19274	1.06701	0.03817	0.05053	0.18639	0.00187	0.00077	0.40371	0.01178
2024-05-31	6.97766	2.22119	1.07741	0.03559	0.04570	0.19333	0.00202	0.00097	0.42834	0.01201

197 rows × 10 columns

05.1 Корреляция временных рядов выраженная в унциях золота

corr_gold

[К оглавлению](#)

In [105...]: # Корреляция в показателях пересчитанных в унции золота.

```
corr_gold = df_gold.corr()  
corr_gold
```

Out[105]:

	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
NASDAQ	1.00000	0.96479	-0.27300	-0.55457	0.19988	-0.51896	-0.35184	-0.34978	NaN	-0.77273	-0.68937
SP	0.96479	1.00000	-0.01443	-0.45189	0.20758	-0.45181	-0.26121	-0.22286	NaN	-0.69261	-0.72542
aluminum	-0.27300	-0.01443	1.00000	0.72397	0.27523	0.44937	0.78086	0.59951	NaN	0.62091	0.35409
brent	-0.55457	-0.45189	0.72397	1.00000	0.28786	0.80867	0.84423	0.77926	NaN	0.79832	0.58681
coal	0.19988	0.20758	0.27523	0.28786	1.00000	0.39389	0.34499	0.68677	NaN	-0.11502	-0.00433
corn	-0.51896	-0.45181	0.44937	0.80867	0.39389	1.00000	0.72601	0.76097	NaN	0.66637	0.60717
cupper	-0.35184	-0.26121	0.78086	0.84423	0.34499	0.72601	1.00000	0.75807	NaN	0.79303	0.53434
gaz	-0.34978	-0.22286	0.59951	0.77926	0.68677	0.76097	0.75807	1.00000	NaN	0.72065	0.27451
gold	NaN	NaN	NaN	NaN							
platinum	-0.77273	-0.69261	0.62091	0.79832	-0.11502	0.66637	0.79303	0.72065	NaN	1.00000	0.65703
silver	-0.68937	-0.72542	0.35409	0.58681	-0.00433	0.60717	0.53434	0.27451	NaN	0.65703	1.00000

In [106...]: # график корреляции средн мес знач выраж в унциях золота

```
sns.set(rc = {'figure.figsize':(12,6)})  
sns.heatmap(corr_gold, cmap='Blues', annot=True).set_title('Корреляции показателей, выраженных в унциях золота')
```

Out[106]: Text(0.5, 1.0, 'Корреляции показателей, выраженных в унциях золота')



Октябрь 2022:

Посмотрим как трансформировалась корреляция при переходы с долл. на унции золота. Заметно снижение:

Уголь - Газ 0,90 --> 0,79

Золото - Серебро 0,85 --> ---

Уголь - Зерно 0,83 --> 0,65

Медь - Серебро 0,79 --> 0,37

Нефть - Алюминий 0,76 --> 0,70

Газ - Нефть 0,75 --> 0,64

Газ - Зерно 0,75 --> 0,63

А вот, что с платиной идет увеличение:

Платина - Серебро 0,36 --> 0,67

июнь 2023:

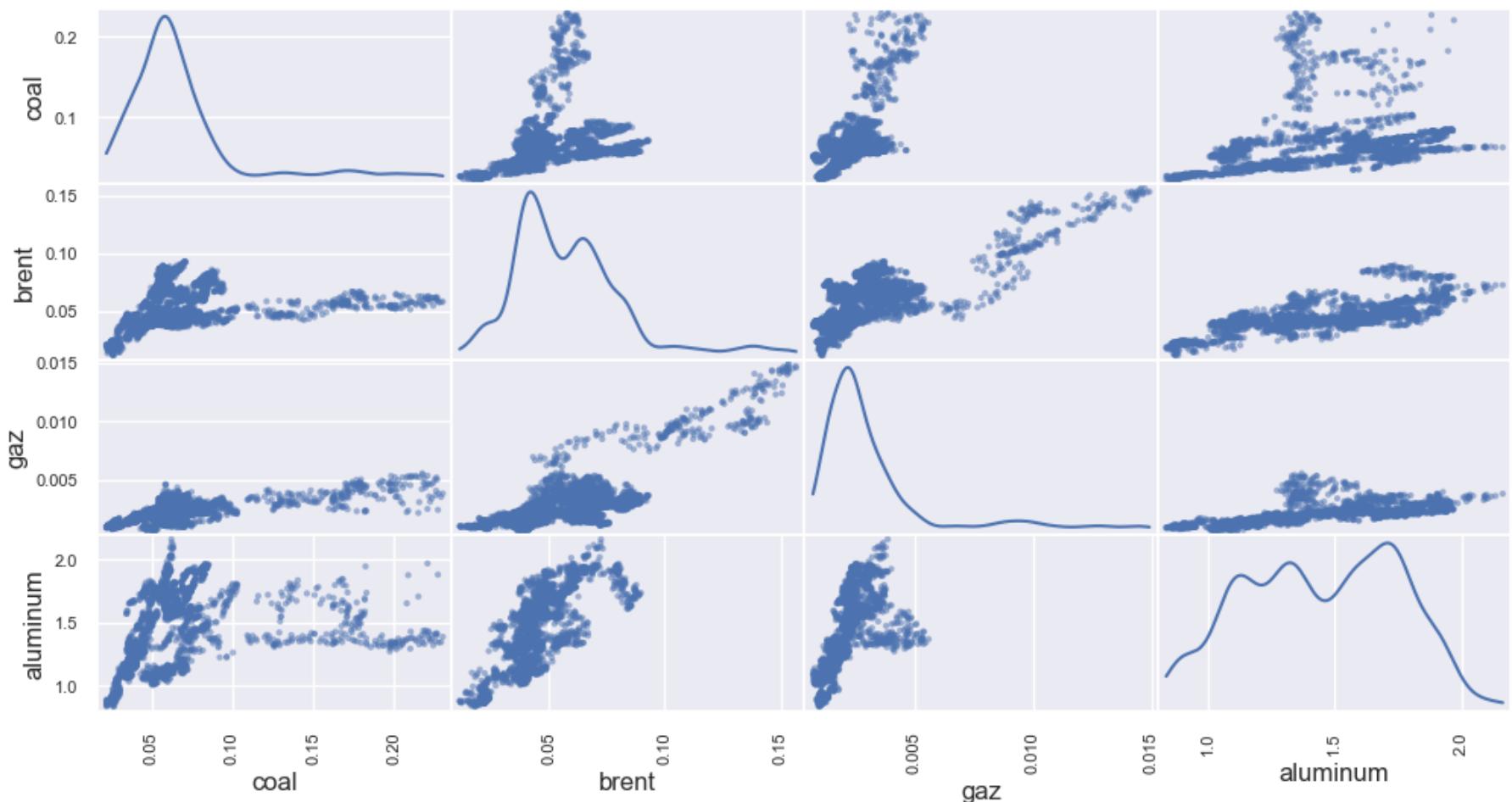
Уголь - Газ 0,83 --> 0,71

май 2023:

Уголь - Газ 0,83 --> 0,71 --> 0.69

```
In [107...]: # Корреляция график scatter_matrix по отдельным временным рядам, выраженных в унциях золота  
df1_gold = df_gold[['coal', 'brent', "gaz", 'aluminum']]  
  
print(df1_gold.corr())  
pd.plotting.scatter_matrix(df1_gold, diagonal='kde'); # , diagonal='kde'
```

	coal	brent	gaz	aluminum
coal	1.00000	0.28786	0.68677	0.27523
brent	0.28786	1.00000	0.77926	0.72397
gaz	0.68677	0.77926	1.00000	0.59951
aluminum	0.27523	0.72397	0.59951	1.00000



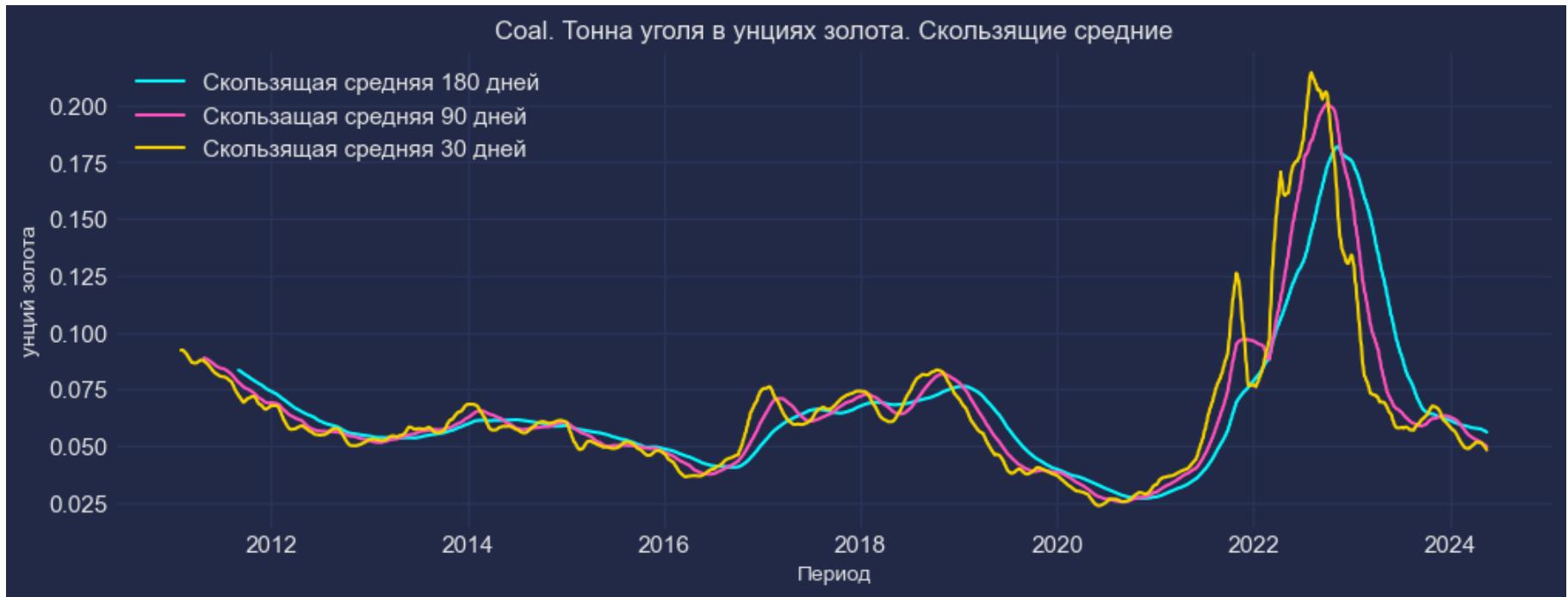
```
In [108]: # Временной ряд стоимости угля в унциях золота  
coal_g = df_gold['coal']  
  
coal_g.dropna(how='any', inplace=True)  
  
coal_g
```

```
Out[108]: Date
2010-12-17    0.08886
2010-12-20    0.08842
2010-12-21    0.08878
2010-12-22    0.09050
2010-12-23    0.09076
...
2024-05-13    0.04527
2024-05-14    0.04472
2024-05-15    0.04417
2024-05-16    0.04471
2024-05-17    0.04587
Name: coal, Length: 3265, dtype: float64
```

```
In [109...]: # Уголь в унциях золота
# Скользящие средние разных периодов в унциях золота
```

```
coal_g_rol180 = coal_g.rolling(180).mean()
coal_g_rol90 = coal_g.rolling(90).mean()
coal_g_rol30 = coal_g.rolling(30).mean()

# график скользящих средних
plt.style.use("cyberpunk")
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(coal_g_rol180, label='Скользящая средняя 180 дней')
plt.plot(coal_g_rol90, label='Скользящая средняя 90 дней')
plt.plot(coal_g_rol30, label='Скользящая средняя 30 дней')
plt.title('Coal. Тонна угля в унциях золота. Скользящие средние')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("унций золота", fontsize = 10)
plt.legend()
plt.show()
```



Вышепредставленный график достаточно близко похож к скользящим средним стоимости угля в долл., но есть особенность, что короткая скользящая средняя в золоте не пересекает длинную, в отличии от оценки в долл.

```
In [110]: plt.style.use("cyberpunk")
plt.figure(figsize=(8, 4))
plt.subplot(211)
plt.plot(coal_rol180, label='Скользящая средняя 180 дней в долл.')
plt.plot(coal_rol30, label='Скользящая средняя 30 дней в долл.')
plt.legend()

plt.subplot(212)
plt.plot(coal_g_rol180, label='Скользящая средняя 180 дней в золоте')
plt.plot(coal_g_rol30, label='Скользящая средняя 30 дней в золоте')
plt.legend()

plt.show()
```



In [111...]

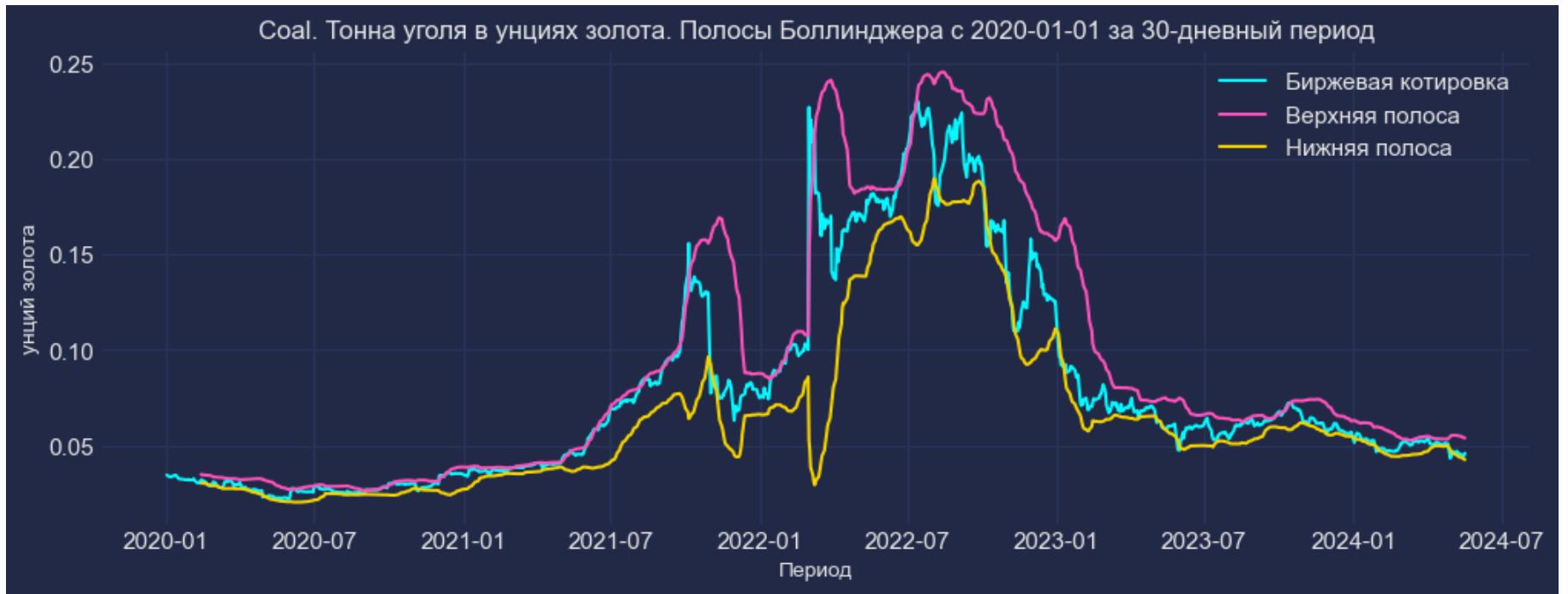
```
# Уголь в унциях золота
# полосы Боллинджера с заданной даты

n = 30
date = '2020-01-01'

coal_g_d = coal_g[date:]
coal_g_ma = coal_g_d.rolling(window=n).mean()
coal_g_sd = coal_g_d.rolling(window=n).std()

coal_g_line1 = coal_g_ma + (2 * coal_g_sd)
coal_g_line2 = coal_g_ma - (2 * coal_g_sd)

# График
plt.figure(figsize=(12, 4))
plt.grid(1)
plt.plot(coal_g_d, label='Биржевая котировка')
plt.plot(coal_g_line1, label='Верхняя полоса')
plt.plot(coal_g_line2, label='Нижняя полоса')
plt.title(f'Coal. Тонна угля в унциях золота. Полосы Боллинджера с {date} за {n}-дневный период')
plt.xlabel("Период", fontsize = 10)
plt.ylabel("унций золота", fontsize = 10)
plt.legend()
plt.show()
```



```
In [112]: # Нормализовываю знач в унциях золота

# series:
series_scaler_gold = scalerMMS.fit_transform(df_gold)

series_scaler_gold

# DataFrame с нормализованными данными:
df_gold_scaler = pd.DataFrame(series_scaler_gold, index=df_gold.index)

df_gold_scaler.columns = col

df_gold_scaler
```

Out[112]:

	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
	Date											
2008-01-02	0.23347	0.52651		NaN	0.70676	NaN	0.56300	0.69179	0.59509	0.0	0.73104	0.41843
2008-01-03	0.22816	0.51769		NaN	0.69632	NaN	0.56037	0.72342	0.57383	0.0	0.71783	0.42084
2008-01-04	0.21494	0.50095		NaN	0.69283	NaN	0.56461	0.71650	0.58981	0.0	0.71973	0.42197
2008-01-07	0.21569	0.50670		NaN	0.67675	NaN	0.56696	0.71515	0.59551	0.0	0.71453	0.41679
2008-01-08	0.19910	0.47583		NaN	0.66992	NaN	0.57113	0.74376	0.58906	0.0	0.71263	0.42666
...
2024-05-13	0.74749	0.78961		0.17385	0.16638	0.11130	0.06465	0.22925	0.02304	0.0	0.02110	0.17746
2024-05-14	0.74758	0.78686		0.16721	0.16171	0.10867	0.05963	0.24408	0.02141	0.0	0.02693	0.17846
2024-05-15	0.74669	0.78339		0.16867	0.15922	0.10600	0.06081	0.23657	0.02250	0.0	0.02875	0.18927
2024-05-16	0.74761	0.78507		0.17017	0.16159	0.10859	0.05848	0.22889	0.02509	0.0	0.03004	0.19393
2024-05-17	0.73201	0.76867		0.17989	0.15959	0.11420	0.05135	0.24271	0.02803	0.0	0.03249	0.22254

4124 rows × 11 columns

In [113...]: # График шкалированных значений временных рядов изначально выраженных в унциях золота
df_gold_scaler.plot(figsize=(12, 6), grid=True, title='Шкалирование значения в унциях золота');



На вышепредставленном графике характерно видно общее падение до минимальных значений в первой половине 2020 года - начало COVID-19. Напоминю, что стоимость одной унции золота взлетела тогда до 1750\$.

06. Переход к измерению в баррелях нефти

df_brent

[К оглавлению](#)

```
In [114]: # Прересчет в баррели нефти
```

```
df_brent = pd.DataFrame()

for column in df02.columns:
    df_brent[column] = df02[column] / df02['brent']

df_brent
```

```
Out[114]:
```

	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Date											
2008-01-02	26.67242	14.79109	NaN	1.0	NaN	4.72711	0.03118	0.08023	8.75920	15.81153	0.15502
2008-01-03	26.66680	14.82746	NaN	1.0	NaN	4.77459	0.03251	0.07863	8.87705	15.79713	0.15760
2008-01-04	25.87716	14.58446	NaN	1.0	NaN	4.82230	0.03246	0.08101	8.91724	15.90144	0.15855
2008-01-07	26.48014	15.00350	NaN	1.0	NaN	4.93961	0.03311	0.08347	9.10690	16.14790	0.16082
2008-01-08	25.54438	14.55087	NaN	1.0	NaN	5.01099	0.03426	0.08339	9.18987	16.26125	0.16440
...
2024-05-13	196.59597	62.63699	29.91543	1.0	1.26859	5.50024	0.05764	0.02856	28.02423	12.05974	0.33854
2024-05-14	200.42705	63.68876	30.24399	1.0	1.27762	5.50801	0.06013	0.02845	28.56761	12.61593	0.34578
2024-05-15	202.32496	64.14683	30.61631	1.0	1.27492	5.58912	0.06005	0.02920	28.86646	12.84955	0.35666
2024-05-16	200.53226	63.61355	30.37108	1.0	1.27777	5.48817	0.05875	0.02996	28.58172	12.79452	0.35625
2024-05-17	198.73715	63.16425	30.99690	1.0	1.32206	5.39245	0.06053	0.03142	28.82087	13.03835	0.37845

4124 rows × 11 columns

In [115...]

```
# графики стоимости выраженных в баррелях нефти
fig, axes = plt.subplots(nrows=5, ncols=2, dpi=100, figsize=(12,10))
for i, ax in enumerate(axes.flatten()):
    data_temp = df_brent[df_brent.columns[i]]
    ax.plot(data_temp, linewidth=1) # , color='black'
    # Decorations
    ax.set_title(df_brent.columns[i] + ' in barrel')
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=6)
    ax.grid(1)

plt.tight_layout();
```





In [116...]: # удаляю столбцы с постоянными значениями

```
df1_brent = df_brent.drop('brent', axis=1)

# делаю прореживание данных на среднемесячные значения
# так же делая для измерения в золоте и нефти

df_brent_month = df1_brent.resample('M').mean()
df_brent_day = df1_brent.resample('D').mean()

df_brent_month
```

Out[116]:

	NASDAQ	SP	aluminum	coal	corn	cupper	gaz	gold	platinum	silver
Date										
2008-01-31	26.25301	14.97273	NaN	NaN	5.31251	0.03480	0.08685	9.69002	17.26151	0.17454
2008-02-29	24.61958	14.33885	NaN	NaN	5.45693	0.03791	0.09125	9.78282	21.68186	0.18649
2008-03-31	21.95016	12.82061	NaN	NaN	5.32954	0.03709	0.09363	9.36973	19.76718	0.18636
2008-04-30	21.47466	12.42912	NaN	NaN	5.38302	0.03573	0.09314	8.25180	18.28613	0.15858
2008-05-31	20.00439	11.30610	NaN	NaN	4.81634	0.03042	0.09158	7.14813	NaN	0.13688
...
2024-01-31	190.48301	60.69068	27.67904	1.34843	5.71108	0.04816	0.03438	25.62786	11.63991	0.28897
2024-02-29	193.76242	61.42996	26.63341	1.19192	5.19352	0.04654	0.02205	24.82874	10.93511	0.27779
2024-03-31	191.61022	61.09181	26.18289	1.32002	5.07370	0.04697	0.02066	25.54423	10.77239	0.28978
2024-04-30	179.22809	57.44857	27.98589	1.32427	4.88450	0.04902	0.02013	26.21441	10.58049	0.30898
2024-05-31	196.12513	62.43053	30.27704	1.28434	5.43343	0.05677	0.02727	28.10728	12.04509	0.33774

197 rows × 10 columns

06.1 Корреляция временных рядов выраженная в баррелях нефти

[corr_brent](#)

[К оглавлению](#)

In [117]: # Корреляция в показателях пересчитанных в баррели нефти.

```
corr_brent = df_brent.corr()
corr_brent
```

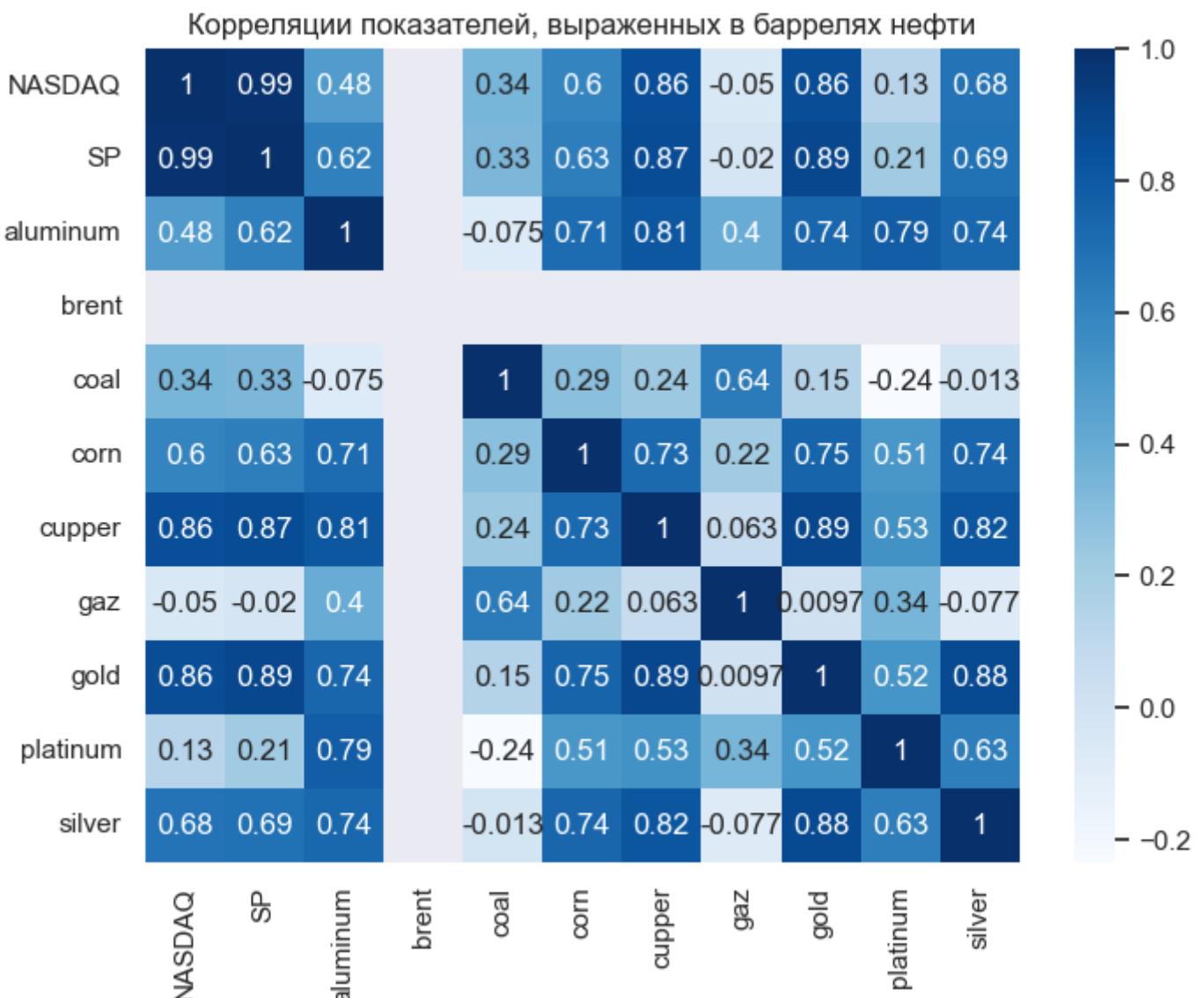
Out[117]:

	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
NASDAQ	1.00000	0.98516	0.48040	NaN	0.34370	0.60028	0.85856	-0.05021	0.86368	0.13493	0.67762
SP	0.98516	1.00000	0.61918	NaN	0.33113	0.63146	0.86949	-0.02001	0.89285	0.21355	0.68655
aluminum	0.48040	0.61918	1.00000	NaN	-0.07504	0.71203	0.81211	0.39560	0.74352	0.78504	0.73839
brent	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
coal	0.34370	0.33113	-0.07504	NaN	1.00000	0.28842	0.23518	0.64392	0.14506	-0.23558	-0.01274
corn	0.60028	0.63146	0.71203	NaN	0.28842	1.00000	0.72670	0.21556	0.74966	0.50918	0.74230
cupper	0.85856	0.86949	0.81211	NaN	0.23518	0.72670	1.00000	0.06298	0.89226	0.53397	0.81933
gaz	-0.05021	-0.02001	0.39560	NaN	0.64392	0.21556	0.06298	1.00000	0.00972	0.34418	-0.07657
gold	0.86368	0.89285	0.74352	NaN	0.14506	0.74966	0.89226	0.00972	1.00000	0.51876	0.87552
platinum	0.13493	0.21355	0.78504	NaN	-0.23558	0.50918	0.53397	0.34418	0.51876	1.00000	0.63012
silver	0.67762	0.68655	0.73839	NaN	-0.01274	0.74230	0.81933	-0.07657	0.87552	0.63012	1.00000

In [118]: # график корреляции средн мес знач выраж в баррелях нефти.

```
sns.set(rc = {'figure.figsize':(8,6)})
sns.heatmap(corr_brent, cmap='Blues', annot=True).set_title('Корреляции показателей, выраженных в баррелях нефти')
```

Out[118]: Text(0.5, 1.0, 'Корреляции показателей, выраженных в баррелях нефти')



Октябрь 2022:

К корреляции в долл., затем в золоте добавляем корреляцию в барелях:

Уголь - Газ 0,90 --> 0,79 --> 0,70

Золото - Серебро 0,85 --> --- --> 0,93

Уголь - Зерно 0,83 --> 0,65 --> 0,18

Медь - Серебро 0,79 --> 0,37 --> 0,78

Нефть - Алюминий 0,76 --> 0,70 --> ---

Газ - Нефть 0,75 --> 0,64 --> ---

Газ - Зерно 0,75 --> 0,63 --> 0,4

А вот, что с платиной:

Платина - Серебро 0,36 --> 0,67 --> 0,80

Интерпритацию данных значений оставлю на последующий анализ и осмысление, что бы не попасть в ловушку "ложных корреляций" и поверхностного подхода.

июнь 2023:

Уголь - Газ 0,83 --> 0,71 --> 0,72

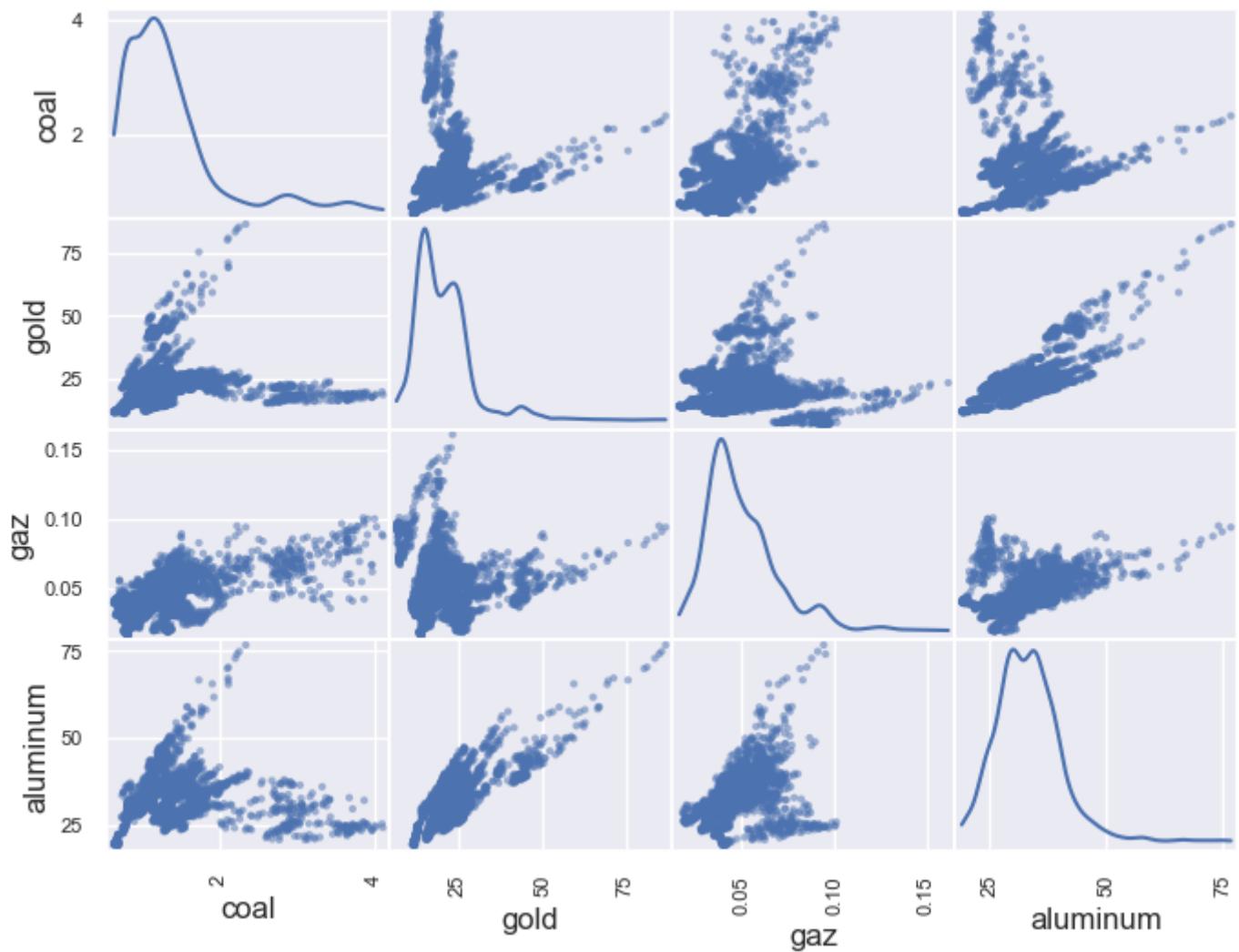
май 2024:

Уголь - Газ 0,83 --> 0,71 --> 0,72 --> 0,64

```
In [119... # Корреляция график scatter_matrix по отдельным временным рядам, выраженных в баррелях неефти.
```

```
df1_brent = df_brent[['coal', 'gold', 'gaz', 'aluminum']]  
print(df1_brent.corr())  
pd.plotting.scatter_matrix(df1_brent, diagonal='kde'); # , diagonal='kde'
```

	coal	gold	gaz	aluminum
coal	1.00000	0.14506	0.64392	-0.07504
gold	0.14506	1.00000	0.00972	0.74352
gaz	0.64392	0.00972	1.00000	0.39560
aluminum	-0.07504	0.74352	0.39560	1.00000



```
In [120...]: # Уголь в барелях нефти
```

```
# Временной ряд стоимости угля в унциях золота  
coal_b = df_brent['coal']
```

```
coal_b.dropna(how='any', inplace=True)
```

```
coal_b
```

```
coal_b_rol180 = coal_b.rolling(180).mean()  
coal_b_rol90 = coal_b.rolling(90).mean()  
coal_b_rol30 = coal_b.rolling(30).mean()
```

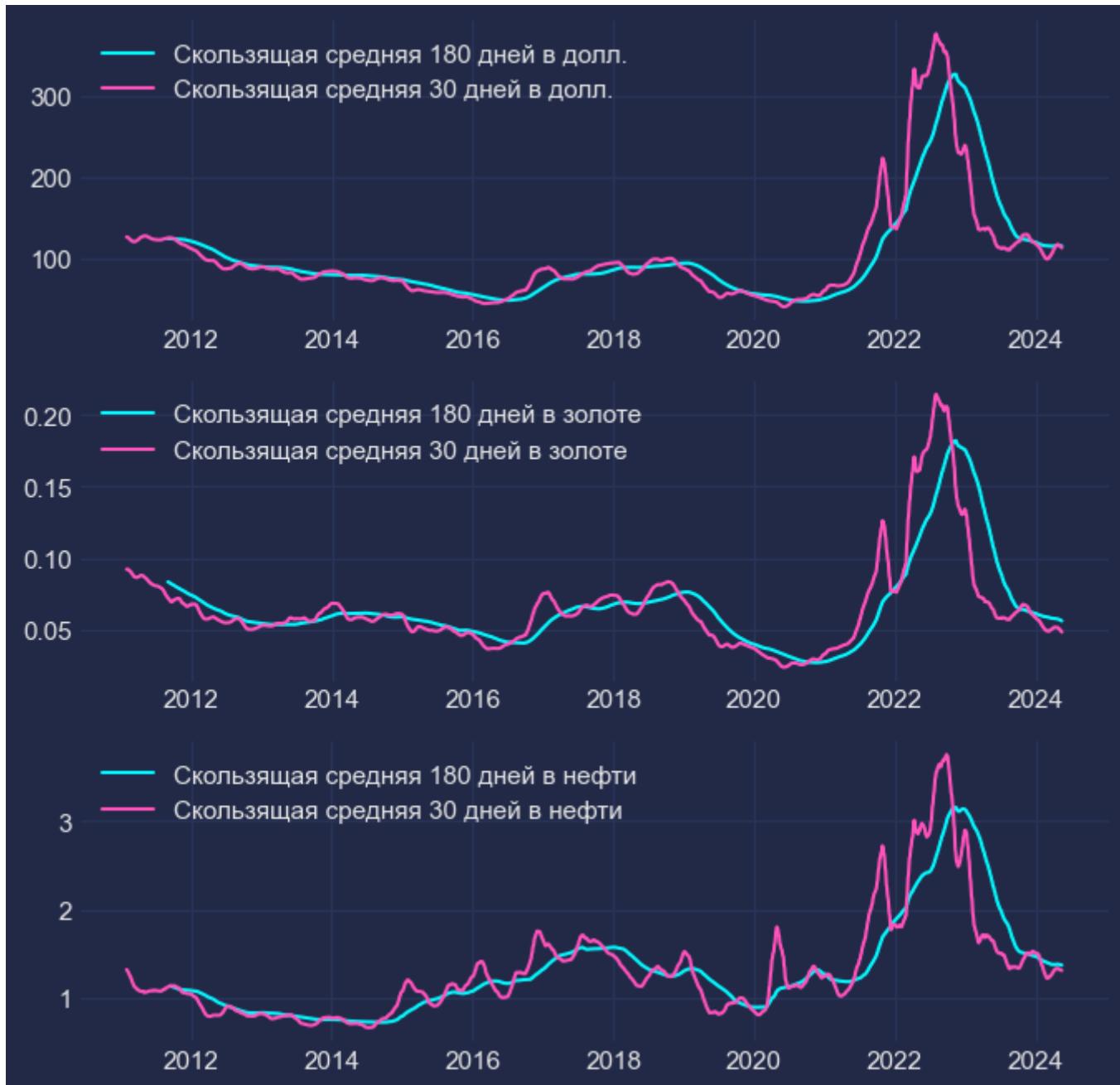
```
In [121...]: # Продолжение внизу скользящих средних в сравнении
```

```
plt.style.use("cyberpunk")  
plt.figure(figsize=(8, 8))  
plt.subplot(311)  
plt.plot(coal_b_rol180, label='Скользящая средняя 180 дней в долл.')  
plt.plot(coal_b_rol30, label='Скользящая средняя 30 дней в долл.')  
plt.legend()
```

```
plt.subplot(312)  
plt.plot(coal_g_rol180, label='Скользящая средняя 180 дней в золоте')  
plt.plot(coal_g_rol30, label='Скользящая средняя 30 дней в золоте')  
plt.legend()
```

```
plt.subplot(313)  
plt.plot(coal_b_rol180, label='Скользящая средняя 180 дней в нефти')  
plt.plot(coal_b_rol30, label='Скользящая средняя 30 дней в нефти')  
plt.legend()
```

```
plt.show()
```



In [122...]

```
# Нормализовываю знач в баррелях неефти.

# series:
series_scaler_brent = scalerMMS.fit_transform(df_brent)

series_scaler_brent

# DataFrame с нормализованными данными:
df_brent_scaler = pd.DataFrame(series_scaler_brent, index=df_brent.index)

df_brent_scaler.columns = col

df_brent_scaler
```

Out[122]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Date											
2008-01-02	0.02682	0.04619	NaN	0.0	NaN	0.11258	0.05602	0.44105	0.02959	0.26156	0.07219
2008-01-03	0.02681	0.04645	NaN	0.0	NaN	0.11632	0.07031	0.43000	0.03105	0.26110	0.07611
2008-01-04	0.02493	0.04466	NaN	0.0	NaN	0.12008	0.06974	0.44641	0.03155	0.26439	0.07755
2008-01-07	0.02637	0.04775	NaN	0.0	NaN	0.12931	0.07672	0.46336	0.03391	0.27217	0.08099
2008-01-08	0.02414	0.04441	NaN	0.0	NaN	0.13493	0.08912	0.46278	0.03494	0.27575	0.08643
...
2024-05-13	0.43010	0.39939	0.18814	0.0	0.18358	0.17345	0.33989	0.08537	0.26908	0.14317	0.35062
2024-05-14	0.43919	0.40716	0.19383	0.0	0.18617	0.17407	0.36665	0.08462	0.27584	0.16072	0.36159
2024-05-15	0.44370	0.41054	0.20028	0.0	0.18540	0.18045	0.36584	0.08973	0.27955	0.16809	0.37811
2024-05-16	0.43944	0.40660	0.19603	0.0	0.18622	0.17250	0.35183	0.09501	0.27601	0.16636	0.37748
2024-05-17	0.43518	0.40329	0.20688	0.0	0.19897	0.16497	0.37100	0.10504	0.27899	0.17405	0.41117

4124 rows × 11 columns

In [123...]

```
# График шкалированных значений временных рядов изначально выраженных в баррелях нефти.  
# настройка графиков  
plt.style.use("cyberpunk")  
df_brent_scaler.plot(figsize=(12, 6), grid=True, title='Шкалирование значения в баррелях нефти.');
```



В расчете на баррель нефти график выглядит противоположно в сравнении с графиком в унции золота. Как мы помним одновременно с началом COVID-19 стоимость барреля нефти снижалась практически до нуля и, следовательно, стоимость биржевых товаров взлетает до максимальных значений.

07. Представление шкалированной стоимости угля в разных ед.измерения

df2

[К оглавлению](#)

Ниже представленный график шкалированной стоимость угля в разных ед.измерения необходимо интерпретировать.

На обсуждение: м.б. стоит это использовать для последующего прогнозирования.

In [124...]

```
# График Шкалированной стоимость угля в разных ед.измерения
plt.figure(figsize=(14,4))
plt.plot(df_scaler['coal']['2020-01-01':], label='Coal in $', color='b')
plt.plot(df_gold_scaler['coal']['2020-01-01':], label='Coal in gold', color='r')
plt.plot(df_brent_scaler['coal']['2020-01-01':], label='Coal in brent', color='g')
plt.title('Шкалированная стоимость угля в разных ед.измерения')
plt.legend()
plt.show()
```



Обратите внимание на два "всплеска": начало COVID-19, и февраль-март 2022 - начало СВО России. Скоординированный "всплеск" в октябре 2021 пока не проанализирован.

In [125...]

```
# dataframe со шкалированной стоимостью угля при разных ед изм.  
df2 = pd.concat([df_scaler['coal'], df_gold_scaler['coal'], df_brent_scaler['coal']], axis=1)  
  
df2.columns = ['in_doll', 'in_gold', 'in_brent']  
  
df2.dropna(how='any', inplace=True)  
  
df2
```

Out[125]:

	in_doll	in_gold	in_brent
Date			
2010-12-17	0.20992	0.32102	0.20307
2010-12-20	0.20992	0.31889	0.19864
2010-12-21	0.21180	0.32067	0.19908
2010-12-22	0.21743	0.32890	0.20416
2010-12-23	0.21680	0.33018	0.20094
...
2024-05-13	0.16802	0.11130	0.18358
2024-05-14	0.16677	0.10867	0.18617
2024-05-15	0.16740	0.10600	0.18540
2024-05-16	0.16965	0.10859	0.18622
2024-05-17	0.18115	0.11420	0.19897

Date	in_doll	in_gold	in_brent
2010-12-17	0.20992	0.32102	0.20307
2010-12-20	0.20992	0.31889	0.19864
2010-12-21	0.21180	0.32067	0.19908
2010-12-22	0.21743	0.32890	0.20416
2010-12-23	0.21680	0.33018	0.20094
...
2024-05-13	0.16802	0.11130	0.18358
2024-05-14	0.16677	0.10867	0.18617
2024-05-15	0.16740	0.10600	0.18540
2024-05-16	0.16965	0.10859	0.18622
2024-05-17	0.18115	0.11420	0.19897

3231 rows × 3 columns

07.1 Корреляция стоимости угля выраженная в разных единицах измерения

corr2

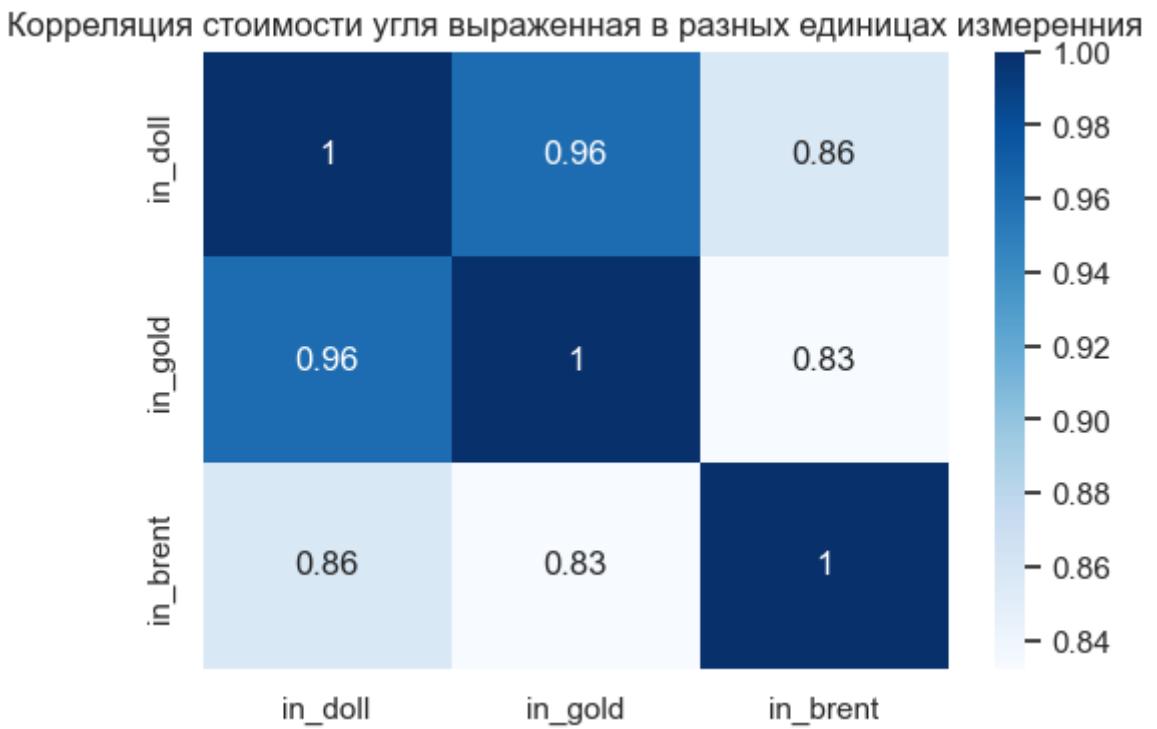
[К оглавлению](#)

```
In [126]: # Корреляция стоимости угля выраженная в разных единицах измерения  
corr2 = df2.corr()  
corr2
```

```
Out[126]:      in_doll  in_gold  in_brent  
in_doll    1.00000  0.96072  0.85761  
in_gold    0.96072  1.00000  0.83239  
in_brent   0.85761  0.83239  1.00000
```

```
In [127]: sns.set(rc = {'figure.figsize':(6,4)})  
sns.heatmap(corr2, cmap='Blues', annot=True).set_title('Корреляция стоимости угля выраженная в разных единицах изме
```

```
Out[127]: Text(0.5, 1.0, 'Корреляция стоимости угля выраженная в разных единицах измерения')
```



Часть вторая

08. Построение прогнозов стоимости угля на основании использования библиотеки SKTIME

sktime

[К оглавлению](#)

08.1 Метод прогнозирования SKTIME ARIMA

[sktime ARIMA](#)

Прогноз только по самому временному ряду стоимости угля.

```
In [128]: # Расчет прогноза по самому временному ряду стоимости угля  
y_d = df_day['coal'] # ежедневные данные  
y_d
```

```
Out[128]: Date  
2008-01-02      NaN  
2008-01-03      NaN  
2008-01-04      NaN  
2008-01-05      NaN  
2008-01-06      NaN  
...  
2024-05-13    105.75  
2024-05-14    105.25  
2024-05-15    105.50  
2024-05-16    106.40  
2024-05-17    111.00  
Freq: D, Name: coal, Length: 5981, dtype: float64
```

```
In [133]: ### на конкретную дату  
y_d['2023-06-01']
```

```
Out[133]: 93.8499984741211
```

In [129...]

```
forecaster = ARIMA()

# forecaster = AutoARIMA(sp=12, d=0, max_p=2, max_q=2, suppress_warnings=True)

# или так можно задать:
# forecaster.fit(y, fh=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# задание периодов прогноза

fh = np.arange(1, 5)

# прогноз по дни
forecaster.fit(y_d)

y_pred_ARIMA_d = forecaster.predict(fh=fh)

y_pred_ARIMA_d

### 2023-06-09
# oduleNotFoundError: ARIMA requires package 'pmdarima' to be present in the python environment,
# but 'pmdarima' was not found. 'pmdarima' is a soft dependency and not included in the base sktime installation.
# Please run: `pip install pmdarima` to install the pmdarima package.
# To install all soft dependencies, run: `pip install sktime[all_extras]
```

```
Out[129]: 2024-05-18    110.96677
2024-05-19    110.93367
2024-05-20    110.90070
2024-05-21    110.86787
Freq: D, Name: coal, dtype: float64
```

Прогноз от октября 2022:

2022-11-05 222.22353

2022-11-06 221.78808

2022-11-07 221.35449

2022-11-08 220.92274

Прогноз от 15 марта 2023:

2023-03-16 135.40383

2023-03-17 135.28157

2023-03-18 135.15985

2023-03-19 135.03866

Прогноз от 2023-06-09:

2023-06-10 112.41548

2023-06-11 112.37348

2023-06-12 112.33164

2023-06-13 112.28997

Прогноз от 2024-05-19:

2024-05-18 110.96677

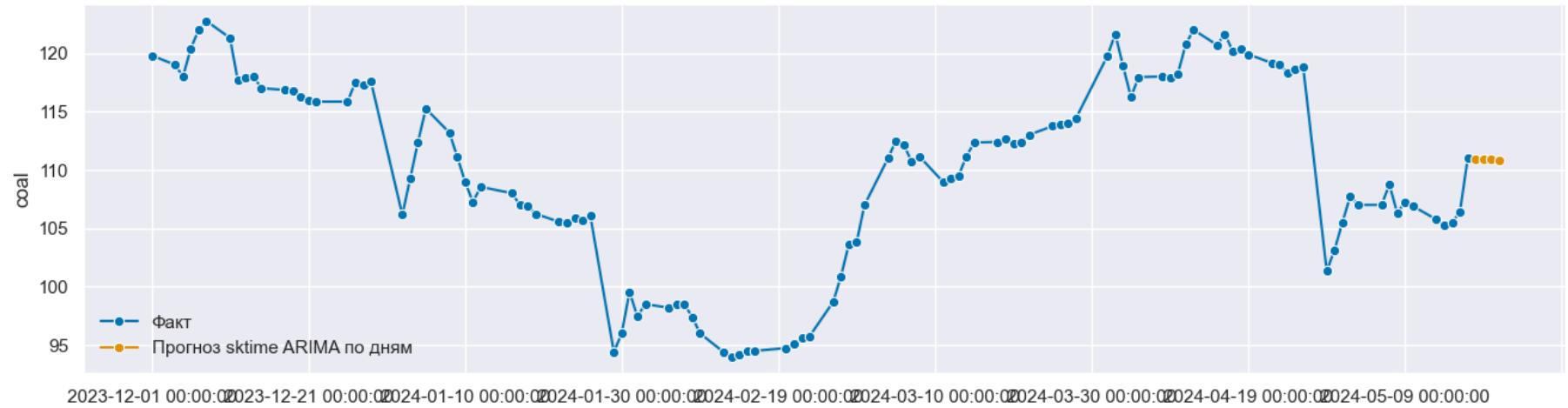
2024-05-19 110.93367

2024-05-20 110.90070

2024-05-21 110.86787

```
In [130]: plot_series(y_d['2023-12-01':], y_pred_ARIMA_d, labels=["Факт", "Прогноз sktime ARIMA по дням"])
```

Out[130]: (<Figure size 1600x400 with 1 Axes>, <AxesSubplot: ylabel='coal'>)



```
In [131]: y_m = df_month['coal'] # ежемес данные
```

```
y_m
```

```
Out[131]: Date
2008-01-31      NaN
2008-02-29      NaN
2008-03-31      NaN
2008-04-30      NaN
2008-05-31      NaN
...
2024-01-31    106.62000
2024-02-29    97.22600
2024-03-31   111.73200
2024-04-30   117.84682
2024-05-31   106.95000
Freq: M, Name: coal, Length: 197, dtype: float64
```

```
In [132...]: # прогноз по месяцам
forecaster.fit(y_m)

fh_m = np.arange(1, 4)

y_pred_ARIMA_m = forecaster.predict(fh=fh_m)

y_pred_ARIMA_m
```

```
Out[132]: 2024-06-30    106.78758
2024-07-31    106.63402
2024-08-31    106.48882
Freq: M, Name: coal, dtype: float64
```

Прогноз по месяца от октября 2022:

2022-12-31 225.08039

2023-01-31 221.93905

2023-02-28 218.90548

Прогноз от 15 марта 2023:

2023-04-30 128.35539

2023-05-31 126.93890

2023-06-30 125.60092

Прогноз от 2023-06-09:

2023-07-31 105.38935

2023-08-31 105.26898

2023-09-30 105.15516

Прогноз от 2024-05-19:

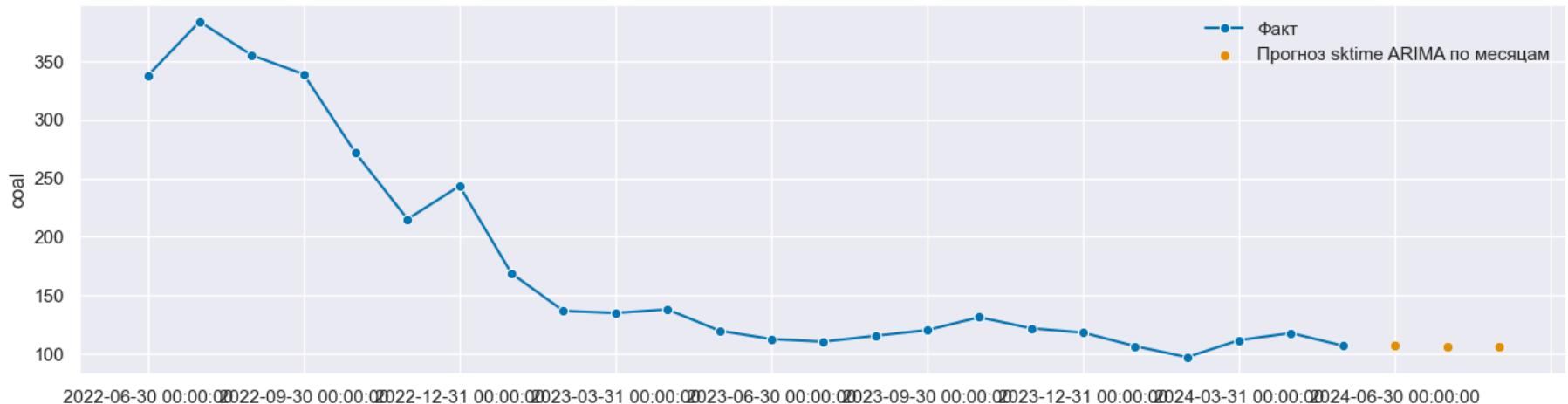
2024-06-30 106.78758

2024-07-31 106.63402

2024-08-31 106.48882

```
In [134]: plot_series(y_m['2022-06-01':], y_pred_ARIMA_m, labels=["Факт", "Прогноз sktime ARIMA по месяцам"])
```

```
Out[134]: (<Figure size 1600x400 with 1 Axes>, <AxesSubplot: ylabel='coal'>)
```



```
In [135]: # forecaster.forecasters_
```

```
In [136]: # вероятностный прогноз
coverage = 0.2
y_pred_ints = forecaster.predict_interval(coverage=coverage)
y_pred_ints
```

```
Out[136]:
```

	coal	
	0.2	
	lower	upper
2024-06-30	101.78594	111.78923
2024-07-31	99.75062	113.51742
2024-08-31	98.28056	114.69707

Прогноз от октября 2022:

2022-12-31 219.76729 230.39348

2023-01-31 214.55297 229.32513

2023-02-28 210.01143 227.79952

Прогноз от 15 марта 2023:

2023-04-30 123.52096 133.18981

2023-05-31 120.28877 133.58902

2023-06-30 117.67445 133.52740

Прогноз от 2023-06-09:

2023-07-31 100.23679 110.54192

2023-08-31 98.17787 112.36008

2023-09-30 96.69923 113.61108

Прогноз 2024-05-19:

2024-06-30 101.78594 111.78923

2024-07-31 99.75062 113.51742

2024-08-31 98.28056 114.69707

In [137...]

```
plt.figure(figsize=(10, 4))
plt.plot(y_m)
plt.plot(y_pred_ARIMA_m)
plt.plot(y_pred_ints)
plt.show()
```



`predict_quantiles` - квантильные прогнозы sktime предлагает `predict_quantiles` в качестве унифицированного интерфейса для возврата квантильных значений прогнозов. Аналогично `predict_interval`.

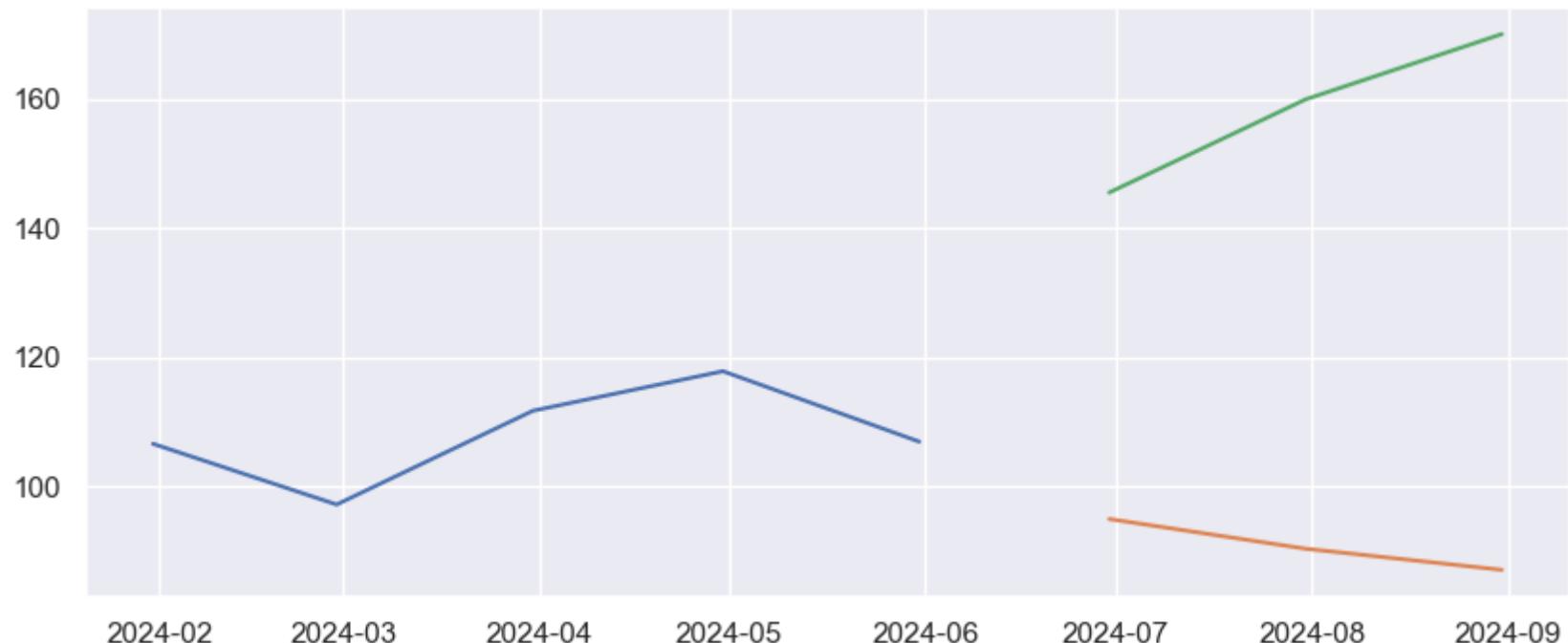
`predict_quantiles` имеет аргумент `alpha`, содержащий запрашиваемые значения квантилей. Подобно случаю с `predict_interval`, `alpha` может быть значением с плавающей точкой или списком значений с плавающей точкой.

```
In [138]: y_pred_quantiles = forecaster.predict_quantiles(alpha=[0.275, 0.975])  
y_pred_quantiles
```

```
Out[138]:
```

	coal	
	0.275	0.975
2024-06-30	94.98645	145.48170
2024-07-31	90.39297	159.88594
2024-08-31	87.12184	169.99016

```
In [140]: plt.figure(figsize=(10, 4))  
plt.plot(y_m['2024-01-01':])  
plt.plot(y_pred_quantiles)  
plt.show()
```



```
In [141...]: # predict_var выдает предсказания дисперсии:  
y_pred_var = forecaster.predict_var()  
y_pred_var
```

```
Out[141]: coal  
2024-06-30    389.75671  
2024-07-31    738.20061  
2024-08-31   1049.71070
```

```
In [ ]:
```

```
In [142...]: # predict_proba - прогнозы распределения  
y_pred_proba = forecaster.predict_proba()  
y_pred_proba
```

```
Out[142]: ▼  
Normal  
Normal(columns=Index(['coal'], dtype='object'),  
       index=DatetimeIndex(['2024-06-30', '2024-07-31', '2024-08-31'], dtype='datetime64[ns]', freq  
='M'),  
       mu=  
          coal  
2024-06-30    106.78758  
2024-07-31    106.63402  
2024-08-31    106.48882,  
       sigma=  
          coal  
2024-06-30    19.74226
```

08.2 Sktime. Метод прогнозирования VAR (векторная авторегрессия) в долл.

sktime.var

[К оглавлению](#)

```
In [143...]: # Напоминание:  
# df со средн мес знач в долл.  
# df_month  
  
# df со средн мес знач в золоте.  
# df_gold_month  
  
# df со средн мес знач в нефти.  
# df_brent_month
```

```
In [144...]: # Создание df со средн дневн и сред мес значениями  
  
y_m = df_month['2015-01-01':] # ежемес знач  
y_d = df_day['2015-01-01':] # ежедн знач
```

```
In [145...]: df_day1 = df_day.dropna(how='any')  
df_day1.fillna(0, inplace=True)  
  
y_d1 = df_day1['2015-01-01':] # ежедн знач без пропусков
```

```
/tmp/ipykernel_85268/3963326005.py:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df_day1.fillna(0, inplace=True)
```

```
In [146...]: # df среднемес знач  
y_m
```

Out[146]:	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	copper	gaz	gold	platinum	silver
	Date											
	2015-01-31	4673.70149	2028.17851	2220.80000	49.68000	59.47632	388.10000	2.65322	2.92935	1252.92499	1245.72000	17.20000
	2015-02-28	4854.25583	2082.19579	2174.71053	58.44053	61.71176	383.68421	2.62363	2.75495	1224.91052	1199.63684	16.72021
	2015-03-31	4938.01045	2079.99044	2056.14773	56.93500	61.54950	382.94318	2.70793	2.74691	1177.76818	1140.41363	16.19436
	2015-04-30	4985.95143	2094.86284	2005.79762	60.96524	59.24471	373.96429	2.75276	2.59133	1199.82858	1154.72381	16.32014
	2015-05-31	5029.43147	2111.94352	1953.55000	65.60100	59.20000	359.23750	2.89338	2.85595	1198.21501	1141.61501	16.79985

	2024-01-31	15081.38816	4804.49151	2190.63095	79.19714	106.62000	451.84524	3.81207	2.71500	2028.04285	920.87619	22.86643
	2024-02-29	15808.93501	5011.96143	2172.20000	81.62400	97.22600	423.37500	3.79663	1.79550	2025.21499	891.93000	22.66120
	2024-03-31	16216.29551	5170.57249	2216.12500	84.66550	111.73200	429.50000	3.97643	1.74730	2161.87000	911.55000	24.52955
	2024-04-30	15950.86355	5112.49274	2489.32955	89.00000	117.84682	434.62500	4.36091	1.79123	2332.59090	941.63636	27.49568
	2024-05-31	16330.81318	5198.45162	2521.21154	83.27154	106.95000	452.42308	4.72654	2.27115	2340.45386	1002.95385	28.12438

113 rows × 11 columns

In [147...]: `y_m.isna().sum()`

```
Out[147]: Ticker
NASDAQ      0
SP           0
aluminum    0
brent        0
coal         0
corn         0
copper       0
gaz          0
gold         0
platinum    0
silver       0
dtype: int64
```

```
In [148... # y_m.dropna(how='any', inplace=True)
```

```
y_m.fillna(0, inplace=True)
```

```
/tmp/ipykernel_85268/268557846.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
y_m.fillna(0, inplace=True)
```

```
In [149... # y_m.info()
```

```
# y_m.isna().sum()
y_m.index
```

```
Out[149]: DatetimeIndex(['2015-01-31', '2015-02-28', '2015-03-31', '2015-04-30',
                           '2015-05-31', '2015-06-30', '2015-07-31', '2015-08-31',
                           '2015-09-30', '2015-10-31',
                           ...
                           '2023-08-31', '2023-09-30', '2023-10-31', '2023-11-30',
                           '2023-12-31', '2024-01-31', '2024-02-29', '2024-03-31',
                           '2024-04-30', '2024-05-31'],
                          dtype='datetime64[ns]', name='Date', length=113, freq='M')
```

```
In [150...]: # df дневн знач с пропусками  
y_d
```

Out[150]:	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	copper	gaz	gold	platinum	silver
	Date											
	2015-01-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	2015-01-02	4726.81006	2058.19995	2168.50	56.42	NaN	395.75	2.8385	3.003	1186.00000	1203.00000	15.734
	2015-01-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	2015-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	2015-01-05	4652.56982	2020.57996	2148.25	53.11	63.65	406.00	2.7930	2.882	1203.90002	1210.09998	16.179

	2024-05-13	16388.24023	5221.41992	2493.75	83.36	105.75	458.50	4.8045	2.381	2336.10010	1005.29999	28.221
	2024-05-14	16511.17969	5246.68018	2491.50	82.38	105.25	453.75	4.9535	2.344	2353.39990	1039.30005	28.485
	2024-05-15	16742.39062	5308.14990	2533.50	82.75	105.50	462.50	4.9695	2.416	2388.69995	1063.30005	29.514
	2024-05-16	16698.32031	5297.10010	2529.00	83.27	106.40	457.00	4.8920	2.495	2380.00000	1065.40002	29.665
	2024-05-17	16685.97070	5303.27002	2602.50	83.96	111.00	452.75	5.0825	2.638	2419.80005	1094.69995	31.775

3425 rows × 11 columns

```
In [151...]: y_d.info()  
# y_d.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3425 entries, 2015-01-01 to 2024-05-17
Freq: D
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   NASDAQ      2360 non-null    float64
 1   SP          2360 non-null    float64
 2   aluminum    2322 non-null    float64
 3   brent       2354 non-null    float64
 4   coal         2309 non-null    float64
 5   corn         2356 non-null    float64
 6   copper       2358 non-null    float64
 7   gaz          2359 non-null    float64
 8   gold         2358 non-null    float64
 9   platinum    2341 non-null    float64
 10  silver       2357 non-null    float64
dtypes: float64(11)
memory usage: 321.1 KB
```

In [152...]

```
# df дневн знач без пропусков
y_d1.fillna(0, inplace=True)
y_d1
```

```
/tmp/ipykernel_85268/740862418.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
y_d1.fillna(0, inplace=True)
```

Out[152]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	copper	gaz	gold	platinum	silver
Date											
2015-01-05	4652.56982	2020.57996	2148.25	53.11	63.65	406.00	2.7930	2.882	1203.90002	1210.09998	16.179
2015-01-06	4592.74023	2002.60999	2144.50	51.10	62.70	405.00	2.8040	2.938	1219.30005	1220.80005	16.603
2015-01-07	4650.47021	2025.90002	2165.00	51.15	60.90	396.25	2.7960	2.871	1210.59998	1220.69995	16.510
2015-01-08	4736.18994	2062.13989	2202.75	50.96	60.60	394.25	2.8070	2.927	1208.40002	1221.69995	16.351
2015-01-09	4704.06982	2044.81006	2189.00	50.11	60.15	400.25	2.7905	2.946	1216.00000	1229.09998	16.386
...
2024-05-13	16388.24023	5221.41992	2493.75	83.36	105.75	458.50	4.8045	2.381	2336.10010	1005.29999	28.221
2024-05-14	16511.17969	5246.68018	2491.50	82.38	105.25	453.75	4.9535	2.344	2353.39990	1039.30005	28.485
2024-05-15	16742.39062	5308.14990	2533.50	82.75	105.50	462.50	4.9695	2.416	2388.69995	1063.30005	29.514
2024-05-16	16698.32031	5297.10010	2529.00	83.27	106.40	457.00	4.8920	2.495	2380.00000	1065.40002	29.665
2024-05-17	16685.97070	5303.27002	2602.50	83.96	111.00	452.75	5.0825	2.638	2419.80005	1094.69995	31.775

2253 rows × 11 columns

In [153...]

```
# y_d1.info()
y_d1.isna().sum()
# y_d1.index
```

```
Out[153]: Ticker  
NASDAQ      0  
SP           0  
aluminum    0  
brent        0  
coal         0  
corn          0  
copper       0  
gaz          0  
gold         0  
platinum    0  
silver       0  
dtype: int64
```

```
In [154]: y_d1.asfreq('D')
```

```
Out[154]:   Ticker    NASDAQ      SP  aluminum  brent  coal  corn  copper  gaz  gold  platinum  silver
```

	Date										
2015-01-05	4652.56982	2020.57996	2148.25	53.11	63.65	406.00	2.7930	2.882	1203.90002	1210.09998	16.179
2015-01-06	4592.74023	2002.60999	2144.50	51.10	62.70	405.00	2.8040	2.938	1219.30005	1220.80005	16.603
2015-01-07	4650.47021	2025.90002	2165.00	51.15	60.90	396.25	2.7960	2.871	1210.59998	1220.69995	16.510
2015-01-08	4736.18994	2062.13989	2202.75	50.96	60.60	394.25	2.8070	2.927	1208.40002	1221.69995	16.351
2015-01-09	4704.06982	2044.81006	2189.00	50.11	60.15	400.25	2.7905	2.946	1216.00000	1229.09998	16.386
...
2024-05-13	16388.24023	5221.41992	2493.75	83.36	105.75	458.50	4.8045	2.381	2336.10010	1005.29999	28.221
2024-05-14	16511.17969	5246.68018	2491.50	82.38	105.25	453.75	4.9535	2.344	2353.39990	1039.30005	28.485
2024-05-15	16742.39062	5308.14990	2533.50	82.75	105.50	462.50	4.9695	2.416	2388.69995	1063.30005	29.514
2024-05-16	16698.32031	5297.10010	2529.00	83.27	106.40	457.00	4.8920	2.495	2380.00000	1065.40002	29.665
2024-05-17	16685.97070	5303.27002	2602.50	83.96	111.00	452.75	5.0825	2.638	2419.80005	1094.69995	31.775

3421 rows × 11 columns

```
In [155]: # y_d1.index  
y_d1.isna().sum()
```

```
Out[155]: Ticker
NASDAQ      0
SP           0
aluminum    0
brent        0
coal         0
corn          0
cupper       0
gaz          0
gold         0
platinum    0
silver       0
dtype: int64
```

```
In [156...]: # df для обучения беру без последних 2 значений
i = 2

y_m_train = y_m[:-i]

y_m_train
```

Out[156]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Date											
2015-01-31	4673.70149	2028.17851	2220.80000	49.68000	59.47632	388.10000	2.65322	2.92935	1252.92499	1245.72000	17.20000
2015-02-28	4854.25583	2082.19579	2174.71053	58.44053	61.71176	383.68421	2.62363	2.75495	1224.91052	1199.63684	16.72021
2015-03-31	4938.01045	2079.99044	2056.14773	56.93500	61.54950	382.94318	2.70793	2.74691	1177.76818	1140.41363	16.19436
2015-04-30	4985.95143	2094.86284	2005.79762	60.96524	59.24471	373.96429	2.75276	2.59133	1199.82858	1154.72381	16.32014
2015-05-31	5029.43147	2111.94352	1953.55000	65.60100	59.20000	359.23750	2.89338	2.85595	1198.21501	1141.61501	16.79985
...
2023-11-30	13913.16057	4460.06331	2210.32143	82.02773	121.86190	468.28571	3.72111	3.05000	1985.65000	905.72857	23.52100
2023-12-31	14690.47397	4685.05149	2188.11250	77.32400	118.18250	468.93750	3.85152	2.53885	2031.67499	947.90499	23.91370
2024-01-31	15081.38816	4804.49151	2190.63095	79.19714	106.62000	451.84524	3.81207	2.71500	2028.04285	920.87619	22.86643
2024-02-29	15808.93501	5011.96143	2172.20000	81.62400	97.22600	423.37500	3.79663	1.79550	2025.21499	891.93000	22.66120
2024-03-31	16216.29551	5170.57249	2216.12500	84.66550	111.73200	429.50000	3.97643	1.74730	2161.87000	911.55000	24.52955

111 rows × 11 columns

In [157...]:

```
# df для тестирования беру последние i значений

y_m_test = y_m[-i:]

y_m_test
```

Out[157]:

Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	silver
Date											
2024-04-30	15950.86355	5112.49274	2489.32955	89.00000	117.84682	434.62500	4.36091	1.79123	2332.59090	941.63636	27.49568
2024-05-31	16330.81318	5198.45162	2521.21154	83.27154	106.95000	452.42308	4.72654	2.27115	2340.45386	1002.95385	28.12438

```
In [158]: # y_m_train.index  
y_m_test.index
```

```
Out[158]: DatetimeIndex(['2024-04-30', '2024-05-31'], dtype='datetime64[ns]', name='Date', freq='M')
```

```
In [159]: # прогноз VAR на train для следующих i периодов  
fh=[1, 2]  
forecaster = VAR()  
forecaster.fit(y_m_train, fh=fh) # y_m_train  
y_m_pred = forecaster.predict()  
y_m_pred
```

```
Out[159]:      Ticker    NASDAQ        SP  aluminum   brent     coal     corn  copper     gaz     gold  platinum     silver  
               Date  
2024-04-30  16681.43985  5271.03561  2272.92765  84.52511  95.42934  433.14285  4.04723  1.71961  2213.22116  889.55538  24.65649  
2024-05-31  17137.85507  5364.54903  2323.76050  85.21838  90.07562  437.33493  4.11464  1.56847  2267.85474  873.48619  24.92883
```

```
In [155]: # Прогноз от октября 2022

# Ticker      NASDAQ  SP      aluminum      brent      coal      corn      copper      gaz      gold      platinum      silver
# 2022-10-31   11022.47480    3747.62382    2054.62424    83.10507    298.17192    665.47727    3.3
# 2022-11-30   10653.74568    3639.38105    1907.67892    76.07661    280.64174    645.59887    3.1

# Прогноз от 15 октября 2023:

# Ticker      NASDAQ  SP      aluminum      brent      coal      corn      copper      gaz      gold      platinum      silver
# 2023-02-28   11064.81593    3944.29704    2516.89268    87.82867    164.75180    648.04233    4.0
# 2023-03-31   11149.86778    3922.83904    2463.51579    89.88762    172.29638    622.62755    3.8

# Прогноз от 2023-06-09:
# Ticker      NASDAQ  SP      aluminum      brent      coal      corn      copper      gaz      gold      platinum      silver
# Date
# 2023-05-31   12356.03401    4163.85714    2320.69926    82.47391    139.48830    648.95043    3.9
# 2023-06-30   12605.97197    4196.86109    2282.52453    81.67902    145.56486    641.98883    3.9

# Прогноз от 2024-05-19:
# Ticker      NASDAQ  SP      aluminum      brent      coal      corn      copper      gaz      gold      platinum      silver
# Date
# 2024-04-30   16681.43985    5271.03561    2272.92765    84.52511    95.42934    433.14285    4.0
# 2024-05-31   17137.85507    5364.54903    2323.76050    85.21838    90.07562    437.33493    4.1
```

```
In [160]: df_delta = y_m_test - y_m_pred
df_delta
```

	Ticker	NASDAQ	SP	aluminum	brent	coal	corn	copper	gaz	gold	platinum	silver
	Date											
2024-04-30	-730.57631	-158.54287	216.40190	4.47489	22.41748	1.48215	0.31368	0.07162	119.36973	52.08099	2.83919	
2024-05-31	-807.04190	-166.09741	197.45104	-1.94684	16.87438	15.08815	0.61190	0.70268	72.59912	129.46766	3.19556	

```
In [161... df_delta_pr = (df_delta / y_m_test * 100).round(2)  
df_delta_pr
```

```
Out[161]:    Ticker  NASDAQ   SP  aluminum  brent  coal  corn  copper  gaz  gold  platinum  silver  
              Date  
2024-04-30      -4.58  -3.1       8.69   5.03  19.02   0.34     7.19   4.00   5.12    5.53  10.33  
2024-05-31      -4.94  -3.2       7.83  -2.34  15.78   3.33    12.95  30.94   3.10   12.91  11.36
```

```
In [162... # прогноз VAR на полном df  
  
fh=[1, 2]  
forecaster = VAR()  
forecaster.fit(y_m, fh=fh)  
y_m_pred1 = forecaster.predict()  
y_m_pred1
```

```
Out[162]:    Ticker  NASDAQ       SP  aluminum  brent  coal  corn  copper  gaz  gold  platinum  silver  
              Date  
2024-06-30  16938.02339  5307.90109  2593.02346  88.06541  109.34565  464.94852  4.82841  1.81746  2373.40555  998.39256  28.43420  
2024-07-31  17609.54092  5441.26128  2645.03880  91.95599  111.31175  477.11326  4.91036  1.79664  2402.43148  990.14817  28.76339
```

In [163...]	# Прогноз от октября 2022												
	# Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	sil	
	# 2022-12-31	10095.38815		3653.12931	2207.13735		91.36506		227.72750		656.21026	3.3	
	# 2023-01-31	9622.75128		3525.90529	2129.62842		87.41021		225.74037		628.53211	3.1	
	# Прогноз от 15 октября 2022:												
	# Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	sil	
	# 2023-04-30	11678.61052		3958.30899	2285.67372		81.95777		128.63773		594.41993	3.8	
	# 2023-05-31	11925.07310		3966.62690	2236.61238		80.66609		131.30668		562.18753	3.7	
	# Прогноз от 2023-06-09:												
	# Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	sil	
	# Date												
	# 2023-07-31	13349.03776		4282.96826	2258.71315		73.88578		107.08447		593.86030	3.7	
	# 2023-08-31	13513.72796		4299.25512	2268.67219		72.97452		112.75701		595.76091	3.8	
	# то есть в в июне прогноз на июль: 107.08, август: 112.74												
	# 2023-08-16 факт 115.85, а прогноз на сентябрь 98.13												
	# Ticker	NASDAQ	SP	aluminum	brent				coal	corn	cupper	gaz	gol
	# Date												
	# 2023-09-30	14037.25108		4505.02031	2198.67279		80.36289		98.12955		472.33075	3.7	
	# 2023-10-31	14186.75855		4514.51324	2262.85160		77.19405		91.74047		470.03268	3.7	
	# Прогноз 2024-05-19:												
	# Ticker	NASDAQ	SP	aluminum	brent	coal	corn	cupper	gaz	gold	platinum	sil	
	# Date												
	# 2024-06-30	16938.02339		5307.90109	2593.02346		88.06541		109.34565		464.94852	4.8	
	# 2024-07-31	17609.54092		5441.26128	2645.03880		91.95599		111.31175		477.11326	4.9	

```
In [165...]: print('Факт ст-ть угля: ', (y_m['coal'][-2:]).round(0))
print('----')
print('Прогноз ст-ти угля: ', (y_m_pred1['coal']).round(0))

plt.figure(figsize=(12, 4))
plt.plot(y_m['coal']['2024-01-01':], label='Факт')
plt.plot(y_m_pred['coal'], label='Тест прогноз на факт.периоде')
plt.plot(y_m_pred1['coal'], label='Прогноз на будущ.периоды')
plt.title('Факт и прогноз (методом VAR из sktime) стоимости угля по среднемесячным значениям в долл.')
plt.ylabel('долл.')
plt.legend()
plt.show()
```

Факт ст-ть угля: Date
2024-04-30 118.0
2024-05-31 107.0
Freq: M, Name: coal, dtype: float64

Прогноз ст-ти угля: Date
2024-06-30 109.0
2024-07-31 111.0
Name: coal, dtype: float64



In []:

[Finish](#)[Окончание](#)[К оглавлению](#)

Коммерсант2022-10-27 № 200

Торг уместен, но не очень велик

По мере ослабления глобального роста мировые цены на сырьевые товары также продолжат снижаться, но из-за укрепления доллара США в местных валютах они по-прежнему будут оставаться высокими, что будет оказывать дополнительное давление на инфляцию, говорится в обзоре сырьевых рынков от Всемирного банка (ВБ). Так, стоимость нефти марки Brent с февраля снизилась на 6%, но 60% стран-импортеров нефти и 90% импортеров зерна столкнулись с ростом цен, выраженных в национальных валютах. В ВБ ждут, что цены на энергоресурсы после роста в этом году на 60% снизятся на 11% в 2023-м и еще на 12% в 2024-м, но и после этого останутся на 50% выше, чем в среднем за пять лет. Этот же фактор скажется на увеличении стоимости электроэнергии и повышенных транспортных издержек. Цены на газ в Европе и на уголь, перевозимый по морю, уже превышают средние за пять лет показатели на 420% и 180% — удорожание природного газа также приведет к увеличению потребления угля. При этом средняя цена нефти марки Brent, как ожидается, снизится со \$100 за баррель в 2022 году до \$92 за баррель в 2023-м и \$80 за баррель в 2024-м.

Цены же на неэнергетическое сырье в третьем квартале этого года уже снизились на 13% к предыдущему кварталу, в том числе на ценные металлы — на 9%, на сельхозсырье — на 11% (сейчас они превышают прошлогодний уровень на 9%). По итогам года индекс стоимости неэнергетического сырья, как ожидается, вырастет на 10,5%, в следующем году цены могут снизиться на 8,1%. Так, стоимость металлов после снижения в этом году на 2% по прогнозу ВБ упадет на 15% (на это влияет в том числе спад на рынке недвижимости в Китае), а затем стабилизируется. Цены на продовольствие в этом году вырастут на 13,4% (в том числе на зерно — на 20,6%), а в следующем могут снизиться на 5%, ожидают в банке. Этот прогноз, однако, подвержен значительным рискам как в краткосрочной, так и среднесрочной перспективе, полагают в ВБ. Более высокие цены на энергоресурсы, к примеру, могут поддержать повышение цен и на другое сырье, в частности, на удобрения: их стоимость за год увеличится на 66,1%, в следующем году ожидается ее снижение на 12,4%. Также прогноз предусматривает возвращение на рынок большего объема поставок продовольствия с Украины. Татьяна Едовина

2024-05-19

Сделал еще раз.

