федеральное государственное автономное образовательное учреждение   
высшего образования

«Национальный исследовательский Томский политехнический университет»

|  |  |
| --- | --- |
| Школа / филиал | ИЯТШ ТПУ |
| Обеспечивающее подразделение | ОММФ |
| Направление подготовки / специальность | 01.03.02  Прикладная математика и информатика |
| Образовательная программа (направленность (профиль)) | Прикладная математика в инженерии |

**ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ**

**на тему:**

Исследование динамики цен акций (курсов валют) на Московской бирже методами регрессионного анализа.

Выполнил

Студент группы 0В21 Дзебан А.А.

Проверил  
доцент, ОММФ ИЯТШ Шинкеев М.Л.

Томск 2024

**Цель работы**: построение модели регрессионного анализа, используя метод Фурье. Оценка параметров модели. Предсказание поведения цен акций.

**Задание:**

1. Используя открытые источники (например, finam.ru), провести сбор исходных статистических данных для анализа: получить данные о цене закрытия акций / курсе валют (сlose) на Московской бирже за выбранный период с заданной периодичностью (см. варианты заданий).

Исходя из **варианта 6** задания получим необходимые данные:

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Сбербанк | 01.02.2024 ̶ 30.04.2024 | день |

Получим данные в формате .csv, оставим только цену закрытия и время (возьмем первую запись за момент времени 0 и будем считать от неё) и выведем первые 10 значений:

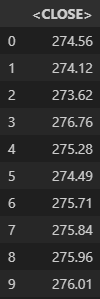


Рисунок 1, данные в виде pd.Dataframe

Выведем график цен за выделенный период:

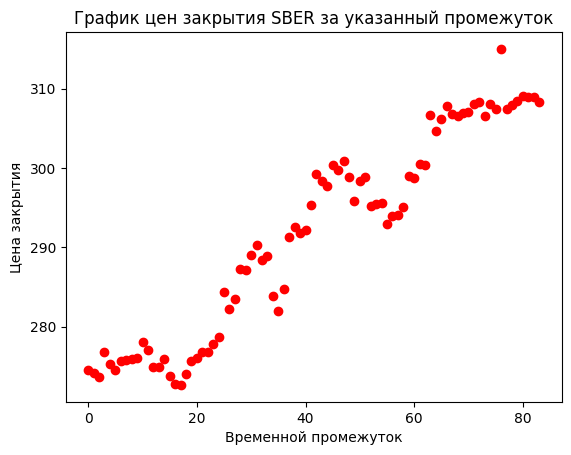


Рисунок 2

Здесь уже заметен стабильный “бычий” тренд, при этом можно ожидать хорошую аппроксимацию простой линейной регрессией.

1. Построить подходящую регрессионную модель вида:

,

характеризующую зависимость средней цены акций y от времени t. Параметры гармоник , нелинейно входящие в модель, найти, используя либо спектральный анализ (последовательно добавляем в модель гармоники с частотами, соответствующие наибольшим амплитудам), либо используя численные методы, минимизируя сумму квадратов остатков (в любом случае в модели должны присутствовать только значимые гармоники).

**Будем действовать следующим образом:**

1. Построим линейную часть для модели регрессии, оценим значимость модели и её параметры. Вычтем линейную часть из отклика и получим остатки, с которыми можно работать для выявления нелинейной части тренда.

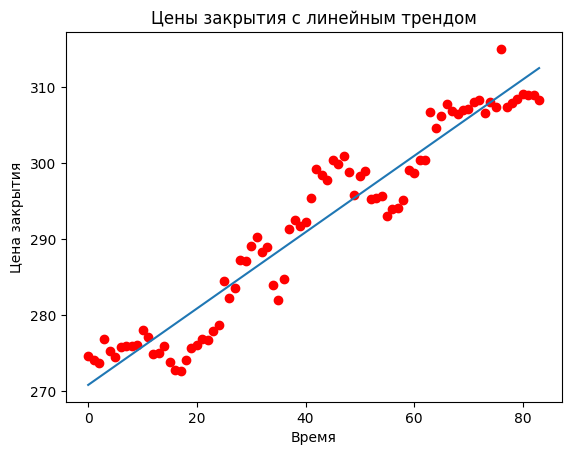
Параметры модели:

Значимость коэффициента модели:

То есть коэффициенты модели являются значимыми,

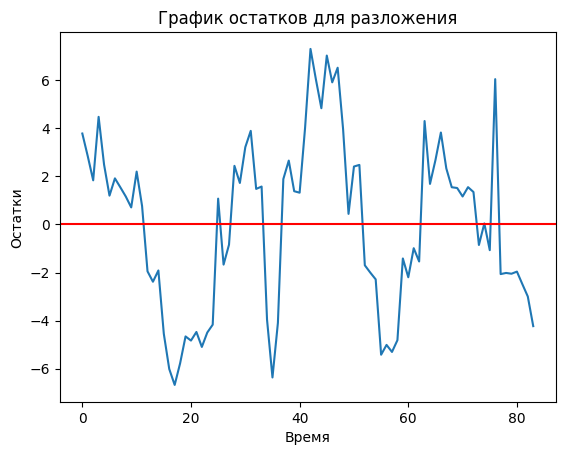
Значимость самой модели:

Построим график цен с закрытия с предсказанной линией тренда:

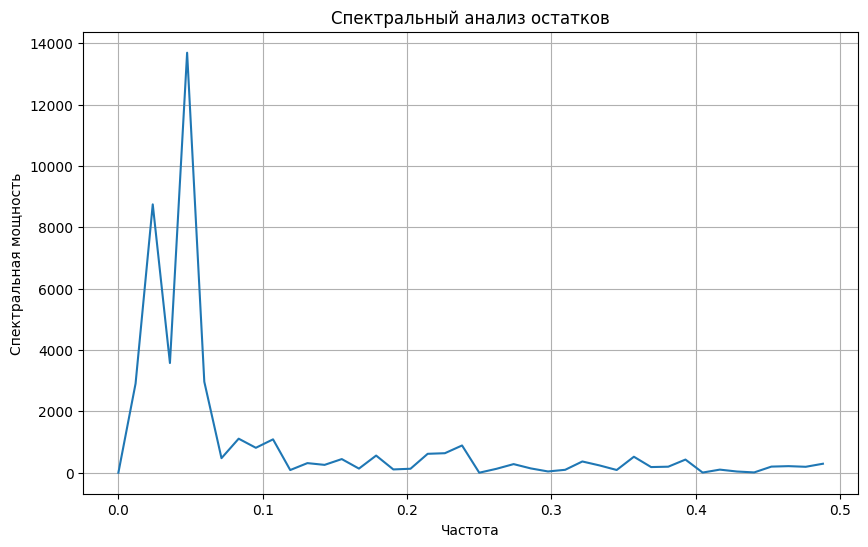


Здесь метрика , то есть даже линейная модель вполне хорошо описывает поведение цены за указанный промежуток.

График поведения остатков (нелинейная часть поведения):



1. Для получения параметров гармоник, воспользуемся методом спектрального анализа, то есть проведем дискретную трансформацию Фурье. Отобразим график положительных частот и спектральной мощности:



Для определения частот будем считать модели от наиболее “мощных” частот к менее мощным. Спектральные частоты, при которых модель оказалась незначимой:

[0.047619047619047616,

0.023809523809523808,

0.03571428571428571,

0.05952380952380952,

0.011904761904761904,

0.08333333333333333,

0.10714285714285714,

0.23809523809523808,

0.09523809523809523,

0.22619047619047616,

0.21428571428571427,

0.17857142857142855]

Для определения коэффициентов будем действовать так:

1. Считаем модель разложения в ряд Фурье на частоте
2. Если коэффициент при косинусе и синусе на частоте значим – добавляем их в модель, если при косинусе или синусе не значим – зануляем соответствующий коэффициент.
3. Продолжаем пока модель перестанет быть значимой.

**Листинг кода и выводы для каждой модели представлены в приложении.**

В моем случае на шаге 12 (12 частоте) модель перестала быть значимой: полученные значения для финальной модели, объясняющей поведение нелинейной части данных:

F-statistic: 3.059

Prob (F-statistic): 0.0523

При этом все коэффициенты, включенные в модель, то есть не обнуленные при вычислении были статистически значимы.

Сопоставим коэффициенты и их статистическую значимость в сводную таблицу:

Обозначим частоты как:

|  |  |
| --- | --- |
| ω1 | 0.047619047619047616 |
| ω2 | 0.023809523809523808 |
| ω3 | 0.03571428571428571 |
| ω4 | 0.05952380952380952 |
| ω5 | 0.011904761904761904 |
| ω6 | 0.08333333333333333 |
| ω7 | 0.10714285714285714 |
| ω8 | 0.23809523809523808 |
| ω9 | 0.09523809523809523 |
| ω10 | 0.22619047619047616 |
| ω11 | 0.21428571428571427 |
| ω12 | 0.17857142857142855 |

Тогда итоговая модель:

3. Оценить коэффициент детерминации, значимость модели по критерию Фишера, остаточную дисперсию, значимость коэффициентов модели ,  (в предположении, что остатки независимые нормальные случайные величины с одинаковой дисперсией, а коэффициенты  известны и равны найденным оценкам).

Метрики модели:

Построим график предсказанных и реальных значений:



Заметно, что модель хорошо\* описывает поведение реальных данных на промежутке, что и видно, исходя из графиков моделей.

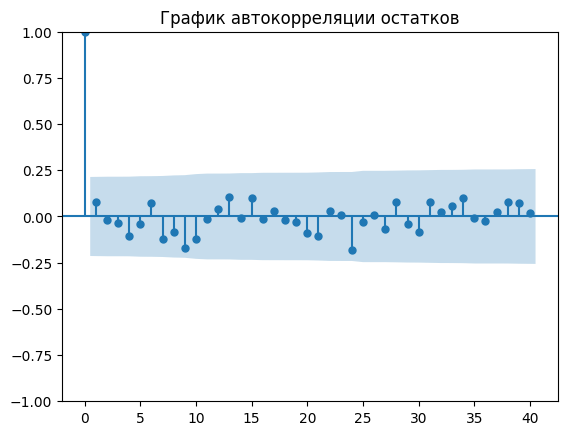
\*не считая малого числа выбросов

4.Исследовать свойства остатков модели (проверить гипотезы о независимости, гомоскедастичности, нормальности), сделать выводы.

Для исследования независимости остатков (отсутствия автокорреляции) подсчитаем статистику Дарбина-Уотсона:

Статистика Дарбина-Уотсона: DW = 1.76616704284849

Здесь при приближении статистики к двум можно сделать вывод об отсутствии автокорреляции – это и наблюдается в нашем случае, то есть остатки можно считать независимыми.



Для исследования остатков на гомоскедастичность будем пользоваться тестом Бреуша-Пагана:

Тест Бреуша-Пагана: LM-статистика = 13.664585725089866, p-value = 0.47498686008856805 >>0.05, то есть нулевую гипотезу о гомоскедастичности остатков следует принять.



Для исследования нормальности остатков воспользуемся тестом Шапиро-Уилка.

Тест Шапиро-Уилка: W-статистика = 0.9747980996084719, p-value = 0.09813935556420914 > 0.05, то есть принимаем гипотезу у нормальности распределения остатков.

**Вывод:** исходя из того, что

А) Остатки обладают малым коэффициентом автокорреляции

Б) Остатки гомоскедастичны

В) Остатки распределены нормально

Г) Модель обладает высоким значением

Использование этой модели можно считать статистически обоснованным.

1. Получить прогноз для средней цены акции (курса валют) на 4 шага вперед, построить доверительные границы для прогноза (стоимости акции / курса валют) и сравнить результат с фактической стоимостью на данный период.

Полученный прогноз на следующие 4 дня:



Построим доверительные границы, изобразим их на графике вместе с реально полученными значениями:



Здесь предсказание оказалось некорректным, поскольку тренд в конце наблюдений потерял свою силу, а обучение производилось на данных, обладающие сильным линейным трендом.

**Вывод:** были исследована модель, построенная по принципу преобразований Фурье, она показала высокую эффективность на обучающей выборке, однако поскольку сами данные имели сильный восходящий тренд, а последующее поведение можно было бы описать как “выход на плато”, то будущие данные были предсказаны не точно.

**Приложение:**

# %%

import numpy as np

import pandas as pd

import statsmodels.api as sm

import matplotlib.pyplot as plt

from scipy.fft import fft, fftfreq

# %%

# Загрузка данных

data = pd.read\_csv("SBER\_240101\_240430.csv", delimiter=";")

data.drop(columns=['<PER>', '<TIME>', '<TICKER>', '<DATE>'], inplace=True)

# %%

# График цен закрытия

plt.plot([i for i in range(len(data))], data["<CLOSE>"].to\_numpy(), 'ro')

plt.xlabel("Временной промежуток")

plt.ylabel("Цена закрытия")

plt.title("График цен закрытия SBER за указанный промежуток")

plt.show()

# %%

# Линейная регрессия

time = np.array([i for i in range(len(data))])

X = np.array([i for i in range(len(data))]).reshape(-1, 1)

y = data['<CLOSE>']

# Модель линейной регрессии

X\_with\_const = sm.add\_constant(X)

sm\_model = sm.OLS(y, X\_with\_const).fit()

print(sm\_model.summary())

# Построение тренда

model\_params = np.array(sm\_model.params)

plt.plot([i for i in range(len(data))], data["<CLOSE>"].to\_numpy(), 'ro',

         [i for i in range(len(data))], [(i \* model\_params[1] + model\_params[0]) for i in range(len(data))])

plt.xlabel("Время")

plt.ylabel("Цена закрытия")

plt.title("Цены закрытия с линейным трендом")

plt.show()

# %%

# Остатки

residuals = y.to\_numpy() - [(i \* model\_params[1] + model\_params[0]) for i in range(len(data))]

plt.plot(time, residuals)

plt.axhline(0, color='red')

plt.xlabel("Время")

plt.ylabel("Остатки")

plt.title("График остатков для разложения")

plt.show()

# Дискретное преобразование Фурье для остатков

N = len(residuals)

frequencies = fftfreq(N)  # Частоты

amplitudes = fft(residuals)  # Преобразование

power = np.abs(amplitudes)\*\*2  # Спектральная мощность

# Отображаем положительные частоты

positive\_frequencies = frequencies[:N // 2]

positive\_power = power[:N // 2]

# Построение спектра

plt.figure(figsize=(10, 6))

plt.plot(positive\_frequencies, positive\_power)

plt.title("Спектральный анализ остатков")

plt.xlabel("Частота")

plt.ylabel("Спектральная мощность")

plt.grid()

plt.show()

# %%

fourier\_params = {}

freq\_params = []

# %%

# Создаем копии массивов, чтобы не изменять оригинальные данные

remaining\_power = positive\_power.copy()

remaining\_frequencies = positive\_frequencies.copy()

# %%

# Цикл для добавления синусоидальных компонент в модель

for step in range(1, 40):

# Находим индекс частоты с максимальной амплитудой

    top\_index = np.argmax(remaining\_power)

    max\_frequency = remaining\_frequencies[top\_index]

    freq\_params.append(max\_frequency)

    # Удаляем использованную частоту и её амплитуду из рассмотрения

    remaining\_power = np.delete(remaining\_power, top\_index)

    remaining\_frequencies = np.delete(remaining\_frequencies, top\_index)

    # Формируем компоненты для текущей частоты

    cos\_component = np.cos(2 \* np.pi \* max\_frequency \* time).reshape(-1, 1)

    sin\_component = np.sin(2 \* np.pi \* max\_frequency \* time).reshape(-1, 1)

    X\_forstep = np.hstack((cos\_component,sin\_component))

    model\_cosandsin = sm.OLS(residuals,X\_forstep).fit()

    print(f"Результаты модели для ряда фурье при шаге {step} и частоте {max\_frequency}:")

    print(model\_cosandsin.summary())

    # Проверка значимости для косинуса

    if model\_cosandsin.pvalues[0] < 0.05 and model\_cosandsin.pvalues[1] <0.05 and model\_cosandsin.f\_pvalue <0.05:

        print(f"\nКоэф при косинусе и синусе {model\_cosandsin.params}  значимы, значимость модели - {model\_cosandsin.f\_pvalue}")

        fourier\_params.update({f"Sin{step} & Cos{step}" :model\_cosandsin.params})

        residuals -= model\_cosandsin.fittedvalues

    elif model\_cosandsin.pvalues[0] >= 0.05 and model\_cosandsin.pvalues[1] < 0.05 and model\_cosandsin.f\_pvalue <0.05:

        model\_cosandsin = sm.OLS(residuals,sin\_component).fit()

        print(model\_cosandsin.summary())

        residuals -= model\_cosandsin.fittedvalues

        fourier\_params.update({f"Sin{step}" :model\_cosandsin.params})

        print(f"\nКоэф при косинусе не значим, при синусе - значим {model\_cosandsin.params}, значимость модели - {model\_cosandsin.f\_pvalue}")

    elif model\_cosandsin.pvalues[0] < 0.05 and model\_cosandsin.pvalues[1] >= 0.05 and model\_cosandsin.f\_pvalue <0.05:

        model\_cosandsin = sm.OLS(residuals,cos\_component).fit()

        fourier\_params.update({f"Cos{step}" :model\_cosandsin.params})

        print(f"\nКоэф при косинусе {model\_cosandsin.params} значим, при синусе - незначим, значимость модели - {model\_cosandsin.f\_pvalue}")

        residuals -= model\_cosandsin.fittedvalues

    elif model\_cosandsin.pvalues[0] >= 0.05 and model\_cosandsin.pvalues[1] >= 0.05 and model\_cosandsin.f\_pvalue < 0.05:

        print(f"Оба коэффициента на шаге {step} оказались не значимыми")

        continue

    else:

        print(f"Модель перестала быть значимой на шаге {step}")

        print(model\_cosandsin.summary())

        break

# %%

fourier\_params

# %%

freq\_params

# %%

len(freq\_params)

# %%

#Итоговая модель:

# %%

cos\_component = np.cos(2 \* np.pi \* max\_frequency \* time).reshape(-1, 1)

sin\_component = np.sin(2 \* np.pi \* max\_frequency \* time).reshape(-1, 1)

# %%

model\_params

# %%

features= []

# %%

# Линейная часть

linear\_trend = model\_params[0] + model\_params[1] \* time

# Инициализация итоговой модели

final\_model = linear\_trend.copy()

# Добавление синусоидальных компонент

for i, freq in enumerate(freq\_params):

    cos\_component = np.cos(2 \* np.pi \* freq \* time)

    sin\_component = np.sin(2 \* np.pi \* freq \* time)

    # Учет параметров из fourier\_params

    if f'Sin{i+1}' in fourier\_params:

        final\_model += fourier\_params[f'Sin{i+1}'][0] \* sin\_component

    if f'Cos{i+1}' in fourier\_params:

        final\_model += fourier\_params[f'Cos{i+1}'][0] \* cos\_component

    if f'Sin{i+1} & Cos{i+1}' in fourier\_params:

        sin\_coeff, cos\_coeff = fourier\_params[f'Sin{i+1} & Cos{i+1}']

        final\_model += sin\_coeff \* sin\_component + cos\_coeff \* cos\_component

# Построение графика итоговой модели

plt.plot(time, data["<CLOSE>"].to\_numpy(), 'ro', label='Исходные данные')

plt.plot(time, final\_model, label='Итоговая модель')

plt.xlabel("Время")

plt.ylabel("Цена закрытия")

plt.title("Итоговая модель: линейный тренд + синусоидальные компоненты")

plt.legend()

plt.show()

# %%

# Создание матрицы признаков

X\_final = sm.add\_constant(time)  # Константа и линейный тренд

# Добавление синусоидальных компонент

for i, freq in enumerate(freq\_params):

    cos\_component = np.cos(2 \* np.pi \* freq \* time)

    sin\_component = np.sin(2 \* np.pi \* freq \* time)

    # Учет параметров из fourier\_params

    if f'Sin{i+1}' in fourier\_params:

        X\_final = np.column\_stack((X\_final, sin\_component))

    if f'Cos{i+1}' in fourier\_params:

        X\_final = np.column\_stack((X\_final, cos\_component))

    if f'Sin{i+1} & Cos{i+1}' in fourier\_params:

        X\_final = np.column\_stack((X\_final, sin\_component, cos\_component))

# Подгонка модели

final\_ols\_model = sm.OLS(y, X\_final).fit()

# Вывод summary

print(final\_ols\_model.summary())

# %%

# Получение предсказанных значений

predicted\_values = final\_ols\_model.predict(X\_final)

# Построение графика

plt.figure(figsize=(10, 6))

plt.plot(time, y, 'ro', label='Реальные значения')  # Реальные значения

plt.plot(time, predicted\_values, 'b-', label='Предсказанные значения')  # Предсказанные значения

plt.xlabel("Время")

plt.ylabel("Цена закрытия")

plt.title("Сравнение реальных и предсказанных значений")

plt.legend()

plt.grid()

plt.show()

# %%

from statsmodels.stats.stattools import durbin\_watson

# Вычисление статистики Дарбина-Уотсона

dw\_test = durbin\_watson(final\_ols\_model.resid)

print(f"Статистика Дарбина-Уотсона: {dw\_test}")

# %%

#Здесь значение близкое к 2 указывает на отсутствие автокорреляции, то есть остатки можно считать независимыми

# %%

from statsmodels.graphics.tsaplots import plot\_acf

# График автокорреляции остатков

plot\_acf(final\_ols\_model.resid, lags=40)

plt.title("График автокорреляции остатков")

plt.show()

# %%

from statsmodels.stats.diagnostic import het\_breuschpagan

# Тест Бреуша-Пагана

bp\_test = het\_breuschpagan(final\_ols\_model.resid, final\_ols\_model.model.exog)

print(f"Тест Бреуша-Пагана: LM-статистика = {bp\_test[0]}, p-value = {bp\_test[1]}")

# %%

#Принимаем нулевую гипотезу о гомоскедастичности (то есть дисперсия является постоянной)

# %%

# График остатков vs предсказанные значения

plt.scatter(final\_ols\_model.fittedvalues, final\_ols\_model.resid)

plt.axhline(y=0, color='r', linestyle='--')

plt.xlabel("Предсказанные значения")

plt.ylabel("Остатки")

plt.title("Остатки vs Предсказанные значения")

plt.show()

# %%

from scipy.stats import shapiro

# Тест Шапиро-Уилка

shapiro\_test = shapiro(final\_ols\_model.resid)

print(f"Тест Шапиро-Уилка: W-статистика = {shapiro\_test[0]}, p-value = {shapiro\_test[1]}")

# %%

#Данные распределены нормально (P-value > 0.05)

# %%

import statsmodels.api as sm

# QQ-plot остатков

sm.qqplot(final\_ols\_model.resid, line='s')

plt.title("QQ-plot остатков")

plt.show()

# %%

#Поскольку остатки независимы, гомоскедастичны и нормально распределены, а так же R^2 = 0.986, то есть использование модели можно считать более чем обоснованным.

# %%

# Шаг 1: Создание данных для прогноза

future\_time = np.arange(1, len(time) + 4)  # Временные метки для 4 шагов вперед

X\_future = sm.add\_constant(future\_time)  # Константа и линейный тренд

# Добавление синусоидальных компонент для будущих временных меток

for i, freq in enumerate(freq\_params):

    cos\_component = np.cos(2 \* np.pi \* freq \* future\_time)

    sin\_component = np.sin(2 \* np.pi \* freq \* future\_time)

    if f'Sin{i+1}' in fourier\_params:

        X\_future = np.column\_stack((X\_future, sin\_component))

    if f'Cos{i+1}' in fourier\_params:

        X\_future = np.column\_stack((X\_future, cos\_component))

    if f'Sin{i+1} & Cos{i+1}' in fourier\_params:

        X\_future = np.column\_stack((X\_future, sin\_component, cos\_component))

# Шаг 2: Прогнозирование

predictions = final\_ols\_model.get\_prediction(X\_future)

predicted\_values = predictions.predicted\_mean  # Точечный прогноз

confidence\_intervals = final\_ols\_model.conf\_int()  # Доверительные интервалы

# %%

coefficients = final\_ols\_model.params  # Точечные оценки коэффициентов

# Шаг 2: Вывод результатов

results\_df = pd.DataFrame({

    'Коэффициент': coefficients,

    'Нижняя граница': confidence\_intervals[0],

    'Верхняя граница': confidence\_intervals[1]

})

print("Коэффициенты и их доверительные интервалы:")

print(results\_df)

# %%

actual\_values = y

# %%

data2topred = pd.read\_csv('SBER\_240424\_240428.csv', delimiter=',')

realval4 = data2topred["<CLOSE>"].to\_numpy()

data2topred.head()

# %%

# Шаг 4: Визуализация

plt.figure(figsize=(10, 6))

plt.plot(time, y.to\_numpy(), 'ro', label='Исторические данные')  # Исторические данные

plt.plot(future\_time, predicted\_values, 'bo-', label='Прогноз')  # Прогноз

plt.plot(future\_time[-5:], realval4, label = 'Будущие данные')

plt.xlabel("Время")

plt.ylabel("Цена закрытия")

plt.title("Прогноз на 4 шага вперед")

plt.legend()

plt.grid()

plt.show()

# %%

#Здесь не получилось достоврено предсказать (тренд потерял свою силу)

# %%

data.head(10)

# %%

len(time)

# %%

future\_time = np.arange(len(time), len(time) + 4)  # Временные метки для 4 шагов вперед

X\_future = sm.add\_constant(future\_time)  # Константа и линейный тренд

# Добавление синусоидальных компонент для будущих временных меток

for i, freq in enumerate(freq\_params):

    cos\_component = np.cos(2 \* np.pi \* freq \* future\_time)

    sin\_component = np.sin(2 \* np.pi \* freq \* future\_time)

    if f'Sin{i+1}' in fourier\_params:

        X\_future = np.column\_stack((X\_future, sin\_component))

    if f'Cos{i+1}' in fourier\_params:

        X\_future = np.column\_stack((X\_future, cos\_component))

    if f'Sin{i+1} & Cos{i+1}' in fourier\_params:

        X\_future = np.column\_stack((X\_future, sin\_component, cos\_component))

# Шаг 2: Прогнозирование

predictions = final\_ols\_model.get\_prediction(X\_future)

predicted\_values = predictions.predicted\_mean  # Точечный прогноз

confidence\_intervals = predictions.conf\_int()  # Доверительные интервалы

# Шаг 3: Визуализация

plt.figure(figsize=(10, 6))

plt.plot(time, y, 'ro', label='Исторические данные')  # Исторические данные

plt.plot(future\_time, predicted\_values, 'bo-', label='Прогноз')  # Прогноз

plt.fill\_between(future\_time, confidence\_intervals[:, 0], confidence\_intervals[:, 1], color='blue', alpha=0.2, label='Доверительный интервал')  # Доверительный интервал

plt.plot([84,85,86,87,88], realval4, label = 'Будущие данные')

plt.xlabel("Время")

plt.ylabel("Цена закрытия")

plt.title("Прогноз на 4 шага вперед с доверительными интервалами")

plt.legend()

plt.grid()

plt.show()