МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация

БАКАЛАВРСКАЯ РАБОТА

на тему **Сравнительный анализ методов прогнозирования временных рядов**

Студент Т.А. Ионова _____
(И.О. Фамилия) (личная подпись)
Руководитель М.А. Тренина _____
(И.О. Фамилия) (личная подпись)
Консультанты Н.В. Андрюхина _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

Аннотация

Темой данной бакалаврской работы является «Сравнительный анализ методов прогнозирования временных рядов».

Работа выполнена студентом Тольяттинского государственного университета, института математики, физики и информационных технологий, группы ПМИб-1501, Ионовой Татьяной Алексеевной.

Объект работы: задача прогнозирования временных рядов.

Предмет исследования: методы решения задачи прогнозирования временных рядов.

Цель работы: провести сравнительный анализ различных моделей прогнозирования временных рядов.

Для достижения цели работы необходимо решить следующие задачи:

1) Рассмотреть основные методы сглаживания и прогнозирования временных рядов.

2) Реализовать распространенные и востребованные методы на языке Python.

4) Сделать сравнительный анализ реализованных методов при помощи выбранных метрик.

Отчет состоит из введения, трех глав и заключения.

В первой главе представлены основные определения и методы прогнозирования временных рядов.

Во второй главе описано проектирование моделей прогнозирования временных рядов.

В третьей главе проводится тестирование реализованных методов и анализ результатов.

Бакалаврская работа представлена на 62 страницах, включает 27 иллюстраций, 4 приложения, список используемой литературы содержит 16 источников.

ABSTRACT

The title of the bachelor's thesis is «Comparative Analysis of Methods for Forecasting Time Series».

The object of this work is the problem of forecasting time series.

The subject of this work is comparative analysis of implementations of algorithms for solving the forecasting time series problem in the programming language Python.

The forecasting time series is an important task that arises in different fields of science and technology: economy and finances, analysis of political and social actions.

We start with the statement of the problem and then follow through with its possible solutions and give full coverage to methods for analysis and forecasting time series.

Comparison of algorithms in this graduation work is based on the results of simulation experiments for various algorithms.

The first part of the work describes the main definition and methods for forecasting time series, for example exponential smoothing, autoregression, moving average.

The second part of the thesis focuses on implementation of basic methods for forecasting time series.

The third part of the work describes the test results and analysis of methods that have been implemented.

The graduation project consists of an explanatory note on 62 pages, including 27 figures, the list of 16 references including 7 foreign sources and 4 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проблема предсказывания будущего волнует уже не первое поколение людей. Анализ ориентированных на время данных и прогнозирование будущих значений временного ряда являются одними из наиболее важных проблем, с которыми сталкиваются аналитики во многих областях, начиная от финансов и экономики, управления производственными операциями, до анализа политических и социальных действий, для исследования влияния людей и политических решений на окружающую среду [2].

Изменения экономики страны, курса валюты, прибыли компании – всё это требует тщательного анализа при выборе адекватной модели для построения прогноза. К настоящему времени существует множество методов, которые используют данные из прошлого для построения прогноза в будущем.

Прогнозирование – одна из наиболее значимых основ для эффективного развития как предприятий и отраслей, так и экономики региона или страны в целом. Вовремя полученный достоверный прогноз дает возможность избегать ситуаций, которые могут негативно повлиять на состояние экономики, или смягчить их отрицательное воздействие [1, 2].

Правильный выбор метода, оценка его достоверности и экономическая интерпретация результатов прогнозов требуют знакомства с методологией прогнозирования и возможностями конкретного метода.

Актуальность выпускной квалификационной работы обусловлена необходимостью выбора и оценки модели прогнозирования временных рядов.

Объектом исследования является задача прогнозирования временных рядов.

Предмет исследования: методы решения задачи прогнозирования временных рядов.

Цель работы: провести сравнительный анализ различных моделей прогнозирования временных рядов.

Для достижения цели работы необходимо решить следующие задачи:

1) Рассмотреть основные методы сглаживания и прогнозирования временных рядов.

2) Реализовать распространенные и востребованные методы на языке программирования Python.

3) Сделать сравнительный анализ реализованных методов при помощи выбранных метрик.

1 ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

1.1 Основные понятия

Временной ряд – совокупность наблюдений за одной или несколькими величинами за определенный промежуток времени [7]. Примером временного ряда могут выступать:

- количество осадков за день;
- изменение курса доллара за год;
- прибыль компании за последние 5 лет.

Временные ряды могут содержать два вида компонент – систематическую (тренд, сезонность и цикличность) и случайную составляющие [7].

Тренд – систематическая линейная или нелинейная компонента, изменяющаяся во времени. Пример изображен на рисунке 1.1.

Рисунок 1.1 – Пример временного ряда с восходящим трендом

Сезонность – периодические колебания уровней ряда внутри года, изменяющееся регулярно в течение заданного периода (рисунок 1.2).

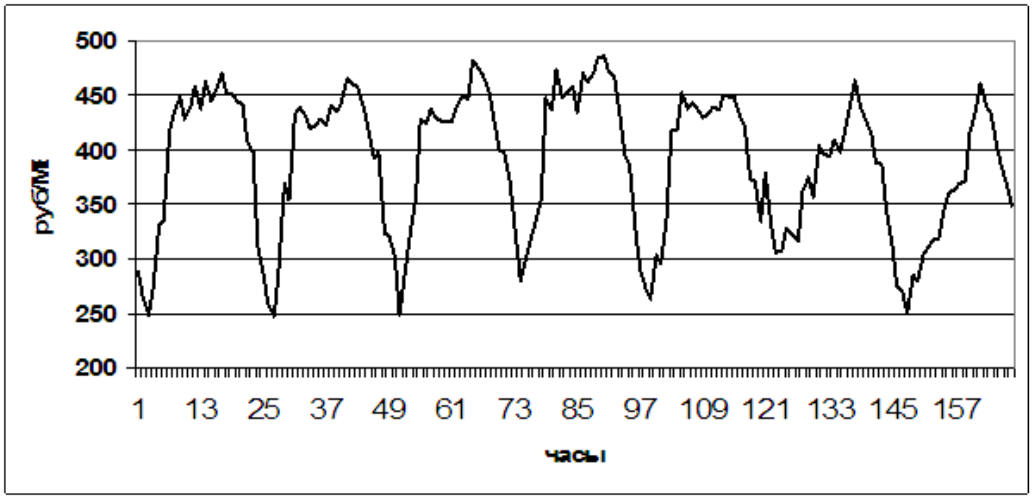

Рисунок 1.2 – Пример временного ряда со сезонной компонентой

Цикличность – периодические колебания временного ряда, выходящие за рамки одного года. Длиной цикла называют промежуток времени между двумя соседними впадинами или вершинами в масштабах года.

Случайная составляющая – случайный шум или ошибка, которая воздействует на временной ряд нерегулярно (рисунок 1.3).

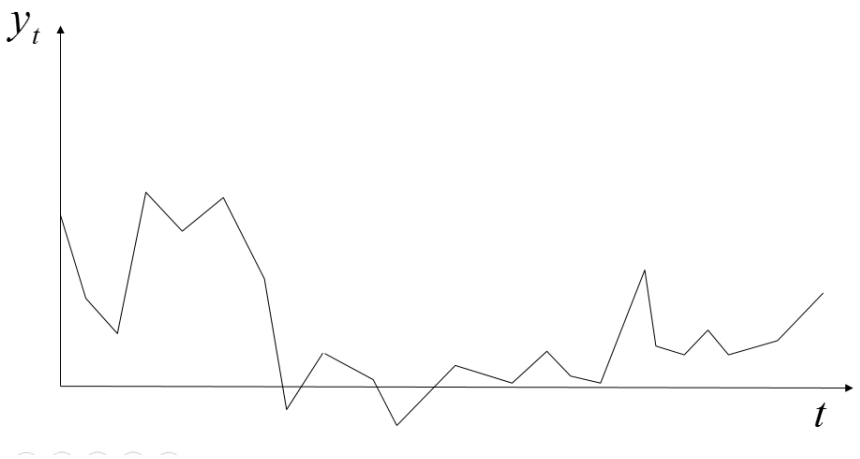

Рисунок 1.3 – Пример временного ряда со случайной составляющей

Автокорреляция элементов временного ряда – корреляционная зависимость между последовательными элементами временного ряда. Возникает тогда, когда каждое соседнее значение связано с предыдущим.

Лаг – величина сдвига между элементами временного ряда.

Лаговый оператор *B* принимает на вход значение элемента временного ряда и возвращает элемент, на 1 единицу времени смещенный в прошлое.

При повторном использовании лагового оператора, значение сдвигается на несколько временных шагов.

$$By_t = y_{t-1},$$
$$B\ By_t = B^2 y_t = y_{t-2}, \quad (1.1)$$
$$B^p y_t = y_{t-p}.$$

Временные ряды характеризуются несколькими величинами:

1) Математическое ожидание – среднее значение случайной величины при стремлении количества ее измерений (количества испытаний) к бесконечности.

2) Дисперсия – разброс значений случайной величины относительно ее математического ожидания.

3) Автокорреляционная функция – последовательность коэффициентов автокорреляции с лагами = 1, 2, 3, …

В общем случае, временные ряды можно разделить на два вида:

1) Стационарные временные ряды. Временные ряды, математическое ожидание и дисперсия которых равны постоянному значению (константе), а значение автокорреляционной функции зависит лишь от длины лага. Пример такого ряда изображен на рисунке 1.4.

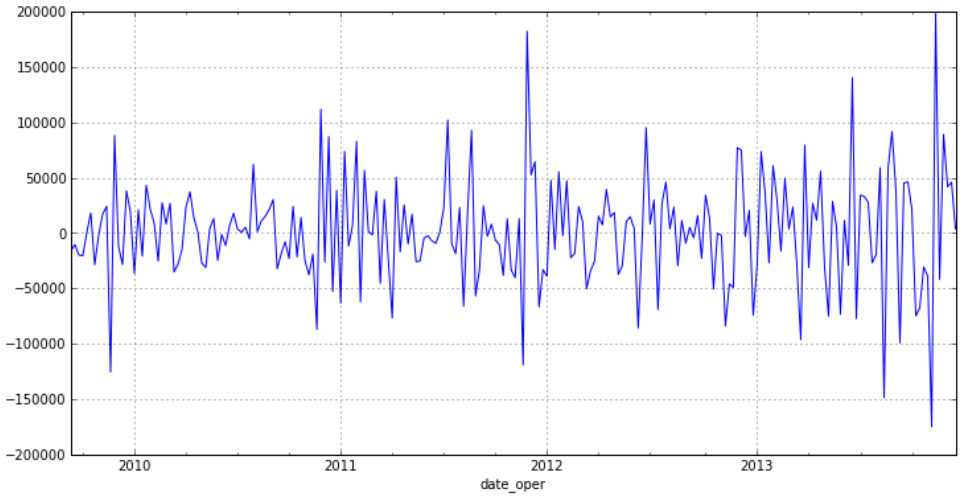

Рисунок 1.4 – Пример стационарного ряда

2) Нестационарные временные ряды. Ряд, отличающийся от стационарного наличием неслучайной составляющей (рисунок 1.5).

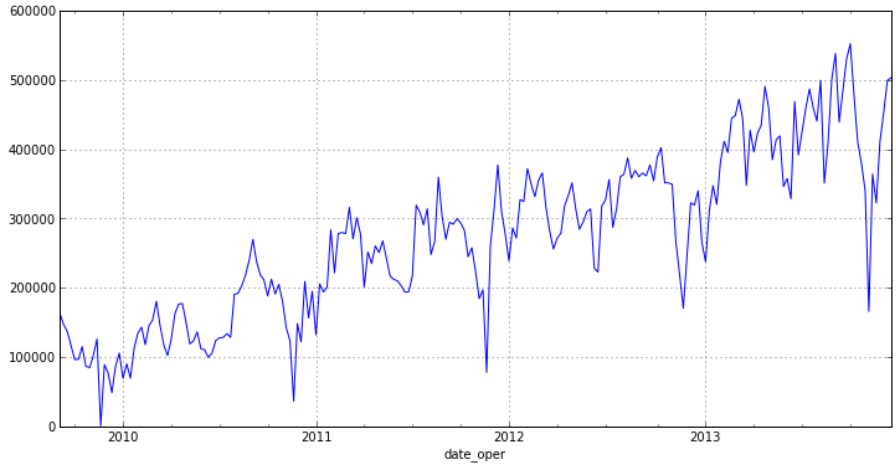

Рисунок 1.5 – Пример нестационарного временного ряда

Для приведения нестационарного ряда к стационарному можно применить один из методов:

1) Если нестационарный временной ряд подвержен экспоненциальному росту, то к нему можно применить простое логарифмирование:

$$y_t^* = \ln\ y_t\ , \tag{1.2}$$

Или логарифмирование цепных индексов:

$$y_t^* = \ln\ \frac{y_t}{y_{t-1}}\ = \ln y_t - \ln y_{t-1}. \tag{1.3}$$

2) Также иногда применяется расчет темпов прироста:

$$y_t^* = \frac{y_t - y_{t-1}}{y_{t-1}} = \frac{y_t}{y_{t-1}} - 1. \tag{1.4}$$

3) Чаще всего для преобразования стационарных рядов применяют интегрирование или взятие конечных разностей различных порядков временного ряда [7, 9]. Интегрирование первого порядка имеет вид:

$$\Delta y_t = y_t - y_{t-1}. \tag{1.5}$$

Интегрирование порядка *d* можно записать в виде уравнения:

$$\Delta^d y_t = \Delta^{d-1} y_t - \Delta^{d-1} y_{t-1}. \tag{1.6}$$

Анализ временных рядов включает в себя методы анализа временных рядов, для извлечения значимых (полезных) структур.

Прогнозирование временных рядов – использование модели для предсказания будущих значений временного ряда на основании предыдущих.

Уровнем временного ряда называются наблюдения $y_t(t = 1,2,3, \dots, n)$, из которых состоит данный ряд.

Остатки временного ряда – разница между фактическим и предсказанным значением. С помощью остатков можно оценить метод прогнозирования. Хороший метод прогнозирования имеет остатки, обладающие следующими свойствами:

- Остатки не коррелируют друг с другом, то есть в них не осталась полезная информация, которая может быть использована для улучшения прогноза;
- Остатки имеют нулевое среднее значение, то есть прогноз не является смещенным.

Важно проанализировать исходный временной ряд и выбрать наиболее подходящую модель для его построения, так как не существует универсальной модели, подходящей для всех типов рядов.

1.2 Модели экстраполяции на основе кривых роста

Среди компонент временного ряда чаще всего исследуется тренд, так как он позволяет делать краткосрочные и долгосрочные прогнозы. Выделяют несколько видов моделей тренда [1, 6, 9].

Пусть y_t – уровень (значение) временного ряда в t момент времени, а X_t – момент времени ($X_t = t = (0, 1, 2, \dots, n)$.

Модель линейного тренда – простейшая модель, применяемая для прогнозирования и имеет вид:

$$y_t = b_0 + b_1 * X_t, \qquad (1.7)$$

где b_i – параметры модели.

Данная модель имеет вид прямой линии и используется для описания процессов, развивающихся во времени однородно [17].

Модель полиномиального тренда – простейший нелинейный тренд, имеющий вид:

$$y_t = b_0 + b_1 * X_t + b_2 * X_t^2 + \cdots + b_p * X_t^p, \qquad (1.8)$$

где p – порядок полинома.

Стоит заметить, что при $p = 1$ модель полиномиального тренда принимает вид линейного тренда.

Коэффициента многочленов невысоких степеней имеют конкретную интерпретацию в зависимости от содержания временного ряда. Например:

b_0 – уровень ряда при $t = 0$, b_1 – скорость роста, b_2 – ускорение (его значение равно половине ускорения значения показателя), b_3 – изменение ускорения.

Для нахождения оценок коэффициентов моделей можно применить метод наименьших квадратов, суть которого состоит в нахождении таких параметров, при которых сумма квадратов отклонений расчетных значений уровней от фактических была бы минимальной, то есть оценки параметров находятся в результате минимизации выражения:

$$\sum_{t=1}^{n} (y_t - y_t)^2 \to min, \qquad (1.9)$$

где n – длина временного ряда;

y_t – фактические значения уровней временного ряда;

y_t – расчетные значения.

Система нормальных уравнений для определения параметров модели выглядит следующим образом:

$$\begin{matrix} b_0 n + b_1 & t + b_2 & t^2 + \cdots + b_p & t^p = & y_t \\ b_0 & t + b_1 & t^2 + b_2 & t^3 + \cdots + b_p & t^{p+1} = & y_t * t \\ b_0 & t^{p-1} + b_1 & t^p + b_2 & t^{p+1} + \cdots + b_p & t^{2p-1} = & y_t * t^{p-} \\ & & \cdots\cdots\cdots\cdots & & & \\ b_0 & t^p + b_1 & t^{p+1} + b_2 & t^{p+2} + \cdots + b_p & t^{2p} = & y_t * t^p \end{matrix} \qquad (1.10)$$

В экономических исследованиях обычно применяют полиномы не выше третьего порядка, так как полиномы более высоких степеней отражают случайные отклонения [1, 5].

Гиперболическая модель тренда используется для описания убывающих процессов, имеющих асимптоту при $y = 0$. Записывается в виде выражения:

$$y_t = b_0 + \frac{b_1}{X_t}. \tag{1.11}$$

Тогда система нормальных уравнений для оценки параметров b, будет иметь вид:

$$\begin{matrix} nb_0 + b_1 & \frac{1}{X_t} = & y_t \\ b_0 & \frac{1}{X_t} + b_1 & \frac{1}{X_t^2} = & \frac{1}{X_t} * y_t \end{matrix}. \tag{1.12}$$

Часто результат экстраполяции прогнозируемых значений ряда выполняется не точечным, а интервальным прогнозом. Для этого определяется доверительный интервал прогноза по формуле:

$$y_{n+l} \pm t_\alpha S_y \sqrt{\frac{n+1}{n} + \frac{t_l - t\ ^2}{(t-t)^2}}, \tag{1.13}$$

где S_y – среднее квадратическое отклонение фактических данных от расчетных

$$S_y = \sqrt{\frac{(y_t - y_t)^2}{n-k}}; \tag{1.14}$$

t_α – значение t-статистики Стьюдента;

k – число оцениваемых параметров кривой;

t – порядковый номер уровня ряда;

t – порядковый номер уровня, стоящего в середине ряда ($t = \frac{n+1}{2}$).

Несмотря на простоту данных моделей, они часто используются для прогнозирования сложных процессов. Полученные прогнозы практически

всегда надежны, если модель тренда адекватно отражает основную тенденцию развития.

1.3 Адаптивные методы прогнозирования

Главная особенность адаптивных методов прогнозирования – это их способность учитывать изменения изучаемых процессов и подстраиваться под них [4, 5].

Идея адаптивных методов состоит в возможности построения корректирующих моделей, которые способны учитывать прогноз, сделанный на предыдущем шаге [8, 13]. Параметр адаптации (сглаживания) описывает быстроту реакции модели на изменение динамики временного ряда. Благодаря этому, адаптивные методы часто используются при оперативном и краткосрочном прогнозировании. Наиболее известными являются три типа моделей:

- Однопараметрическая модель Брауна (экспоненциальное сглаживание);
- Двухпараметрическая модель Хольта (двойное экспоненциальное сглаживание);
- Трехпараметрическая модель Хольта-Винтерса (тройное экспоненциальное сглаживание).

Рассмотрим данные модели подробнее.

Модель Брауна или экспоненциальное сглаживание. Сущность данной модели заключается в том, что временной ряд сглаживается с помощью взвешенной скользящей средней, в которой веса подчиняются экспоненциальному закону (более ранним наблюдениям придаются меньшие веса, чем более поздним) [7].

Сглаживание можно рассматривать как метод, позволяющий выделить из данных, содержащих шумы, полезную информацию (рисунок 1.6).

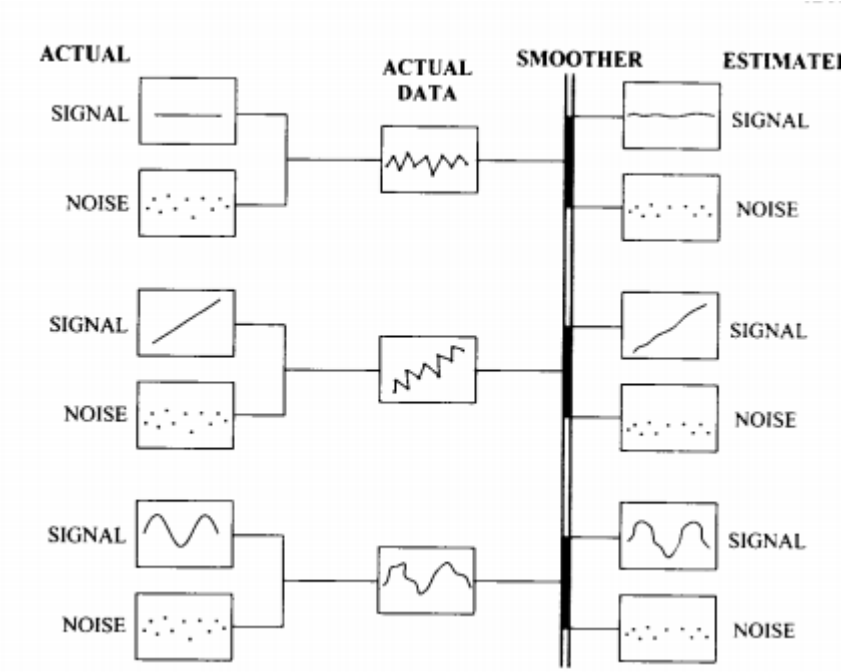

Рисунок 1.6 – Процесс сглаживания данных

Модель Брауна предусматривает постоянное обновление модели за счет новых данных. Прогноз можно представить в виде выражения:

$$y_{t+1} = \alpha y_t + \ 1 - \alpha \ y_t, \qquad (1.15)$$

где y_{t+1} – прогнозируемое значение;

α – параметр сглаживания ($0 < \alpha < 1$);

y_t – фактическое значение ряда в момент времени t;

y_t – предыдущий прогноз.

Раскрыв скобки в выражении 1.15, получим:

$$y_{t+1} = y_t + \alpha(y_t - y_t). \qquad (1.16)$$

Выражение 1.16 можно интерпретировать таким образом: экспоненциальное сглаживание есть ни что иное, как старый прогноз с некоторым уточнением α, умноженное на ошибку последнего прогноза.

В качестве начального значения y_0 можно взять либо среднее арифметическое уровней ряда, либо первое значение временного ряда [7, 8].

Выбор параметра сглаживания α является отдельной задачей и зависит от исходного ряда. Если требуется, чтобы прогнозные значения были стабильны и случайные отклонения сглаживались, в качестве параметра α следует взять малое значение. Наоборот, если нужно, чтобы модель резко

реагировала на изменения динамики временного ряда, параметр α следует брать большим. В отдельных случаях Браун предложил формулу для определения параметра сглаживания:

$$\alpha = \frac{2}{m+1}, \tag{1.17}$$

где m – число наблюдений, входящих в интервал сглаживания (в случае коротких временных рядов можно взять $m = n$ (длина ряда)).

Для поиска оптимального параметра сглаживания можно использовать критерий минимума среднего квадрата ошибок. Сначала выбирается α со значением близким к 0, затем с некоторым шагом, например 0.1, берутся следующие значения (0.1, 0.2, 0.3, …., 0.9). Для каждого значения рассчитывается средний квадрат ошибок и выбирается α с минимальным значением ошибки.

Для построения прогноза на *l* шагов вперед, применяется формула:

$$y_{t+l} = y_{t+l-1}. \tag{1.18}$$

Данная модель не учитывает линейный тренд, поэтому была придумала модификация модели Брауна – модель Хольта (двойное экспоненциальное сглаживание).

Модель Хольта учитывает локальный линейный тренд. Если в исходном временном ряду есть тенденция к росту, то наряду с оценкой текущего уровня нужна и оценка наклона [6]. Данная модель имеет три отдельных уравнения, которые работают вместе, чтобы сформировать прогноз.

Первое уравнение – базовое уравнение сглаживания, которое сглаживает исходный временной ряд:

$$a_t = \alpha_1 y_t + \ 1 - \alpha_1 \ \ a_{t-1} + b_{t-1} \ . \tag{1.19}$$

Второе уравнение – тренд, который обновляется с течением времени и выражается в виде разницы между двумя сглаженными значениями:

$$b_t = \alpha_2 \ a_t - a_{t-1} \ + \ 1 - \alpha_2 \ b_{t-1}. \tag{1.20}$$

Последнее уравнение используется для прогнозирования на l шагов вперед и совмещает в себе значения двух предыдущих уравнений:

$$y_{t+l} = a_t + lb_t. \quad (1.21)$$

Подбор параметров сглаживания α_1 и α_2 выполняется так же как и в модели Брауна.

Начальные значения a_0 и b_0 находятся из уравнения линейной регрессии:

$$y_{t+1} = c_0 + c_1 t, \quad (1.22)$$

где $a_0 = c_0, b_0 = c_1$

Данная модель учитывает тренд, но не сезонность. Для построения модели с учетом тренда и сезонности, используется метод тройного экспоненциального сглаживания или модель Хольта-Винтерса.

Модель Хольта-Винтерса использует три составляющие временного ряда: значение, тренд и сезонность, и является улучшением модели Хольта. Данная модель делает предсказания, комбинируя эффекты всех трех состояний и представляет собой четыре уравнения [12, 19].

1) Уравнение сглаживания ряда:

$$L_t = \alpha \frac{y_t}{S_{t-s}} + \ 1 - \alpha \ \ L_{t-1} + T_{t-1} \ . \quad (1.23)$$

2) Оценка тренда:

$$T_t = \beta \ L_t - L_{t-1} \ + \ 1 - \beta \ T_{t-1}. \quad (1.24)$$

3) Оценка сезонности:

$$S_t = \gamma \frac{y_t}{L_t} + \ 1 - \gamma \ S_{t-s}. \quad (1.25)$$

4) Прогноз на l периодов вперед:

$$y_{t+l} = \ L_t \ + lT_t \ S_{t-s+l}, \quad (1.26)$$

где s – период сезонности (4 для квартальных данных, 12 для годовых);

α, β, γ – параметры сглаживания.

Для построения модели нужно выбрать параметры сглаживания α, β, γ, которые можно найти путем минимизации средней квадратической ошибки. Начальные значения L_0, T_0 можно выбрать из уравнения линейной регрессии, как в модели Хольта, а значение S_0 принять равным единице [7].

Модели экспоненциального сглаживания обладают способностью хорошо предсказывать значения и являются легко настраиваемыми.

1.4 Модель авторегрессии

В модели регрессии, мы делаем прогноз, используя линейную комбинацию предикторов – признаков $X_t (t = 0,1, \dots, n)$.

Модель авторегрессии же прогнозирует значение y в момент времени t, используя прошлые наблюдения из $t-1, t-2, \dots$ момента времени. То есть, мы предполагаем, что прогнозируемое значение имеет связь с предыдущими наблюдениями. Термин авторегрессия указывает на то, что это регрессия переменной против себя самой [7].

Обычно модель авторегрессии обозначают как *AR(p)*, где p – порядок авторегрессии.

Модель авторегрессии порядка p имеет вид:

$$y_t = c + \varphi_1 y_{t-1} + \cdots + \varphi_p y_{t-p} + \varepsilon_t, \qquad (1.27)$$

где φ – коэффициенты авторегрессии;

c – константа (можно взять равной 0 для упрощения).

ε_t – белый шум (непрерывный во времени случайный процесс, математическое ожидание и дисперсия которого равны 0).

Используя лаговый оператор B уравнение 1.27 можно записать в виде:

$$1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p \;\; y_t = \varepsilon_t. \qquad (1.28)$$

Для оценки параметров авторегрессии чаще всего используется метод наименьших квадратов, который минимизирует сумму квадратов отклонений:

$$\sum_{t=p+1}^{n}\left(y_t - \sum_{j=1}^{p}\varphi_j y_{t-j}\right)^2 \to min. \qquad (1.29)$$

Это приводит к системе нормальных уравнений:

$$\begin{cases} \varphi_1 \sum_{t=p+1}^{n} y_{t-1}^2 + \varphi_2 \sum_{t=p+1}^{n} y_{t-1}y_{t-2} + .. + \varphi_p \sum_{t=p+1}^{n} y_{t-1}y_{t-p} = y_t y_{t-1} \\ \varphi_1 \sum_{t=p+1}^{n} y_{t-1}y_{t-2} + \varphi_2 \sum_{t=p+1}^{n} y_{t-2}^2 + .. + \varphi_p \sum_{t=p+1}^{n} y_{t-2}y_{t-p} = y_t y_{t-2}. \\ \ldots \\ \varphi_1 \sum_{t=p+1}^{n} y_{t-1}y_{t-p} + \varphi_2 \sum_{t=p+1}^{n} y_{t-2}y_{t-p} + .. + \varphi_p \sum_{t=p+1}^{n} y_{t-p}^2 = y_t y_{t-p} \end{cases} \qquad (1.30)$$

Условие стационарности накладывает ограничение на коэффициенты авторегрессии: например, для уравнения авторегрессии первого порядка коэффициент φ_1 должен быть по абсолютной величине меньше 1, а для коэффициентов авторегрессии второго порядка $-1 < \varphi_2 < 1$; $\varphi_1 + \varphi_2 < 1$; $\varphi_2 - \varphi_1 < 1$.

Модель авторегрессии хорошо описывает временные ряды, но подходит лишь для стационарных процессов. Поэтому перед началом построении модели, нужно проверить исходный ряд y на стационарность, используя тест Дики-Фуллера.

Тест Дики-Фуллера – проверка временного ряда на стационарность, в основе которого лежит проверка уравнения регрессии первого порядка на наличие единичного корня.

Уравнение авторегрессии первого порядка:

$$y_t = c + \varphi_1 y_{t-1} + \varepsilon_t. \qquad (1.31)$$

Выражение 1.31 можно записать в виде:

$$\Delta y_t = b y_{t-1} + \varepsilon_t, \qquad (1.32)$$

где $b = \varphi_1 - 1$,

$\Delta y_t = y_t - y_{t-1}$.

Тест Дики-Фуллера включает в себя проверку:

- нулевой гипотезы H_0: $b = 1$;
- альтернативной гипотезы H_1: b *меньше 0*.

Распределение *DF*-статистики называется распределением Дики-Фуллера и выражается не через распределение *t*-статистики, а через винеровский процесс.

Однако бывают ситуации, когда процесс может описываться авторегрессией не первого, а более высокого порядка, тогда используют расширенный тест Дики-Фуллера, в основу которого входит включение лагов первых разностей.

Модель авторегрессии часто очень хорошо описывает временные ряды, но подходит лишь для стационарных временных рядов. Поэтому перед началом построения модели следует проверить исходный ряд на стационарность или привести его к стационарному виду, используя разные методы (логарифмирование, интегрирование).

1.5 Модель скользящего среднего

Используя модель скользящего среднего, мы делаем предположение, что в ошибках модели в предыдущий момент времени содержится полезная информация об исследуемом ряде [6]. Следовательно, модель скользящего среднего первого порядка, представляет собой выражение:

$$y_t = b\varepsilon_{t-1} + \varepsilon_t, \tag{1.33}$$

где b – параметр модели;

ε_t – белый шум.

Тогда модель скользящего среднего порядка q (обозначение: *MA(q))* представляется в виде:

$$y_t = \sum_{j=0}^{q} b_j \varepsilon_{t-j}. \tag{1.34}$$

Так как данную модель чаще всего используют в виде дополнительной модели, описывающей случайные ошибки рядов, то константу можно считать параметром основной модели.

Модель скользящего среднего является одной из самых важных моделей временных рядов, так как согласно теореме Волда любой стационарный временной ряд можно представить как процесс *MA(q=∞)* с некоторыми коэффициентами, причем сумма модулей этих коэффициентов должна быть конечной.

Главной проблемой модели скользящего среднего является выбор метода для оценки ее параметров. Так как сумма квадратов остатков не выражается аналитически, применение обычного метода наименьших квадратов затруднено, однако можно воспользоваться методом максимального правдоподобия или некоторыми альтернативными подходами:

- Предположим, что в периоды до наших наблюдений значения ε равны 0, тогда задача сводится к минимизации суммы квадратов остатков последовательных выражений:

$$\varepsilon_1 = y_1, \varepsilon_2 = y_2 - b_1\varepsilon_1, \varepsilon_3 = y_3 - b_1\varepsilon_1 - b_2\varepsilon_2. \quad (1.35)$$

- Предположим, что ошибки в предшествующие и начальный моменты времени равны 0, тогда остальные значения ошибок можно найти, используя отклонения фактических значений уровней ряда от значений, полученных путем построения уравнения авторегрессии первого порядка.

Как и в модели авторегрессии, условие сходимости накладывает на коэффициенты модели скользящего среднего ограничение: b_j по абсолютной величине должны быть меньше единицы.

1.6 Модель авторегрессии скользящего среднего и интегрированная модель авторегрессии скользящего среднего

В целом модели временных рядов могут иметь множество форм и представлять разные стохастические процессы. Чаще всего используют модели авторегрессии *(AR)* и скользящего среднего *(MA)*, но комбинируя эти две модели можно получить не менее популярные модели авторегрессии скользящего среднего *(ARMA* или *APCC)* и авторегрессии интегрированного скользящего среднего *(ARIMA)* [8, 18].

Модель *ARMA(p,q)* является комбинацией моделей авторегрессии порядка p и скользящего среднего порядка q и подходит для построения одномерных временных рядов. Данная модель описывается выражением:

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}, \quad (1.36)$$

где p, q – порядки авторегрессии и скользящего среднего;

c, φ_i, θ_j – параметры модели.

Обычно ARMA модель записывают, используя лаговый оператор.

$$\varphi\ L\ y_t = \theta\ L\ \varepsilon_t, \quad (1.37)$$

где $\varphi\ L\ = 1 - \sum_{i=1}^{p} \varphi_i L^i$, $\theta\ L\ = 1 + \sum_{j=1}^{q} \theta_j L^j$.

Данная модель хорошо описывает стационарные ряды, однако не подходит для нестационарных временных рядов. Но зная, что при интегрировании нестационарного временного ряда можно получить стационарный, была придумана улучшенная модель *ARIMA*. *ARIMA(p,d,q)* – интегрированная модель авторегрессии скользящего среднего. Данная модель имеет вид:

$$\Delta^d y_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i \Delta^d y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}, \quad (1.38)$$

где c, φ_i, θ_j – параметры модели;

Δ^d – оператор взятия разности *d* порядка.

Выражение 1.43 можно записать, используя лаговый оператор:

$$\varphi\ L\quad 1 - L\ ^d y_t = \theta\ L\ \varepsilon_t. \tag{1.39}$$

Модель *ARIMA* с параметрами *p=0, d=1, q=0* представляет собой процесс случайного блуждания: $y_t = y_{t-1} + \varepsilon_t$.

Стоит заметить, что в модели ARIMA в результате прогнозирования мы получаем не само значение y_{t+1}, а изменение относительно предыдущих дней. Реальное значение y_{t+1} модели *ARIMA(p,1,q)* можно получить, используя формулу:

$$y_{t+1} = y_t + \Delta y_{t+1}, \tag{1.40}$$

где y_{t+1} – прогнозируемое значение;

y_t – известное значение временного ряда в предыдущий момент времени;

Δy_{t+1} – значение, полученное с помощью модели *ARIMA(p,1,q).*

Для модели *ARIMA(p,2,q):*

$$y_{t+1} = 2y_t - y_{t-1} + \Delta^2 y_{t+1}. \tag{1.41}$$

Для построения модели *ARIMA* часто используется методология Бокса-Дженкинса, которая представляет собой алгоритм (рисунок 1.7) [7, 8].

Рисунок 1.7 – Методология Бокса-Дженкинса

Рассмотрим методологию Бокса-Дженкинса подробнее:

1) Проверка исходного ряда на стационарность проходит с использованием формальных тестов, например, тест Дики-Фуллера или критерия *KPSS*. Также можно проанализировать коррелограмму ряда, для этого строится график выборочной автокорреляционной функции (*ACF*). Если ряд является стационарным, то коррелограмма быстро убывает, иначе ряд является нестационарным.

2) Преобразование ряда. Если исходный временной ряд нестационарный, то следует преобразовать его в стационарный. Чаще всего используется интегрирование уровней ряда. Обычно порядок интегрирования не превышает 2.

3) Выбор модели для экспериментальной проверки. После преобразования ряда, нужно выбрать остальные параметры модели – параметры p и q. Для этого можно воспользоваться двумя методами:

- Изучение графиков *ACF* и *PACF*.

ACF – автокорреляция. Показывает тесноту связи между значениями уровня ряда y_{t+1} и y_t. Автокорреляционная функция порядка *t* (ACF_t) имеет вид:

$$ACF_t = \frac{\sum_{i=1}^{T-t} y_i - y\ (y_{i+t} - y)}{\sum_{i=1}^{T} y_i - y\ ^2}, \quad (1.42)$$

где y_i – уровни ряда;

y – среднее значение уровней ряда;

t – лаг.

PACF – частная (частичная) автокорреляция. Показывает тесноту линейной стохастической зависимости между значениями в текущий момент времени от значений временного ряда на k моментов времени назад. Частная автокорреляция похожа на автокорреляцию, однако удаляет зависимость промежуточных значений: $y_{t-1}, y_{t-2}, \dots, y_{t-k+1}$

$$PACF_t = \frac{ACF_t - \sum_{j=1}^{t-1} PACF_{t-1,j} * ACF_{t-j}}{1 - \sum_{j=1}^{t-1} PACF_{t-1,j} * ACF_j},$$

$$PACF_1 = ACF_1, \quad (1.43)$$

$$PACF_{t,t} = ACF_t,$$

$$PACF_{t,j} = PACF_{t-1,j} - PACF_t * PACF_{t-1,t-j}.$$

Параметры p, q выбираются в зависимости от наблюдаемого поведения АКФ и ЧАКФ. В общем случае, можно использовать таблицу 1.1.

Обычно выбираются такие значения p и q, которые совпадают с лагами, при которых коэффициенты автокорреляционной и частной автокорреляционной функции значимы [7, 14].

Таблица 1.1– Правила подбора параметров p, q

Шаблон поведения АКФ и ЧАКФ		Возможная модель
АКФ	ЧАКФ	
Имеет вид убывающей функции, которая приближается к нулю.	Имеет значимые величины на первых p лагах.	*AR(p)*
Имеет значимые величины на первых q лагах.	Имеет вид убывающей функции, которая приближается к нулю.	*MA(q)*
Имеет вид убывающей функции, которая приближается к нулю.	Имеет вид убывающей функции, которая приближается к нулю.	Недостаточно информации, для определения параметров модели *ARMA*.

Данный метод лишь является отправной точкой и дает весьма субъективную оценку [8].

- Критерии выбора оптимальной модели. Первый из критериев – информационный критерий AIC (Акаике). Данный критерий штрафует модель за большое количество параметров. Значение данного критерия находится по формуле:

$$AIC = \ln\ \frac{\sigma^2}{n}\ + \frac{p+q}{n}, \qquad (1.44)$$

где σ^2 – сумма квадратов ошибок;

n – число наблюдений (ошибок);

p, q – параметры модели.

Наилучшей считается модель с наименьшим значением критерия Акаике.

Также используют критерий Шварца, который похож на критерий Акаике и минимизирует следующее выражение:

$$BIC = \ln\ \frac{\sigma^2}{n}\ + \frac{p+q\ \ln\ n}{n}. \quad (1.45)$$

Данные критерии позволяют выбрать модель с минимальной ошибкой и наименьшим количеством параметров или выбрать лучшую модель из списка моделей с одинаковыми значениями ошибок.

4) Оценка параметров экспериментальной модели. Как правило, для оценивания параметров модели *ARIMA (p, d, q)* используется метод наименьших квадратов, который дает хороший результат.

Пусть $Y = \ y_1, y_2, \dots, y_N\ ^T$– вектор уровней исходного временного ряда. Требуется найти оценки параметров φ_i, θ_i модели *ARIMA(p, d, q),* используя метод наименьших квадратов.

$y = \ y_{p+1}, y_{p+2}, \dots, y_N\ ^T$ – вектор фактических значений ряда, с помощью которых находятся оценки параметров модели *ARIMA(p, d, q).*

Тогда, следуя определению модели, обозначим матрицу X_1, содержащую значения ряда в предыдущие моменты времени *1, 2, 3, …, p-1*.

$$X_1 = \begin{matrix} 1 & y_p & y_{p-1} & y_{p-2} & \cdots & y_1 \\ 1 & y_{p+1} & y_p & y_{p-1} & \cdots & y_2 \\ 1 & y_{p+2} & y_{p+1} & y_p & \cdots & y_3 \\ 1 & y_{p+3} & y_{p+2} & y_{p+1} & \cdots & y_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & y_{N-1} & y_{N-2} & y_{N-3} & \cdots & y_{N-p} \end{matrix}. \quad (1.46)$$

Матрица X_2 содержит информацию о предыдущих ошибках модели.

$$X_2 = \begin{matrix} \varepsilon_q & \varepsilon_{q-1} & \varepsilon_{q-2} & \cdots & \varepsilon_1 \\ \varepsilon_{q+1} & \varepsilon_q & \varepsilon_{q-1} & \cdots & \varepsilon_2 \\ \varepsilon_{q+2} & \varepsilon_{q+1} & \varepsilon_q & \cdots & \varepsilon_3 \\ \varepsilon_{q+3} & \varepsilon_{q+2} & \varepsilon_{p+1} & \cdots & \varepsilon_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \varepsilon_{N-1} & \varepsilon_{N-2} & \varepsilon_{N-3} & \cdots & \varepsilon_{N-q} \end{matrix}. \quad (1.47)$$

Основная матрица *X* состоит из соединенных матриц X_1 и X_2:

$$X = cat\ X_1, X_2\ , \quad (1.48)$$

где cat – процедура конкатенации (объединения матриц).

Тогда:

$$y = X\beta + \varepsilon, \tag{1.49}$$

где $\varepsilon = \ \varepsilon_1, \varepsilon_2, \dots, \varepsilon_N\ ^T$ - вектор ошибок;

$\beta = (c, \varphi_1, \varphi_2, \dots, \varphi_p, \theta_1, \theta_2, \dots, \theta_q)$ – вектор параметров модели.

Метод наименьших квадратов минимизирует выражение:

$$\varepsilon^T\varepsilon = \ y - X\beta\ ^T(y - X\beta). \tag{1.50}$$

Тогда вектор оценок коэффициентов модели можно найти по формуле:

$$\beta = \ X^TX\ ^{-1}X^Ty. \tag{1.51}$$

5) Проверка модели на адекватность. Чтобы модель была адекватная, нужно, чтобы ее остатки:

- имели случайный характер;
- их математическое ожидание было равно нулю;
- отсутствовала автокорреляционная зависимость;
- подчинялись нормальному закону распределения.

Чтобы оценить данные критерии, можно воспользоваться тестом Жарка-Бера, который определяет отклонение от нормальности и основан на значениях эксцесса и асимметрии [7]. Данный тест подходит только для длинных временных рядов (количество уровней ряда больше 2000). Статистика JB вычисляется по формуле:

$$JB = \frac{n}{6}\ \ A^2 + \frac{Ex^2}{4}\ , \tag{1.52}$$

где n – количество наблюдений;

A – асимметрия ($A = \frac{(x_i - x)^3}{n\sigma^3}$);

E – эксцесса ($E = \frac{(x_i - x)^4}{n\sigma^4} - 3$);

Статистика Жарка-Бера имеет асимптотическое хи-квадрат распределение с двумя степенями свободы. Обычно проверяется нулевая гипотеза о том, что остатки принадлежат нормальному распределению.

Проверку незначимости коэффициентов автокорреляционной функции можно провести при помощи Q-статистики Бокса-Пирса [6, 7]:

$$Q = n \sum_{t=1}^{k} r_t^2, \tag{1.53}$$

Или Q-статистики Бокса-Льюнга:

$$Q = n\ (n+2) \sum_{t=1}^{k} \frac{r_t^2}{n-t}. \tag{1.54}$$

Значение Q сравнивается со значением хи-квадрат при *v* степенях свободы (*v = k – p – q)* и уровнем значимости *α (κ* рекомендуется брать равным *n/4).*

Если Q > χ^2, то группа первых коэффициентов автокорреляции является значимыми и остатки не являются случайными.

6) Построение прогноза осуществляется путем замены в общем уравнении модели *ARIMA* (формула 1.36) t на $t+1$, будущих ошибок на нули, а прошлых – на остатки [9]. Таким образом, мы получим уравнение для построения прогноза на 1 шаг вперед:

$$y_{t+1} = c + \varphi_1 y_t + \cdots + \varphi_p y_{t+1-p} + \theta_1 \varepsilon_t + \ldots + \theta_q \varepsilon_{t-1+q}. \tag{1.55}$$

Доверительный интервал для моделей *ARIMA* сложен для вычисления. 95% доверительный интервал для первого прогноза вычисляется по формуле:

$$y_{t+1} \pm 1.96\sigma, \tag{1.56}$$

где σ – стандартное отклонение остатков.

Также можно вычислить доверительный интервал для многошагового прогноза модели *ARIMA*(0,0,q):

$$\sigma_h = \sigma^2 \left[1 + \sum_{i=1}^{h-1} \theta_i^2\right], \quad (1.57)$$

где $h = 2, 3, \ldots$.

Тогда 95% доверительный интервал можно рассчитать по формуле:

$$y \pm 1.96\sigma_h. \quad (1.58)$$

По данной формуле также можно вычислить доверительный интервал для модели *AR*(1).

С помощью методологии Бокса-Дженкинса выбирается наилучшая модель из класса моделей *ARMA*.

Для выбора модели с наименьшей ошибкой прогноза можно использоваться разные меры оценки точности прогноза.

1.7 Метрики оценки точности прогноза

В общем случае процесс построения модели и прогноза для всех типов моделей одинаков и включает несколько этапов:

1) Сбор и получение данных, их исследование. Исходные данные должны быть полные, не иметь пропусков и иметь одну единицу измерения.

2) Выбор начальных параметров модели (порядок авторегрессии, скользящего среднего, коэффициента сглаживания).

3) Исходные данные должны быть разделены на 2 части. Первая часть (примерно 70-80% от начального объема данных) – обучающее множество, используется для настройки модели (нахождения оценок коэффициентов). Вторая часть исходных данных – тестовое множество, используется для проверки точности настроенной модели. Так как

построенная модель может идеально описывать исходные данные, но абсолютно не подходить для прогноза.

4) Сравниваются разные модели и выбирается та, которая показала наилучшие оценки точности.

Для оценки точности прогноза используется несколько метрик, каждая из которых обладает своими особенностями.

Пусть была идентифицирована модель прогнозирования временных рядов и построен прогноз. Получен вектор ошибок $e = y - y$.

MFE – средняя ошибка прогноза. Отображает среднее отклонение фактических данных от прогнозируемых, но положительные и отрицательные ошибки сокращают друг друга. Чем ближе значение данной метрики к нулю, тем прогноз точнее.

$$MFE = \frac{1}{n}\sum_{i=1}^{n} e_i. \quad (1.59)$$

MAE (MAD) – средняя абсолютная ошибка (отклонение). Отображает среднее абсолютное отклонение фактических данных от прогнозируемых. В отличии от средней ошибки прогноза (MFE) положительные и отрицательные ошибки не сокращают друг друга. Чем ближе значение данной метрики к нулю, тем прогноз точнее.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} e_i \ . \quad (1.60)$$

MAPE – средняя абсолютная процентная ошибка. Отображает процент отклонения фактических значений от прогнозируемых, однако применяется для рядов, среднее значение которых больше единицы. Чем меньше значение данной метрики, тем модель лучше.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{e_i}{y_i}\right| * 100\%. \quad (1.61)$$

MPE – средняя процентная ошибка. Отображает процент отклонения фактических значений от прогнозируемых, но в отличии от MAPE

положительные и отрицательные ошибки сокращают друг друга. Чем меньше процент ошибок, тем модель лучше.

$$MPE = \frac{1}{n}\sum_{i=1}^{n}\frac{e_i}{y_i} * 100\%. \quad (1.62)$$

MSE – средняя квадратическая ошибка. Отображает среднее квадратическое отклонение фактических значений от прогнозируемых. Большие ошибки имеют большое влияние на результат. Чем меньше данная метрика, тем лучше модель строит прогноз.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2. \quad (1.63)$$

Точность прогноза – обратное определению ошибки прогноза. Применяется очень редко, так как чаще всего оценивают именно ошибку прогноза (MAPE, MAE). Точность прогноза определяется в процентах.

$$\text{Точность прогноза} = 100\% - MAPE. \quad (1.64)$$

Оценивая точность прогнозирования построенной модели, обычно используют несколько метрик. Чаще всего в качестве меры сравнения разных моделей между собой используются метрики: *MSE* (среднее квадратическое отклонение) и *MAPE* (средняя абсолютная процентная ошибка).

2 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ МЕТОДОВ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

2.1 Обзор используемого инструментария разработки

Для реализации методов будем использовать объектно-ориентированный язык программирования Python версии 3.7 и интегрированную среду разработки PyCharm. Данный язык программирования был выбран из-за ряда следующих причин:

- Данный язык обладает простым синтаксисом;
- Python является мультиплатформенным языком программирования;
- Python имеет огромную базу библиотек для работы со всеми видами информации: для работы с файлами, математикой, построение графиков, а также статистические библиотеки для анализа временных рядов.

Для анализа временных рядов и реализации моделей их прогнозирования будут использоваться следующие библиотеки:

- Библиотека NumPy. Данная библиотека используется для работы с матрицами и массивами, например, нахождение обратной матрицы, умножение одной матрицы на другую, и включает в себя методы численных оптимизаций и статические процедуры [3, 14].
- Библиотека Pandas. Данная библиотека позволяет работать и хранить разные типы данных. Включает в себя возможность чтения информации из файлов разного типа: csv, excel, txt. Имеет методы для анализа неполных данных.
- Библиотека Matplotlib предоставляет функции для отображения двумерных и трехмерных графиков и диаграмм. Позволяет легко настроить цвет, форму и вид графиков.
- Библиотеки Statsmodels и Scipy. Данные библиотеки включают себя возможность работы с данными со статистической точки зрения.

Позволяет проводить различные тесты и строить графики корреляционных и автокорреляционных функций [14, 20].

Все модели прогнозирования временных рядов будут реализованы в модуле models.py. Модуль – файл с расширением py, который содержит определения и операторы Python. Импорт модулей происходит с помощью ключевого слова *import*.

Перед началом реализации моделей, загрузим данные для тестирования работоспособности реализованных моделей, используя библиотеку Pandas [14]. Так, как все исходные данные хранятся в формате csv воспользуемся методом read_csv, которому передаются параметры: путь к файлу, знак разделения строк и формат цифр. Затем приведем данные к формату ndarray с помощью метода to_numpy(), чтобы можно было работать с информацией в виде массивов. Для отрисовки данных воспользуемся классом pyplot библиотеки matplotlib и методом plot(data), которая принимает массив значений для отображения их на графике.

2.2 Реализация модели полиномиального тренда

Модель полиномиального тренда будет реализована в виде класса под названием PolynomialRegression. UML-диаграмма данного класса изображена на рисунке 2.1.

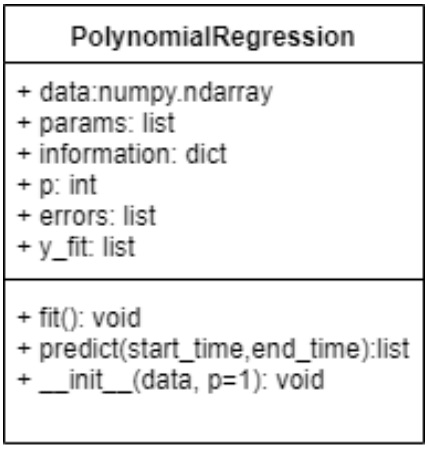

Рисунок 2.1 – UML-диаграмма класса PolynomialRegression

Класс PolynomialRegression содержит методы:

- __init__(data, p=1) – конструктор класса, который принимает на вход одномерный массив и степень *p* полинома. Если степень полинома не задана, берется значение по умолчанию равное 1.
- fit() – метод, который находит оценки коэффициентов по методу наименьших квадратов и делает подсчет основных метрик *MSE, MAE, MAPE.*
- predict(start_time, end_time) – метод, принимающий на вход два целочисленных значений *start_time* и *end_time*, который производит построение прогноза в моменты времени *t+start_time, t+start_time+1, …, t+end_time.*

Класс PolynomialRegression содержит поля:

- data – одномерный массив, который содержит исходный временной ряда.
- params – коэффициенты полиномиального тренда, найденные с помощью метода наименьших квадратов.
- information – словарь, который содержит основные значения метрик *MSE, MAE, MAPE*, а также значение коэффициента детерминации и скорректированного коэффициента детерминации.
- y_fit – массив значений, найденных с помощью модели полиномиального тренда.
- errors – массив, который содержит остатки модели полиномиального тренда.

Для построения модели, сначала нужно создать объект класса PolynomialRegression, затем вызвать метод fit() для нахождения коэффициентов регрессии и построить прогноз, используя метод predict.

Проверим работоспособность данного класса. В качестве p возьмем значения из диапазона [1, 2, 3, 4, 5]. Отобразим исходный временной ряд и

временные ряды, полученные с помощью класса PolynomialRegression (рисунок 2.2).

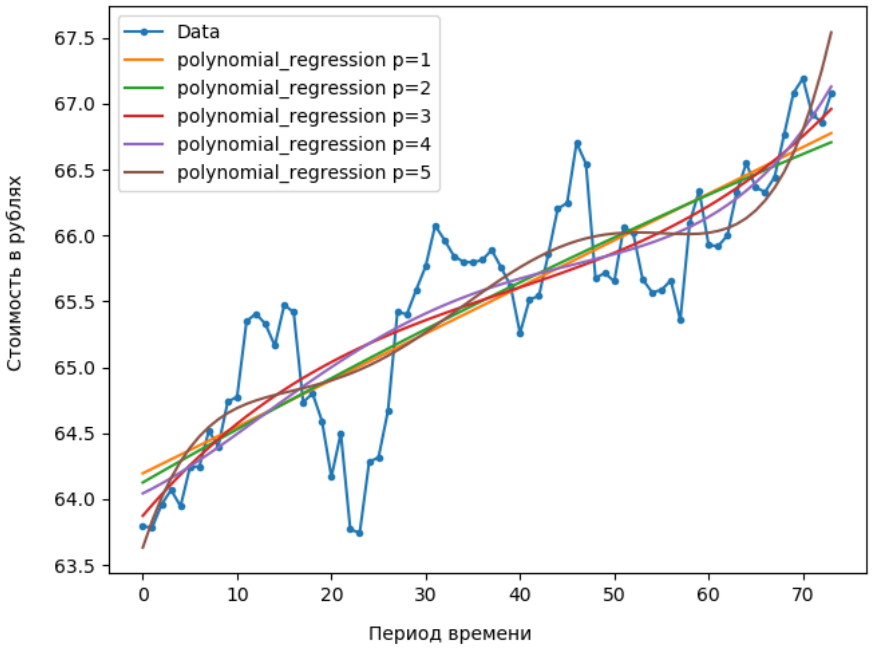

Рисунок 2.2 – График исходного временного ряда и полиномиального тренда

Так как для нахождения коэффициентов большинства моделей будет использоваться метод наименьших квадратов, реализуем его в методе least_square_method, принимающий на вход массивы *x* и *y*. С помощью метода inv класса linalg библиотеки numpy будем находить обратную матрицу. Атрибут T объекта класса ndarray содержит транспонированную матрицу. Метод dot выполняет умножение двух матриц.

2.3 Реализация моделей экспоненциального сглаживания

Модель экспоненциального сглаживания (модель Брауна) реализуем в классе ExponentialSmoothing, который содержит методы:

- __init__(time_series, alpha=None, first_ = ‘average’) – конструктор класса. Принимает на вход массив исходного временного ряда, значение коэффициента сглаживания и вид первого значения

сглаженного временного ряда. Если alpha не задано, то значение находится по формуле:

$$alpha = \frac{2}{len\ time\_series\ + 1}. \quad (2.1)$$

Если значение first не задано, то в качестве первого значения сглаженного ряда берется первое значение исходного ряда, иначе – среднее значение исходного временного ряда.

- fit() – метод, который находит значения сглаженного временного ряда и рассчитывает основные метрики ошибок модели (MSE, MAE, MAPE).
- predict(h) – метод, который делает прогнозирование на h периодов вперед.

Класс ExponentialSmoothing содержит поля:

- alpha – коэффициент сглаживания модели.
- information – словарь, в котором содержится информация об ошибках построенной модели.
- residuals – одномерный массив, который содержит остатки модели.
- y_fit – сглаженный временной ряд.

UML-диаграмма данного класса изображена на рисунке 2.3.

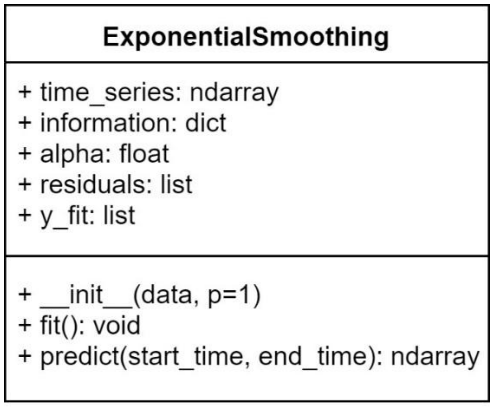

Рисунок 2.3 – UML-диаграмма класса ExponentialSmoothing

Модели Хольта и Хольта-Винтерса реализуем в классах DoubleExponentialSmoothing и HoltWinters, которые будут иметь те же

методы и поля, что и класс ExponentialSmoothing, за исключение метода _find_coefficients_() и конструктора __init__:

- _find_coefficients_() – метод, который возвращает коэффициенты линейной регрессии, найденные по первым пяти элементам исходного временного ряда.
- __init__(data, alphas) – конструктор класса DoubleExponentialSmoothing, принимающий на вход исходный временной ряд и массив из двух значений – коэффициентов сглаживания.
- __init__(data, s, alphas) – конструктор класса HoltWinters, принимающий на вход исходный временной ряд, значение периода сезонности и массив из трех значений – коэффициентов сглаживания.

Листинги данных классов приведены в приложении А.

UML-диаграммы классов DoubleExponentialSmoothing и HoltWinters можно увидеть на рисунках 2.4 и 2.5.

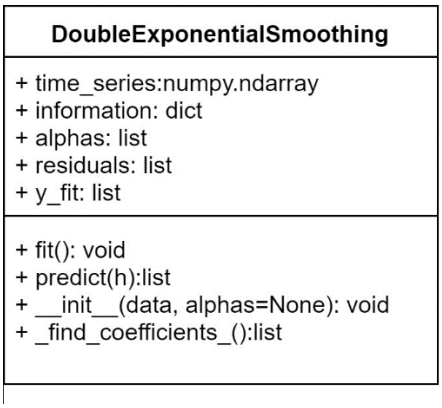

Рисунок 2.4 – UML-диаграмма класса DoubleExponentialSmoothing

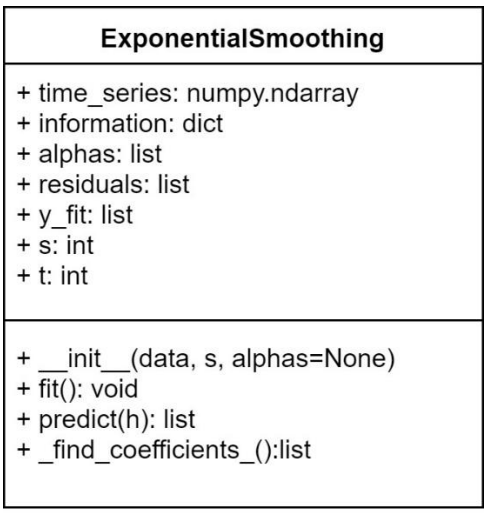

Рисунок 2.5 – UML-диаграмма класса HoltWinters

Проверим работоспособность всех трех функций. В качестве значений альфа возьмем 0.2. На рисунке 2.6 можно увидеть график исходного временного ряда и временных рядов, полученных при помощи вышеперечисленных классов.

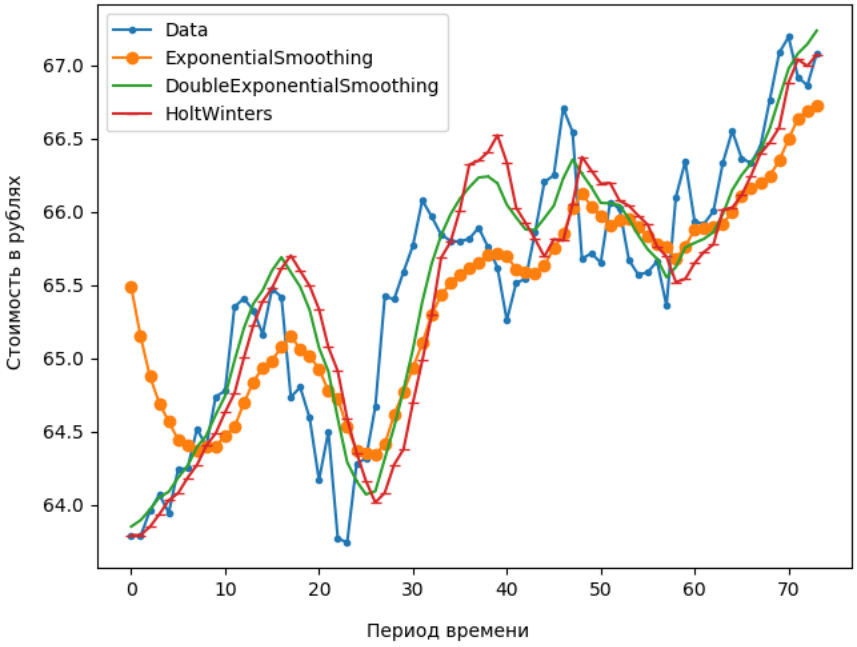

Рисунок 2.6 – График исходного и сглаженных временных рядов

Так как параметры сглаживания были выбраны случайно, трудно судить о качестве построенных моделей. Можно сделать вывод о том, что модели примерно описывают исходный временной ряд, значит программа работает верно.

2.4 Реализация модели ARIMA

Перед реализацией основного класса модели, следует реализовать метод для получения из исходного временного ряда интегрированный и наоборот. Для этого создадим три функции:

- integration(data, d) - функция, принимающая на вход исходный временной ряд *data* и порядок интегрирования *d*, и возвращающая массив интегрированных значений.
- from_integration_data(last_values, d, integr_data) – функция, принимающая last_values - последнее фактическое значение временного ряда, d - порядок интегрирования, integr_data – интегрированное значение. Возвращает не интегрированное значение.
- from_integration_data_list(not_integr_data, integr_data, d) – функция, похожая на from_integration_data, но переводит не одно интегрированное значение, а целый массив значений.

Также стоит заметить, что при нахождении коэффициентов модели *ARIMA* через метод наименьших квадратов, матрица X – предикторов может состоять из значений, близких к нулю, так как ошибки модели – значения эпсилон – белый шум с нулевым математическим ожиданием [12]. Тогда нахождение обратной матрицы будет затруднено. Для решения этой проблемы оценочные коэффициенты будем находить по формуле:

$$\beta = X^{+}y, \qquad (2.2)$$

где X^{+} – псевдо-обратная матрица, найденная с помощью метода *pinv* библиотеки *numpy*.

Теперь можно приступить к реализации модели *ARIMA*. Для этого создадим класс ARIMA с методами:

- __init__(data, p, d, q, without_constant = False) – конструктор объекта класса, принимающий на вход массив, содержащий исходный временной ряд, порядок авторегрессии *p*, порядок интегрирования *d*,

порядок скользящего среднего q. Сразу в конструкторе присваиваем атрибуту time_series интегрированный временной ряд. Переменная without_constant принимает False, если модель содержит константу и True, если не содержит.

- _find_errors_ar_() – метод, который находит массив начальных остатков модели, используя модель авторегрессии первого порядка.
- lag_data(data, lag) – возвращает массив данных data, сдвинутых на lag.
- fit() – метод, находящий коэффициенты модели по методу наименьших квадратов, а также подсчитывающий значения *MSE, MAE, AIC, MAPE*. Разберем этот метод подробнее.

Для начала мы создаем массив остатков *residuals*. С помощью метода _find_errors_ar_() находим остатки модели в *t* моменты времени.

Затем создаем массив *x*, состоящий из лагов исходных (или интегрированных) данных и соединяем две матрицы – *x* и *residuals* при помощи функции concatenate. Далее мы находим коэффициенты модели, используя функцию least_square_method(). Затем проверяем получившиеся коэффициенты и всю модель на стационарность, используя определение единичного корня. Для этого мы находим корни полинома:

$$a\ z\ = 1 - \sum_{i=1}^{p} \alpha_i z^i, \quad (2.3)$$

где α_i – найденные коэффициенты.

Если полученные корни лежат внутри единичного круга (меньше единицы), то модель можно считать стационарной.

Далее мы находим массив y_fit, используя формулу:

$$y\_fit = x\beta. \quad (2.4)$$

Данный массив содержит предсказанные значения *y* по найденной модели *ARIMA(p,d,q)*. Если порядок интегрирования больше нуля, то

находим не интегрированные значения, используя функцию from_integration_data_list. Затем мы находим значения *MSE, MAE,* а также значения критерия Акаике и значение *MAPE*.

- _forecast_() - метод, который делает предсказание на один шаг вперед и возвращает интегрированное значение, если параметр *d* больше нуля.
- forecast() – метод, вызывающий _forecast_, но возвращающий не интегрированное значение.
- predict(steps) – метод, делающий предсказание на steps шагов вперед и возвращает массив предсказанных значений.

Листинг данного класса представлен в приложении А.

UML-диаграмма данного класса изображена на рисунке 2.7.

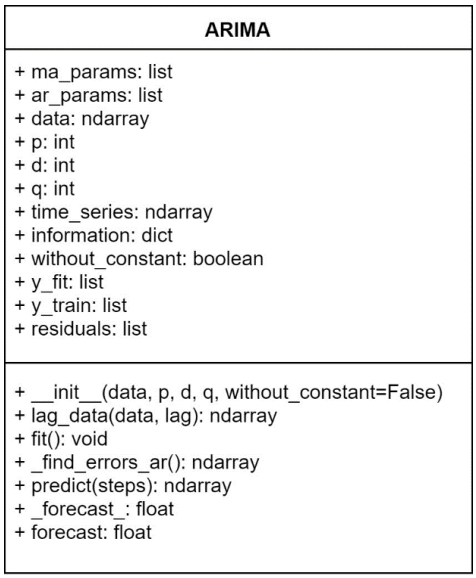

Рисунок 2.7 – UML-диаграмма класса ARIMA

Также были реализованы метрики оценки точности модели в модуле my_statistics.py. Листинг данного модуля приведен в приложении Б.

Так как кроме реализации модели, нам потребуются методы для первичного анализа временного ряда, построения графиков АФК и ЧАФК и тесты для проверки остатков модели, воспользуемся библиотекой statsmodels, которая содержит все вышеперечисленные методы.

Таким образом, были реализованы основные модели сглаживания и прогнозирования временных рядов. Теперь можно приступить к их анализу.

3 СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

3.1 Предобработка исходных данных

В качестве исходных данных для анализа реализованных методов будем использовать несколько данных в формате csv, взятых с сайта http://export.rbc.ru:

- Курс Доллара к Рублю за период: 24.05.2018 – 24.05.2019
- Курс Евро к Рублю за период: 01.02.2019 – 01.05.2019
- Стоимость грамма золота в рублях за период: 28.05.2018 – 28.05.2019
- Индекс РТС нефти и газа: 28.05.2018 – 28.05.2019

Для начала загрузим все исходные данные и изобразим графики временных рядов (рисунок 3.1).

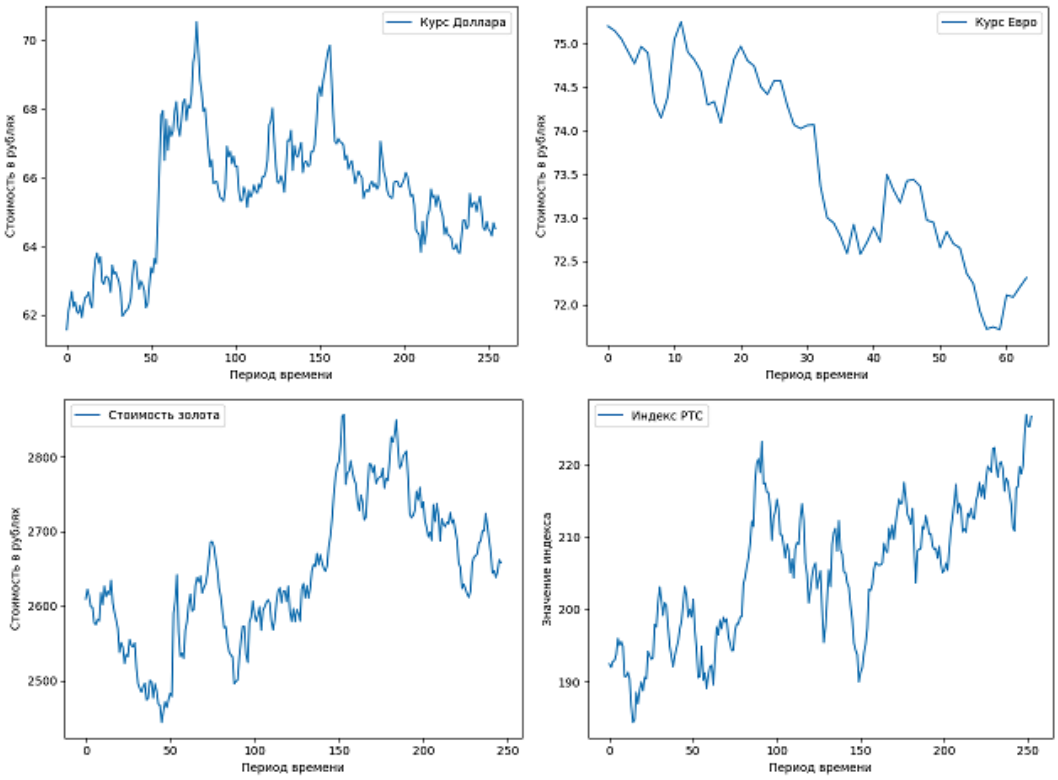

Рисунок 3.1 – Графики исходных временных рядов

Судя по графикам, можно сделать вывод о том, что ни один из временных рядов не является стационарным. Подтвердим нашу гипотезу,

используя тест Дики-Фуллера. Значения статистики *p* приведем в таблицу 3.1. Если *p* больше 0.05, то временной ряд не является стационарным [7].

Таблица 3.1 – Значения *p* для исходных временных рядов

Временной ряд	Значение *p (p-value)*	Ряд стационарный
Курс Доллара к Рублю	0.1048658328044293	-
Курс Евро к Рублю	0.7130586852073865	-
Стоимость грамма золота в рублях	0.4005823069855874	-
Индекс РТС нефти и газа	0.5290816806874549	-

Таким образом, ни один из исходных временных рядов не является стационарным. Возьмем первые разности и изобразим их на графике (рисунок 3.2).

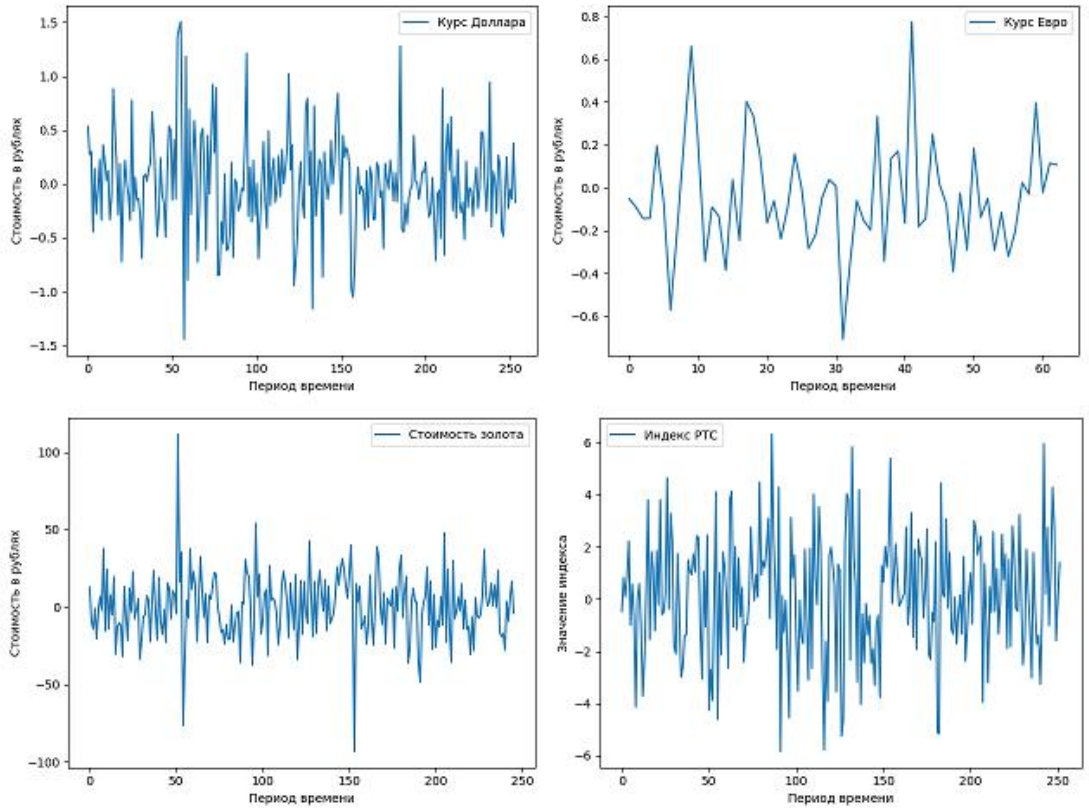

Рисунок 3.2 – Графики интегрированных временных рядов

Таблица 3.2 – Значения *p* для интегрированных временных рядов

Временной ряд	Значение *p (p-value)*	Ряд стационарный
Курс Доллара к Рублю	2.13064805593792e-27	+
Курс Евро к Рублю	3.46518894587761e-10	+
Стоимость грамма золота в рублях	3.98964510311376e-15	+
Индекс РТС нефти и газа	1.39998520505992e-28	+

Таким образом, мы получили стационарность, один раз проинтегрировав исходные временные ряды.

Для построения и тестирования реализованных моделей, разделим данные на две части: обучающая выборка и выборка для тестирования. В качестве обучающей выборки возьмем 80% исходные данных, а в качестве выборки для тестирования – остальные 20%.

3.2 Подбор параметра полиномиальной модели

Подберем значение *p* – степень полиномиальной модели и посчитаем средний квадрат ошибки на тестовой выборке. В таблице 3.3 приведем значения *p* и средний квадрат ошибки.

Таблица 3.3 – Расчетные значения *MSE* для модели полиномиального тренда

Временной ряд	Степень модели (*p*)	MSE
Курс Доллара к Рублю	1	12.01
	2	1.93
	3	0.69
	4	4.02
	5	311.34
Курс Евро к Рублю	1	0.165
	2	0.094
	3	1.84
	4	17.92
	5	7.9

Продолжение таблицы 3.3

Временной ряд	Степень модели (p)	MSE
Стоимость грамма золота в рублях	1	15476.549
	2	74368.590
	3	18182.959
	4	4082.198
	5	1281496.956
Индекс РТС нефти и газа	1	20.902
	2	140.986
	3	64.14
	4	885.476
	5	675.802

Выберем лучшую модель для каждого временного ряда, исходя из результатов ошибки из таблицы 3.3. Отобразим график исходного временного ряда и ряда, полученного по выбранной модели на рисунке 3.3.

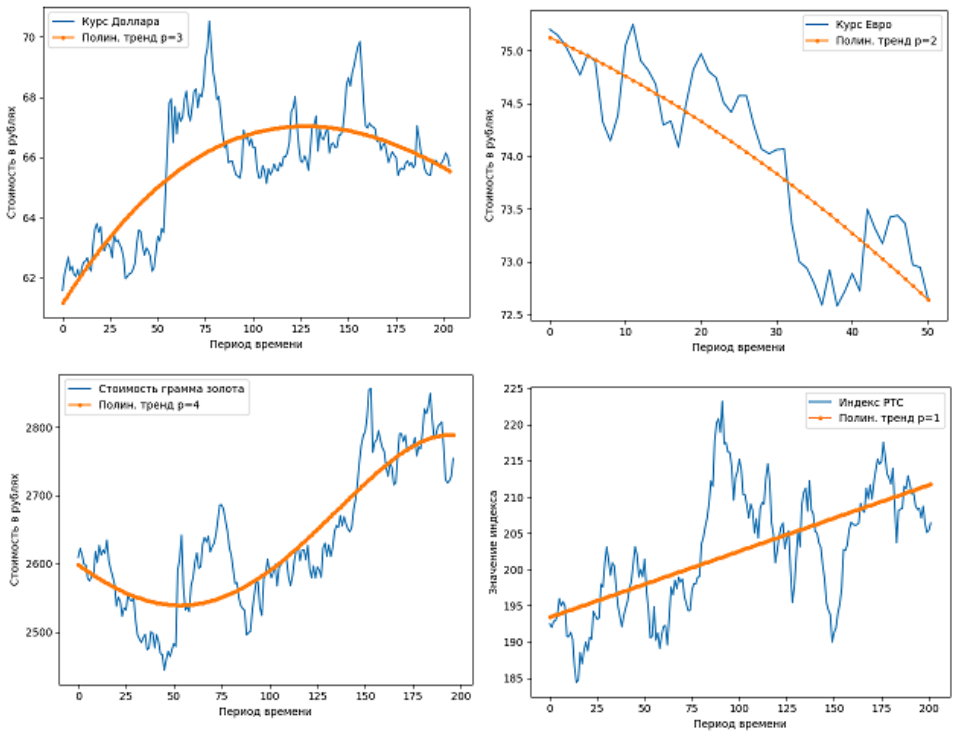

Рисунок 3.3 – График исходных временных рядов и прогнозов, построенных по полиномиальной модели тренда

Теперь нужно проверить выбранные модели на адекватность. Для этого взглянем на значения скорректированного и обычного коэффициента детерминации (таблица 3.4).

Таблица 3.4 – Значения коэффициента детерминации

Временной ряд	Коэффициент детерминации	Скорректированный коэффициент детерминации
Курс Доллара к Рублю	0.5966	0.5885
Курс Евро к Рублю	0.7521	0.7363
Стоимость грамма золота в рублях	0.7511	0.7461
Индекс РТС нефти и газа	0.40	0.396

Модель считается адекватной, если ее коэффициент детерминации хотя бы не меньше 0.5. Можно сказать, что все подобранные модели адекватно описывают исходный временной ряд, кроме модели временного ряда 4 (индекс РТС), так как она описывает только 40% временного ряда.

Теперь проверим дисперсию остатков модели, используя *F*-критерий Фишера. Для этого рассчитаем *F*-значение, используя формулу:

$$FR = \frac{Dy}{Dad}, \tag{3.1}$$

где Dy – дисперсия зависимой переменной;

$$Dy = \frac{y - mean\ y\ ^2}{N - 1}, \tag{3.2}$$

N – длина выборки;

Dad – дисперсия остатков модели;

$$Dy = \frac{y - y\ ^2}{N - k}, \tag{3.3}$$

k – число параметров модели;

y – расчетные значения ряда, полученные по модели.

Получив значение FR мы сравниваем его с критическим значением критерия Фишера, с доверительной вероятностью 95%, степенями свободы – *N-1, N-k*. Для этого используем функцию criterion_fisher, которая возвращает

рассчитанное значение FR и табличное значение критерия Фишера. Листинг данной функции приведен в приложении Б. Результаты расчетов приведены в таблице 3.5.

Таблица 3.5 – Значения F-критерия

Временной ряд	Расчетный F-критерий	Критическое значение
Курс Доллара к Рублю	2.455	1.261
Курс Евро к Рублю	3.953	1.604
Стоимость грамма золота в рублях	3.956	1.2669
Индекс РТС нефти и газа	1.655	1.26

Так как все расчетные значения F оказались больше табличных значений, то с вероятностью 95% можно считать построенные модели статистически адекватными.

Теперь проверим остатки на отсутствие автокорреляции с помощью критерия Дарбина-Уотсона. Для этого рассчитаем d-значение, используя формулу:

$$d = \frac{\sum_{i=1}^{N-1} (e_i - e_{i+1})^2}{\sum_{i=1}^{N} e_i^2}, \quad (3.4)$$

где e_i – остатки модели.

Теоретические значения d_L и d_U берутся из таблицы с числом наблюдений N, количеством параметров модели k и уровнем значимости α.

Если d меньше нижней границы d_L то гипотеза о независимости случайных величин отклоняется и в остатках присутствует корреляция.

Если d больше верхней границы, d_U то гипотеза не отклоняется.

Если d находится между d_L и d_U, то тест некорректен.

Когда d превышает 2, то сравнивается значение $(4 - d)$.

Рассчитаем значения d для каждой модели и отобразим их в таблицу 3.6.

Таблица 3.6 – Значения *d*-критерия для остатков модели

Временной ряд	Расчетный d-критерий	d_L	d_U
Курс Доллара к Рублю	0.1314	1.73	1.79
Курс Евро к Рублю	0.411	1.462	1.624
Стоимость грамма золота в рублях	0.18	1.728	1.810
Индекс РТС нефти и газа	0.12	1.758	1.778

Так как все получившиеся значения оказались меньше нижней границы фактических значений, то мы отвергаем гипотезу о независимости остатков с вероятностью 95%. Следовательно, в остатках осталась полезная информация о временных рядах.

3.3 Подбор параметров модели экспоненциального сглаживания

Для подбора параметров экспоненциального сглаживания построим сетку на диапазоне (0, 1) с шагом 0.05. В таблице 3.7 приведем значения лучших параметров сглаживания модели Брауна для каждого исходного ряда, которые минимизируют средний квадрат ошибки на тестовой выборке.

Таблица 3.7– Значения параметров сглаживания α и средний квадрат ошибок

Временной ряд	Параметр сглаживания α	MSE
Курс Доллара к Рублю	0.95	1.269
Курс Евро к Рублю	0.95	0.348
Стоимость грамма золота в рублях	0.45	4361.8978
Индекс РТС нефти и газа	0.05	89.328

На рисунке 3.4 изображены графики исходных временных рядов и прогнозов, найденных с помощью модели экспоненциального сглаживания.

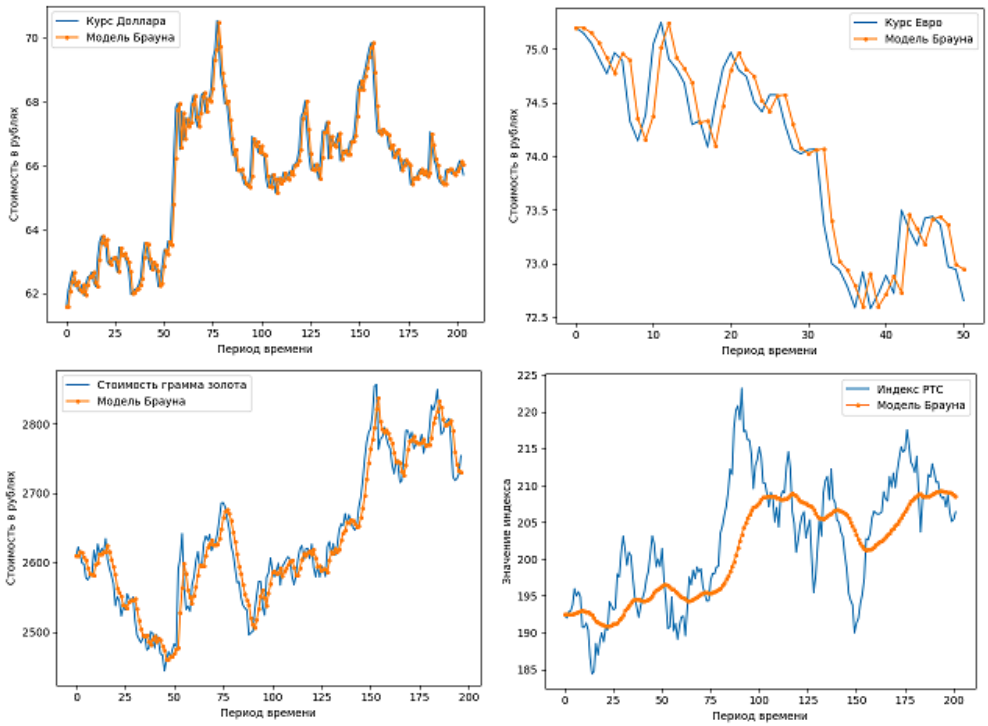

Рисунок 3.4 – Графики исходных и сглаженных временных рядов

В модели двойного экспоненциального сглаживания (модель Хольта) используется два параметра сглаживания: α_1 и α_2. Поэтому для подбора параметров будем использовать две сетки (0, 1) с шагом 0.5. В таблицу 3.8 запишем лучшие параметры сглаживания, которые минимизируют средний квадрат ошибок.

Таблица 3.8 – Значения параметров сглаживания α и средний квадрат ошибок

Временной ряд	α_1	α_2	MSE
Курс Доллара к Рублю	0.1	0.05	0.408
Курс Евро к Рублю	0.95	0.1	0.0821
Стоимость грамма золота в рублях	0.65	0.25	903.355
Индекс РТС нефти и газа	0.05	0.95	11.229

На рисунке 3.5 изобразим графики исходных временных рядов и сглаженных временных рядов, полученных при помощи модели двойного экспоненциального сглаживания.

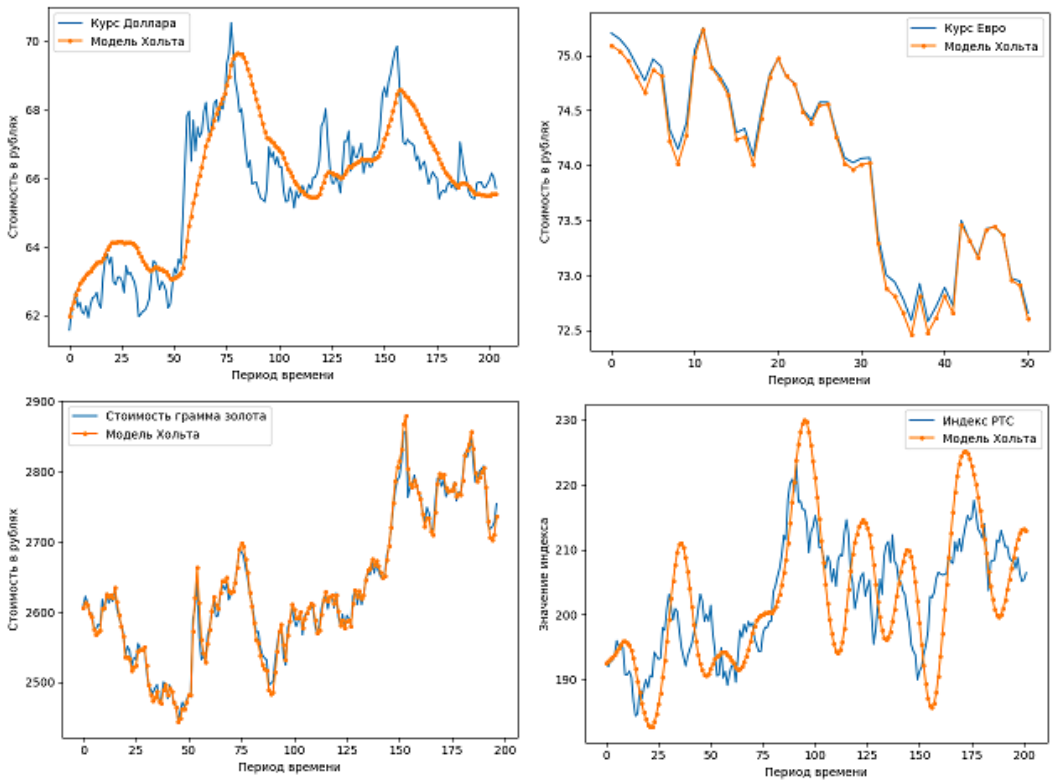

Рисунок 3.5 – График исходных временных рядов и сглаженных временных рядов, построенных с помощью модели Хольта

В модели тройного экспоненциального сглаживания нужно указать период сезонности s и три параметра сглаживания: α_1, α_2 и α_3. Выберем модель с параметрами, которые минимизируют средний квадрат ошибок и запишем в таблицу 3.9.

Таблица 3.9 – Значения параметров сглаживания α и средний квадрат ошибок

Временной ряд	s	α_1	α_2	α_3	MSE
Курс Доллара к Рублю	24	0.1	0.1	0.25	0.408
Курс Евро к Рублю	12	0.75	0.15	0.95	0.0813
Стоимость грамма золота в рублях	24	0.5	0.35	0.25	862.252
Индекс РТС нефти и газа	24	0.05	0.65	0.4	10.74

На рисунке 3.6 можно увидеть график исходных уровней временного ряда и полученных при помощи модели Хольта-Винтерса значений сглаженного временного ряда.

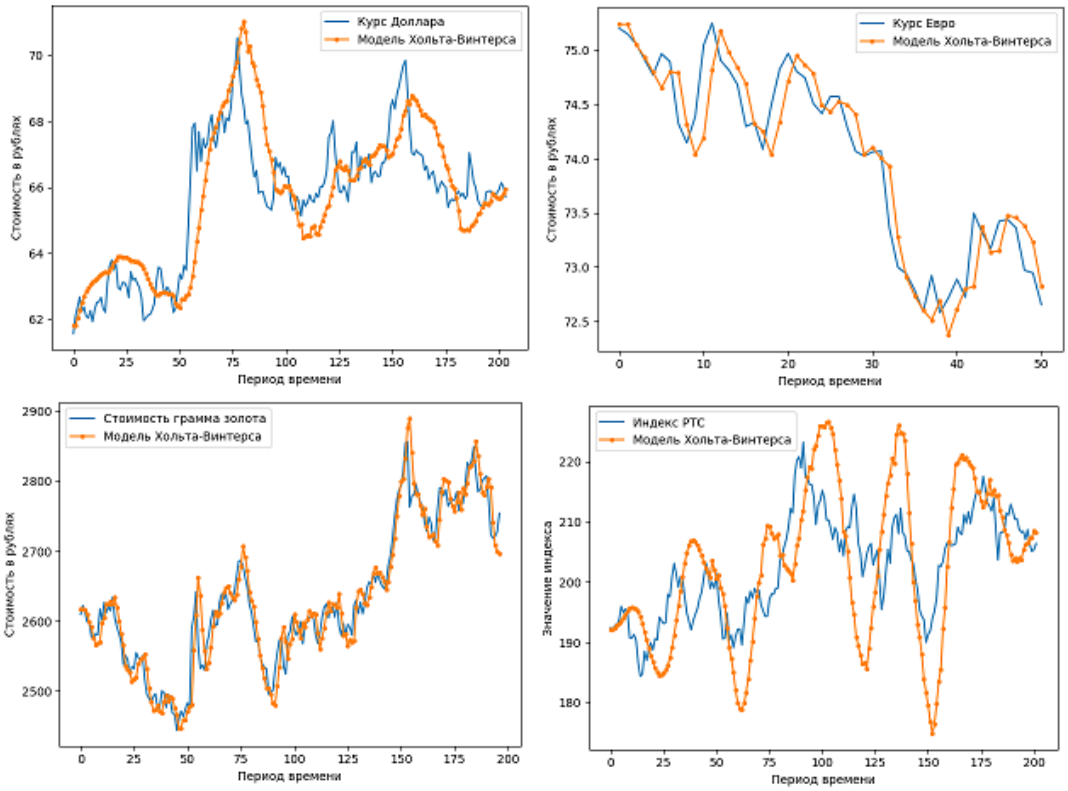

Рисунок 3.6 – Графики исходных временных рядов и сглаженных временных рядов, полученных с помощью модели Хольта-Винтерса

Таким образом, мы выбрали лучшие параметры для моделей экспоненциального сглаживания, которые минимизируют средний квадрат ошибки тестированных данных.

3.4 Подбор параметров модели ARIMA

В главе 3.1 мы выяснили, что ни один из исходных рядов не обладает свойством стационарности. Чтобы получить стационарные ряды, мы взяли первые разности уровней ряда. Именно с этими разностями мы и будем работать.

Для начала отобразим графики автокорреляционной и частной автокорреляционной функции для временного ряда 1 (курс доллара) с лагом 30 (рисунок 3.7).

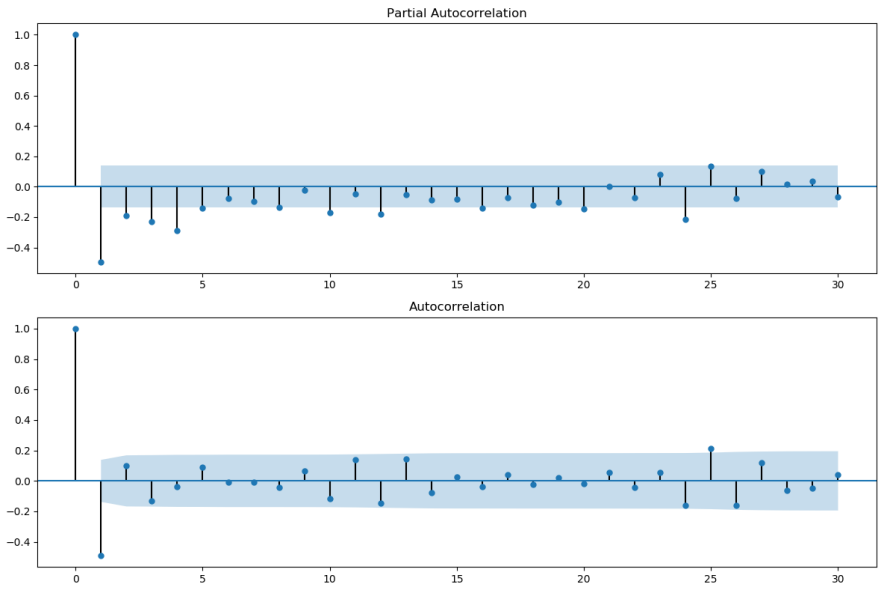

Рисунок 3.7 – Графики АКФ и ЧАКФ

На данных графиках видно, что коэффициенты 1, 2, 3 и 24 ЧАКФ являются значимыми. Значит, в качестве начального порядка процесса авторегрессии мы можем взять p = 1, 2, 3, 4, 24. На графике автокорреляционной функции значимы коэффициенты при 1 и 25 лагах, значит в качестве порядка процесса скользящего среднего можно взять значения q = 1 или 25.

Оценив графики АКФ и ЧАКФ остальных временных рядов и средний квадрат ошибки модели на тестовой выборке, были получены следующие модели *ARIMA*:

Таблица 3.10 – Значения параметров *p,d,q* модели и средний квадрат ошибок

Временной ряд	Параметры модели *ARIMA*	MSE	AIC
Курс Доллара к Рублю	20, 1, 7	0.3034	-1.28138
	1,1,1 без константы	0.275	-1.53216
	25,1,1 без константы	0.236	-1.41808
	4,2,3	0.313	-1.3358
Курс Евро к Рублю	1,1,0	0.0927	-2.56423
	13,1,0	0.0536	-2.61356
	1,1,1 без константы	0.0895	-2.5262
	17,2,3	0.0777	-1.927

Продолжение таблицы 3.10

Временной ряд	Параметры модели *ARIMA*	MSE	AIC
Стоимость грамма золота в рублях	6,1,7	873.003	6.205
	6,2,0	913.099	6.3218
	9,2,1 без константы	882.139	6.2987
	15,1,0	1091.28	6.4019
Индекс РТС нефти и газа	1,2,14	10.8369	1.8065
	19,1,7	19.057	2.3178
	2,1,5 без константы	11.8126	1.74712
	3,1,2 без константы	11.822	1.7514

Из приведенных выше моделей выберем модель с наименьшим значением критерия Акаике. Выбранные модели являются балансом между точностью прогнозирования и количеством параметров, используемых для построения модели. Изобразим графики исходных временных рядов и прогноза, полученного при помощи выбранной модели *ARIMA* (рисунок 3.8).

Выбрав наилучшие модели, проверим их адекватность, оценив остатки используя *Q*-статистику Льюнга-Бокса. Для этого надо рассчитать значения автокорреляции.

Если полученное значение *Q*-статистики больше значения хи-квадрат, то нулевая гипотеза о том, что данные являются случайными (являются белым шумом) отклоняется. В таблице 3.11 приведены рассчитанные значения *Q*-статистики и значения хи-квадрат.

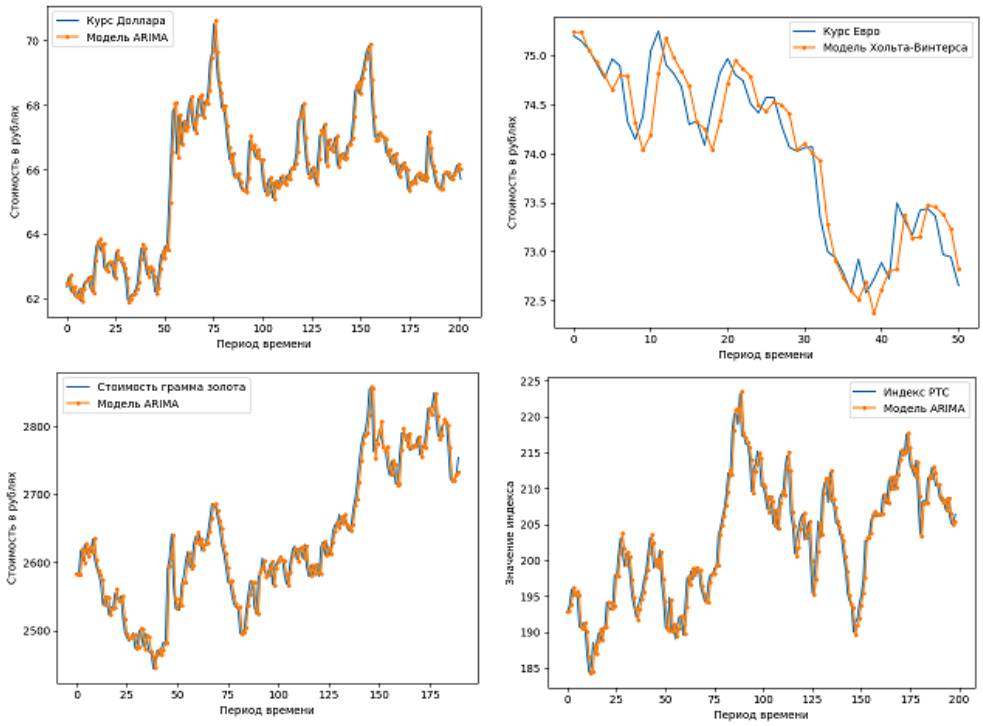

Рисунок 3.8 – График исходных временных рядов и временных рядов, полученных по модели ARIMA

Таблица 3.11 – Рассчитанные значения *Q*-статистики и значения хи-квадрат

Временной ряд	Выбранная модель	*Q*	χ^2
Курс Доллара к Рублю	*ARIMA(1,1,1)* без константы	157.547	288.955
Курс Евро к Рублю	*ARIMA(13,1,0)*	45.3	65.1707
Стоимость грамма золота в рублях	*ARIMA(6,1,7)*	162.857	277.13
Индекс РТС нефти и газа	*ARIMA(2,1,5)* без константы	162.234	287.8815

Так как все рассчитанные значения *Q*-статистики меньше χ^2, то нулевая гипотеза о независимости остатков моделей принимается. Следовательно, все выбранные модели адекватны и могут использоваться для решения задачи прогнозирования.

3.5 Сравнительный анализ выбранных моделей

Будем анализировать и сравнивать точность прогноза выбранных моделей при краткосрочном (шаг в 1 день), среднесрочном (шаг в 10 дней) и долгосрочном (шаг от 20 дней) прогнозировании. Сравнение будет производиться с использованием метрик *MSE, MAE* и *MAPE*.

Произведем краткосрочное прогнозирование, используя выбранные лучшие модели, и рассчитаем метрики. Запишем результаты в таблицу 3.12.

Таблица 3.12 – Расчет ошибок краткосрочного прогнозирования моделей

Временной ряд	Модели	*MSE*	*MAE*	*MAPE*
Курс Доллара к Рублю	Полиномиальный тренд	0.00869	0.09319	0.1422%
	Модель Брауна	0.0747	0.2733	0.416%
	Модель Хольта	0.00731	0.0855	0.1305%
	Модель Хольта- Винтерса	0.00288	0.05363	0.082%
	Модель *ARIMA*	0.00081	0.0285	0.0435%
Курс Евро к Рублю	Полиномиальный тренд	0.0389	0.1973	0.272%
	Модель Брауна	0.0289	0.17	0.234%
	Модель Хольта	0.0554	0.2353	0.324%
	Модель Хольта- Винтерса	0.0489	0.2211	0.305%
	Модель *ARIMA*	0.00083	0.0289	0.0396%
Стоимость грамма золота в рублях	Полиномиальный тренд	2148.099	46.348	1.662%
	Модель Брауна	1.734	1.316	0.048%
	Модель Хольта	32.725	5.721	0.209%
	Модель Хольта- Винтерса	1525.294	39.055	1.447%
	Модель *ARIMA*	209.618	14.48	0.525%
Индекс РТС нефти и газа	Полиномиальный тренд	40.402	6.356	3%
	Модель Брауна	8.95	2.992	1.43%
	Модель Хольта	56.354	7.506	3.525%
	Модель Хольта- Винтерса	11.226	3.35	1.605%
	Модель *ARIMA*	3.436	1.8538	0.0894%

Чем ближе к 0% значение *MAPE*, тем точнее модель делает краткосрочный прогноз. Изобразим процентную ошибку прогноза модели на диаграмме (рисунок 3.9).

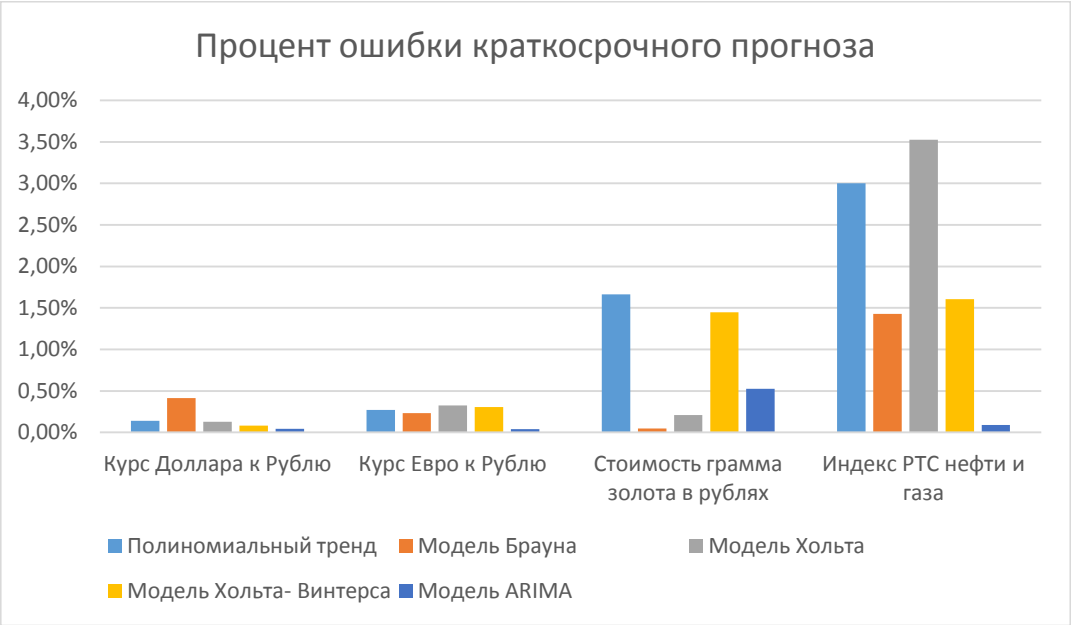

Рисунок 3.9 – Процентная ошибка краткосрочного прогноза моделей

Изучая результаты можно сделать вывод о том, что модель *ARIMA* лучше делает краткосрочные прогнозы, однако для третьего временного ряда модели Брауна и Хольта делают прогноз точнее.

Проведем среднесрочный прогноз, сделав прогнозирование на 10 дней вперед. Запишем результаты в таблицу 3.13.

Анализируя результаты, можно сделать вывод о том, что для среднесрочного прогнозирования лучше всего выбирать модель *ARIMA*, так как именно она дает наименьшую процентную ошибку прогноза. Однако процент ошибки не на много меньше ошибок других моделей.

Таблица 3.13 - Расчет ошибок среднесрочного прогнозирования моделей

Временной ряд	Модели	*MSE*	*MAE*	*MAPE*
Курс Доллара к Рублю	Полиномиальный тренд	0.7685	0.7424	1.137%
	Модель Брауна	1.492	1.0924	1.662%
	Модель Хольта	0.841	0.777	1.1897%
	Модель Хольта- Винтерса	1.358	1.01	1.538%
	Модель *ARIMA*	0.334	0.47304	0.728%
Курс Евро к Рублю	Полиномиальный тренд	0.083	0.2454	0.34%
	Модель Брауна	0.384	0.508	0.7%
	Модель Хольта	0.084	0.2476	0.343%
	Модель Хольта- Винтерса	0.0965	0.25971	0.36%
	Модель *ARIMA*	0.0574	0.21	0.29%
Стоимость грамма золота в рублях	Полиномиальный тренд	4728.683	65.236	2.342%
	Модель Брауна	927.561	23.997	0.876%
	Модель Хольта	425.76	18.029	0.661%
	Модель Хольта- Винтерса	649.77	20.064	0.742%
	Модель *ARIMA*	540.212	20.393	0.742%
Индекс РТС нефти и газа	Полиномиальный тренд	9.7683	2.567	1.21%
	Модель Брауна	25.43	4.439	2.13%
	Модель Хольта	11.104	2.48	1.162%
	Модель Хольта- Винтерса	14.3829	3.094	1.465%
	Модель *ARIMA*	8.288	2.3954	1.135%

Изобразим процентную точность моделей на диаграмме (рисунок 3.10)

Проведем анализ моделей для долгосрочного прогноза (на 20 дней вперед). Результаты тестирования запишем в таблицу 3.14.

Графики построенных прогнозов для тестовых данных и исходные временные ряды можно увидеть в приложении В.

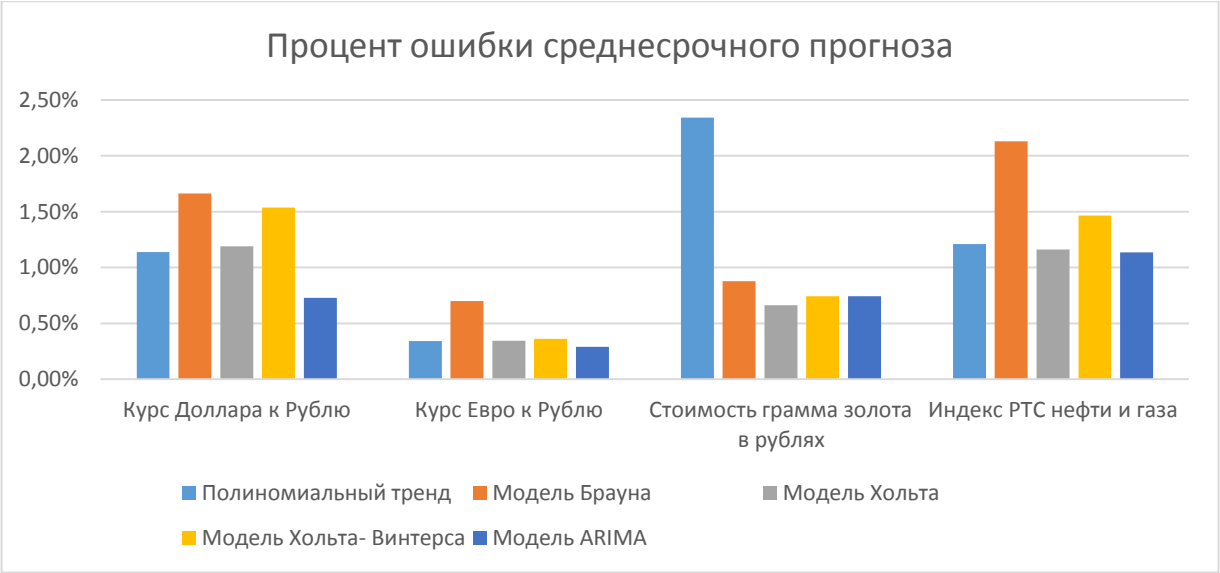

Рисунок 3.10 – Процентная ошибка среднесрочного прогноза моделей

Таблица 3.14 - Расчет ошибки долгосрочного прогноза моделей

Временной ряд	Модели	*MSE*	*MAE*	*MAPE*
Курс Доллара к Рублю	Полиномиальный тренд	0.452	0.533	0.817%
	Модель Брауна	0.90	0.793	1.2067%
	Модель Хольта	0.458	0.508	0.778%
	Модель Хольта- Винтерса	0.714	0.679	1.035%
	Модель *ARIMA*	0.2823	0.4355	0.671%
Курс Евро к Рублю	Полиномиальный тренд	1.727	1.234	1.74%
	Модель Брауна	0.941	0.819	1.12%
	Модель Хольта	0.785	0.749	1.022%
	Модель Хольта- Винтерса	0.906	0.790	1.079%
	Модель *ARIMA*	0.0844	0.24	0.332%
Стоимость грамма золота в рублях	Полиномиальный тренд	4154.3	61.77	2.222%
	Модель Брауна	900.93	25.383	0.926%
	Модель Хольта	323.632	15.0706	0.555%
	Модель Хольта- Винтерса	1420.95	31.5014	1.1766%
	Модель *ARIMA*	371.89	16.2718	0.598%
Индекс РТС нефти и газа	Полиномиальный тренд	13.659	2.96	1.385%
	Модель Брауна	56.79	2.29	3.203%
	Модель Хольта	8.14	2.28	1.06%
	Модель Хольта- Винтерса	8.717	2.43	1.136%
	Модель *ARIMA*	6.78	2.18	1.01%

Изобразим процент ошибок долгосрочного прогнозирования на диаграмме (рисунок 3.11).

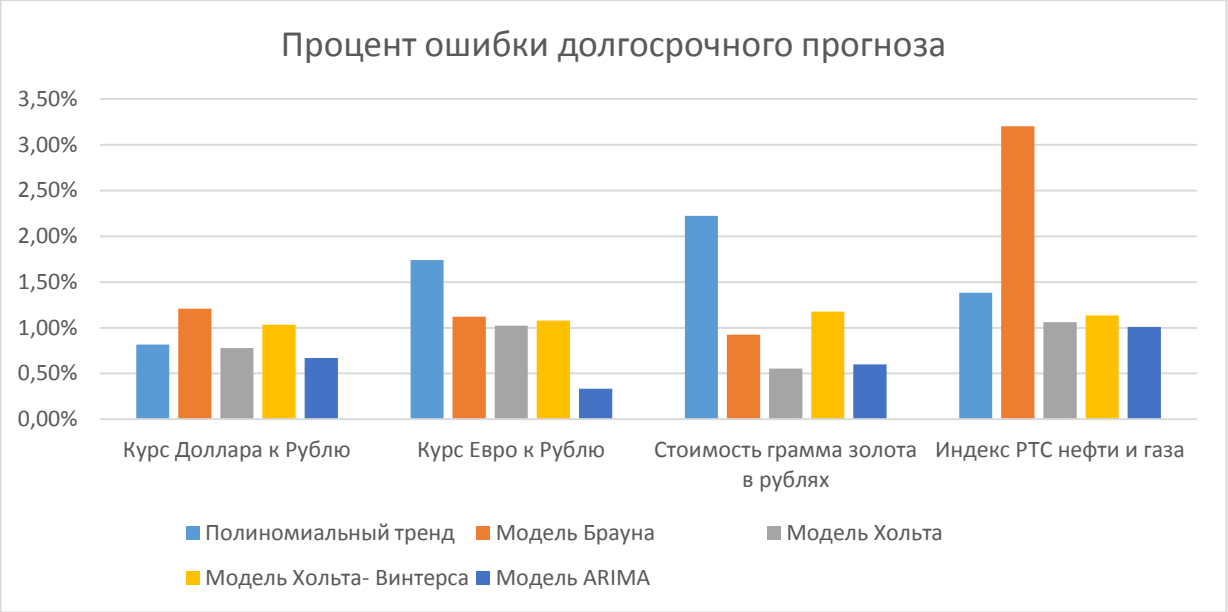

Рисунок 3.11 – Процентная ошибка долгосрочного прогноза моделей

Для удобства, найдем среднюю от найденных процентных ошибок каждой модели и отобразим ее на диаграмме (рисунок 3.12).

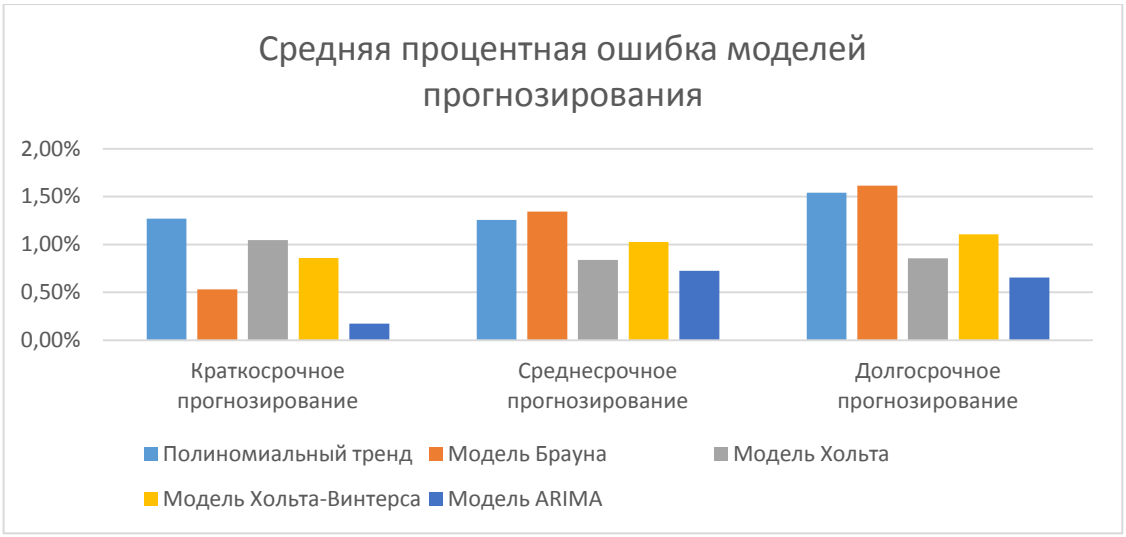

Рисунок 3.12 – Диаграмма среднего значения *MAPE* для каждой модели

Изучая эту диаграмму можно сказать, что для краткосрочного прогнозирования лучше всего выбирать модели *ARIMA* или Брауна, а для среднесрочного или долгосрочного – модели Хольта, Хольта-Винтерса и *ARIMA*. Самой лучшей моделью прогнозирования исходных временных рядов является *ARIMA*, которая хорошо подходит для всех видов прогнозов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной бакалаврской работы были рассмотрены теоретические вопросы анализа и прогнозирования временных рядов, а также рассмотрены метрики оценки точности прогнозирования и методы оценки адекватности построенных моделей.

На объектно-ориентированном языке программирования Python были реализованы модели полиномиального тренда, экспоненциального сглаживания и авторегрессии скользящего среднего.

Был проведен анализ реализованных методов на реальных экономических данных.

В ходе вычислительного эксперимента было выявлено, что модель *ARIMA* выдает самый маленький процент ошибок для всех видов прогноза. Более простые модели – Хольта и Хольта-Винтерса также подходят для среднесрочного и долгосрочного прогнозирования, уступая моделям *ARIMA* в среднем на 0.2%-0.5%.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Айвазян С.А. Методы эконометрики: учебник / С.А. Айвазян. – Магистр: ИНФРА-М, 2010. – 512 с.
2. Арефьева Н.Т. Прогнозирование и его социокультурные цели / Н.Т. Арефьева // Электронный журнал «Знание. Понимание. Умение». – 2010. – № 4 – С. 1.
3. Доусон М. Программируем на Python / М. Доусон. – СПб.: Питер, 2014. – 416 с.
4. Коломыйцев О.А. Обзор методов и подходов к прогнозированию финансовых временных рядов / О.А. Коломыйцев, В.Ю. Шелепов // Сборник научных трудов по материалам международной научно-практической конференции. – 2008. – Т. 8. № 1 – С. 71-73.
5. Орлова И.В. Экономико-математические методы и модели: компьютерное моделирование: учебное пособие / И.В. Орлова, В.А. Половников. – Москва: Вузовский учебник, 2007. – 365 с.
6. Рашка С. Python и машинное обучение / С. Рашка; пер. с англ. А.В. Логунова. – Москва: ДМК Пресс, 2017. – 418 с.
7. Сажин Ю.В. Анализ временных рядов и прогнозирование: учебник / Ю. В. Сажин, А. В. Катынь, Ю. В. Сарайкин. – Саранск: Изд-во Мордов. ун-та, 2013. – 192 с.
8. Светуньков И.С. Методы социально-экономического прогнозирования. Том 1. Теория и методология. / И.С. Светунькова. – Москва: Юрайт, 2015. – 351 с.
9. Трегуб А.В. Методика построения модели ARIMA для прогнозирования динамики временных рядов / А. В. Трегуб // Лесной вестник. – 2011. – № 5. – С. 179-183.
10. Уткин В.Б. Эконометрика / В.Б. Уткин. – Москва: Дашков и К, 2017. – 564 с.

11. Чураков Е.П. Прогнозирование эконометрических временных рядов: учебное пособие / Е.П. Чураков. – Москва: Финансы и статистика, 2008. – 208 с.
12. Dettlng M. Applied Time Series analysis / Dr. M. Dettling: ETH. – 2014. – 176 p.
13. Faraway J. Linear Models with R. / J.J. Faraway: Chapman & Halll/CRC. – 2009. – 255 p.
14. McKinney W. Python for Data Analysis, 2nd Edition / W. McKinney: O'Reilly Media. – 2018. – 541 p.
15. Robert H. Time series analysis and its applications, 3rd Edition / R.H. Shumway, D.S. Stoffer: Springer New York. – 2011. – 202 p.
16. Wayne A. Applied Time Series Analysis with R, 2nd Edition / W.A. Woodward, H.L. Gray, A.C. Elliot: CRC Press. – 2016. – 618 p.
17. William W. S. Time Series Analysis: univariate and multivariate methods, 2nd Edition / W.S. Wei: Addison-Wesley. – 2006. – 634 p.

Электронные ресурсы

18. Доронина А.И. Модели временного ряда: AR (p), MA (q), ARIMA (p,d,q). Пример исследования потребления нефтепродуктов во Франции [Электронный ресурс] / Финансовый Университет при Правительстве Российской Федерации. – Режим доступа: http://www.scienceforum.ru/2014/pdf/1560.pdf
19. Hyndman R.J Forecasting: principles and practice, 2nd Edition [Электронный ресурс] / Melbourne, Australia. – Режим доступа: https://otexts.com/fpp2/
20. Statsmodels's Documentation [Электронный ресурс]. – Режим доступа: https://www.statsmodels.org/

ПРИЛОЖЕНИЕ А.

Листинг модуля models.py

```
import numpy as np
import my_statistics as stat
from matplotlib import pyplot as plt


""" Реализация модели ARIMA
p, d, q - параметры модели
without_constant - ключ, определяющий вид модели: с константой или без нее
"""
class ARIMA:
    def __init__(self, data, p, d, q, without_constant=False):
        self.ma_params = []
        self.ar_params = []
        self.data = data
        self.p = p
        self.d = d
        self.q = q
        self.time_series = list(integration(data=self.data, d=self.d))
        self.size_ts = len(self.time_series)
        self.information = dict()
        self.without_constant = without_constant
        self.y_fit = list()
        self.y_train = list()

    @staticmethod
    def lag_data(data, lag):
        lag_matrix = np.zeros((len(data), lag))
        for l in range(lag):
            for i in range(len(data)):
                if not (i-l-1) < 0:
                    lag_matrix[i, l] = data[i-l-1]
        return lag_matrix

    """
    Метод для нахождения коэффициентов модели ARIMA
    """

    def fit(self):
        # находим массив значений эпсилон (предыдущих ошибок модели)
```

```
        self.errors = ARIMA.lag_data([0] + list(self._find_errors_ar_()),
self.q)
        # определяем матрицы X и Y
        x = ARIMA.lag_data(self.time_series, self.p)
        y = self.time_series[self.p:]
        ones_temp = np.ones((len(x), 1))
        x = np.concatenate([ones_temp, x, self.errors], axis=1)[self.p:]
        # находим оценочный коэффициенты модели
        b = list()
        if self.without_constant:
            b.append(0)
            b = np.append(b, least_square_method(x[:, 1:], y))
        else:
            b = least_square_method(x, y)
        if not np.all(np.abs(np.roots(np.r_[1, -b])) < 1):
             raise ValueError("AR process isn't stationary. Try other
parameters")
        y_fit = x.dot(np.array(b))
        self.errors = y - y_fit
        y_fit = from_integration_data_list(self.data[self.p:], y_fit, self.d)
        y_ = self.data[self.d + self.p:]
        # расчитываем ошибку
        errors = [y - y_ for y_, y in zip(y_fit, y_)]
        self.information = {'MSE': stat.MSE(y_fit, y_),
                            'MAE': stat.MAE(y_fit, y_),
                            'AIC': stat.AIC(self.p + self.q, errors),
                            'MAPE': stat.MAPE(y_fit, y_)}
        # параметры AR модели
        self.ma_params = b[self.p + 1:]
        # параметры MA модели
        self.ar_params = b[:self.p + 1]
        self.y_fit = y_fit
        self.y_train = y_
    """
    Метод для расчеты ошибок модели с помощью модели авторегрессии
    """
    def _find_errors_ar_(self) -> np.ndarray:
        x = np.ones((self.size_ts - 1, 2))
        x[:, 1] = self.time_series[:self.size_ts - 1]
        y = self.time_series[1:]
        b = least_square_method(x, y)
        return y - x.dot(b)
```

```
    """
    Метод для построения прогноза на steps шагов вперед
    """
    def predict(self, steps):
        prediction_list = []
        for _ in range(steps):
            forecast_value = self._forecast_()
            self.time_series.append(forecast_value)
            forecast_value = from_integration_data(self.data[-self.d:],
self.d, forecast_value)
            prediction_list.append(forecast_value)
            self.data = np.append(self.data, forecast_value)
            self.errors = np.append(self.errors, 0)
        return prediction_list

    """
    Метод для построения прогноза на 1 шаг вперед
    """
    def _forecast_(self):
        forecast_value = 0
        for i in range(1, self.p + 1):
            forecast_value += self.ar_params[i] * self.time_series[-i]
        for j in range(1, self.q):
            forecast_value += self.ma_params[j] * self.errors[-j]
        forecast_value += self.ar_params[0]
        return forecast_value

    """
    Метод, переводящий интегрированное значение прогноза delta^d(y)
    в неинтегрированное - y
    """
    def forecast(self):
        forecast_value = self._forecast_()
        return from_integration_data(self.data[-self.d:], self.d,
forecast_value)


class ExponentialSmoothing:
    def __init__(self, time_series, alpha=None, first_='first'):
        self.time_series = time_series
        if alpha is None:
```

```
            self.alpha = 2 / (len(time_series) + 1)
        else:
            self.alpha = alpha
        self.information = dict()
        self.first_val = first_
        self.errors = list()
        self.y_fit = list()

    def fit(self):
        N = len(self.time_series)
        U = np.zeros(N)
        if self.first_val is 'average':
            U[0] = np.mean(self.time_series)
        else:
            U[0] = self.time_series[0]
        for i in range(1, N):
            U[i] = self.alpha * self.time_series[i - 1] \
                   + (1 - self.alpha) * U[i - 1]
        self.y_fit = U
        self.information = {'MSE': stat.MSE(U, self.time_series),
                            'MAE': stat.MAE(U, self.time_series),
                            'MAPE': stat.MAPE(U, self.time_series)}

    def predict(self, h):
        forecast_values = []
        for i in range(h):
            forecast_values.append(self.y_fit[-1] * (1 - self.alpha)
                                   + self.alpha * self.time_series[-1])
        return forecast_values


# Метод Хольта
class DoubleExponentialSmoothing:
    def __init__(self, data, alphas: list):
        self.time_series = data
        if len(alphas) != 2:
            raise ValueError("Must be two alpha, but was get
{0}".format(len(alphas)))
        else:
            self.alpha = alphas
        self.errors = list()
        self.y_fit = list()
```

```
        self.information = dict()
        self.L, self.T = 0, 0

    def fit(self):
        [L, T] = self._find_coefficients_()
        y_pred = list()
        for i in range(len(self.time_series)):
            old_L = L
            L = self.alpha[0]*self.time_series[i] + (1-
self.alpha[0])*(old_L+T)
            T = self.alpha[1]*(L-old_L)+(1-self.alpha[1])*T
            y_pred.append(L+T)
        self.L, self.T = L, T
        self.information = {'MSE': stat.MSE(y_pred, self.time_series),
                            'MAE': stat.MAE(y_pred, self.time_series),
                            'MAPE': stat.MAPE(y_pred, self.time_series)}
        self.errors = [y - y_ for y, y_ in zip(self.time_series, y_pred)]
        self.y_fit = y_pred

    def predict(self, h):
        _pred = list()
        for i in range(h):
            _pred.append(self.L+(i+1)*self.T)
        return _pred

    def _find_coefficients_(self) -> list:
        n = 5
        x = np.ones((n, 2))
        x[:, 1] = [x for x in range(1, n+1)]
        return least_square_method(x, self.time_series[:n])


# Метод Хольта-Винтерса
class HoltWinters:
    def __init__(self, data, s, alphas=None):
        if alphas is None:
            alphas = [0.1, 0.1, 0.1]
        self.data = data
        self.alphas = alphas
        self.s = s
        self.t = len(data)
        self.information = dict()
```

```
        self.errors = list()
        self.y_fit = list()

    def fit(self):
        L = []
        T = []
        S = [1] * self.s
        l, t = self._find_coefficients_()
        L.append(l)
        T.append(t)
        alpha = self.alphas[0]
        betta = self.alphas[1]
        gamma = self.alphas[2]
        y_pred = [L[0] + T[0]]

        for i in range(1, self.s):
            L.append(alpha*(self.data[i]/S[0])+(1-alpha)*(L[i-1]+T[i-1]))
            T.append(betta*(L[i]-L[i-1])+(1-betta)*T[i-1])
            y_pred.append(L[i-1]+T[i-1])

        for i in range(self.s, len(self.data)):
            L.append(alpha * (self.data[i] / S[i-self.s]) + (1 - alpha) *
(L[i - 1] + T[i - 1]))
            T.append(betta * (L[i] - L[i - 1]) + (1 - betta) * T[i - 1])
            S.append(gamma*(self.data[i]/L[i])+(1-gamma)*S[i-self.s])
            y_pred.append((L[i - 1] + T[i - 1])*S[i-self.s])

        self.L = L
        self.T = T
        self.S = S

        self.information = {'MSE': stat.MSE(y_pred, self.data),
                            'MAE': stat.MAE(y_pred, self.data),
                            'MAPE': stat.MAPE(y_pred, self.data)}

        self.y_fit = y_pred
        self.errors = self.data - y_pred

    def _find_coefficients_(self) -> list:
        n = 5
        x = np.ones((n, 2))
        x[:, 1] = [x for x in range(1, n+1)]
```

```
        return least_square_method(x, self.data[:n])

    def predict(self, h):
        _pred = list()
        h_ = int(h/self.s)+1
        for i in range(h):
            _pred.append((self.L[-1]+(i+1)*self.T[-1])*self.S[(i+1)+h-
self.s*h_])
        return _pred

class PolynomialRegression:
    def __init__(self, data, p=1):
        self.data = data
        self.params = list()
        self.information = dict()
        self.p = p
        self.y_fit = list()
        self.errors = list()

    def fit(self):
        x = np.ones((len(self.data), self.p+1))
        for i in range(1, self.p+1):
            x[:, i] = [x_**i for x_ in range(1, len(x)+1)]
        b = least_square_method(x, self.data)
        y_fit = x.dot(b)
        self.params = b
        self.information = {'MSE': stat.MSE(y_fit, self.data),
                            'MAE': stat.MAE(y_fit, self.data),
                            'MAPE': stat.MAPE(y_fit, self.data),
                            'R-quad': stat.R_quad(self.data, y_fit),
                            'correct r-quad': stat.cor_R_quad(self.data,
y_fit, len(x[1,:]-1))}
        self.errors = self.data - y_fit
        self.y_fit = y_fit

    def predict(self, start_time, end_time):
        predict_val = []
        for i in range(start_time, end_time):
            val = 0
            for j in range(1, self.p+1):
                val += i**j*self.params[j]
            val += self.params[0]
```

```
            predict_val.append(val)
        return predict_val



def integration(data, d):
    if d == 0:
        return data
    data_ = data
    for _ in range(d):
        integr_data = np.empty(len(data_)-1)
        for i in range(len(data_) - 1):
            integr_data[i] = (data_[i + 1] - data_[i])
        data_ = integr_data
    return integr_data


def from_integration_data(last_values, d, integr_data):
    if d == 0:
        return integr_data
    if d == 1:
        return integr_data + last_values
    elif d == 2:
        return integr_data + 2 * last_values[1] - last_values[0]
    else:
        raise ValueError("d must be 1 or 2")


def from_integration_data_list(not_integr_data, integr_data, d):
    if d == 0:
        return integr_data
    new_data = list()
    if d == 1:
        for i in range(len(integr_data)):
            new_data.append(not_integr_data[i] + integr_data[i])
        return new_data
    elif d == 2:
        for i in range(len(integr_data)):
            new_data.append(integr_data[i] + 2 * not_integr_data[i+1] -
not_integr_data[i])
        return new_data
    else:
```

```
        raise ValueError("d must be 1 or 2")


def least_square_method(x, y):
    x, y = np.array(x), np.array(y)
    try:
        b = np.linalg.inv(x.T.dot(x)).dot(x.T).dot(y)
    except np.linalg.LinAlgError:
        b = np.linalg.pinv(x).dot(y)
    finally:
        return b
```

ПРИЛОЖЕНИЕ Б.

Листинг модуля my_statistics.py

```
import numpy as np
import statsmodels.api as sm
from scipy.stats import f, chi2
from statsmodels.stats.diagnostic import acorr_ljungbox

# среднеквадратичная ошибка
def MSE(pred, true):
    if len(pred) != len(true):
        raise ValueError("Length prediction values not equal length true
values (%d)!=(%d)" % (len(pred), len(true)))
    mse = 0
    for i in range(len(pred)):
        mse += (pred[i] - true[i])**2
    return mse/len(pred)

# критерий Акаике
def AIC(r, errors):
    L = RSS(errors)
    n = len(errors)
    return r/n + np.log(L/n)

def BIC(r, errors):
    sigma = RSS(errors)
    n = len(errors)
    return r/n*np.log(n) + np.log(sigma/n)

def RSS(errors):
    return sum([x*x for x in errors])

def MAPE(pred, true):
    delta = 0
    for p, t in zip(pred, true):
        delta += abs((p-t)/t)
    return delta*100/len(pred)

def MAE(pred, true):
    mae = 0
    for p, t in zip(pred, true):
        mae += abs(p-t)
```

```
    return mae/len(pred)

def R_quad(y_true, y_pred):
    R = 1
    S_1 = sum([(y-x)**2 for x, y in zip(y_pred, y_true)])
    S_2 = sum([(y-np.mean(y_true))**2 for y in y_true])
    return (R - (S_1/S_2))


def cor_R_quad(y_true, y_pred, k):
    R_2 = R_quad(y_true, y_pred)
    cor_r_quad = 1 - ((len(y_true)-1)/(len(y_true)-k-1))*(1-R_2)
    return cor_r_quad


def DW_criterion(errors):
    DW = 0
    for i in range(1, len(errors)):
        DW += (errors[i] - errors[i-1])**2
    DW = DW/sum([x*x for x in errors])
    return DW


def is_stationary(data):
    test = sm.tsa.adfuller(data)
    print('adf: ', test[0])
    print('p-value: ', test[1])
    print('Critical values: ', test[4])
    return test[0] < test[4]['5%']


def criterion_fisher(y, y_fit, k):
    n = len(y)
    mean = sum(y)/n
    dy = (sum((y-mean)**2))/(n-1)
    dad = (sum((y-y_fit)**2))/(n-k)
    return [dy/dad, f.ppf(0.95, n-1, n-k)]

def ljung_box(residuals):
    q_test = sm.tsa.stattools.acf(residuals, nlags=len(residuals)-2)
    Q = 0
    N = len(residuals)
```

```
print(q_test)
for i in range(1, len(q_test)):
    Q += q_test[i] ** 2 / (N - (i + 1))
Q = N*(N+2)*Q
print(len(q_test))
return [Q, chi2.ppf(0.95, len(residuals)-2)]
```

ПРИЛОЖЕНИЕ В.

Графики исходных временных рядов и выбранных моделей

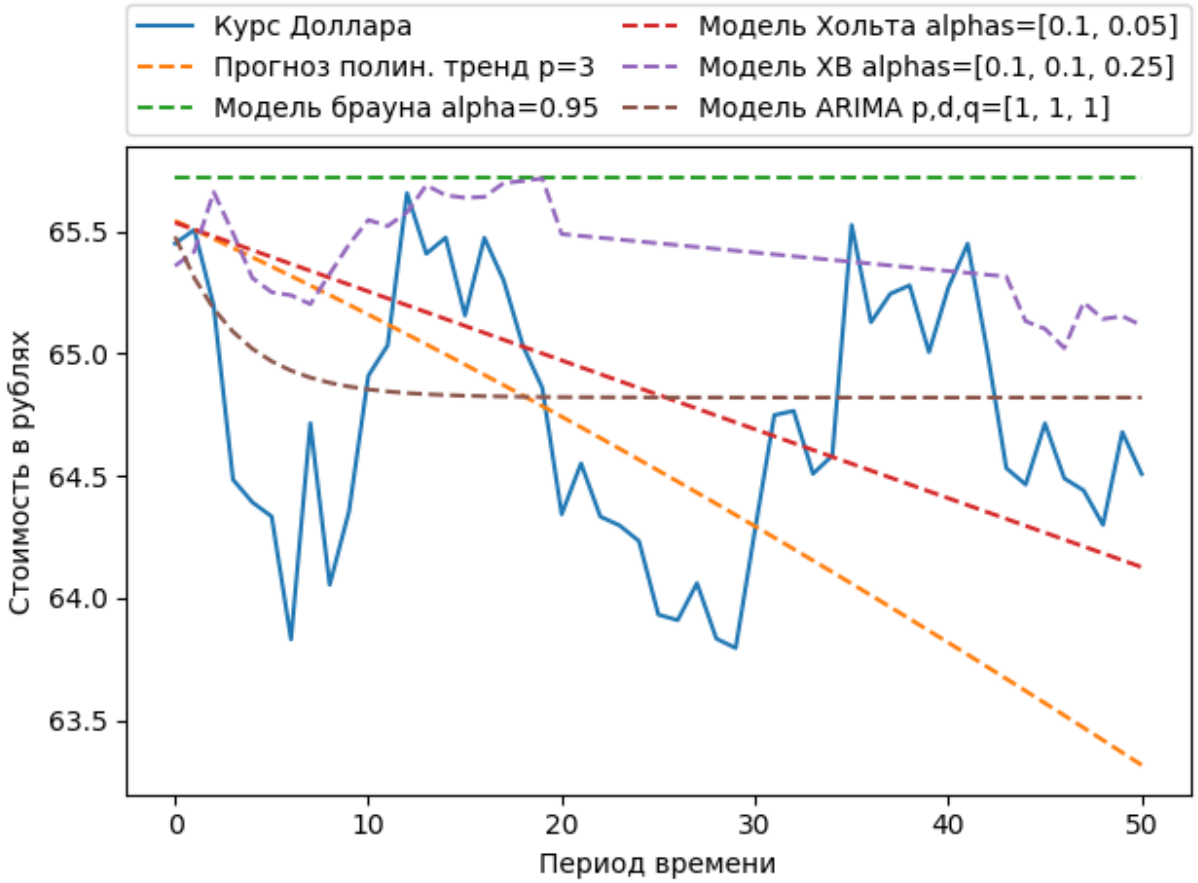

Рисунок В.1 – Исходный ряд «Курс Доллара» для тестирования и прогноз моделей

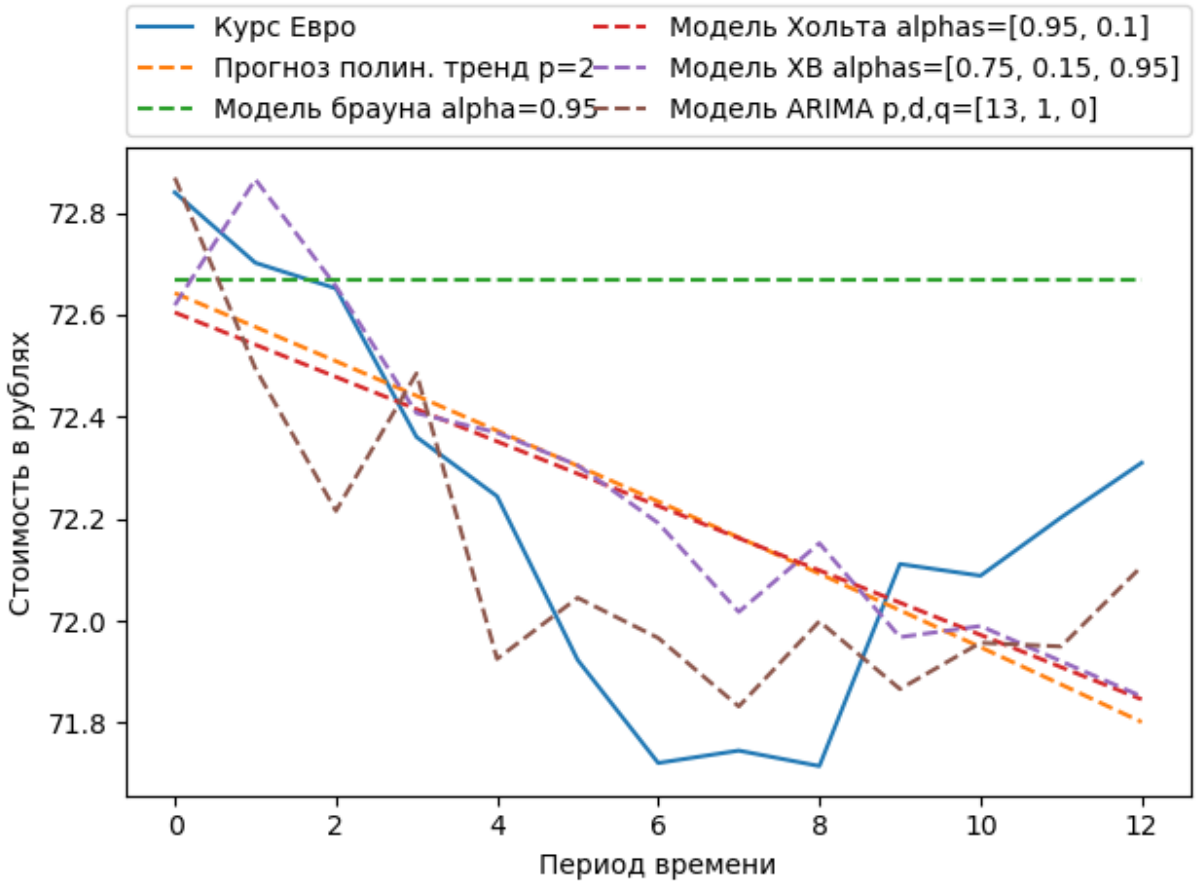

Рисунок В.2 – Исходный ряд «Курс Евро» для тестирования и прогноз моделей

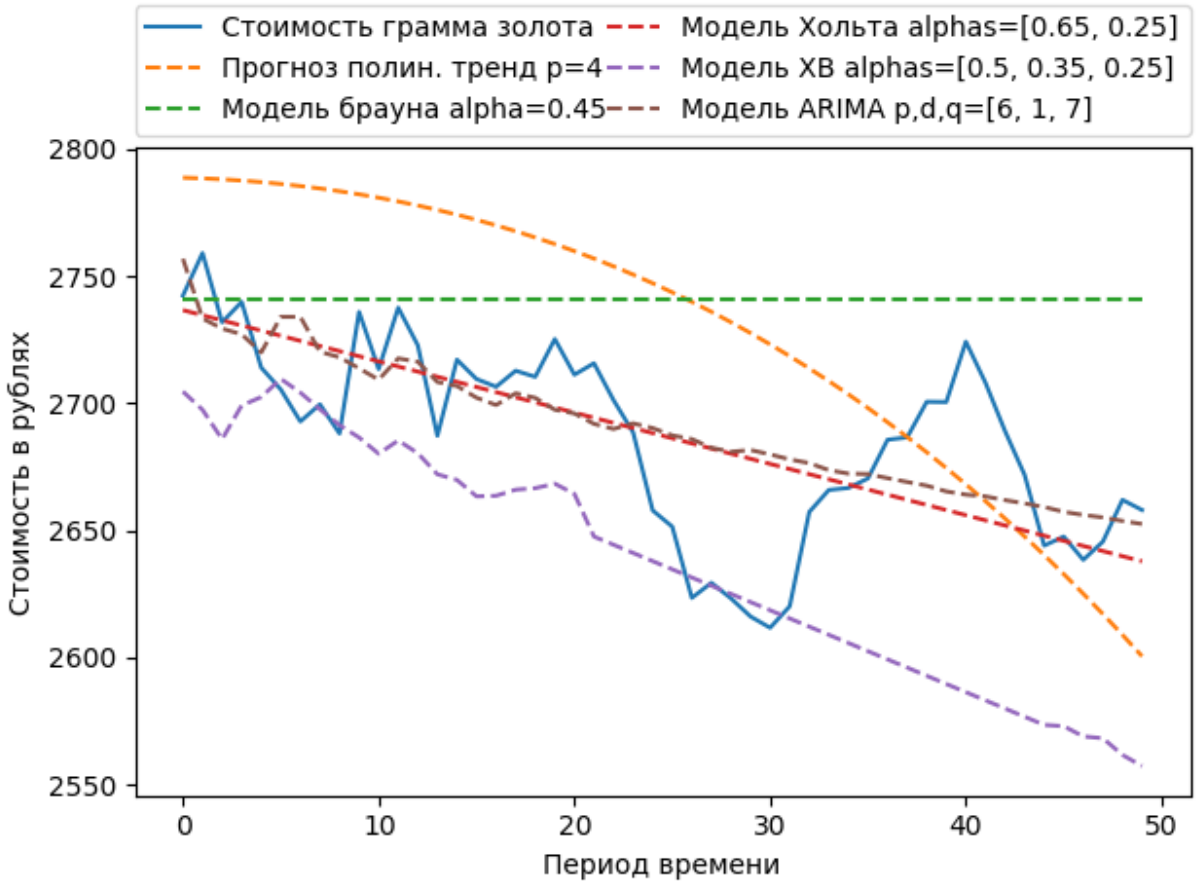

Рисунок В.3 – Исходный ряд «Стоимость грамма золота» для тестирования и прогноз моделей

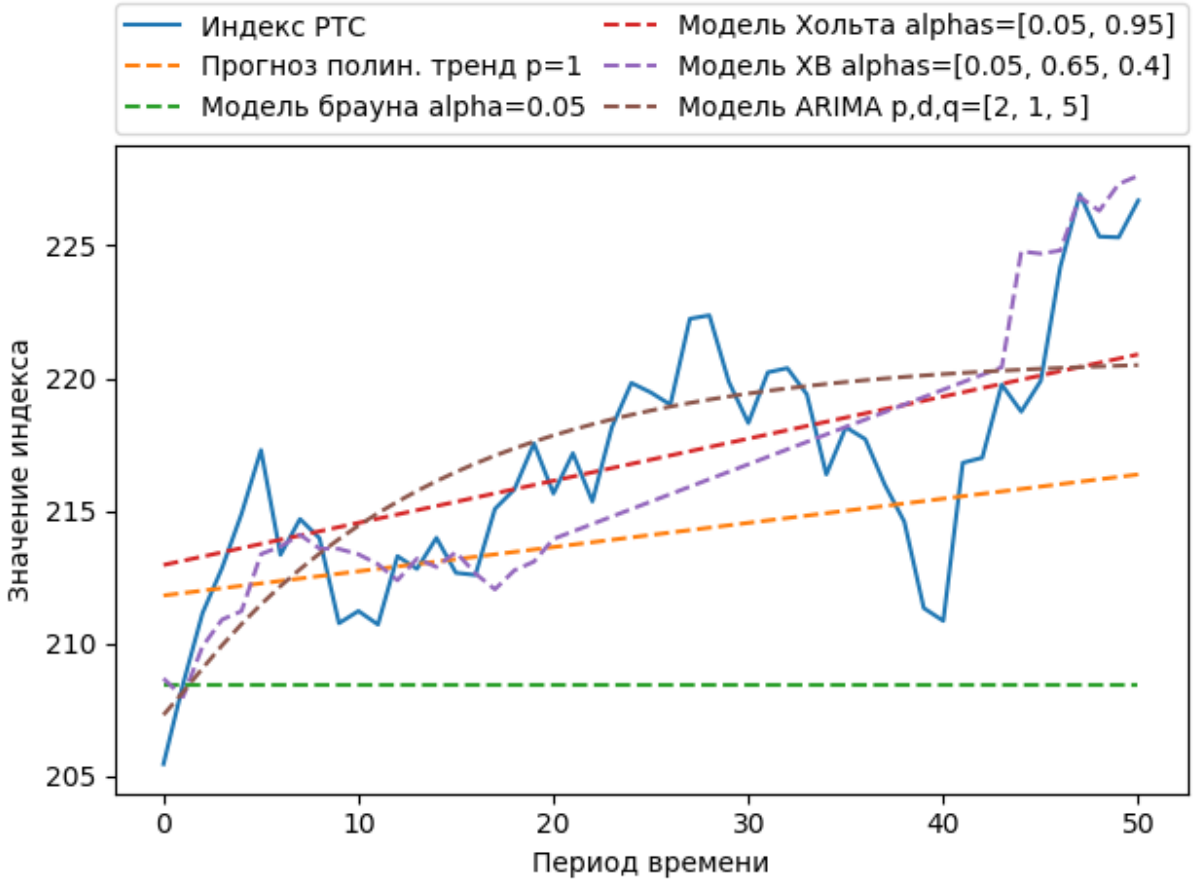

Рисунок В.4 – Исходный ряд «Индекс РТС» для тестирования и прогноз моделей