Оглавление

Теоретическая часть 3

1.1 Временные ряды 3

1.2 Стационарность и нестационарность временных рядов 3

1.3 Дифференцирование и логарифмирование 4

1.4 Алгоритм Бокса-Кокса 4

1.5 Разложение временного ряда 5

1.6 Понятие псевдопоследовательности временного ряда 6

1.7 Алгоритм Кристофа Бергмайра 6

1.8 Ансамблевая модель беггинг 7

1.9 Линейный бутстреп 7

1.9.1 Описание алгоритма 7

1.9.2 Оценка матрицы автоковариации 8

1.9.3 Разложение Холецкого 9

1.10 Модели прогнозирования временных рядов 10

1.10.1 ARIMA 10

1.10.2 ETS 10

1.11 ACF и PACF 11

Практическая часть 13

2.1 Используемые инструменты 13

2.2 Наборы данных 14

2.3 Анализ данных о доходах федерального бюджета 16

2.4 Анализ данных об индексе производства в секторе "Обеспечение электрической энергией, газом и паром" 19

2.5 Анализ данных о кредиторской задолженности предприятия 22

Заключение 25

Список литературы 26

Приложение 27

Глава 1. Теоретическая часть

1.1. Временные ряды

Временной ряд представляет собой набор наблюдений, полученных путем регулярного измерения одной переменной в течение некоторого периода времени.

Основные компоненты временных рядов:

- **Тренд**: Тренд представляет собой долгосрочное направление изменения временного ряда. Он может быть восходящим, нисходящим или плоским (отсутствие явного направления). Тренд может быть линейным или нелинейным.
- **Сезонность**: Сезонность отображает периодические колебания во временном ряде, которые повторяются с постоянным интервалом. Она может быть годовой, квартальной, месячной, недельной или даже более короткой. Сезонность может иметь фиксированную амплитуду или меняться во времени.
- **Цикличность**: Цикличность представляет собой долгосрочные колебания временного ряда, которые не являются периодическими. Циклы могут иметь разную продолжительность и амплитуду, и они могут быть связаны с экономическими, социальными или другими факторами.
- **Шум**: Шум, или случайная составляющая, представляет собой нерегулярные и случайные колебания во временном ряде, которые не поддаются объяснению трендом, сезонностью или цикличностью. Шум может быть вызван случайными событиями или ошибками измерения.

1.2. Стационарность и нестационарность временных рядов

Временные ряды можно разделить на стационарные и нестационарные. Ряд называется **стационарным**, если:

- Среднее значение ряда не меняется со временем и остается постоянным:

$$E(y_1) = E(y_2) = E(y_3) = ...$$

- Дисперсия ряда не зависит от времени и остается постоянной;

$$Var(y_1) = Var(y_2) = Var(y_3) = ... = \gamma_0$$

- Автокорреляция (корреляция между значениями ряда в различные моменты времени) не зависит от времени и остается постоянной;

$$Cov(y_1, y_2) = Cov(y_2, y_3) = Cov(y_3, y_4) = ... = \gamma_1$$

$$Cov(y_1, y_3) = Cov(y_2, y_4) = Cov(y_3, y_5) = ... = \gamma_2$$

$$...$$

Стационарные временные ряды обладают свойством предсказуемости, поскольку их статистические свойства не меняются со временем. Это позволяет применять различные статистические модели для предсказания будущих значений.

Нестационарные временные ряды, в свою очередь, не удовлетворяют вышеперечисленным характеристикам стационарности. Они могут иметь тренд, сезонность и/или изменяющуюся дисперсию. Такие ряды могут быть сложными для прогнозирования, поскольку их статистические свойства изменяются со временем.

1.3. Дифференцирование и логарифмирование

Для анализа и предсказания нестационарных временных рядов часто применяются методы преобразования, такие как дифференцирование и логарифмирование, чтобы сделать ряд стационарным. Это позволяет применять модели, разработанные для стационарных рядов, для прогнозирования будущих значений.

Дифференцирование временных рядов является одним из методов преобразования для достижения стационарности. Оно заключается в вычитании значения ряда в текущем моменте времени из значения ряда в предыдущем моменте времени. Этот процесс приводит к получению нового ряда, называемого разностным рядом.

Дифференцирование помогает устранить тренды и сезонность, которые могут присутствовать в исходном ряде, и сделать его более стационарным. После применения дифференцирования, разностный ряд может отвечать стационарным характеристикам, таким как постоянное среднее значение, постоянная дисперсия и постоянная автокорреляция.

Логарифмирование временных рядов - это еще один метод преобразования, используемый для изменения свойств ряда и достижения стационарности. Он заключается в применении логарифмической функции к значениям временного ряда.

1.4. Алгоритм Бокса-Кокса

Алгоритм Бокса-Кокса - это статистический метод преобразования данных, который используется для стабилизации дисперсии и приближения распределения данных к нормальному. Этот метод был предложен статистиками Джорджем Боксом и Дэвидом Коксом (1964 г.).

Общая формула преобразования Бокса-Кокса имеет вид:

$$w_t = \begin{cases} \ln y_t & \lambda = 0; \\ \frac{y_t^\lambda - 1}{\lambda} & \lambda \neq 0. \end{cases}$$

где y_t - исходные данные, а λ - параметр преобразования. Значение λ выбирается таким образом, чтобы достичь наилучшего приближения к нормальному распределению. Это обычно делается путем поиска значения λ, которое максимизирует логарифм правдоподобия данных.

Процедура выбора оптимального значения λ включает в себя следующие шаги:

1. Определение диапазона значений λ;
2. Вычисление преобразованных данных для каждого значения λ;
3. Оценка параметра λ: Применяются методы оптимизации или статистические методы для оценки параметра λ, который максимизирует логарифм правдоподобия данных;
4. Выбор оптимального преобразования.

1.5. Разложение временного ряда

Разложение временного ряда - метод анализа данных, выделяющий тренд (долгосрочные изменения), сезонность (повторяющиеся паттерны) и случайную составляющую (нерпредсказуемые изменения).

Существуют две основные модели разложения временного ряда:

- **Аддитивная модель**: Исходный временной ряд представляется в виде суммы тренда, сезонности и случайной составляющей.
- **Мультипликативная модель**: Исходный временной ряд представляется в виде произведения тренда, сезонности и случайной составляющей.

STL (Seasonal and Trend decomposition using Loess) - это метод декомпозиции временных рядов, который использует локально взвешенное сглаживание (Loess) для выделения тренда, сезонности и остатка.

Loess (Locally Weighted Scatterplot Smoothing) - это метод сглаживания данных, который использует локальные веса для учета изменения плотности данных в различных участках графика.

Методы Loess и STL обеспечивают гибкое и адаптивное разложение временных рядов, учитывая локальные изменения. Их применение в анализе временных рядов обеспечивает более эффективное управление сложностью данных, выявляя тренд, сезонность и остаток. Эти методы важны для построения точных прогностических моделей и более глубокого понимания структуры временных данных.

1.6. Понятие псевдопоследовательности временного ряда

Псевдопоследовательность временного ряда представляет собой последовательность, которая строится на основе исходного временного ряда с использованием различных методов анализа и модификации. Этот подход позволяет проводить анализ данных с целью выделения определенных свойств или характеристик временного ряда.

Существуют следующие подходы для получения псевдопоследовательностей из временного ряда:

1. Амплитудная модификация: Изменение амплитуды значений временного ряда (умножение на константу) или добавление случайного шума к значениям временного ряда;
2. Генерация случайных данных: Создание псевдопоследовательностей путем генерации случайных данных с использованием различных распределений;
3. Интерполяция: Заполнение пропущенных значений или создание дополнительных точек с использованием методов интерполяции;
4. Экстраполяция: Продление временного ряда за пределы существующих данных.
5. Смешивание временных рядов и др.

Использование псевдопоследовательностей в анализе временных рядов может привести к потере реальных свойств данных, переобучению моделей, ошибочным выводам, неточным прогнозам и искажению статистических свойств. Поэтому необходимо осторожно выбирать методы и алгоритмы при создании псевдопоследовательности, чтобы минимизировать потерю информации и сохранить основные характеристики исходного временного ряда.

1.7. Алгоритм Кристофа Бергмайра

Алгоритм Кристофа Бергмайра для временных рядов - метод декомпозиции временных рядов, разработанный для высокоточного прогнозирования. Он объединяет аддитивную и мультипликативную декомпозицию, а также использует авторегрессионные модели.

Алгоритм состоит из следующих этапов:

1. Преобразование исходного временного ряда с помощью метода Бокса-Кокса;
2. Декомпозиция преобразованного временного ряда на тренд, сезонность и остаток;
3. Применение алгоритма генерации псевдопсоледовательсноти к ряду остатков;
4. Сложение рядов тренда, сезонности и полученного ряда остатков;
5. Применение обратной функции Бокса-Кокса к полученному ряду.

Полученный ряд представляет собой псевдопоследовательность для исходного временного ряда.

1.8. Ансамблевая модель беггинг

Беггинг (Bootstrap Aggregating) - метод ансамблирования в машинном обучении, который спроектирован для улучшения стабильности и точности моделей. Основная идея бэггинга заключается в том, чтобы обучить несколько моделей на различных подмножествах обучающих данных и объединить их прогнозы.

Основные этапы беггинга:

1. Создание бутстреп-выборок: для каждой модели создаются случайные подмножества обучающих данных путем выбора с повторениями (бутстрэп-выборка). Это позволяет одним и тем же данным появляться в различных подмножествах;

2. Обучение моделей: на каждой бутстрэп-выборке обучается отдельная модель. Различия в данных поддерживают вариативность моделей, что делает их более разносторонними;

3. Агрегация прогнозов: прогнозы отдельных моделей агрегируются, часто путем усреднения (для задач регрессии) или голосования (для задач классификации).

1.9. Линейный бутстреп

1.9.1. Описание алгоритма

Линейный бутстреп (Linear Process Bootstrap, LPB) - метод ресемплинга, применяемый для анализа и прогнозирования временных рядов.

Этапы алгоритма LPB для временного ряда $(X_1, \ldots, X_n)$:

1. Найти последовательность отклонений от среднего:

$$Y_i = X_i - \bar{X} \text{ для } i = 1, \ldots, n$$

$$Y = (Y_1, \ldots, Y_n)^T$$

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

2. Вычислить вектор W:

$$W = (\Sigma^{\hat{\epsilon}}_{\kappa,l})^{-1/2} Y$$

где $\Sigma^{\hat{\epsilon}}_{\kappa,l}$ - оценка матрицы автоковариации,

$(\Sigma^{\hat{\epsilon}}_{\kappa,l})^{1/2}$ - нижнетреугольная матрица L разложения Холецкого $\Sigma^{\hat{\epsilon}}_{\kappa,l} = LL^T$

3. Найти Z - стандартизированную версию W, где:

$$Z_i = \frac{W_i - \bar{W}}{\hat{\sigma}_W}$$

$$\bar{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$$

$$\hat{\sigma}_W^2 = \frac{1}{n}\sum_{i=1}^{n}(W_i - \bar{W})^2$$

4. Сгенерировать $Z_1^*, \ldots, Z_n^*$ с помощью бутстрепа с НОР выборками из $Z_1, \ldots, Z_n$
5. Вычислить $Y^* = (\Sigma_{\kappa,l}^{\hat{\epsilon}})^{1/2} Z^*$

1.9.2. Оценка матрицы автоковариации

Пусть $X_1, \ldots, X_n$ - реализация стационарного процесса с нулевым средним $\{X_t\}_{t\in\mathbb{Z}}$, а $\gamma_k = \mathrm{cov}[X_0, X_k]$ - функция автоковариации, тогда матрица автоковариации:

$$\Sigma_n = [\gamma_{|i-j|}]_{i,j=1}^{n}$$

Автоковариация γ_k с k-ым лагом имеет естественную оценку, предоставляемую выборочной автоковариацией:

$$\hat{\gamma}_k = \frac{1}{n}\sum_{i=1}^{n-k} X_i X_{i+k}$$

Однако подстановка $\hat{\gamma}_k$ вместо γ_k в Σ_n не срабатывает, потому что

$$\hat{\Sigma}_n = [\hat{\gamma}_{|i-j|}]_{i,j=1}^{n}$$

не является состоятельной оценкой Σ_n, так как операторная норма $\Sigma_n - \hat{\Sigma}_n$ не сходится к нулю.

Поэтому оценка будет проводиться с помощью матрицы $\hat{\Sigma}_{\kappa,l} = [w_{|i-j|}\hat{\gamma}_{|i-j|}]_{i,j=1}^{n}$, где $w_{|i-j|}$ - весовая функция, которая уменьшает веса значений $\hat{\gamma}_{|i-j|}$ при больших значениях $|i-j|$. Это необходимо, поскольку, известно, что оцененнные ковариации с большими значениями $|i-j|$ менее надежны (см., например, Brockwell и Davis (1991)).

Обозначим весовую функцию $\kappa(\cdot)$ и определим следующим образом:

$$\kappa(x) = \begin{cases} 1, & \text{если } |x| \le 1 \\ g(|x|), & \text{если } 1 < |x| \le c_k \\ 0, & \text{если } |x| > c_k \end{cases}$$

где $g(\cdot)$ - фунция, удовлетворяющая условию $g(|x|) < 1$, а c_k - константа, удовлетворяющая условию $c_k \geq 1$. Локализованная версия весовой функции $\kappa(\cdot)$ с коэффициентом масштабирования l будет обозначаться следующим образом:

$$\kappa_l(x) = \kappa(x/l)$$

Тогда оценка матрицы Σ_n будет выглядеть следующим образом:

$$\hat{\Sigma}_{\kappa,l} = [\kappa_l(|i-j|)\hat{\gamma}_{|i-j|}]_{i,j=1}^{n}$$

Простым примером весовой функции, который мы и будем использовать является трапеция, предложенная Политис и Романо (1995):

$$\kappa(x) = \begin{cases} 1, & \text{если } |x| \leq 1 \\ 2-|x|, & \text{если } 1 < |x| \leq 2 \\ 0, & \text{если } |x| > 2 \end{cases}$$

Однако $\hat{\Sigma}_{\kappa,l}$ не гарантированно является положительно определенной для конечных выборок. Модифицированная оценка достигает этой цели без понижения точности. Рассмотрим спектральное разложение:

$$\hat{\Sigma}^{\epsilon}_{\kappa,l} = T_n D^{\epsilon} T_n^T$$

где $D^{\epsilon} = \text{diag}(d_1^{\epsilon}, \ldots, d_n^{\epsilon})$ и $d_i^{\epsilon} = \max(d_i, \epsilon\hat{\gamma}_0/n^{\beta})$. Здесь β и ϵ - положительные константы, определенные пользователем.

В рамках данной работы было установлено, что $\beta = 1$, $\epsilon = 1$.

1.9.3. Разложение Холецкого

Разложение Холецкого - представление симметричной положительно определенной матрицы A в виде произведения нижнетреугольной матрицы L на ее транспонированную.

Предположим, что у нас есть симметричная положительно определенная матрица A, то есть $A = A^T$ и для любого вектора $x \neq 0$ выполняется $x^T A x > 0$.

Тогда существует нижнетреугольная матрица L такая, что $A = LL^T$. Эта матрица L может быть получена из разложения Холецкого, которое строится следующим образом:

1. Пусть a_{ij} - элемент матрицы A. Для $i = j$ элементы матрицы L вычисляются по формуле:

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{j-1} l_{ik}^2}$$

2. Для $i > j$ элементы матрицы L вычисляются по формуле:

$$l_{ij} = \frac{1}{l_{jj}}(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk})$$

Таким образом, процесс построения разложения Холецкого начинается с первого столбца матрицы L и постепенно заполняет его элементы, а затем переходит ко второму столбцу и так далее.

1.10. Модели прогнозирования временных рядов

1.10.1. ARIMA

ARIMA - это модель, основанная на комбинации трех компонентов: авторегрессии (AR), интегрирования (I) и скользящего среднего (MA).

- **Компонента авторегрессии (AR)** отражает зависимость текущего значения ряда от предыдущих значений. Процесс авторегрессии - это стационарный процесс вида:

$$y_t = c + b_1 y_{t-1} + b_2 y_{t-2} + ... + b_p y_{t-p} + \varepsilon_t$$

- **Компонента интегрирования (I)** применяется для обеспечения стационарности ряда. Если исходный ряд не является стационарным, он подвергается дифференцированию, чтобы устранить тренды и сезонность.
- **Компонента скользящего среднего (MA)** моделирует зависимость текущего значения ряда от случайных ошибок предыдущих моментов времени. Процесс скользящего среднего - процесс, представляемый в виде:

$$y_t = \mu + \varepsilon_t + a_1 \varepsilon_{t-1} + ... + a_q + \varepsilon_{t-q}$$

ARIMA-модель имеет параметры p, d и q, где p относится к порядку авторегрессии, d - к порядку интегрирования и q - к порядку скользящего среднего. Выбор оптимальных значений этих параметров осуществляется с помощью методов, таких как анализ автокорреляционной и частной автокорреляционной функций ряда.

1.10.2. ETS

ETS - это модель экспоненциального сглаживания, которая также используется для анализа и прогнозирования временных рядов. Она основана на простой идеи сглаживания и взвешивания предыдущих значений ряда.

ETS-модель состоит из трех компонентов: уровня, тренда и сезонности. Каждая компонента может быть присутствовать или отсутствовать в модели, что позволяет адаптировать модель к различным типам временных рядов.

- **Компонента уровня** отражает базовый уровень временного ряда и представляет его среднее значение без тренда и сезонности.
- **Компонента тренда** моделирует долгосрочные изменения в ряде. Она может быть линейной или нелинейной, в зависимости от типа тренда в данных.
- **Компонента сезонности** учитывает периодические колебания в ряде. Она может быть аддитивной или мультипликативной, в зависимости от того, как сезонность влияет на вариацию ряда.

ETS-модель также имеет различные варианты, такие как ETS(AAA), ETS(ANM), ETS(MMM) и другие, которые указывают на наличие или отсутствие компонент уровня, тренда и сезонности в модели.

1.11. ACF и PACF

Автокорреляционная функция (ACF)- это инструмент анализа зависимости между значениями временного ряда на различных лагах. ACF измеряет корреляцию между текущим значением ряда и его предыдущими значениями на разных временных отступах. Она рассчитывается по следующей формуле:

$$\rho_h = \frac{\sum_{t=h+1}^{N}(y_t - \bar{y})(y_{t-h} - \bar{y})}{\sum_{t=h+1}^{N}(y_t - \bar{y})^2}$$

- y_t- значение временного ряда в момент времени t
- $\bar{y}$- среднее значение временного ряда
- N- общее количество наблюдений

ACF принимает значения от -1 до 1, где 1 означает положительную корреляцию, -1 означает отрицательную корреляцию, а 0 означает отсутствие корреляции. Относительная сила и статистическая значимость значений ACF помогают анализировать и интерпретировать зависимость между значениями временного ряда на разных лагах.

Частная автокорреляционная функция (PACF) является мерой корреляции между значениями временного ряда на разных лагах, учитывая влияние промежуточных лагов. Она помогает исследовать прямую связь между значениями ряда на определенном лаге, исключая влияние промежуточных лагов.

$$\phi_k = Cor(y_t - P(y_t), y_{t-k} - P(y_{t-k}))$$

$P(y_t)$ - проекция случайной величины y_t на линейную оболочку величин $y_{t-1}, y_{t-2}, \ldots, y_{t-k+1}$

PACF может быть положительной, отрицательной или близкой к нулю. Значение PACF на конкретном лаге позволяет оценить значимость и силу связи между значениями ряда на этом лаге, исключая влияние других лагов. Если значение PACF на лаге h значимо и отлично от нуля, это может указывать на наличие прямой связи между значениями ряда на этом лаге, независимо от других лагов.

Глава 2. Практическая часть

2.1. Используемые инструменты

Для анализа, расчетов, реализации алгоритмов и формирования прогнозов использовался язык программирования Python (v3.10.12) и среда разрабоки Google Colab.

Таблица 2.1 Используемые библиотеки Python

Название библиотеки, модуля или метода	Описание
numpy	Библиотека для работы с многомерными массивами и матрицами
pandas	Библиотека для обработки и анализа данных, предоставляет структуры данных, удобные для манипуляции временными рядами и табличными данными
scipy.stats	Модуль библиотеки SciPy, предоставляющий статистические функции и тесты
scipy.special	Модуль SciPy, предоставляющий специальные математические функции
matplotlib.pyplot	Библиотека для создания статических, интерактивных и анимационных графиков в Python
statsmodels.api	Библиотека для статистического моделирования и тестирования гипотез
statsmodels.tsa.seasonal	Метод разложения временных рядов на тренд, сезонность и остатки
statsmodels.tsa.holtwinters	Модель экспоненциального сглаживания для прогнозирования временных рядов

2.2. Наборы данных

В качестве датасетов для построения моделей и формирования прогнозов были взяты стастистические ряды с сайта sophist.hse.ru:

1. Доходы федерального бюджета;

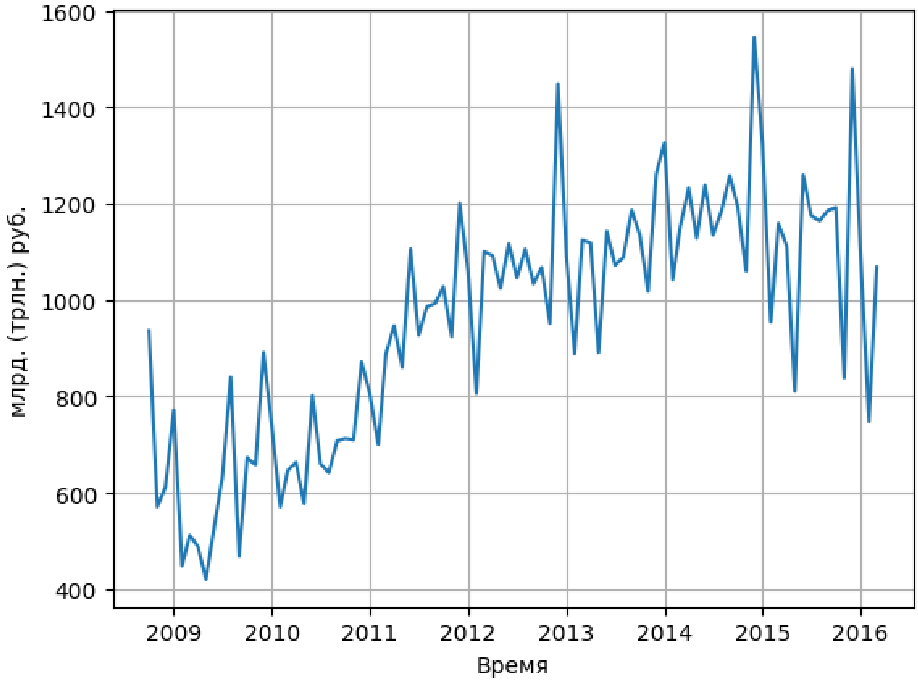

Рис. 2.1. Временной ряд: доходы федерального бюджета

2. Индекс производства в секторе "Обеспечение электрической энергией, газом и паром; кондиционирования воздуха";

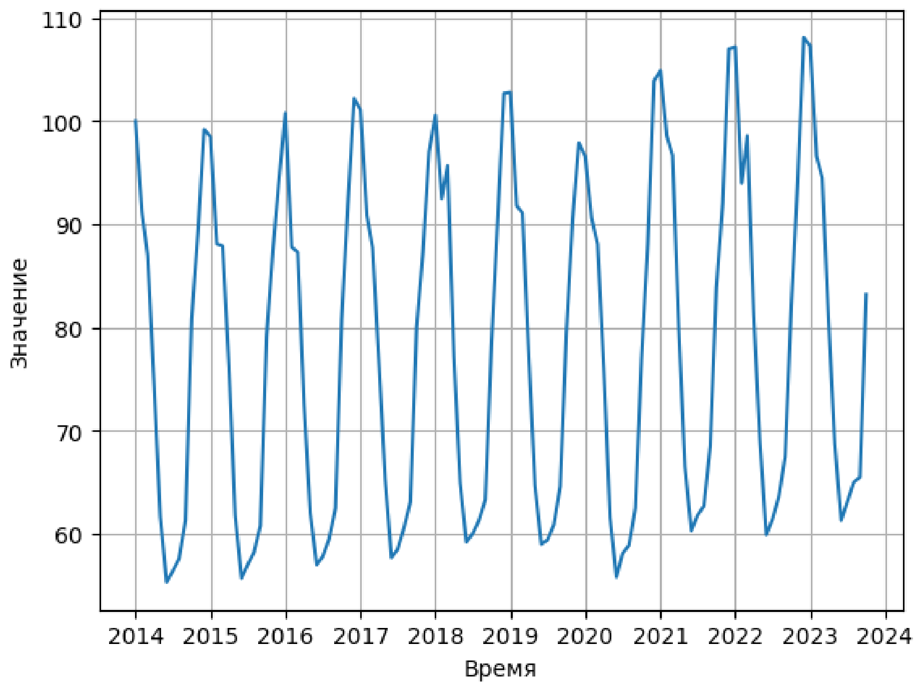

Рис. 2.2. Временной ряд: индекс производства

3. Кредиторская задолженность предприятия;

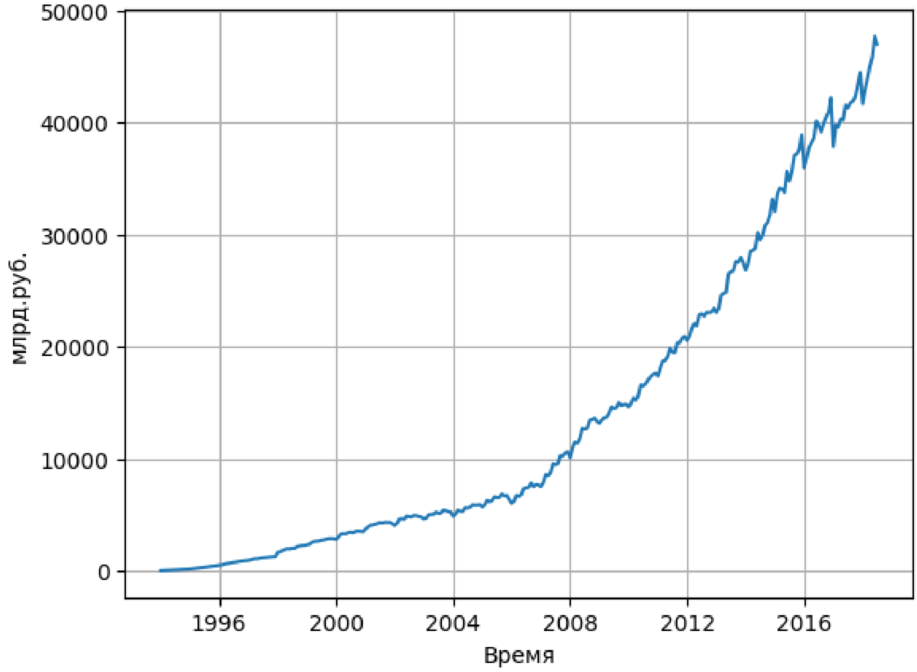

Рис. 2.3. Временной ряд: кредиторская задолженность предприятия

2.3. Анализ данных о доходах федерального бюджета

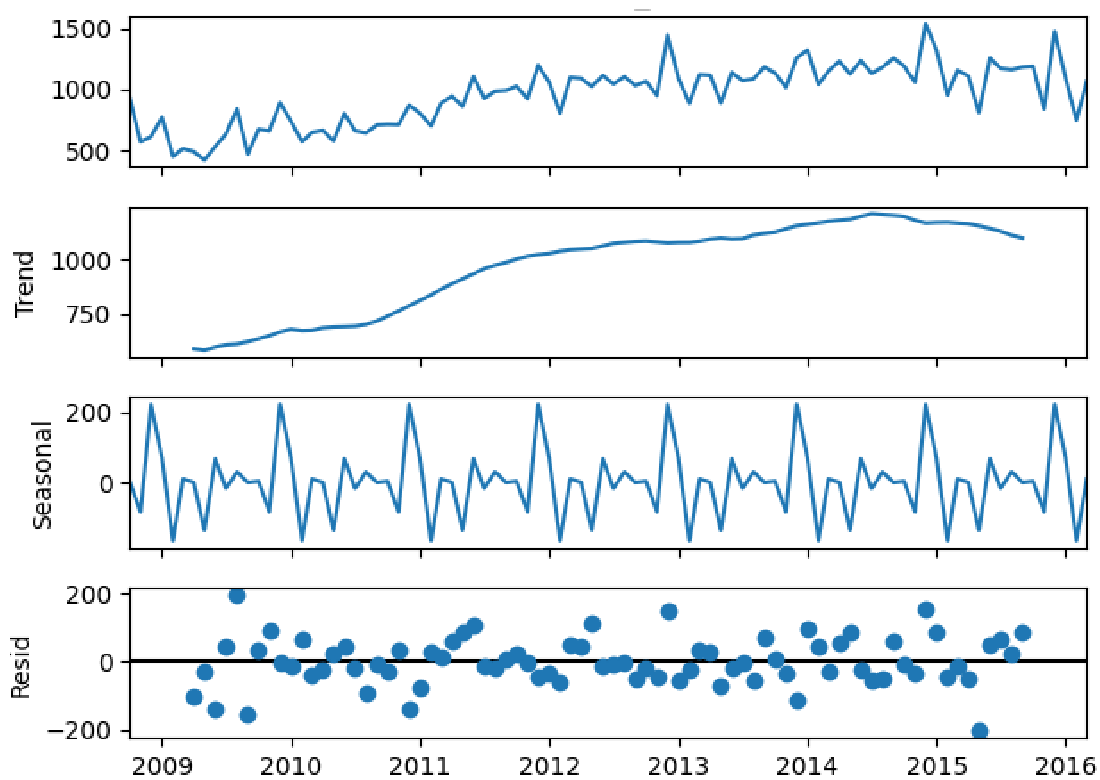

Рис. 2.4. Временной ряд доходов федерального бюджета и его разложение на сезонные компоненты

Для данного ряда был проведен тест ADF, $p = 0.15$, из чего можно сделать вывод, что ряд не является стационарным. Ряд имеет довольно сложную структуру с выраженными сезонностью и трендом. Сгенерируем 5 псевдовыборок с помощью алгоритма LPB и сделаем на их основе прогноз, используя алгоритм беггинга и модели ARIMA и ETS. Построим графики полученных предположений и сравним с предположениями, полученными с помощью моделей от исходного временного ряда.

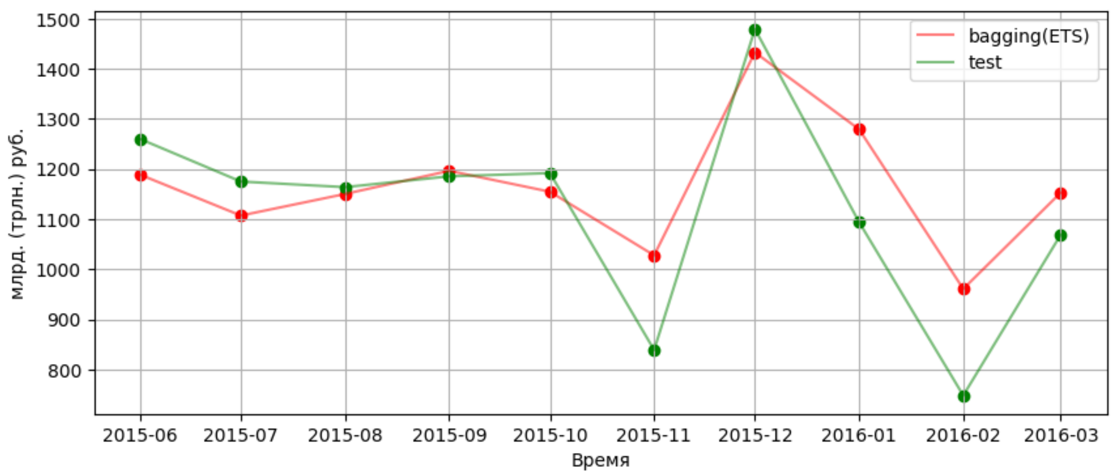

Рис. 2.5. Сравнение прогноза, полученного с помощью беггинга и модели ETS с исходными данными

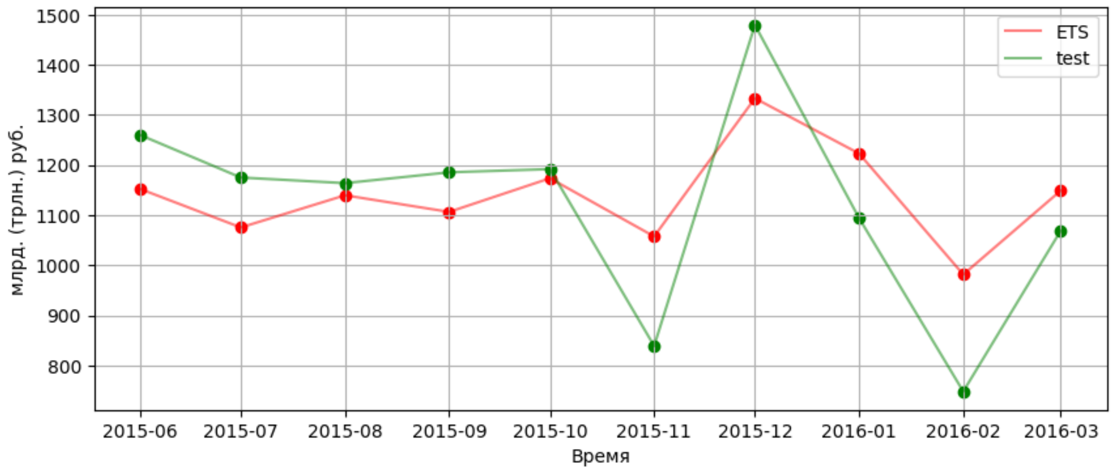

Рис. 2.6. Сравнение прогноза, полученного с помощью модели ETS с исходными данными

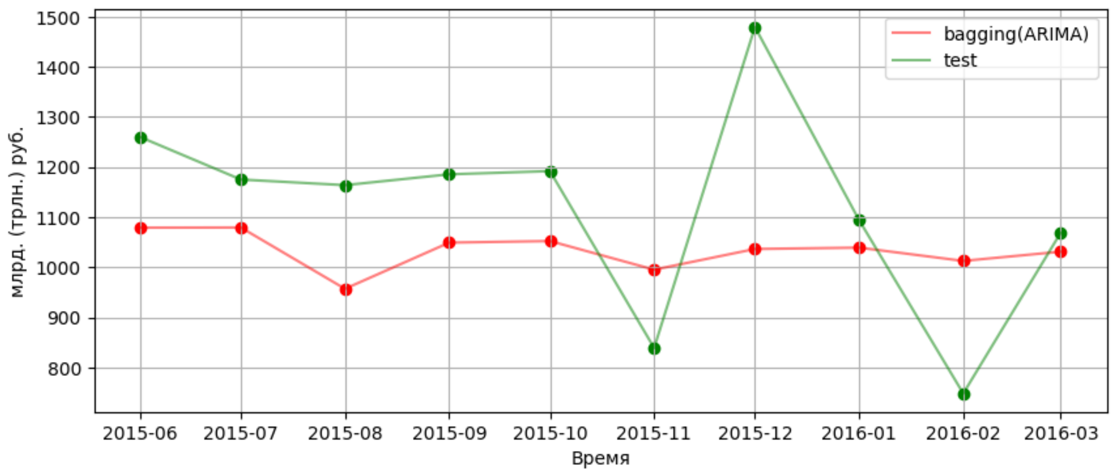

Рис. 2.7. Сравнение прогноза, полученного с помощью беггинга и модели ARIMA с исходными данными

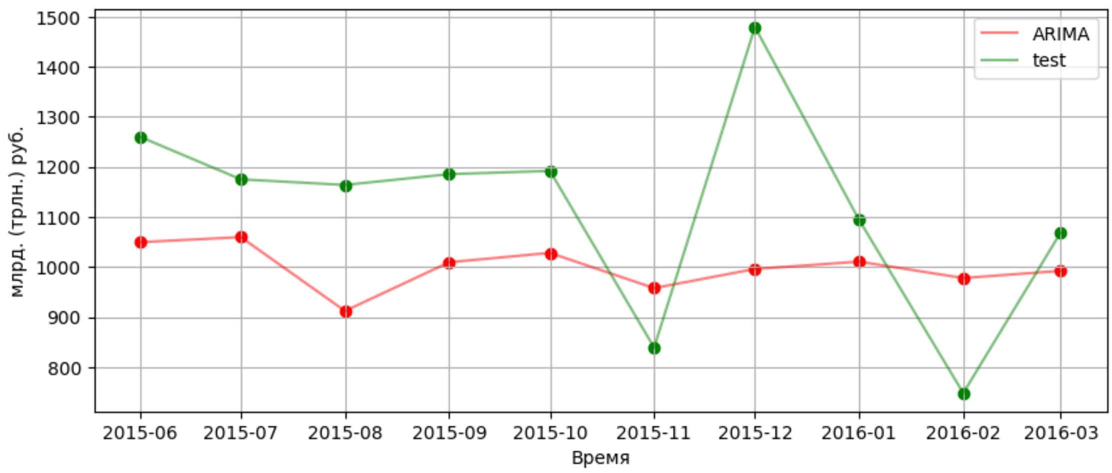

Рис. 2.8. Сравнение прогноза, полученного с помощью модели ARIMA с исходными данными

Таблица 2.2 Доходы федерального бюджета: оценки RMSE и MAE прогнозов различных моделей

Модель	RMSE	MAE
Беггинг (ETS)	116.606174	91.877968
ETS	132.481704	113.660337
Беггинг (ARIMA)	204.521754	171.681403
ARIMA	221.985257	191.051503

Из данной таблицы видно, что беггинг, использующий модель экспоненциального сглаживания, показывает наилучшую производительность с наименьшим значением RMSE и MAE среди всех рассмотренных моделей. Модель ETS, построенная на исходном временном ряде также демонстрирует хорошие результаты с более низкими значениями метрик по сравнению с моделями ARIMA. Оставшиеся две модели показывают менее точные результаты, имея более высокие значения RMSE и MAE. Это может указывать на то, что использование модели ARIMA в данном случае менее эффективно для прогнозирования, однако даже в этом случае применение беггинга в комбинации с моделью ARIMA дало лучший результат, чем построение модели на исходных данных.

2.4. Анализ данных об индексе производства в секторе "Обеспечение электрической энергией, газом и паром"

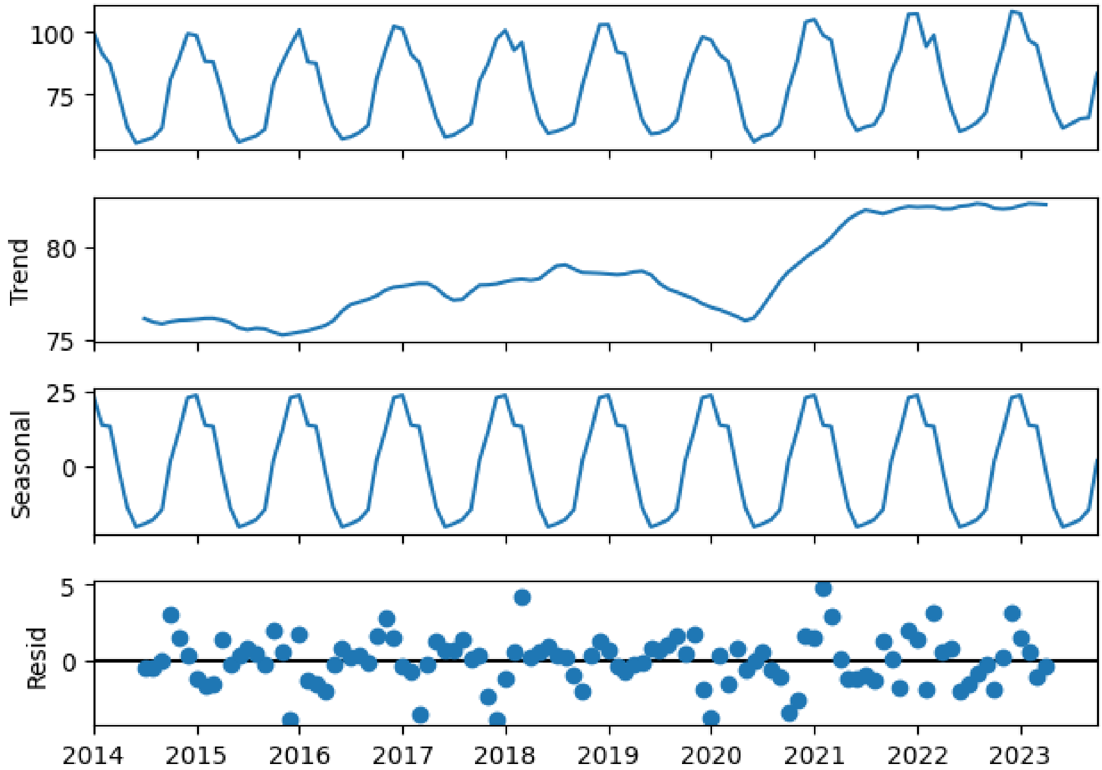

Рис. 2.9. Временной ряд индекса производства и его разложение на сезонные компоненты

Для данного ряда был проведен тест Дики-Фуллера, $p = 0.844$, из чего можно сделать вывод, что ряд не является стационарным. Ряд имеет довольно четкие повторяющиеся паттерны, однако его тренд изменчив. Сгенерируем 5 псевдовыборок с помощью алгоритма LPB и сделаем на их основе прогноз, используя алгоритм беггинга и модели ARIMA и ETS. Построим графики полученных предположений и сравним с предположениями, полученными с помощью моделей от исходного временного ряда.

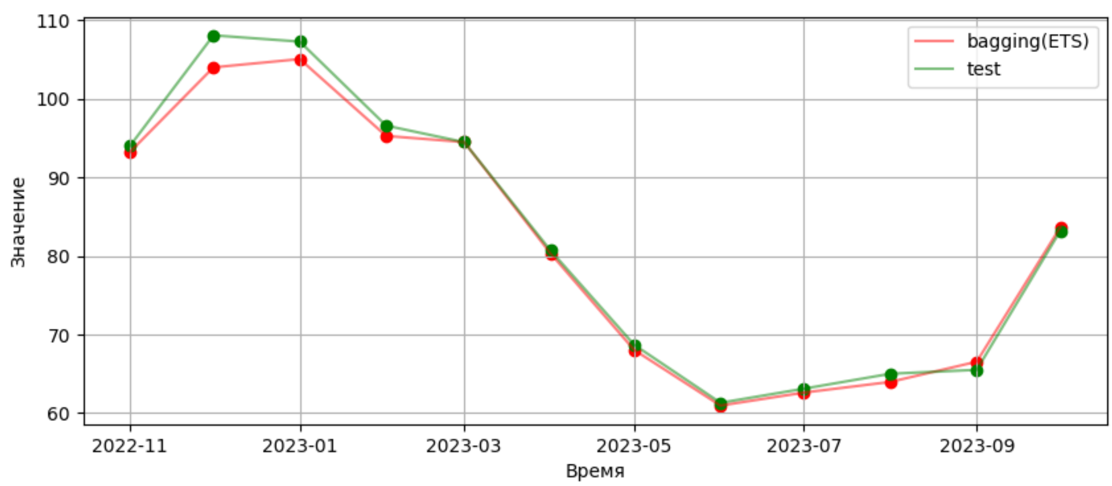

Рис. 2.10. Сравнение прогноза, полученного с помощью беггинга и модели ETS с исходными данными

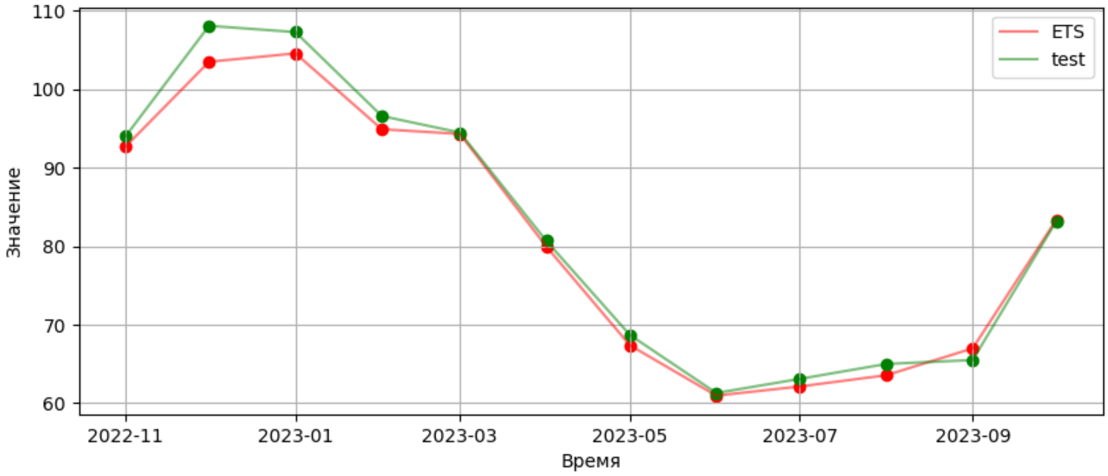

Рис. 2.11. Сравнение прогноза, полученного с помощью модели ETS с исходными данными

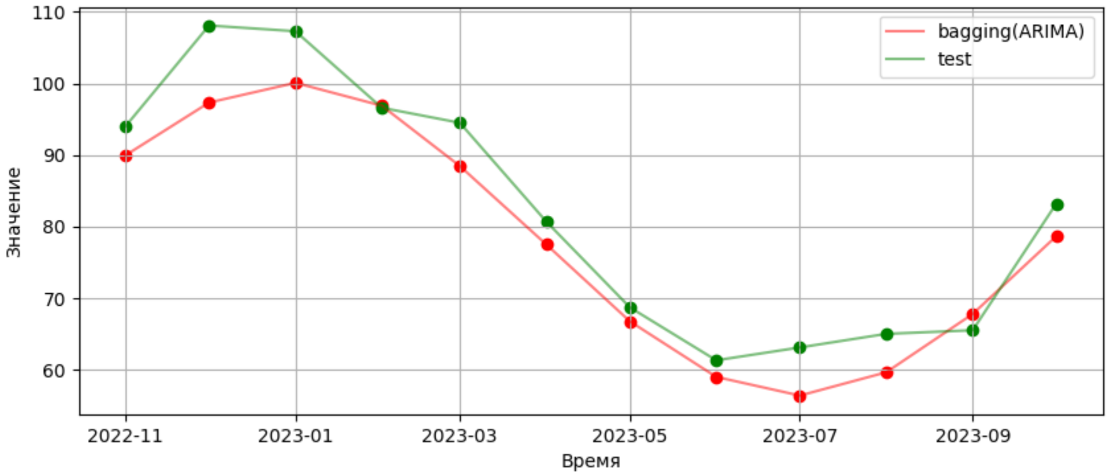

Рис. 2.12. Сравнение прогноза, полученного с помощью беггинга и модели ARIMA с исходными данными

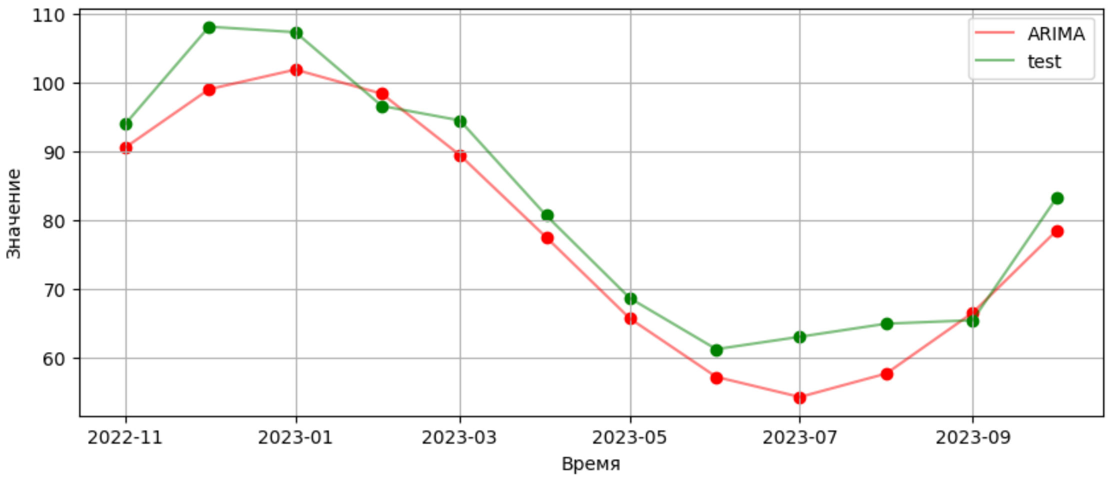

Рис. 2.13. Сравнение прогноза, полученного с помощью модели ARIMA с исходными данными

Таблица 2.3 Индекс производства: оценки RMSE и MAE прогнозов различных моделей

Модель	RMSE	MAE
Беггинг (ETS)	1.505454	1.066151
ETS	1.838627	1.408374
Беггинг (ARIMA)	5.327859	4.564142
ARIMA	5.318232	4.722774

Можно видеть, что беггинг, использующий модель ETS, в этом случае также показывает наилучшие результаты с наименьшими значениями RMSE и MAE. Модель ETS, построенная на исходном временном ряде также демонстрирует довольное хорошие результаты, однако точность ее прогноза уступает алгоритму беггинга с использованием этой же модели. Беггинг с использованием ARIMA и модель, построенная на исходном ряде, показывают довольно близкие результаты, однако они сильно уступают моделям ETS.

2.5. Анализ данных о кредиторской задолженности предприятия

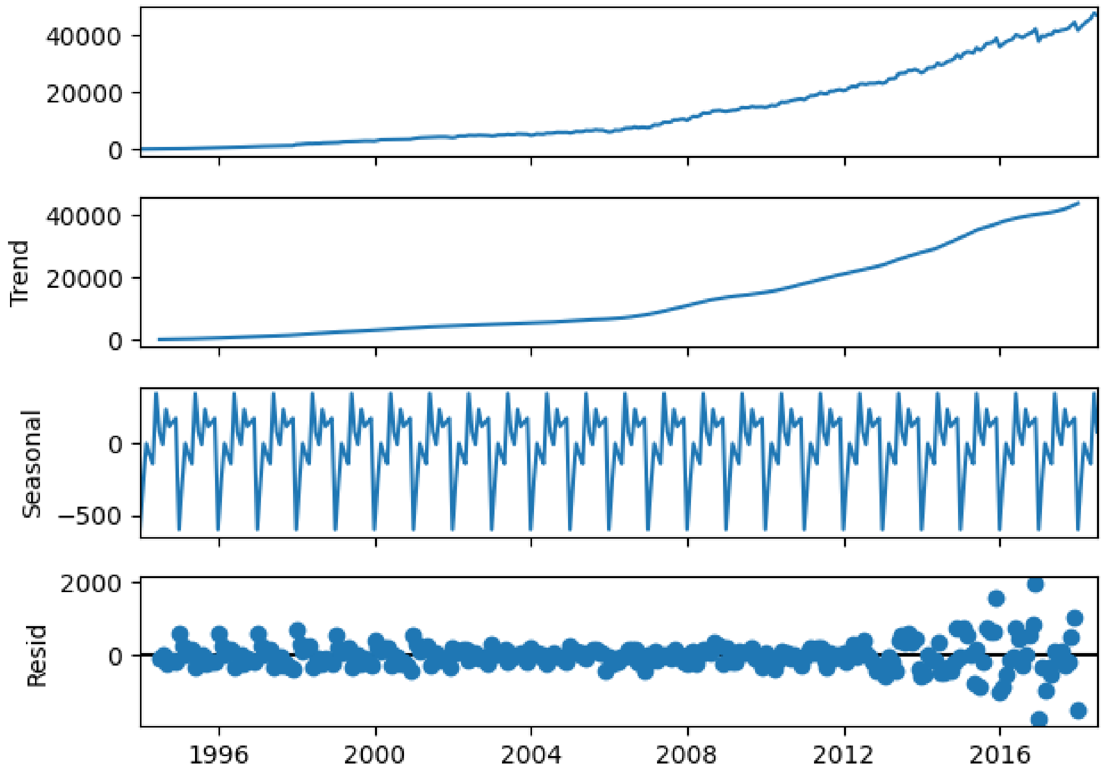

Рис. 2.14. Временной ряд кредиторской задолженности и его разложение на сезонные компоненты

Для данного ряда так же был проведен тест ADF. Было получено значение $p = 0.999$, из чего можно сделать вывод, что ряд не является стационарным. Ряд имеет ярко выраженный растущий тренд. Сгенерируем 5 псевдовыборок с помощью алгоритма LPB и сделаем на их основе прогноз, используя алгоритм беггинга и модели ARIMA и ETS. Построим графики полученных предположений и сравним с предположениями, полученными с помощью моделей от исходного временного ряда.

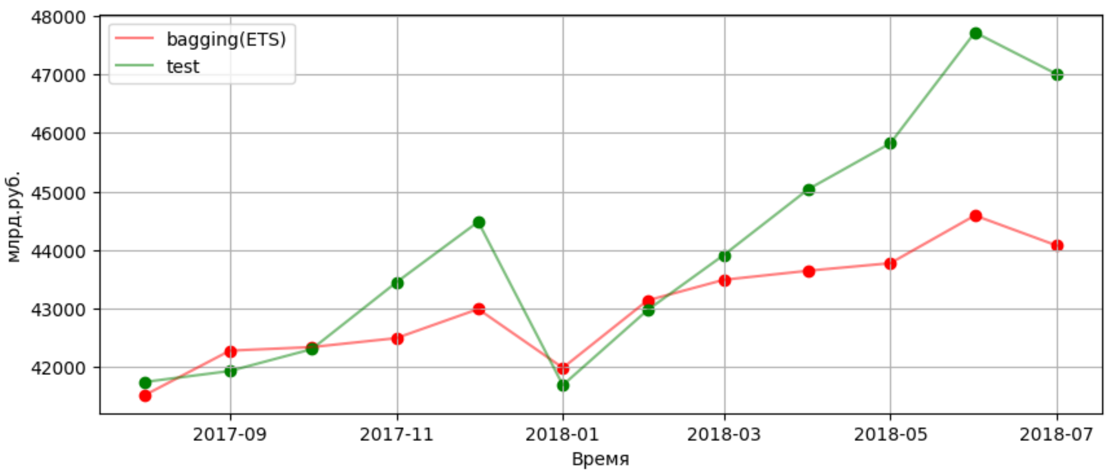

Рис. 2.15. Сравнение прогноза, полученного с помощью беггинга и модели ETS с исходными данными

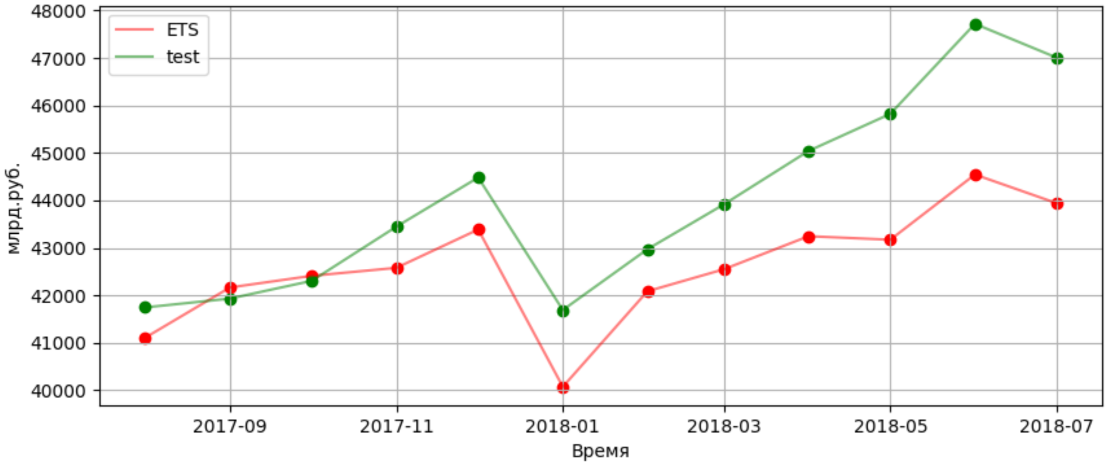

Рис. 2.16. Сравнение прогноза, полученного с помощью модели ETS с исходными данными

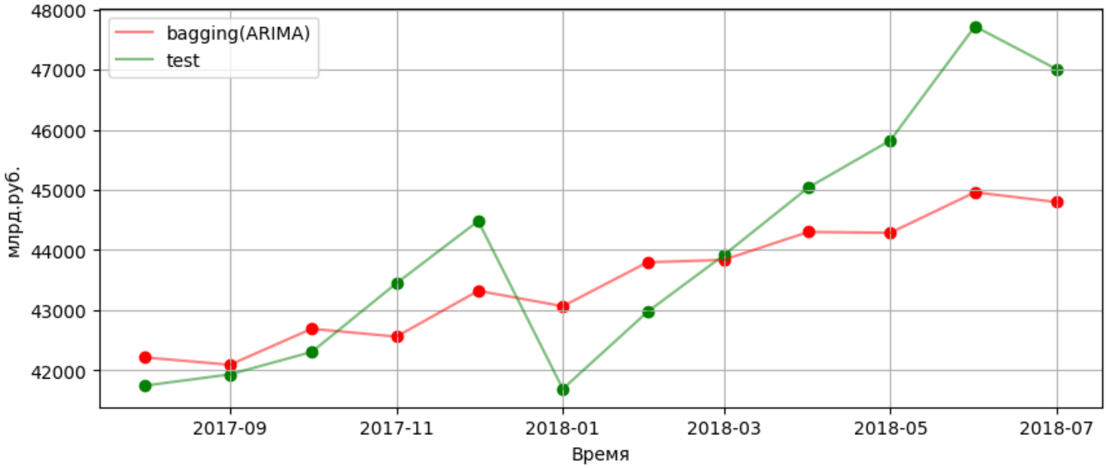

Рис. 2.17. Сравнение прогноза, полученного с помощью беггинга и модели ARIMA с исходными данными

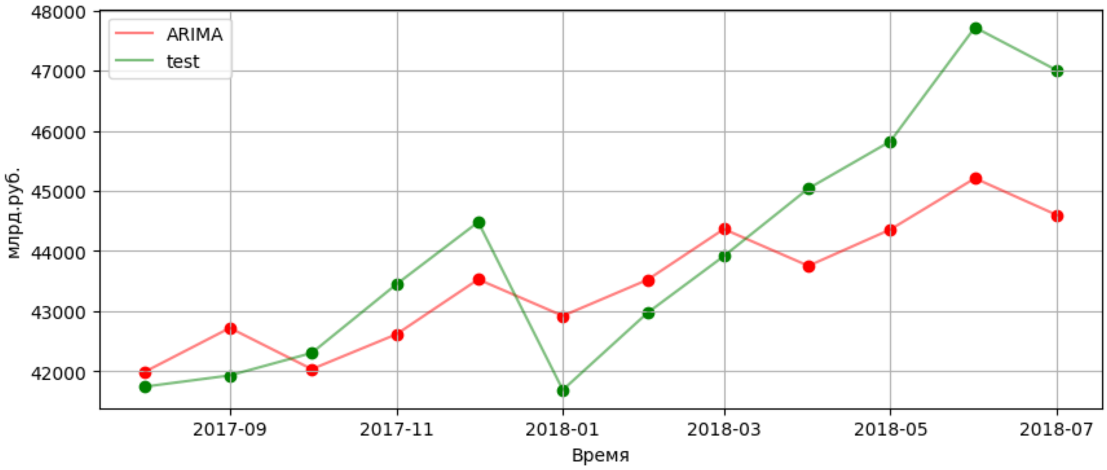

Рис. 2.18. Сравнение прогноза, полученного с помощью модели ARIMA с исходными данными

Таблица 2.4 Индекс производства: оценки RMSE и MAE прогнозов различных моделей

Модель	RMSE	MAE
Беггинг (ETS)	1530.582008	1119.78462
ETS	1765.817015	1457.930387
Беггинг (ARIMA)	1206.865808	948.450332
ARIMA	1299.647913	1082.682861

В данном случае наилучшие показатели демонстрируют модели с использованием ARIMA, а иммено беггинг с использованием ARIMA. Но хотя применение модели ETS в данном случае и дает менее точные предсказания, применение беггинга с этой моделью, улучшают их.

Заключение

В ходе выполнения данной научно-исследовательской работы, мы провели анализ и прогнозирование временных рядов с использованием моделей ARIMA и ETS, изучили основные концепции и подходы к работе с временными рядами, включая разделение на стационарные и нестационарные ряды, метод дифференцирования, оценку автокорреляции и частной автокорреляции, а также реализовали и протестировали метод линейного беггинга.

В ходе тестов мы провели анализ различных временных рядов, проверили их на стационарность и разложили на сезонные компоненты. Затем мы сгенерировали прогнозы, используя модели ARIMA и ETS, а также линейный беггинг в комбинации с этими моделями, после чего построили графики и сравнили с тестовыми данными, а также оценили качество пронозов с помощью метрик RMSE и MAE.

В большинстве случаев применение линейного беггинга дало лучшие результаты, чем простое применение моделей. Таким образом, линейный беггинг сопоставим с другими моделями прогнозирования временных рядов. Однако, прогноз можно улучшить, изменяя весовую функцию и константы, а так же подбирая количество генерируемых псевдовыборок

Список литературы

1. Hyndman R. J., Forecasting: Principles and Practice[Текст]/ Hyndman R. J., Athanasopoulos G. Monash University, Australia. – 2018 - 149 с. [1]
2. Орлов А. И. Прикладная статистика [Текст]/ А. И. Орлов. - Издательство "Экзамен 2004 - 656 с. [2]
3. Box G. E. P. et al. Time series analysis: forecasting and control.[Текст]/ G. E. P. Box, Gwilym M. Jenkins 2015 - John Wiley & Sons - 784 с. [3]
4. Stock J. H. et al. Introduction to econometrics.[Текст]/ J. H. Stock – New York : Pearson, 2012 – Т. 3 [4]
5. Магнус Я. Р. Эконометрика: начальный курс.[Текст]/ Я. Р. Магнус, П. К. Катышев, А. А. Пересецкий – Дело, 1997 [5]
6. Timothy L. McMurry, Dimitris N. Politis, Banded and tapered estimates for autocovariance matrices and the linear process bootstrap[Текст]/ Journal of Time Series Analysis, Wiley Blackwell, vol. 31(6), - 2010 - 471-482 с. [6]
7. Efron, B. Bootstrap Methods: Another Look at the Jackknife [Текст]/ B. Efron. // The Annals of Statistics. 1979. Vol. 7. Bootstrap Methods. № 1. – 1-26 с. [7]

Приложение

Код выполнения работы на языке программирования Python:

```
import numpy as np
import numpy.random as npr
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy.special import boxcox, inv_boxcox
from statsmodels.tsa.seasonal import STL

pip install pmdarima

from statsmodels.tsa.holtwinters import ExponentialSmoothing
from pmdarima.arima import auto_arima
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.nonparametric.smoothers_lowess import lowess

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

from statsmodels.tsa.stattools import adfuller

"""**Linear process bootstrap**"""

# оценка автокорелляционной функции
def wacf(x, lag_max=None):
    if lag_max is None:
        lag_max = len(x) - 1
    n = len(x)
    lag_max = min(lag_max, n - 1)
    if lag_max < 0:
        raise ValueError("'lag_max' must be at least 0")

```

```
    # стандартная оценка acf
    acfest = sm.tsa.acf(x, nlags=n, fft=False)

    # конусная оценка
    s = np.arange(1, n + 1)
    upper = 2 * np.sqrt(np.log10(n) / n)
    ac = np.abs(acfest)
    # находим l
    j = ac < upper
    l = 0
    k = 1
    N = len(j) - 4
    while l < 1 and k <= N:
        if np.all(j[(k-1):(k + 4)]):
            l = k
        else:
            k += 1
    acfest = acfest * kappa(s / l)

    # построим матрицу ковариации
    gamma = acfest
    s = len(gamma)
    Gamma = np.ones((s, s))
    d = np.subtract.outer(np.arange(s), np.arange(s))
    for i in range(1, s):
        Gamma[np.logical_or(d == i, d == -i)] = gamma[i]

    # находим собственные значенияя и векторы с помощью разложения
    eig_vals, eig_vecs = np.linalg.eig(Gamma)

    # сократим собственные значения
    d = np.maximum(eig_vals, 20 / n)

    # строим новую ковариационную матрицу
    Gamma2 = np.dot(np.dot(eig_vecs, np.diag(d)), eig_vecs.T)
    Gamma2 = Gamma2 / np.mean(d)
```

```
    # оцениваем новую ACF
    d = np.subtract.outer(np.arange(s), np.arange(s))
    for i in range(1, s):
        gamma[i] = np.mean(Gamma2[d == i])
    acfest = gamma

    return acfest

# корневая функция
def kappa(x):
    k = np.zeros(len(x))
    x = np.abs(x)
    k[x <= 1] = 1
    k[np.logical_and(x > 1, x <= 2)] = 2 - x[np.logical_and(x > 1, x
  <= 2)]
    return k

# Linear process bootstrap
def lpb(x, nsim=100):
    n = len(x)
    meanx = np.mean(x)
    y = x - meanx
    gamma = wacf(y, n)
    s = len(gamma)
    Gamma = np.ones((s, s))
    d = np.subtract.outer(np.arange(s), np.arange(s))

    for i in range(1, s):
        Gamma[np.logical_or(d == i, d == -i)] = gamma[i]

    L = np.linalg.cholesky(Gamma)
    W = np.linalg.solve(L, np.matrix(y).T)
    out = np.dot(L, npr.normal(size=(n, nsim))) + meanx
    out = np.squeeze(np.asarray(out))
    out = np.transpose(out)
```

```
    return out

"""**Bagging**"""

# алгоритм Бергмайра для формирования выборок из временного ряда
def bergmeir(dataset, is_seasonal, number_of_bs):
  lam = stats.boxcox_normmax(dataset, brack = (0.0, 1.0))
  bc = boxcox(dataset, lam)
  if is_seasonal:
    stl = STL(bc)
    stl = stl.fit()
    seasonal, trend, remainder = stl.seasonal, stl.trend, stl.resid
  else:
    seasonal = pd.Series([0] * len(bc))
    trend, remainder = zip(*lowess(bc.index, bc.values))
    trend = pd.Series(trend)
    remainder = pd.Series(remainder)
  result = [dataset]
  for i in range(number_of_bs):
    restored = trend.values + seasonal.values + lpb(remainder, 1)
    restored = inv_boxcox(restored, lam)
    if np.isnan(sum(restored)): print("restored2 is nan")
    restored = pd.Series(data = restored, index=dataset.index)
    result.append(restored)
  return result

# предсказание с помощью ETS
def ets_forecast(train, test):
  model = ExponentialSmoothing(train, trend="add", seasonal="add")
  model_fit = model.fit()
  forecast = model_fit.forecast(len(test))
  return forecast

def arima_forecast(train, test, model_params):
  model = auto_arima(train, **model_params)
  forecast = model.predict(len(test))
```

```
  return forecast

# предсказание с помощью бэггинга
def bagging_forecast(train, test, model_name):
  forecasts = []
  if model_name == "arima":
    arima = auto_arima(train[0])
    arima_params = arima.get_params()
  # получаем предсказания для псевдовыборок
  for i, sample in enumerate(train):
    if model_name == "ets":
      fc = ets_forecast(train[i], test)
    if model_name == "arima":
      fc = arima_forecast(train[i], test, arima_params)
    forecasts.append(fc)
    #print(f"forecast for train[{i}]:", fc)
  return np.mean(forecasts, axis = 0)

def printer(t, p):
  print("Pred: \t", p)
  print("Test: \t", t.values)
  RMSE = mean_squared_error(t, p, squared = False)
  print('RMSE: ', RMSE)
  MAE = mean_absolute_error(t, p)
  print('MAE: ', MAE)
  plt.figure(figsize=(10,4))
  plt.scatter(t.index, p, color = 'red')
  plt.plot(t.index, p, color = 'red', alpha = 0.5)
  plt.scatter(t.index, t, color = 'green')
  plt.plot(t.index, t, color = 'green', alpha = 0.5)
  plt.show()

def test_stationarity(timeseries):
    # Определение статистики ряда
    rolling_mean = timeseries.rolling(window=12).mean()
    rolling_std = timeseries.rolling(window=12).std()
```

```
    # Визуализация скользящего среднего и стандартного отклонения
    plt.plot(timeseries, label='Original')
    plt.plot(rolling_mean, label='Rolling Mean')
    plt.plot(rolling_std, label='Rolling Std')
    plt.legend()
    plt.show()

    # Тест ДикиФуллера-
    result = adfuller(timeseries, autolag='AIC')
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])
    print('Critical Values:', result[4])

    # Проверка стационарности по p-value
    if result[1] <= 0.05:
        print("Ряд стационарен")
    else:
        print("Ряд нестационарен")

"""Загрузка датасетов"""

class SophistHSE:
    def __init__(self):
        self.url = 'http://sophist.hse.ru/hse/1/tables/'

    #Transform str index to datetime index
    def __time_to_datetime__(self, df):
        current_year = df.index[0].split()[0]
        new_index = []
        for index, row in df.iterrows():
            lst = index.split()
            if len(lst) == 2:
                current_year = lst[0]
                new_index.append(pd.to_datetime(index))
            else:
```

```
                    new_index.append(pd.to_datetime(current_year + ' ' +
    lst[0]))

        df.index = new_index
        return df

    #Parse Table
    def get_table(self, table):
        df = pd.read_html(self.url + table + '.htm', index_col = 0,
    decimal = ',', thousands = None,  na_values = ' ')[0]
        df.rename(columns = df.iloc[0], inplace=True)
        df = df[df.index.notna()]
        df = df.drop(index = ['T'])
        df = self.__time_to_datetime__(df).astype(float)
        return df

"""dataset #1: Доходы федерального бюджета"""

HSE = SophistHSE()
table = 'GOV_M' #Put here name of dataset
df = HSE.get_table(table)

df

df = df['FBREV_M']

df = df.drop(df.index[-90:])

df = df.tail(90)

plt.plot(df)

# Добавьте пояснения к осям
plt.xlabel('Время')
plt.ylabel('млрд. трлн(.) руб.')
plt.grid(True)
```

```
# Отобразите график
plt.show()

train = df.head(80)
test = df.tail(10)

from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df, model='additive', period=12)
plt.figure()
result.plot()
plt.gcf().suptitle('')
plt.show()

bs = bergmeir(train, is_seasonal=True, number_of_bs=5)
print("ets:")
bagging_ets_forecast = bagging_forecast(bs, test, "ets")
ets_pred = ets_forecast(bs[0], test)
print("arima:")
bagging_arima_forecast = bagging_forecast(bs, test, "arima")

model = auto_arima(train)
arima_pred = model.predict(start=test.index[0], end=test.index[-1])

train_diff = train.diff(1).dropna()
model = auto_arima(train_diff)
arima_pred = model.predict(start=test.index[0], end=test.index[-1])
cumsum = train.iloc[0] + train_diff.cumsum()

printer(test, bagging_ets_forecast)

t = test
p = bagging_ets_forecast
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='bagging(ETS)')
plt.scatter(t.index, t, color = 'green')
```

```
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрд. трлн(.) руб.')
plt.legend()
plt.grid(True)
plt.show()

printer(test, ets_pred)

t = test
p = ets_pred
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='ETS')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрд. трлн(.) руб.')
plt.legend()
plt.grid(True)
plt.show()

printer(test, arima_pred)

t = test
p = arima_pred
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='ARIMA')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрд. трлн(.) руб.')
plt.legend()
plt.grid(True)
plt.show()
```

```
printer(test, bagging_arima_forecast)

t = test
p = bagging_arima_forecast
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5,
    label='bagging(ARIMA)')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрд. трлн(.) руб.')
plt.legend()
plt.grid(True)
plt.show()

"""dataset#2 Индекс производства в секторе Обеспечение" электрической энергией,
    газом и паром; кондиционирования воздуха"
"""

table2 = 'IP2_DEA_M'
df2 = HSE.get_table(table2)

df2

df2 = df2['IP2_DEA_M']

df2 = df2.drop(df2.index[-10:])

df2 = df2.drop(df2.index[:10])

plt.plot(df2)

# Добавьте пояснения к осям
plt.xlabel('Время')
```

```
plt.ylabel('Значение')
plt.grid(True)
# Отобразите график
plt.show()

train2 = df2.head(-12)
test2 = df2.tail(12)

result = seasonal_decompose(df2, model='additive', period=12)
result.plot()
plt.show()

test_stationarity(df2)

bs2 = bergmeir(train2, is_seasonal=True, number_of_bs=5)
print("ets:")
bagging_ets_forecast_2 = bagging_forecast(bs2, test2, "ets")
ets_pred_2 = ets_forecast(bs2[0], test2)
print("arima:")
bagging_arima_forecast_2 = bagging_forecast(bs2, test2, "arima")
model = auto_arima(train2)
arima_pred_2 = model.predict(len(test2))

printer(test2, bagging_ets_forecast_2)

t = test2
p = bagging_ets_forecast_2
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='bagging(ETS)')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)
```

```
plt.show()

printer(test2, ets_pred_2)

t = test2
p = ets_pred_2
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='ETS')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)
plt.show()

printer(test2, bagging_arima_forecast_2)

t = test2
p = bagging_arima_forecast_2
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5,
    label='bagging(ARIMA)')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)
plt.show()

printer(test2, arima_pred_2)

t = test2
```

```
p = arima_pred_2
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='ARIMA')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)
plt.show()

"""dataset #3: Кредиторская задолженность предприятия"""

table3 = 'FINENT_M'
df3 = HSE.get_table(table3)

df3

df3 = df3['LIAB_T_M']

df3 = df3.drop(df3.index[-60:])

df3 = df3.drop(df3.index[:30])

df3 = df3.dropna()

plt.plot(df3)

# Добавьте пояснения к осям
plt.xlabel('Время')
plt.ylabel('млрдруб..')
plt.grid(True)
# Отобразите график
plt.show()

```

```python
test_stationarity(df3)

train3 = df3.head(-12)
test3 = df3.tail(12)

result = seasonal_decompose(df3, model='additive', period=12)
result.plot()
plt.show()

bs3 = bergmeir(train3, is_seasonal=True, number_of_bs=20)
print("ets:")
bagging_ets_forecast_3 = bagging_forecast(bs3, test3, "ets")
ets_pred_3 = ets_forecast(bs3[0], test3)
print("arima:")
bagging_arima_forecast_3 = bagging_forecast(bs3, test3, "arima")
model = auto_arima(train3)
arima_pred_3 = model.predict(len(test3))

printer(test3, bagging_ets_forecast_3)

t = test3
p = bagging_ets_forecast_3
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='bagging(ETS)')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрдруб..')
plt.legend()
plt.grid(True)
plt.show()

printer(test3, ets_pred_3)

t = test3
```

```
p = ets_pred_3
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='ETS')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрдруб..')
plt.legend()
plt.grid(True)
plt.show()

printer(test3, bagging_arima_forecast_3)

t = test3
p = bagging_arima_forecast_3
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5,
    label='bagging(ARIMA)')
plt.scatter(t.index, t, color = 'green')
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрдруб..')
plt.legend()
plt.grid(True)
plt.show()

printer(test3, arima_pred_3)

t = test3
p = arima_pred_3
plt.figure(figsize=(10,4))
plt.scatter(t.index, p, color = 'red')
plt.plot(t.index, p, color = 'red', alpha = 0.5, label='ARIMA')
plt.scatter(t.index, t, color = 'green')
```

```
plt.plot(t.index, t, color = 'green', alpha = 0.5, label='test')
plt.xlabel('Время')
plt.ylabel('млрдруб..')
plt.legend()
plt.grid(True)
plt.show()
```